

# COMPUTER SCIENCE

## HIGHER SECONDARY - SECOND YEAR

### Tools & Objects Technology

Untouchability is a sin  
Untouchability is a crime  
Untouchability is inhuman



TAMILNADU TEXTBOOK AND  
EDUCATIONAL SERVICES CORPORATION  
College Road, Chennai – 600 006.

© Government of Tamilnadu

First Edition – 2006

Reprint – 2017

***Chairman Syllabus Committee***

Dr. Balagurusamy E, Former Vice Chancellor, Anna University, Chennai

***Co-Ordinator Textbook Writing***

Dr. Sankaranarayanan V, Director, Tamil Virtual University, Chennai

***Authors***

Dr. M Ponnavaikko, SRM Institute of Science and Technology, Chennai

Ms. Vasanthi Krishnakumar, M/s Sify Limited, Chennai

Dr. Gopal T V, Anna University, Chennai

Ms. Rukmani K, Padma Seshadri Bala Bhavan, Chennai

Dr. Sankaranarayanan V, Director, Tamil Virtual University, Chennai

***Reviewers***

Dr. Gopal T V, Anna University, Chennai

Ms. Vasanthi Krishnakumar, M/s Sify Limited, Chennai

***Copy-Editor***

Ms. Subha Ravi, Director, M/s Digiterati Consultancy Pvt. Ltd, Chennai

**Price : Rs.**

This book has been prepared by the Directorate of School Education on behalf of the Government of Tamilnadu

This book has been printed on 70 G.S.M. Paper

Printed by Web Offset at :

## FOREWORD

We are in the midst of an information revolution. The driving forces behind this revolution are computer and communication technologies. Individuals and organizations today use computers and computer software tools in managing many of their daily tasks. Computers are making their way into our homes, our schools, our hospitals and almost every facet of our lives. They are changing the way people live and work, the way organizations manage their tasks and resources, the way the governments govern their states and people, and the way many systems behave and operate. The applications of these versatile technologies seem almost limitless.

Application tasks may broadly be classified into the following major categories:

- Data processing and management (commercial use)
- Word Processing (office and educational use)
- Message communication (e-mail)
- Image processing (animation and industrial use)
- Voice recognition (multimedia)
- Numerical computing (scientific use)

This Book presents and discusses software tools that are necessary to carry out some of these tasks in organizations. Intelligent use of these tools would certainly improve the efficiency and effectiveness of organizations.

Many software tools for processing information are available in the open market. In this Book, a popular software package known as StarOffice developed by Sun Microsystems, USA is used to demonstrate the various applications. StarOffice is a collection of tools such as Word Processing, Spreadsheet, Database and Multimedia.

I strongly believe in the dictum:

I hear, I forget

I see, I remember

I DO, I UNDERSTAND

The emphasis throughout the volume is hands-on experience. More than fifty percent of learning time is provided for practising the tools.

Improvements in computer and communication technologies are continuously under way. Young thinkers like you will shape the world of the future. Without your dreams and hard work, the ideas of today won't develop into the technologies of tomorrow.

“Imagination is more important than knowledge” – **Albert Einstein**

The authors have taken great care in making the material presented comprehensive, lucid and practical with many illustrations. The authors, the reviewers, and the officials of the Directorate of School Education deserve our appreciation and thanks for accomplishing this trying task in record time.

“In human affairs we have reached a point where the problems that we must solve are no longer solvable without the aid of computers. I fear not computers but the lack of them”

**-Issac Asimov**

Computers are machines that can help us solve complex scientific, business and administrative problems. They have helped automation of many industrial and business systems. However, we must remember that they are machines, created and managed by men. They have no brain of their own. Anything they do is the result of human instructions. They carry out the instructions obediently as long as the instructions can be executed using the available hardware, no matter whether they are right or wrong. That is, computers lack common sense.

Computers need clear instructions to tell them what to do, how to do, and when to do. The way of providing such instructions to computers is called programming. The language used in construction and communication of these instructions is known as a programming language.

There are over 200 programming languages currently in use. Some were designed for scientific use, some for commercial applications while some others were meant for more general-purposes . A programming language should have features that would facilitate programmers in making and designing the solution steps easily.

We have already learned the C language, which is a procedure- oriented language. As the name implies, the emphasis was on solution procedures. C is a powerful general-purpose language. This book presents the advanced version of C know as C++. C++ supports a totally new concept of object-oriented programming (OOP) and therefore it is classified as an object-oriented technology. We chose C++ because it has become an industry-standard OOP language today.

In OOP languages such as C++, the emphasis is on the entities of the physical world called objects. These objects may represent a person, a car, a table of data, or any item that the program must handle. We human beings normally look at real-life problems as a collection of distinct objects and try to solve them taking into account the relationship among the objects. In a similar way, in C++, programming problems are analyzed in terms of objects and the nature of communications between them.

Numerous examples and illustrative programs presented in the volume are meant to be both simple and educational. It is hoped that the material provided will help the students to quickly move into the world of C++ and object-oriented programming.



This volume also presents many IT enabled applications with a visual support in the form of animated content (in a separate CD). Ethical use of IT has been highlighted in all applications.

I would like to place on record our sincere appreciation and thanks to all the authors, reviewers and the Directorate of School Education officials for their excellent work and co-operation.

**E. BALAGURUSAMY**

Chairman

Syllabus Review Committee

# CONTENTS

<b>CHAPTER 1</b>	<b>AN INTRODUCTION TO STAROFFICE WRITER</b>	
1.1	An Introduction to StarOffice	1
1.2	Creating a New Document using StarOffice Writer	2
1.3	Entering Text in the Document	3
1.4	Saving, Closing and Opening Documents	3
1.5	Moving Around the Document	6
1.6	Scrolling the Document	7
1.7	Correcting Mistakes	8
1.8	Inserting Text	8
1.9	Selecting Text	8
1.10	Moving the Text	10
1.11	Copying the Text	10
1.12	Finding and Replacing the Text	10
	Summary	12
	Exercises	13
<b>CHAPTER 2</b>	<b>TEXT FORMATTING</b>	
2.1	Formatting Options	14
2.2	Paragraph Alignment	17
2.3	Indenting Text	19
2.4	Modifying Line Spacing	22
2.5	Creating Bullets and Numbered List	23
2.6	Formatting Using Styles	25
2.7	StarOffice Help	26
	Summary	28
	Exercises	29

<b>CHAPTER 3</b>	<b>CORRECTING SPELLING MISTAKES</b>	
3.1	Checking Spelling while Typing	30
3.2	Checking the Spelling after the Document is Typed	31
3.3	AutoCorrect Option	32
3.4	Creating Autocorrect Entry	32
	Summary	34
	Exercises	35
 <b>CHAPTER 4</b>	 <b>WORKING WITH TABLES</b>	
4.1	Creating a Simple Table	36
4.2	Entering Data in the Table	37
4.3	Adding or Deleting Rows and Columns	37
4.4	Changing the Row/Column Width	38
4.5	Table Formatting Toolbar	39
	Summary	40
	Exercises	41
 <b>CHAPTER 5</b>	 <b>PAGE FORMATTING</b>	
5.1	Changing the Margin	42
5.2	Changing Page Orientation	43
5.3	Creating a Header and Footer	43
	Summary	46
	Exercises	46
 <b>CHAPTER 6</b>	 <b>SPREADSHEET</b>	
6.1	Introduction	47
6.2	Working with StarOffice Calc	50
6.3	Editing the Data in the Worksheet	54
6.4	Creating Formulae	55
6.5	Fill Command	57
6.6	Cell Referencing	59

6.7	Using Functions	60
6.8	Date Arithmetic	63
6.9	Formatting the Worksheet	63
6.10	Changing Column Width and Row Height	66
6.11	Inserting Cells, Rows and Columns	67
6.12	Deleting Cells, Rows and Columns	69
6.13	Inserting Pictures and Special Characters	70
6.14	Drawing in a Spreadsheet	71
6.15	Inserting Objects	72
6.16	Working with Charts	73
6.17	Working with Multiple Sheets	76
6.18	Printing Worksheets	77
6.19	Database Function in StarOffice Calc	78
	Summary	78
	Exercises	79

## **CHAPTER 7      DATABASE**

7.1	Introduction	81
7.2	Data and Information	81
7.3	Data Processing	82
7.4	Database	85
7.5	Basic Concepts of Database Management Systems (DBMS)	89
7.6	Working with StarOffice Base	89
7.7	Integrating with Office Automation Applications	112
	Summary	120
	Exercises	121

## **CHAPTER 8      INTRODUCTION TO MULTIMEDIA**

8.1	What is Multimedia?	123
8.2	Multimedia Applications	123
8.3	Multimedia Elements - Sound, Animation, and Video	125

8.4	Using Multimedia Elements in Content	130
	Summary	131
	Exercises	132
<b>CHAPTER 9</b>	<b>PRESENTATION</b>	
9.1	Introduction	133
9.2	A Basic Presentation	133
9.3	Managing a Presentation	140
9.4	Customizing a Presentation	154
9.5	Printing Presentations	158
	Summary	167
	Exercises	169
<b>APPENDIX FURTHER READING</b>		170
<b>CHAPTER 10</b>	<b>OBJECT ORIENTED CONCEPTS USING C++</b>	
10.1	Object Oriented Paradigm	171
10.2	Polymorphism	173
10.3	Inheritance	174
10.4	A Practical Example	
	Domestic Waterusage	175
	Exercises	177
<b>CHAPTER 11</b>	<b>OVERVIEW OF C++</b>	
11.1	Introduction	178
11.2	Basic Data Types	178
11.3	Data Types	194
11.4	Variables	202
	Exercises	209

## **CHAPTER 12    BASIC STATEMENT**

12.1	Input /Output Statements	212
12.2	My First C++ Program – Structure or A C++ Program	214
12.3	Declaration Statement	214
12.4	Assignment Statements	215
12.5	Control Structures	215
12.6	Program Development	239
	Exercises	240

## **CHAPTER 13    FUNCTIONS C++ ENHANCEMENTS**

13.1	Introduction	245
13.2	Function Prototyping	246
13.3	Calling a Function	248
13.4	Parameters passing in functions	248
13.5	Returning Values	256
13.6	Inline Functions	259
13.7	Scope rules of variables	260
	Exercises	263

## **CHAPTER 14    STRUCTURED DATA TYPE-ARRAYS**

14.1	Introduction	268
14.2	Single Dimension Array	269
14.3	Strings	273
14.4	Two dimensional arrays	277
14.5	Array of Strings	283
	Exercises	285

## **CHAPTER 15    CLASSES AND OBJECTS**

15.1	Introduction to Classes	290
15.2	Specifying the members of a class	290
15.3	Data Abstraction	292

15.4	Members and Member Functions	293
15.5	Creating Objects of a class	293
15.6	Accessing class members using dot operator	294
15.7	Defining methods of a class	295
15.8	Memory allocation of objects	297
15.9	Static members of a class	298
15.10	Arrays of Objects	300
	Exercises	301
<b>CHAPTER 16</b>	<b>POLYMORPHISM</b>	
16.1	Introduction	304
16.2	Function overloading	304
16.3	Operator overloading	308
	Exercises	315
<b>CHAPTER 17</b>	<b>CONSTRUCTORS AND DESTRUCTORS</b>	
17.1	Introduction	317
17.2	Constructors	317
17.3	Functions of Constructors	318
17.4	Constructor Overloading	318
17.5	Rules for constructor definition and usage	323
17.6	Destructors	323
17.7	The rules for destructor definition and usage are	323
	Exercises	324
<b>CHAPTER 18</b>	<b>INHERITANCE</b>	
18.1	Introduction	325
18.2	Advantages of inheritance	325
18.3	Derived class and Base Classes	326
18.4	Visibility Mode	328
18.5	Types of Inheritance	331
	Exercises	333

## **CHAPTER 19      IMPACT OF COMPUTERS ON SOCIETY**

19.1	Introduction	337
19.2	Computers for Personal Use	337
19.3	Computerized Homes	338
19.4	Home Banking and Shopping	339
19.5	Computers in Education	340
19.6	Computers in Entertainment	342
19.7	Computers in Healthcare	344
19.8	Computers in Agriculture	345
19.9	Internet in real time Applications	346
	Exercises	346

## **CHAPTER 20      IT ENABLED SERVICES**

20.1	Introduction	347
20.2	e-Governance	348
20.3	Call Centres	349
20.4	Data Management	349
20.5	Medical Transcription and Tele-Medicine	350
20.6	Data Digitization	351
20.7	Web Based Services	352
	Exercises	352

## **CHAPTER 21      COMPUTER ETHICS**

21.1	Data Security	353
21.2	Computer Crime	354
21.3	Cracking	356
21.4	Work, Family and Leisure	356
	Exercises	356



## CHAPTER 1

# AN INTRODUCTION TO STAROFFICE WRITER

## 1.1 An Introduction to StarOffice

StarOffice is an application that is designed to work on different operating systems. StarOffice is a full-featured office productivity suite with powerful standalone applications that can also open, edit, and save Microsoft Office documents. Whether you are writing a letter, performing complex calculations, creating a presentation, or creating a database, you'll find that StarOffice meets your needs.

StarOffice consists of several applications. These applications are grouped together into an integrated environment in which one can do several things. Listed below are some of the StarOffice functions,

Create text document using StarOffice Writer

Create spreadsheets using StarOffice Calc

Create presentations using StarOffice Impress

Draw using StarOffice Draw

Create a database using StarOffice Base

To Start StarOffice,

1. On the taskbar, click the **Start** button.
2. Choose All **Programs** → **StarOffice 8** folder.
3. Click the StarOffice program that you want to start.

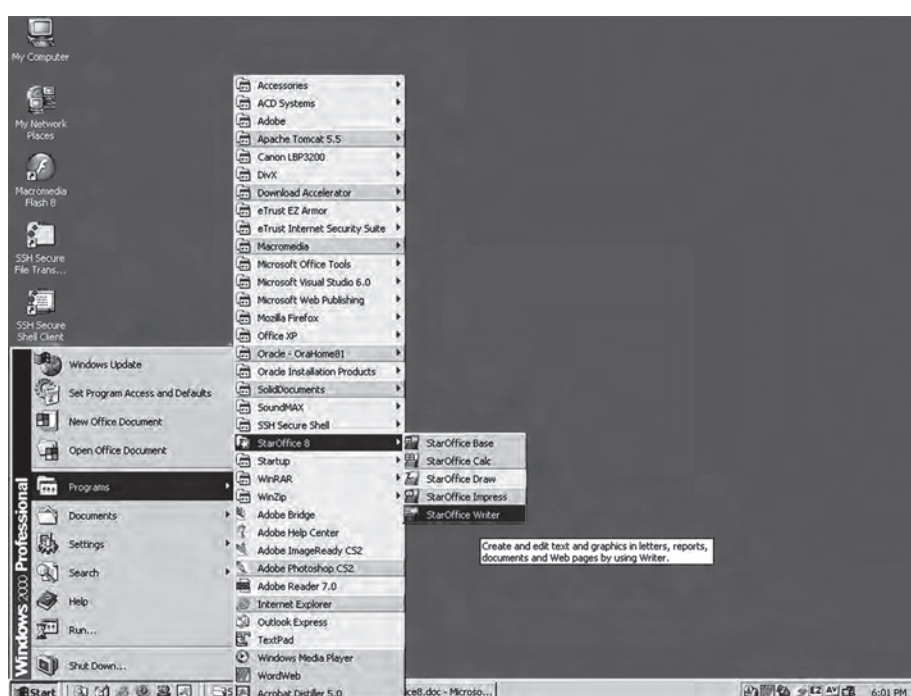


Fig 1.1 Selecting StarOffice Writer from Start Menu

StarOffice Writer is a word processor. The term word processing refers to the activity carried out using a computer and suitable software to create, view, edit, manipulate, transmit, store, retrieve and print documents. A document may contain text, tables, graphs, charts, equations, pictures and drawings.

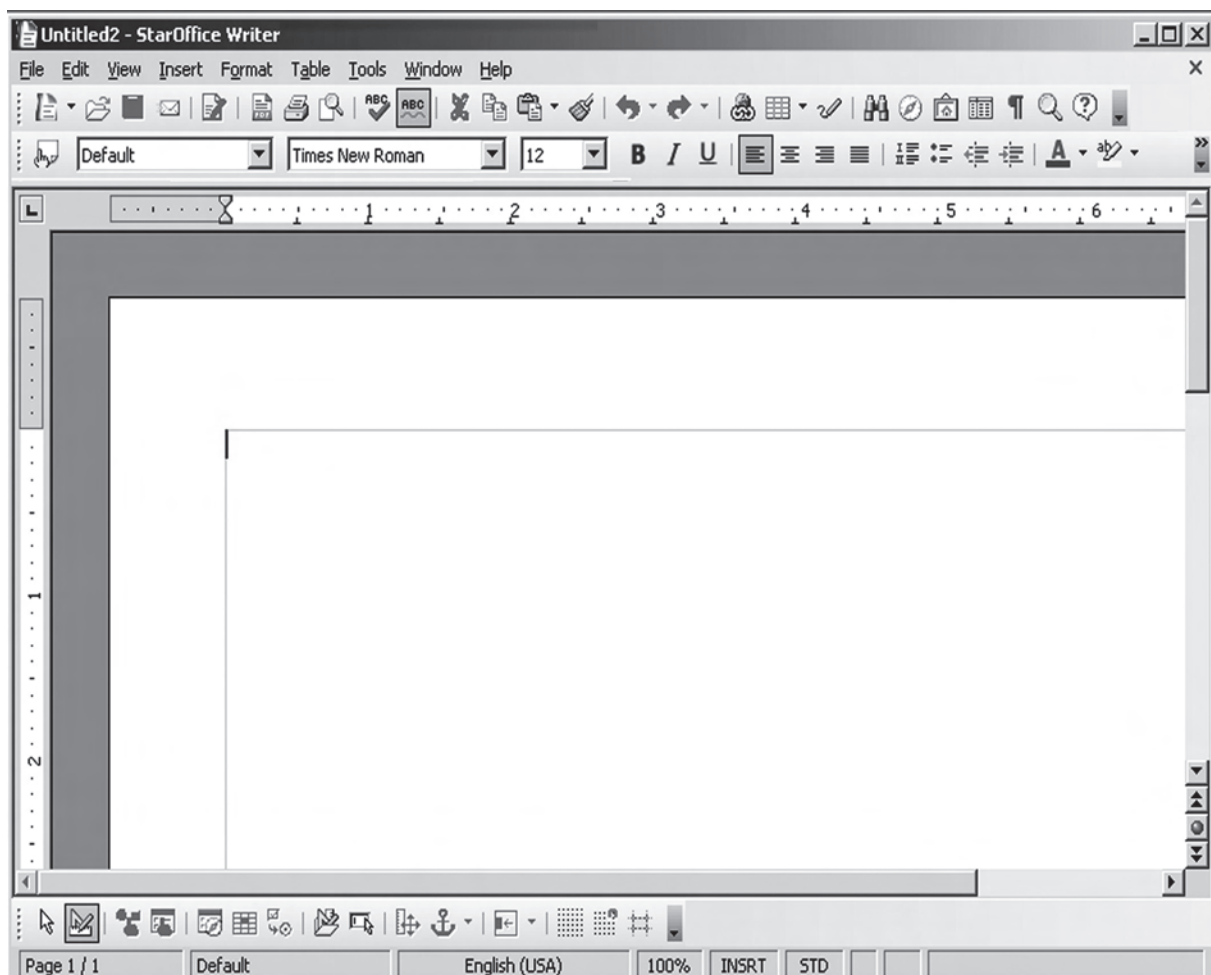
Some of the other commercially available word processing packages are MS Word, Lotus AmiPro, Word Perfect, Word Star and Word Pro.

## 1.2 Creating a New Document Using StarOffice Writer

The StarOffice Writer is opened as shown in figure 1.1.

The **File** → **New** → **Text Document** command can then be used to open a new document.

A blank document appears as shown in figure 1.2



**Fig 1.2 StarOffice Writer window**

Most of the elements of this window are similar to the Windows applications environment studied earlier.

## 1.3 Entering Text in the Document



Once a new document is opened, the text of the document can be typed in big blank area of the screen. To create a document the user can start typing straight away. As the characters are typed they appear on the screen and the flashing vertical bar called the insertion point, moves to the right. This insertion point always indicates where the new text will appear. When the text being typed reaches the end of the line, StarOffice Writer will automatically wrap the text to the next line. The **Enter** key must not be pressed at the end of the each line. The **Enter** key should be pressed only at the end of a paragraph or when a blank line is to be inserted. When a page is filled up, StarOffice Writer automatically creates a new page.

## 1.4 Saving, Closing and Opening Documents

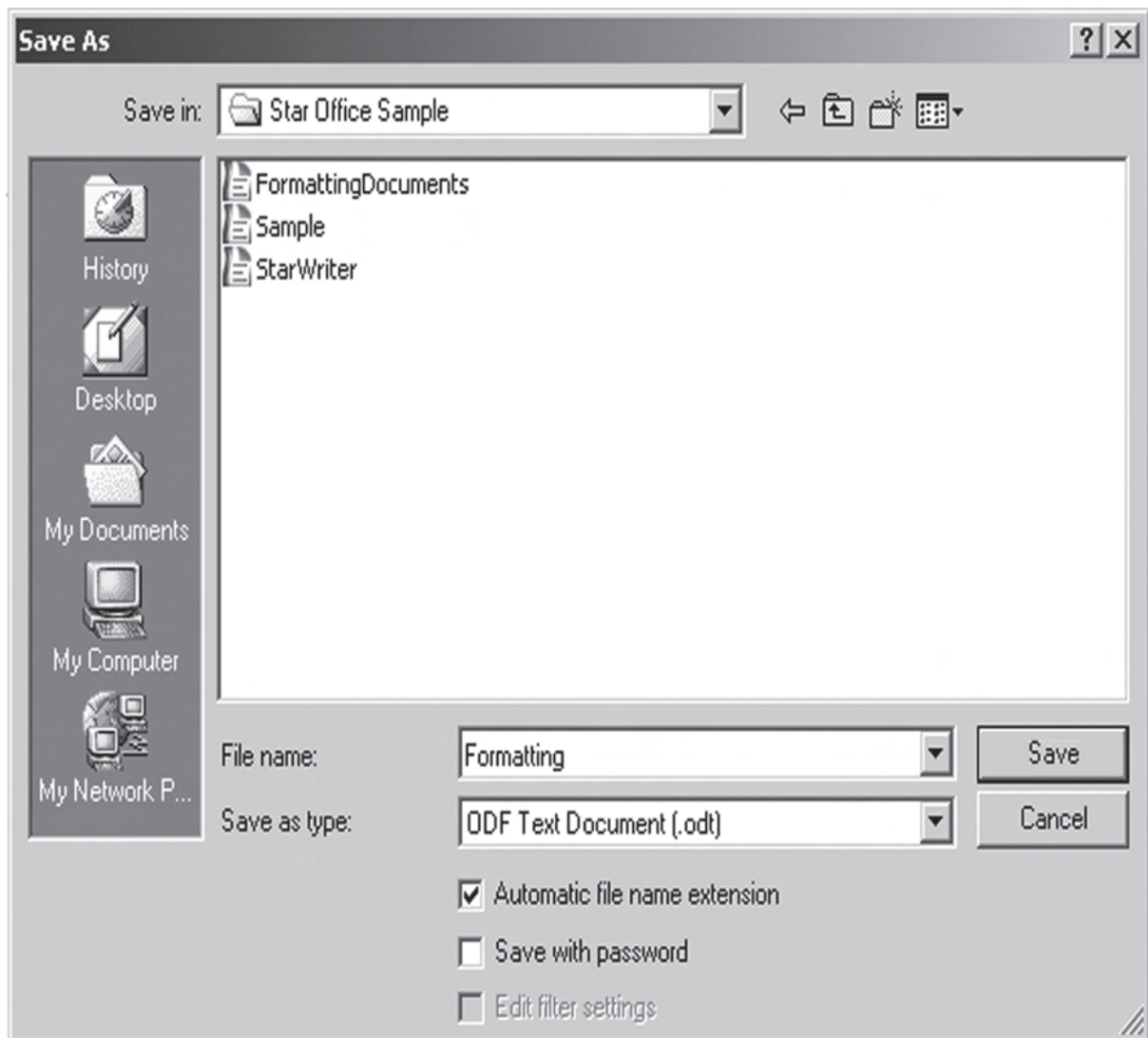
### 1.4.1 Saving a Document

The first time the document is saved, StarOffice Writer prompts for a name. Naming the file enables the user to find and open that file again. One can select the drive and folder where the file will be stored.

To save a document for a first time following steps are used:

1. **File** → **Save** command is selected or  icon is clicked. A Save As dialog box as shown in the figure 1.3 appears on the screen.
2. To select a drive, up one level icon  is clicked, then a list of drives will be displayed. On the list of drives, a double click is made on the required drive. Now a list of folders available on that drive is displayed.
3. A double click is made on the required folder and the file name is given in the **File name** list box.
4. Choose the required document type from the **Save as type** list box and click on the Save button to save the document in that type. The document is now saved and a file name appears in the title bar.

Once a file is saved under a name, to save it again the name need not be entered again. The file can be saved simply by selecting the **File**→ **Save** command or by clicking the Save button. **Ctrl + S** is the keyboard shortcut for saving the document.



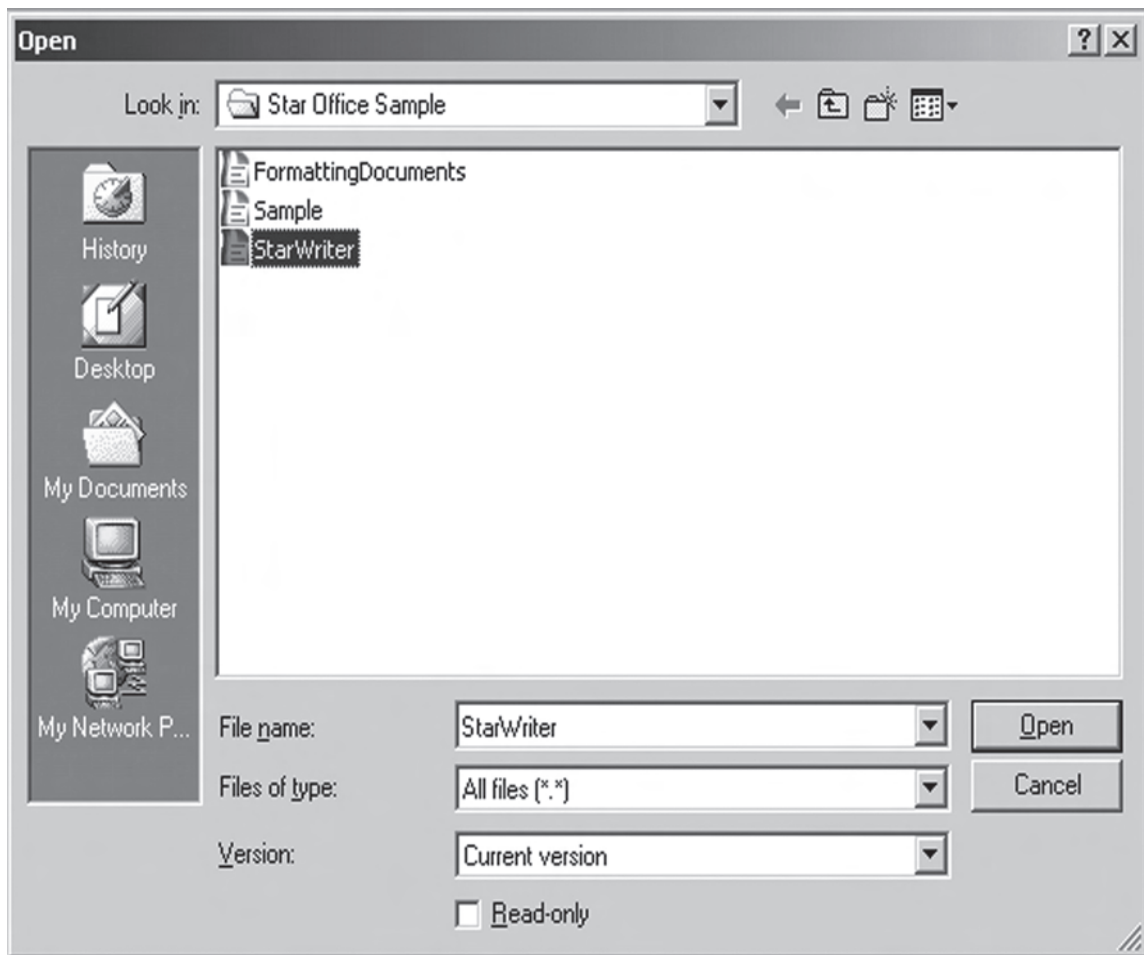
**Fig 1.3 Save As dialog box**

### **1.4.2 Closing a Document**

After a document is saved, it is not closed. It remains open so that the user can continue working. When the work is finished, the user should save and close the document. After saving, the document can be closed using the **File→ Close** command.


### **1.4.3 Opening a Document**

To reopen a document that has been saved and closed, the **File→ Open** command can be used. A dialog box, very similar to the Save As dialog box, appears, as shown in figure 1.4.



**Fig 1.4 The Open dialog box**

The name of the file to be opened can be chosen from the list, which is displayed.

An alternate method of opening files is to click on the **Open** File  icon.

#### **1.4.4 Working with Multiple Documents**

While working with StarOffice Writer, it is possible to have several documents open at the same time. This can be done by opening the documents one after the other. Once the documents are open, you can switch between them in the following two ways.

1. Click on the **Window** menu option and select the file from the list of documents displayed.
2. Click on the document button visible on the taskbar.

These documents can be closed one by one by using the **File** → **Close** command.

## 1.5 Moving Around the Document

To move the insertion point to anywhere in the document either the mouse or the keyboard can be used. The thick horizontal line in the page area is called the end-of-document marker. The insertion point cannot be beyond this line.

To move the insertion point with the mouse, the mouse pointer is moved to the required spot and the mouse button is clicked. The insertion point jumps to that spot. It is to note that mouse pointer is different from the insertion point.

To move the insertion point with the keyboard the arrow keys and other key combination can be used. Table 1.1 lists the common movement keys. If the key combination is joined with a plus sign, the first key must be pressed and held down and the second key is to be pressed.

**Table 1.1 Keyboard movement keys**

To move	Press
One character to the left	LEFT ARROW
One character to the right	RIGHT ARROW
One word to the left	CTRL+LEFT ARROW
One word to the right	CTRL+RIGHT ARROW
One cell to the left (in a table)	SHIFT+TAB
One cell to the right (in a table)	TAB
Up one line	UP ARROW
Down one line	DOWN ARROW
To the end of a line	END
To the beginning of a line	HOME
Up one screen (scrolling)	PAGE UP
Down one screen (scrolling)	PAGE DOWN
To the end of a document	CTRL+END
To the beginning of a document	CTRL+HOME

### Learn by solving

1. Enter the following text.

According to legend, the ancient Olympic Games were founded by Heracles. Yet the first Olympic Games for which we still have written records were held in 776 BCE (though it is generally believed that the Games had been going on for many years already). At this Olympic Games, a naked runner, Coroebus (a cook from Elis), won the sole event at the Olympics, the stade - a run of approximately

192 meters (210 yards). This made Coroebus the very first Olympic champion in history.

The ancient Olympic Games grew and continued to be played every four years for nearly 1200 years.

2. Save the text as Exercise 1 and close the file.
3. Open the document Exercise 1 and add the following paragraph to it. Save the file and close the file.

The very first modern Olympic Games opened in the first week of April 1896. Since the Greek government had been unable to fund construction of a stadium, a wealthy Greek architect, Georgios Averoff, donated one million drachmas (over \$100,000) to restore the Panathenaic Stadium, originally built in 330 BCE, with white marble for the Olympic Games.

---

## **1.6 Scrolling The Document**

If the document is long the text can be scrolled through without moving the insertion point. This can be easily done using the mouse.

---

The scroll arrows and the scroll bar shown in figure 1.2 can be used for this purpose. There are two sets of scroll arrows; one for up and down movement and the other for the left and right movement of the document.

The scrolling procedure is as follows:

1. To scroll left and right the left and right arrow respectively should be clicked.
2. To scroll up and down the up and down arrow respectively should be clicked.
3. To scroll a relative distance in the document the scroll box should be drawn up or down. If there are several pages in the document the user can know the current page number by looking at the pop-up page number that appears next to the scroll bar.

Scrolling a document does not move the insertion point. The mouse click should be used to get the I-beam at the required place.

---

## Learn by solving

Open the previously saved document Exercise 1 and see if all the keyboard option works to move around the document. Also try using the scroll bars and scroll arrow keys.

### 1.7 Correcting Mistakes

All the characters, which are typed, appear on the screen. If a mistake is identified it can be corrected either using the **Backspace** key or the **Delete** key. **Backspace** key deletes the characters to the left of the insertion point. **Delete** key deletes the characters to the right of the insertion point.

### 1.8 Inserting Text

After the entire document is typed if the user wants to add text in the beginning, middle or end of the document it can be easily done.

To add text the insertion point is placed where the new text is to start and the text is typed. The new text will be inserted and the existing text would move to the right.

The user is in the type-over mode, if the text existing to the right of insertion point disappears, while typing a new text. The user can toggle between type-over mode and the insert mode by using the Insert key. The status bar gives the information about the current mode (Insert or Type-over-mode).

As the corrections are being carried out periodically the user should save the document using the **File**→ **Save** command.

### 1.9 Selecting Text

Even though the document is built up by typing one character at a time, while editing and formatting one always work with words, lines, paragraphs and sometimes with the whole document. For this purpose one should learn how to select the text. Once the text is selected, change can be made to that text. The text can be moved, copied and made as bold. The font and colour of the text can also be changed. For selecting text the mouse or the keyboard can be used.

---

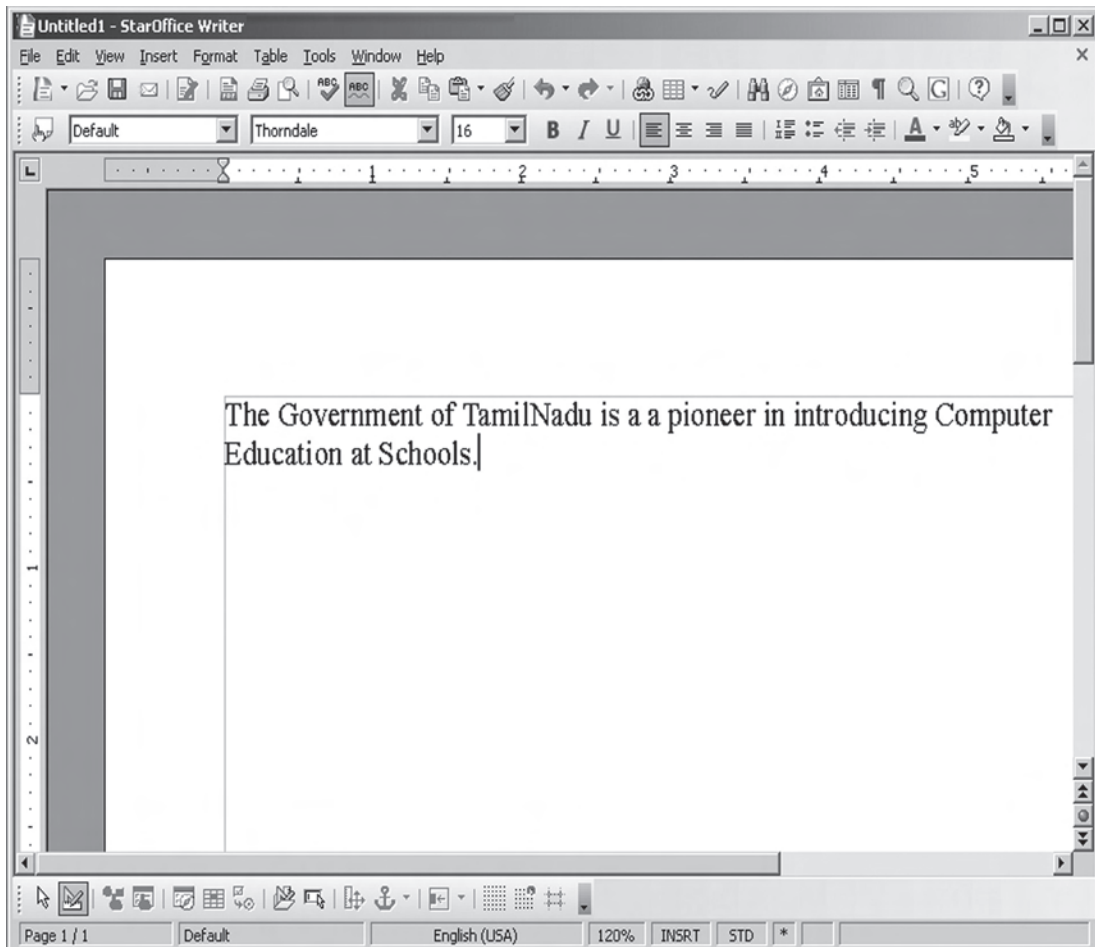
#### 1.9.1 Selecting Text with Mouse

**Following steps are to be followed:**

1. Insertion point is moved to the start of the text to be selected.



2. The left mouse button should be clicked, held down and dragged across the text to be selected.
3. When the intended text is selected, the mouse button should be released. The text appears as shown in figure 1.5.



**Fig 1.5 Selecting the Text**

To unselect the wrongly selected text a click should be made outside the selected text.

### **1.9.2 Selecting Text with Keyboard**

Following are the steps to be followed:

1. Insertion point is moved to the start of the text to be selected.
2. The **Shift** key is pressed down and the movement keys are used to highlight the required text.
3. When the **Shift** key is released, the text is selected.



### 1.9.3 Selection Shortcuts

The following shortcuts can also be used for selection.

Action to be performed	To select what
Double click on a word	To select a word
Click once next to the line	To select the particular line
Press Ctrl + A	To select the entire Document

### 1.10 Moving the Text



The selected text can be easily cut and pasted in the required location. Following steps are to be followed.

1. The text to be moved to a new location is selected.
2. **Edit** → **Cut** is selected or  in the tool bar is selected to cut the selected text.
3. Insertion point is moved to the place where the text is to be pasted.
4. **Edit** → **Paste** is selected or  in the tool bar is selected to paste the text in the new location. The text can also be pasted in this way to another or another type of document.

The following keyboard shortcuts can be used to move text.

Ctrl + X → to Cut Ctrl + V → to Paste
--

### 1.11 Copying the Text

1. The text to be copied is selected.
2. **Edit** → **Copy** is selected or  is clicked.
3. The insertion point is selected where the copy of the text should appear and is  clicked.

The following keyboard shortcuts can also be used for copy and paste:

Ctrl + C → to Copy Ctrl + V → to Paste
---

### 1.12 Finding and Replacing Text

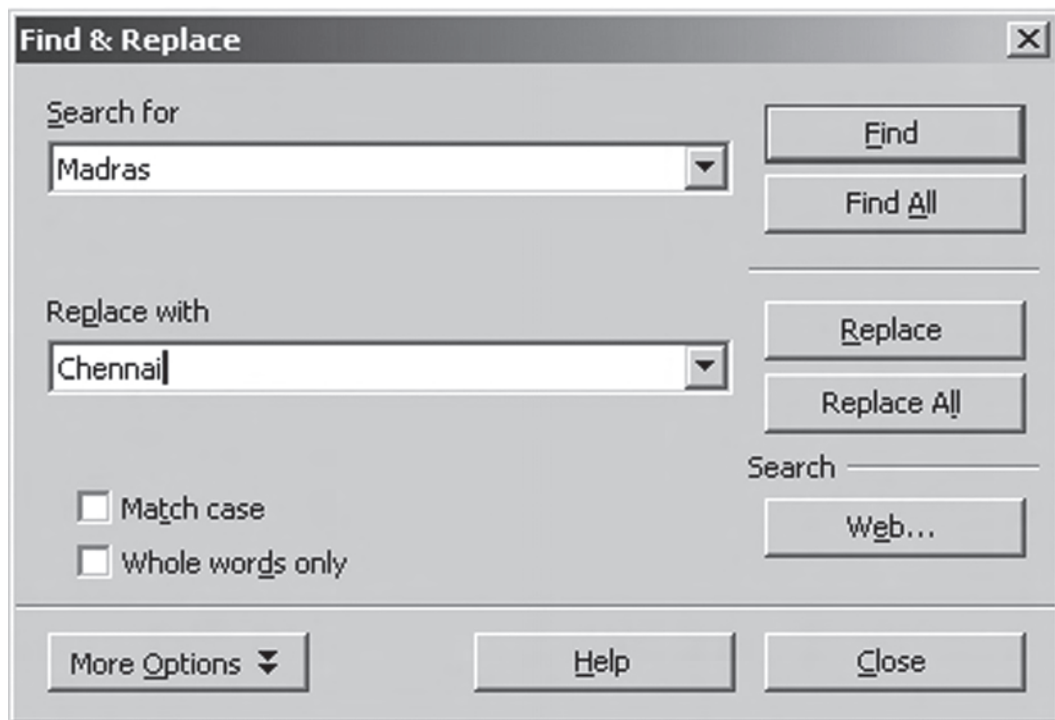
You can use the Find & Replace feature in StarOffice Writer to search for and to replace words in a text document.

## To Find and Replace Text

1. Choose **Edit** → **Find & Replace**.

The Find & Replace dialog box appears as shown in the Fig 1.6.

2. In the **Search for** box, type the text that you want to find in your document.



**Fig 1.6 Find and Replace Dialog Box**

3. In the **Replace with** box, enter the replacement word or phrase.
4. Click **Find** to start the search.
5. When Writer finds the first instance of the word or phrase, do one of the following:
  - To replace the found instance of the text with what you entered in the Replace with box, click **Replace**.
  - To replace all instances of the text with what you entered in the Replace with box, click **Replace All**.
  - To skip the found text and to continue the search, click **Find** again.
6. Click **Close** when you have finished the search.

**Note:** You can also select the word or phrase that you want to search for in the text document, and then choose Edit - Find & Replace. The text that you selected is automatically entered in the Search for box.

---

### **Learn by solving**

1. In the StarOffice Writer document Exercise 1 try to add additional text both at the beginning, middle and end of the document.
2. Make use of the Delete key Backspace key for making correction.
3. Make use of the Insert key to insert a new word and see how the Insert mode changes in the Status bar.
4. Select a particular portion of the document using selection shortcuts, cursor and with the keyboard.
5. Perform the Copy, Cut and Paste operations using the icons as well as the keyboard shortcuts.
6. Make use of the Find and Replace a given word and see how it works.

### **Summary**

- Entering a new text or modifying the existing text in a document is known as text editing.
- The user can move to the various portions of the document using the keyboard shortcuts or the mouse.
- The required word or portion of the text can also be selected using the keyboard shortcuts or using the mouse.
- Copy, Cut, Paste, Find and Replace are some of the commonly used editing functions.

## EXERCISES

### I. Fill in the blanks.

1. The thick horizontal line in the page area is called \_\_\_\_\_,\_\_\_\_\_
2. \_\_\_\_ key deletes the characters to the right of the insertion point.
3. \_\_\_\_\_, \_\_\_\_\_ option is selected to Cut the selected text.
4. \_\_\_\_\_ and \_\_\_\_\_ are the two combo boxes available in the Find and Replace dialog box.
5. \_\_\_\_ key combination is used to move to the end of the document.

### II. State true or false

1. Enter key must be pressed at the end of each line.
2. To select the entire document pressing Ctrl + A is the hot key combination to be used.
3. For searching a given word one need not have to type the required word in the Find and Replace dialog box.
4. To paste the selected word in the desired place insert key is used.
5. Ctrl + X is the keyboard shortcut for copying the selected text.

### III. Answer the following

1. What is meant by text editing?
2. How would you select the required portion of the text in a document?
3. How would you switch over from Insert mode to Type-over mode?
4. What are the steps to be followed for searching a given word?
5. Give the steps involved for replacing a given text.

## CHAPTER 2

# TEXT FORMATTING


A text without any special formatting can have a monotonous appearance. To outline text, to highlight individual words, quotations, or references, or to separate certain parts of the text, various types of formatting can be applied.

### 2.1 Formatting Options

StarOffice Writer offers a number of choices for formatting such as bold or italics, and defining the font, type, and font size. Bold, italic or underlined are the most common types of text formatting.

Almost all the formatting options are available under Format menu. StarOffice Writer also conveniently provides buttons for the most commonly used options. But before these options can be used, the text on which they are to be used has to be selected. Once the desired portion of the text is selected then depending on the need any one of the following buttons are clicked:


Click  to make text **Bold**.

Click  to make text *Italic*.

Click  to make text Underlined.

The same can also be achieved by clicking on **Format** → **Character** and then selecting an option from the **Typeface** list box.

If a particular word or portion of the text is already in bold or italic or underlined style those text properties can be removed by clicking on the respective icons, again.

For example, in order to change the word **Computer** from bold to normal appearance that word is selected and the icon  is pressed once. This will remove the bold property from that text.

Alternatively **Ctrl + B**, **Ctrl + I** and **Ctrl + U** keys can be used to make the selected text bold, italic and underlined respectively.

#### 2.1.1 Changing the Fonts

A font is a set of characters and numbers in a certain style. Each font looks different from other fonts. Some fonts, like the **Times New Roman**, look professional and are suited for business documents. Some fonts, like **FAJITA**, are decorative. Some fonts, like Symbol and Wingdings, are actually sets of symbols. Such a font can be used to insert special characters in the document.

Part of word processing skill is choosing an appropriate font for the document. Text can be selected and any of the fonts available in the system can be used. Font packages can also be purchased to add other font choices to the system.

### Method

Click the down arrow in the Fonts Combo box of font tab in Character dialog box.

Use **Format** → **Character** to open the Character dialog box.

From the list of available fonts, click the required one.



The text changes to the selected font.

### 2.1.2 Font Size

The size of the text is also important. The same size of the text cannot be used for a legal document, and an advertisement material. Similarly the size of the letters used in the main text and footer cannot be the same. The size of a font is measured in points, and there are 72 points to an inch. The larger the point size, the larger the type.

### Method

Click the down arrow in the Size combo box of Fonts tab in Character dialog box.


Use **Format** → **Character** to open the Character dialog box.

The text changes to the selected font size. 

### 2.1.3 Changing the Font Colour

A different colour for selected text can be used. Colour printers are becoming more and more popular. With the help of a colour printer, some splash can be added to the documents by changing the colour of text. In the absence of a colour printer, the document will be displayed on screen in colour. But these changes will be printed as shades of gray on a black-and-white printer.

### Method


To use a different text color, select the text and click the arrow in the **Font Colour** icon. A colour palette is displayed from which the required colour can be selected. 

Alternatively, select the text and click on the Font colour icon, to apply the current colour of the **Font Colour**.

## 2.1.4 Highlighting the Text

Highlighting can be used to call attention to key ideas or pointers in a document. When reading something important in a book, a magazine, a report, or any document, the reader takes a yellow highlighter pen and drags across it. These highlighted sections are used to review or find the key points in the document. The same thing can be done in the StarOffice Writer document. The colour to be used for highlighting can also be selected.

### Method

To highlight the selected text, the **Highlighting** icon is  selected, and then the needed colour is clicked. StarOffice Writer now applies the highlighting. When the mouse button is clicked on the above icon and held down a colour palette is displayed from which the required colour can be selected. To remove the highlighting, select the text and select the No Fill from the colour palette.

As mentioned earlier, the **Character** option under the **Format** menu can also be used to achieve all this. Clicking on **Format** → **Character** displays the Character dialog box as shown in the following figure. This method is used to make several changes at once or to preview the changes before applying them.

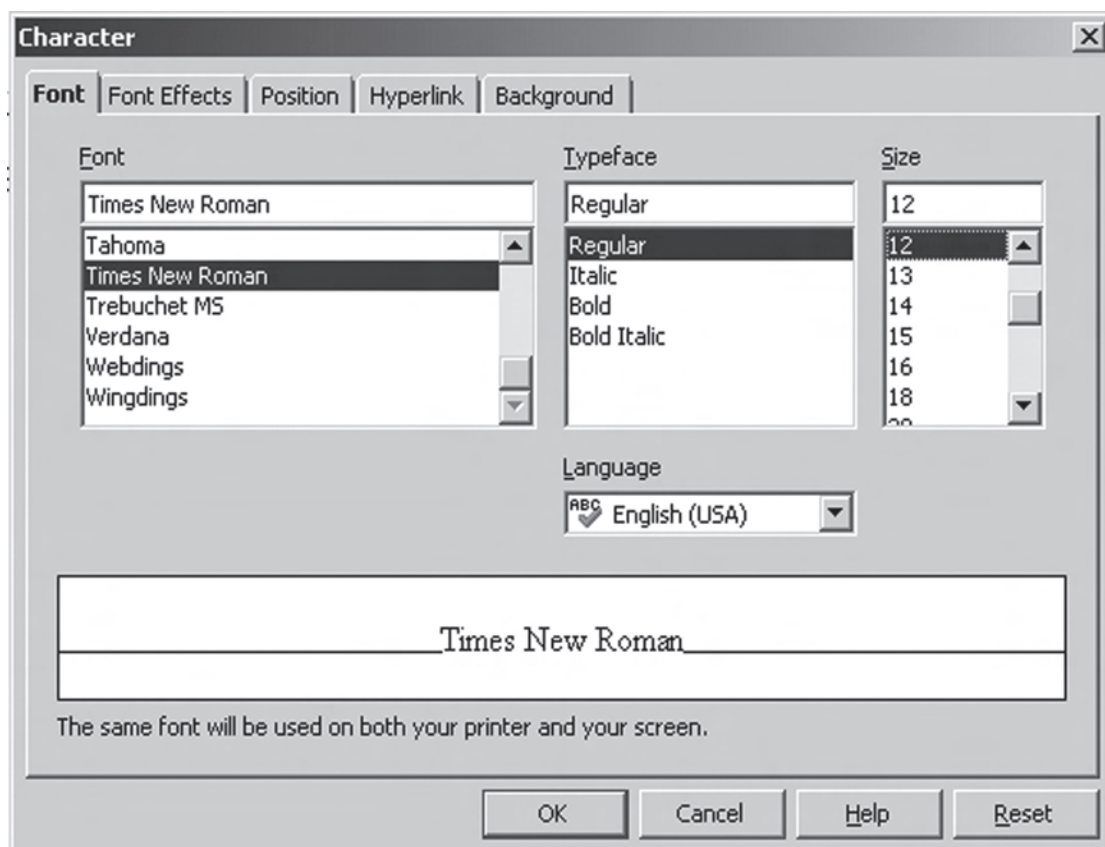


Fig 2.1 Character Dialog Box



## Learn by solving

Select a particular portion in the document Exercise 1 and perform the following tasks:

- Change the text properties (normal to bold, then to italic and underlined). Finally bring it to normal type.
  - Change the font, font size, colour and observe the changes.
- 

## 2.2 Paragraph Alignment

In addition to formatting individual chunks of text, one can also format the paragraphs. One of the most common changes is to change the alignment of a paragraph.

A paragraph is any text followed by a hard return. (A hard return is inserted every time when **Enter** is pressed. Soft returns are inserted as line breaks by StarOffice Writer and are adjusted when text is added or deleted.) A single line, for instance, can be a paragraph. One can apply the paragraph formatting options to a single paragraph or to several paragraphs. Each paragraph, for example, can give different tab settings.

Also, each time **Enter** is pressed, the paragraph options for that paragraph are carried down to the next paragraph. And if a paragraph marker is deleted the paragraph takes on the formatting of the following paragraph. If one types a line, center it, and press **Enter**, the next line is centered as well. If a double-spaced paragraph is followed by a single-spaced paragraph, delete the paragraph mark at the end of the first paragraph, the paragraph will then be single-spaced. This is confusing for beginners who wonder how a paragraph got formatted without anyone making a change. One can think of the paragraph marker as the storehouse for all formatting options for that paragraph.

When typing in StarOffice Writer, all text is left aligned, and the right margin is ragged or uneven. For most text, this alignment works great. For other paragraphs, one may want to make a change. For instance, one can center the document title or right align text like a newspaper banner. The text in the paragraphs can be justified to keep both margins even.

Four types of alignment can be selected, and the best way to make a change is to use the Formatting toolbar. Following steps are used:

1. To change the alignment of one paragraph, first click within that paragraph. To change the alignment of several paragraphs, select the ones needing change.

2. Do one of the following

Click to  left align text.

Click to  right align text

Click to  justify text.

Click to  center text.

**Figure 2.2 shows an example of each type of alignment.**

To undo the change, click the Undo button or select the Edit → Undo command. If a different alignment is required later, select the paragraph (s) and then click another alignment button.

To undo the change, click the Undo button or select the Edit → Undo command. If a different alignment is required later, select the paragraph (s) and then click another alignment button.

To undo the change, click the Undo button or select the Edit → Undo command. If a different alignment is required later, select the paragraph (s) and then click another alignment button.

To undo the change, click the Undo button or select the Edit → Undo command. If a different alignment is required later, select the paragraph (s) and then click another alignment button.

The following keyboard shortcuts can also be used to change the alignment.

To make text...	Press...
Centered	Ctrl+E
Left-aligned	Ctrl+L
Right-aligned	Ctrl+R
Justified	Ctrl+J

## Learn by solving

Open the document Exercise 1. Carry out the following instructions and observe the changes.

1. Add a suitable heading at the top and center it.
  2. Select the second paragraph and use justify it.
  3. Select the third paragraph and right align it.
  4. Add your name, class and school's name at the bottom and right align it.
  5. Add the date at the bottom in the line below the title and right align it.
- 

## 2.3 Indenting Text

Alignment changes are most appropriate for headings or other special paragraphs in the document. But to make a long document easy and inviting to read, some of the paragraph-formatting features described in this section, including indents can be used. For instance, indent the first line of each paragraph. This visual clue helps the reader to see how the document is divided into paragraphs. The paragraphs also indented, such as quotations that are set apart from the main document text. As another option, one may want to use a special kind of indent, called a hanging indent, for numbered lists.

The amount of indent can be varied to the requirement, and one can use either the Formatting toolbar or the Paragraph dialog box to make a change, as covered in this section.

### 2.3.1 Indenting Text with the Toolbar


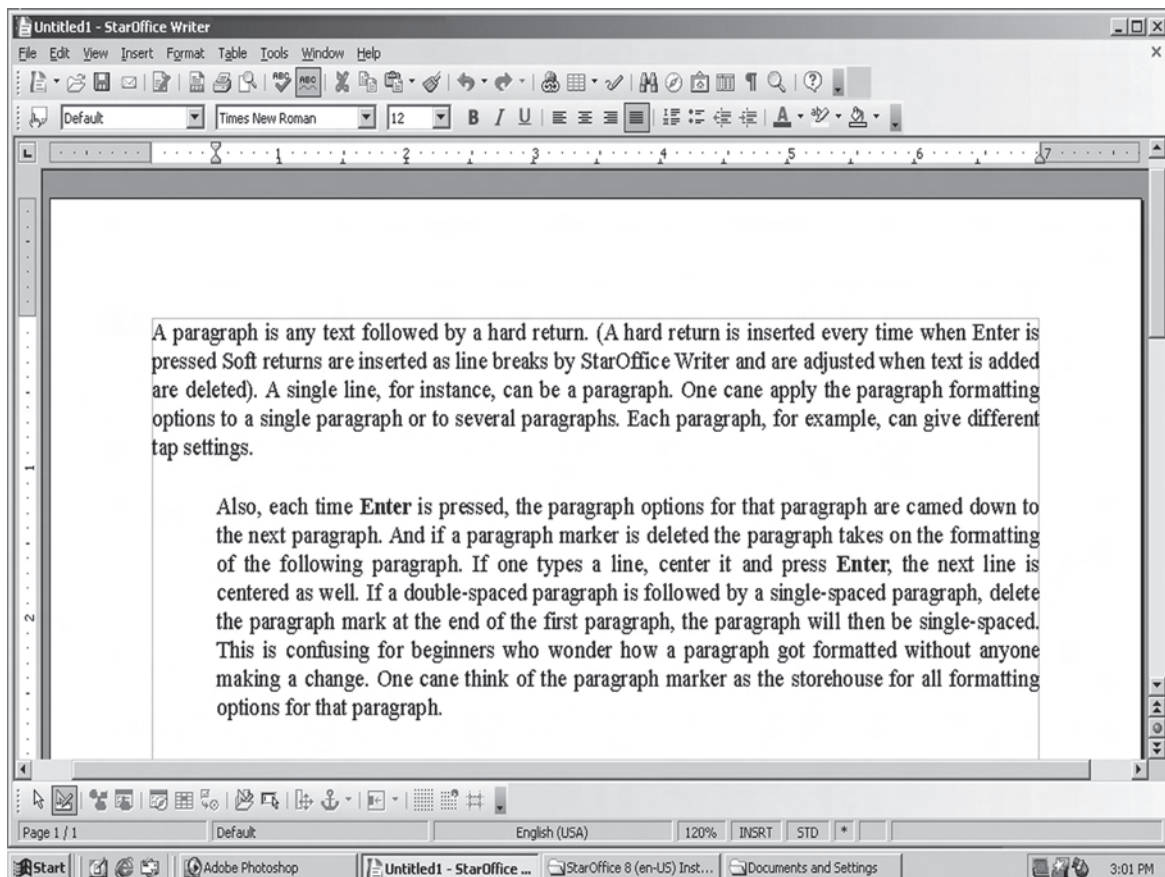
If a left indent is required - useful for setting off a paragraph from the main body text - the toolbar can be used to set the indent. Click the **Increase Indent**  icon; the paragraph is indented 1/2 inch from the left margin. The button again can be clicked to increase the indent. Each time the button is clicked, the paragraph is indented another 1/2 inch.

Figure 2.3 displays a paragraph that has been indented using the Increase indent button.



**Fig 2.3 Paragraph indented using the Increase Indent icon**

If the indent is too much or if one wants to undo the indent, the **Decrease Indent** icon is clicked to decrease or undo the indents.

### 2.3.2 Indenting Text with the Paragraph Dialog Box

The Increase Indent and Decrease Indent icons are useful if it is needed to indent text from the left. If one wants to indent from the left and right or if one wants to create a special kind of indent, a different method is used. The Paragraph dialog box must be used.

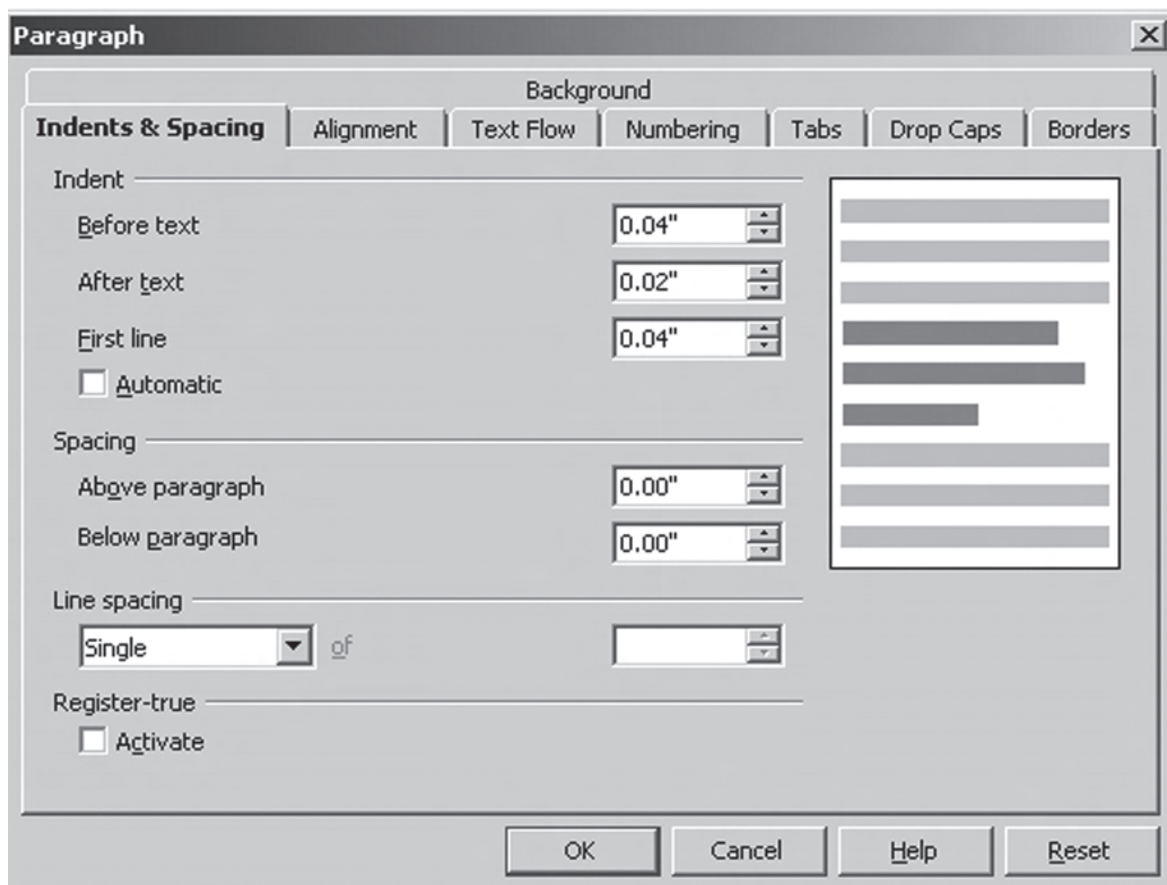
Following steps are used to indent text using the Paragraph dialog box:

1. Move the insertion point to the beginning of the paragraph to be indented. To indent several paragraphs, select those paragraphs.
2. Select **Format** → **Paragraph** command.
3. Click the **Indents & Spacing** tab in the Paragraph dialog box as shown in figure 2.4.

4. Do any of the following.

- i. To indent from the left, type the amount to be indented in the **Before text** spin box or use the spin arrows to select a value.
- ii. To indent text from the right, type the amount or use the **After text** spin arrows to enter the amount in the spin box.
- iii. Click the **OK** button.

The **First line** option can be used to indent the line of the paragraph. Using the spin arrows, if a positive value is specified, the first line will be indented. A negative value will result in a hanging indent; that is, the first line will hang outside the paragraph.



**Fig 2.4 Indents and Spacing Dialog Box**

---

### Learn by solving

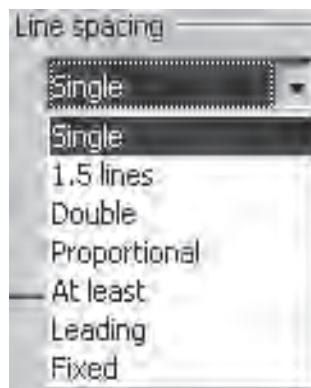
Reproduce the following paragraphs as such using both, the toolbar icons and the menu options.

A table is a collection of rows that are related to each other by a common idea, such as customer table, which contains all the information about the customers.

Structured Query Language is the special language used to access the data contained in a database.

## 2.4 Modifying Line Spacing

By default, StarOffice Writer single-spaces the text in the document. This spacing works well for short documents such as letters and memos, but in longer documents, such as a manuscript, one may want to use a different spacing values. Single line, 1.5 lines, Double, Proportional, At least, Leading and Fixed are the line spacing options available as shown in figure 2.5.



**Fig 2.5 Line spacing Options**

**To change the line spacing, following steps are used:**

1. Select the paragraphs(s) that needs to be changed.
2. Select **Format** → **Paragraph**.
3. If necessary, click the **Indents & Spacing** tab. The Indents & Spacing tab dialog box as shown in figure 2.4.
4. Click on the **Line spacing** drop-down list box, and select the required line spacing option.
5. Click the **OK** button.

---

### Learn by solving

Open the document Exercise 1. Change the line spacing into 1.5 line spacing and double line spacing and observe the change.

---


## 2.5 Creating Bullets and Numbered List

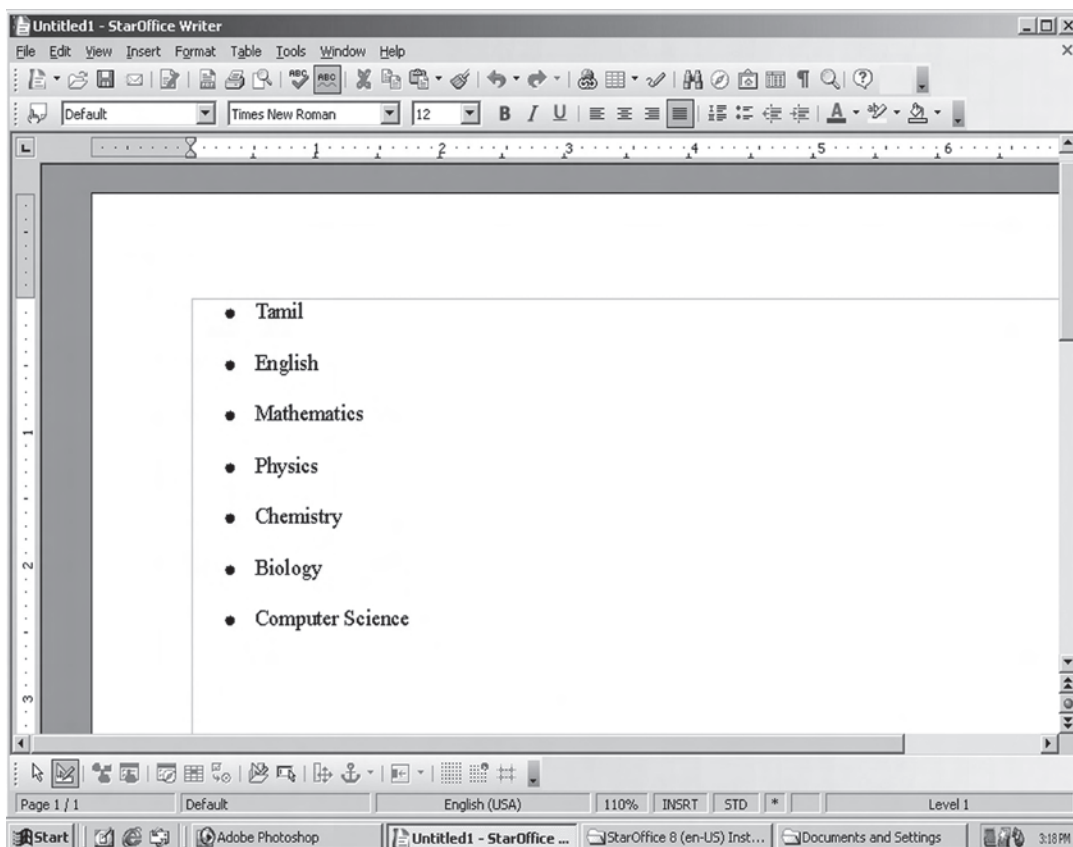
When the document presents a matter using text in the form of long paragraphs the reader may not be able to quickly notice the important points or messages. For this purpose bullets and numbers are used to list the important points and messages.

One way to set off a list of points or topics in a document is to create a bulleted list. A bullet precedes each item in the list, and the text is indented. One can also create a numbered list. Numbered lists work well for directions or other points one want to present in sequence. StarOffice Writer automatically numbers all the items in a list, and the text is indented.

### 2.5.1 Creating a Bulleted List


The fastest way to create a bulleted list is to be use the Bullets button on the Formatting toolbar. For this purpose the following steps are used:

1. Select the text that to which bullets are to be added. The StarOffice Writer will add bullets to each paragraph within the selection, and not to each line. StarOffice Writer will add bullets to any blank lines within the selection.
2. Click on the Bullets icon  from the formatting tool bar, StarOffice Writer creates a bulleted list, as shown in figure 2.6.




**Fig 2.6 Bulleted List**

## 2.5.2 Creating a Numbered List

For items that appear in a specific order, such as a series of steps, one can create a numbered list. StarOffice Writer will add the numbers automatically and also indent the paragraphs so that the text is aligned properly. All one has to do is click the Numbering icon .

Another great thing about a numbered list is that if one adds or deletes an item within the list, StarOffice Writer rennumbers the list.

**Following steps are used to create a numbered list.**

1. Select the text to be numbered. The StarOffice Writer will number each paragraph. Blank lines within the selection will be numbered.
2. Click  icon to create a numbered list.

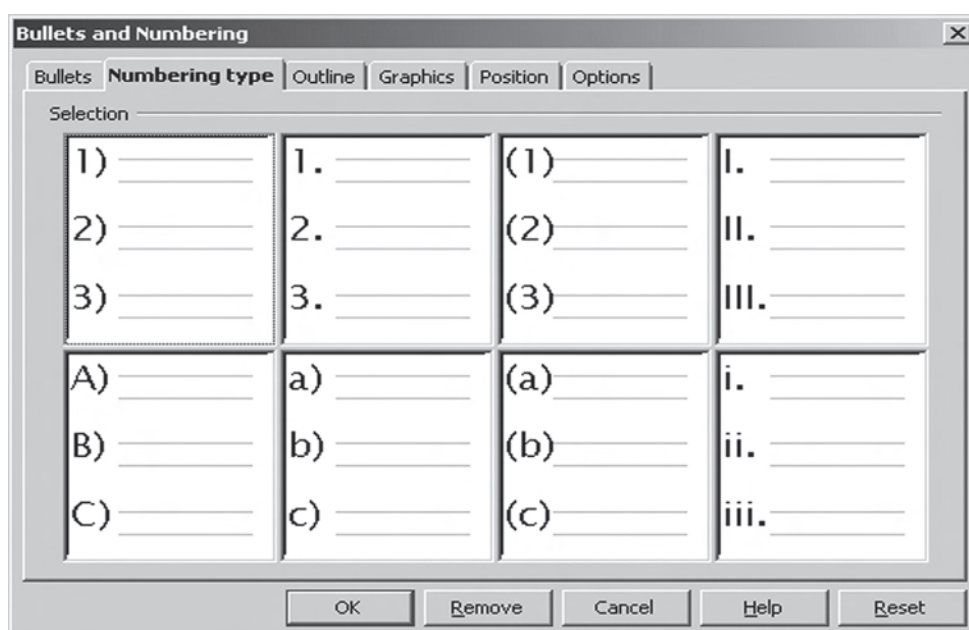
## 2.5.3 Removing the Bullets or Numbers

To remove bullets from a list, the list is selected and the **Bullets** button is clicked again. To remove numbers for a list, select the list and click the **Numbering** button.

## 2.5.4 Applying Different Styles for Bullets and Numbers

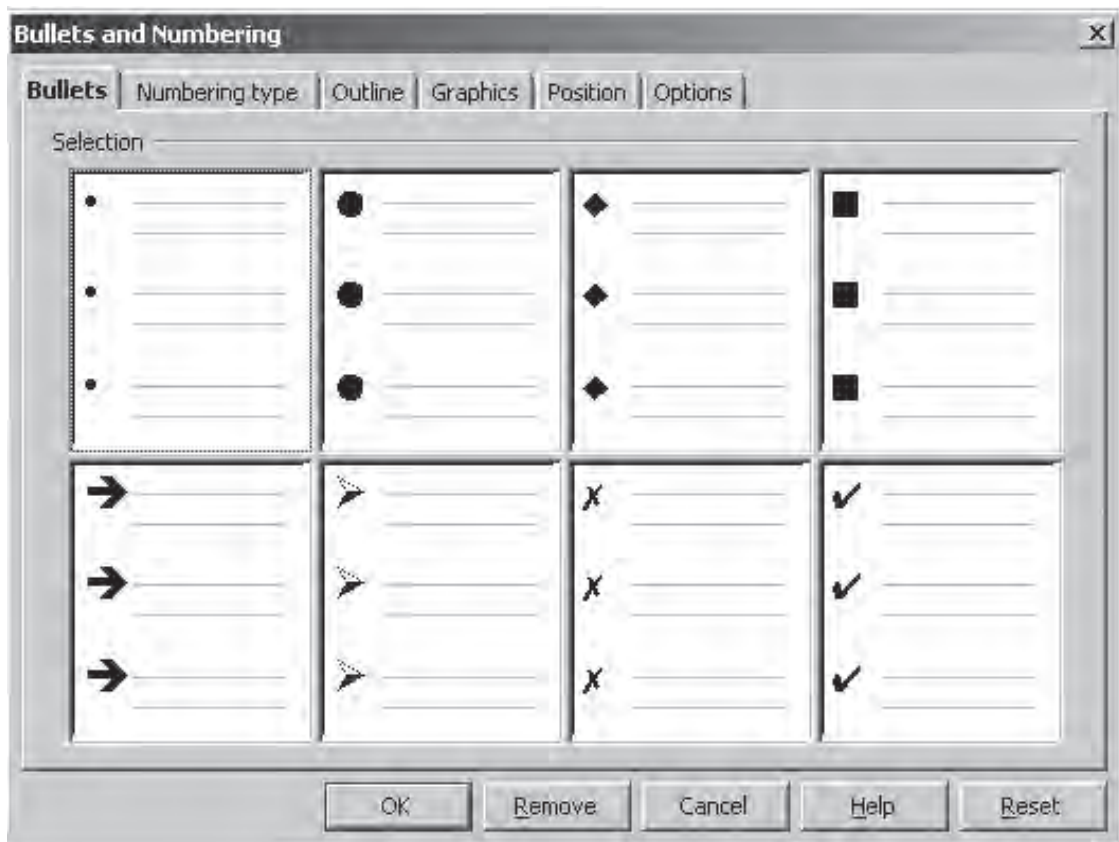
StarOffice Writer provides various styles for the bullets and numbers, which can be selected using **Format** → **Bullets and Numbering**.

**Bullets and Numbering** dialog box appears as shown in the figure 2.7a and 2.7b.



**Fig 2.7a Bullets and Numbering dialog box with the Numbering Style tab selected**





**Fig 2.7b Bullets and Numbering dialog box with the Bullets tab selected.**

Using the various options available for bullets and numbers the user can select the desired one.

---

### Learn by solving

1. Create a list of your favorite games. Number them using the Numbering tool. Modify the list and add two more game. Change the numbers to bullets.
  2. Type a list of ten of your friends. Format them using the bullets option. Change the bullet and observe the result.
- 

## 2.6 Formatting Using Styles

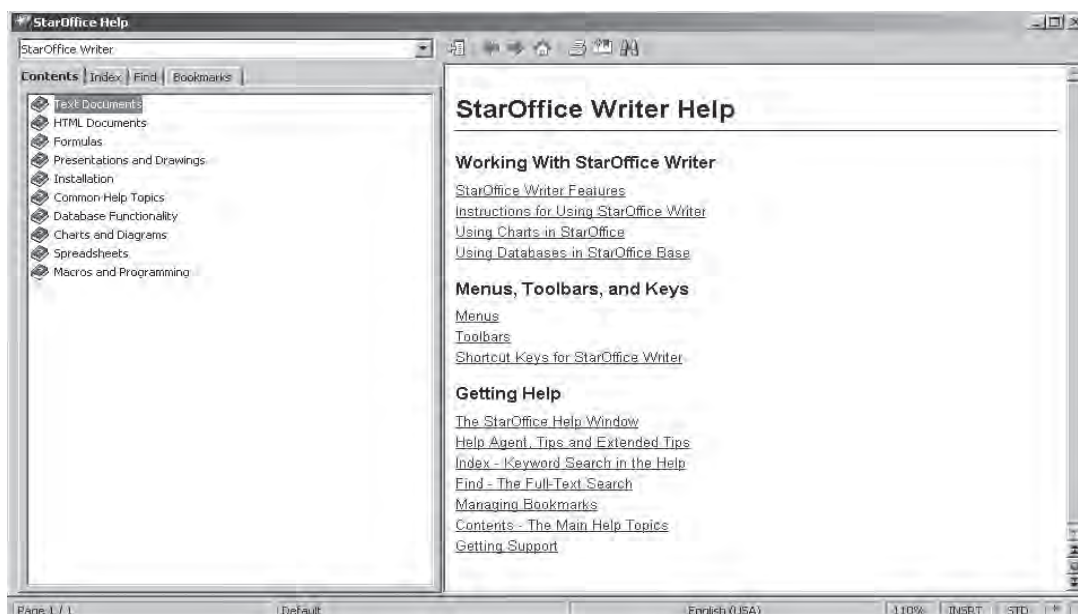
This is an alternative way of formatting text. A style is a named set of defaults for formatting text. The formatting methods discussed earlier are more appropriate for one time use. If, for example, a word is to be made as bold, just the cursor is positioned in the word and the Bold icon is clicked. This works quickly and intuitively.

But if several documents, each running into several pages are to be formatted in a particular style, then a style is more useful. This type of formatting is also easier when a format has to be used at several points within the text. Styles require more advanced planning. The style needs to be created initially only then the style can be applied to the text, using the Typeface list. The advantages of working with styles can really be appreciated when making extensive formatting changes.

## 2.7 StarOffice Help

StarOffice 8 provides several help systems that you can use while you work:

- Online help
- Help Agent
- Help Tips
- Extended Help Tips



**Fig 2.8 StarOffice Writer Help**

The Choose help file box in the toolbar area at the top of the Help window can be used to search for help on other StarOffice elements. This is a list box with the list items shown in figure 2.9



**Fig 2.9 Choose Help file list box**

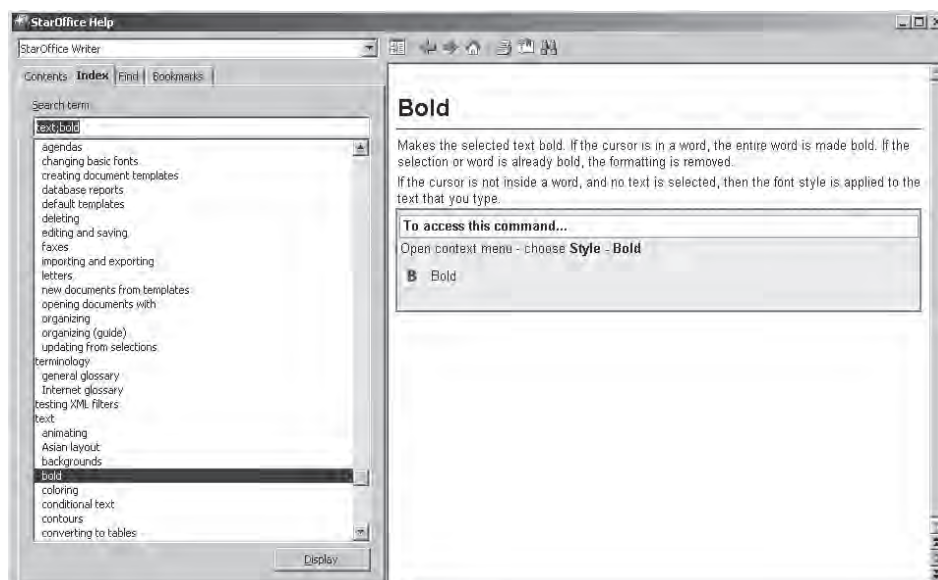
You can access the StarOffice Help in several ways:

- Choose the menu command Help - StarOffice Help, or press F1. This command opens the StarOffice help where you can search for a help topic.
- Click the Help button in any dialog to open the help topic for the dialog.
- Click the Help Agent that appears automatically when you perform a complex task. The Help Agent opens the help topic for the task.
- Rest the mouse pointer over a menu command or over any icon to display a Help Tip.
- To view a brief description of what the command or icon does, press Shift+F1 to display an Extended Tip.
- The Help Agent is displayed on the screen in a small window as shown in figure 2.10.



**Fig 2.10 The Help Agent Window**

The content of Help Agent window changes depending on what is being done in the main window. For example, if some text is highlighted and the Bold icon is clicked the Help Agent displays help about the Bold formatting option as shown in figure 2.11.



**Fig 2.11 Help Agent window with help about Bold formatting.**

## Summary

- There are two types of formatting (i.e.) hard formatting and soft formatting.
- Changing the text style-bold, italic, underlined, and the font properties - size, colour are the commonly used formatting changes.
- Paragraph alignment can be of the following four types
  1. Left alignment
  2. Right alignment
  3. Justification
  4. Centering
- Indentation helps in providing the document more readability.
- The spacing between the lines in a StarOffice Writer document can easily be changed.
- Bullets and Numbers are used to list the important points and messages.
- An alternate method of formatting refers to creating a particular style and applying it quickly to a document instead of making individual changes to the line spacing, paragraph alignment, indentation etc.
- StarOffice has an on-line help facility. This facility can be used by clicking on the Help menu.

## EXERCISES

### I. Fill in the Blanks

1. The size of a font is measured in \_\_\_\_\_ and there are \_\_\_\_\_ points to an inch.
2. The \_\_\_\_\_ printer is required to get the hard copy of the document in colour.
3. \_\_\_\_\_ feature is used to mark the important portions of the document.
4. \_\_\_\_\_ key combination is used for justifying the selected paragraph.
5. The \_\_\_\_\_ is a named set of defaults for formatting text.

### II. State true or false

1. Line spacing should always be an integer value.
2. Indenting text helps to increase the readability.
3. Modifying the line spacing will also modify the spacing between words in a line.
4. To remove bullets from a list the list is selected and the delete button is clicked.
5. A font is a set of characters and numbers in a certain style.

### III. Answer the following


1. What is the difference between hard formatting and soft formatting?
2. What are the formatting changes that can be made with respect to the fonts?
3. What are the various types of paragraph alignments that can be made?
4. What does indenting the next mean?
5. How would you create the bullets and numbered list?

## CHAPTER 3

# CORRECTING SPELLING MISTAKES

The documents prepared using word processing software should be without any spelling mistake. For this purpose StarOffice Writer includes a dictionary and spell-check program. StarOffice Writer can identify the spelling mistakes as the document is typed or after the entire document is typed.

### 3.1 Checking Spelling While Typing

StarOffice Writer has an automatic spell-check feature that can check for possible spelling mistakes even as the document is being typed. This feature can be turned ON or OFF by clicking on the Auto Spellcheck  icon.

When Auto Spellcheck feature is ON, StarOffice Writer compares each word typed with the words in the dictionary and underline words that do not match with a squiggly red line. These red lines can be ignored and the typing work may be continued. The correction can be made after the document is created. These mistakes can be corrected in two ways.

1. **Backspace** key is pressed to delete a misspelled word and the word is retyped.
2. If the right spelling is not known, StarOffice Writer can help the user with some choices. The misspelled word is right clicked. A pop up menu appears on the screen. If the correct spelling is shown, a click is made on that word. StarOffice Writer makes the replacement.

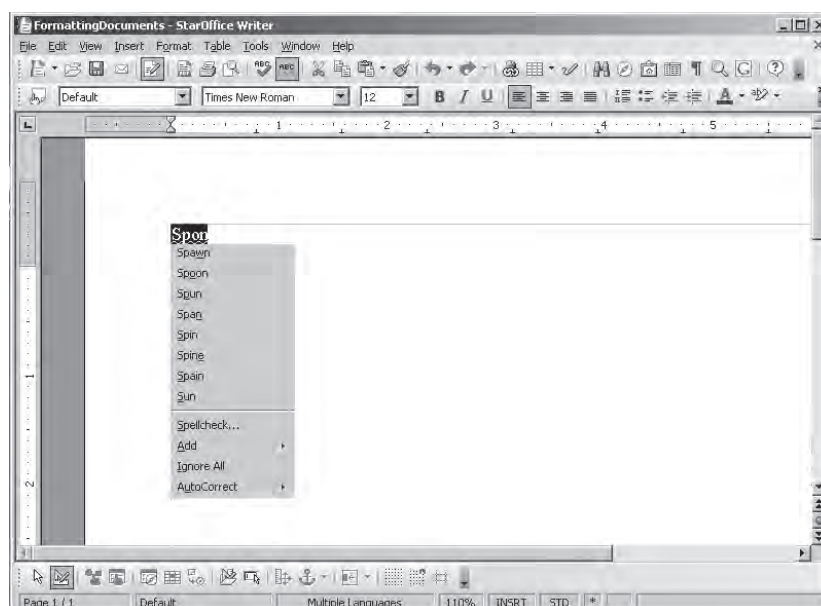



Fig 3.1 A popup menu displaying alternatives for the word “spon”

### 3.2 Checking the Spelling after the Document is Typed

The following steps are used for a spell check.

**Tools** → **Spell** → **Check** is selected or  is clicked. To check a part of the document only that portion is selected. The **F7** key may also be pressed to select the spelling command.

The dialog box shown in figure 3.2 appears on the screen.



**Fig 3.2 Spelling check Dialog Box**

**Not in dictionary** text area displays the misspelled word and the **Suggestions** list displays any alternative spellings.

Any of the following can be done:

- To skip this occurrence but stop on the next one. **Ignore Once** button is clicked. To skip all occurrences of this word. **Ignore All** button is clicked.
- To replace the word with one of the selected spellings, in the suggestions list that spelling is clicked, and **Change** button is clicked to change this occurrence and **Change All** button is clicked to replace all occurrences of the word.



- If none of the replacements is correct, correction can be made manually in the **Not in dictionary** text area. **Add** button is clicked to add the word to the dictionary.
- 

### Learn by solving

1. Open the document Exercise 1. Ensure that the Auto Spellcheck is ON. Add the following paragraph to the existing text. As you are typing make some spelling mistakes intentionally and observe how the Auto Spellcheck works.

It is more than likely that many of today's Olympic disciplines are sophisticated versions of the games of strength and speed that flourished in ancient India and Greece. Chess, Wrestling, Polo, archery and hockey (possibly a fall-out from polo) are some of the games believed to have originated in India.

2. Now, Switch off the Auto Spell Check option and makes some more spelling mistakes. Correct them using the Spelling Correction dialog box.
- 

### 3.3 AutoCorrect Option

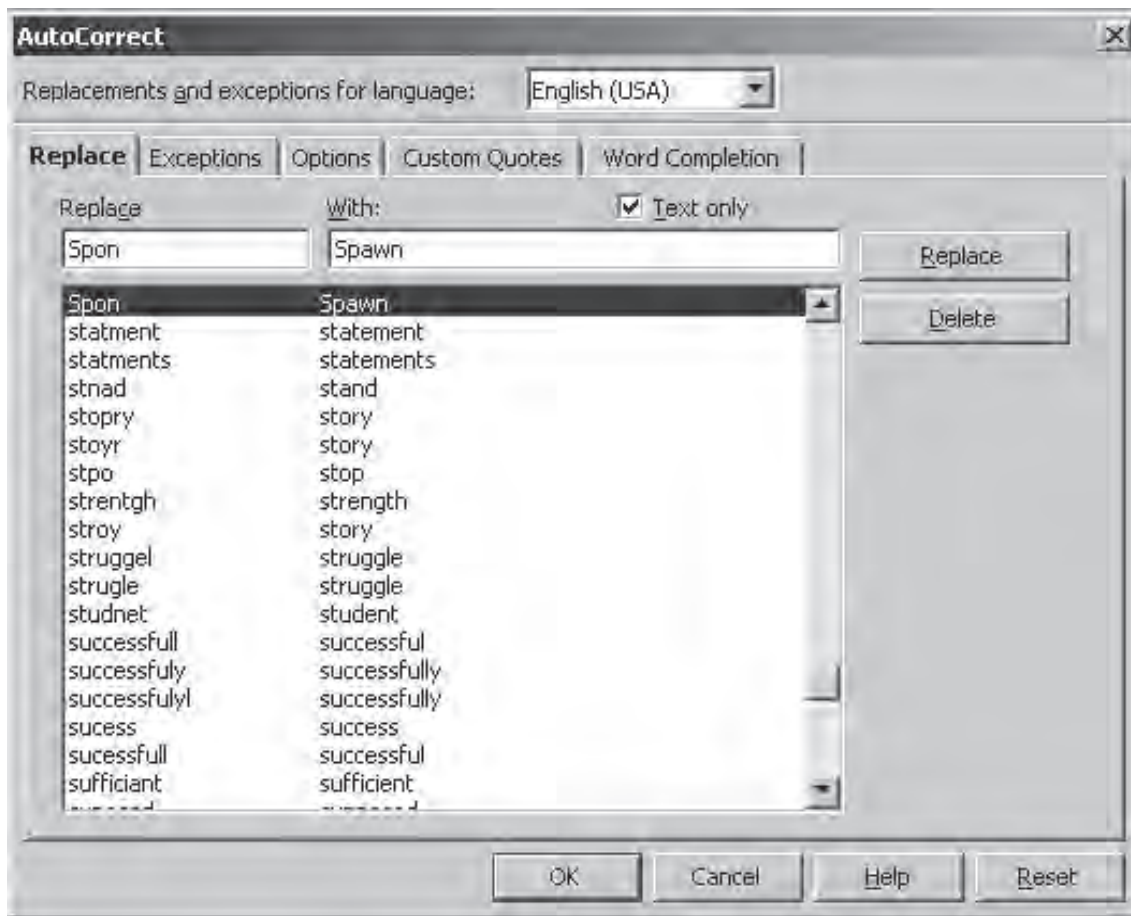
In addition to flagging some words, StarOffice Writer will automatically correct some spelling mistakes. StarOffice Writer recognises some common misspellings and typographical mistakes and makes the replacement automatically. For instance, if a word is typed as “teh”, StarOffice Writer automatically replaces this with the correct word “the”. In addition to the ones StarOffice Writer automatically corrects, the user can add other words to the list. For instance if a user feels that the word “colour” is frequently mistyped as color or in any other wrong way he/she can add an AutoCorrect entry and the StarOffice Writer will make the correct replacement every time that word “colour” is wrongly typed.

### 3.4 Creating AutoCorrect Entry

If the error and its correction are to be added to the Auto Correct list the following command is used: **Tools** → **AutoCorrect**

Now a dialog box as shown in figure 3.3 appears on the screen. In this box, with the **Replace** tab selected the word to be replaced is typed in the **Replace** text box and the replacement word in the **With** text box. When the same mistake is made StarOffice Writer will automatically replace the misspelled word with the correct spelling.





**Fig 3.3 AutoCorrect Dialog Box**

### Learn by solving

- 1) Type the following words and observe how StarOffice Writer's Autocorrect feature corrects them.
  - a) And
  - b) Actually
  - c) After the
  - d) Can
  - e) Clear
  - f) The
  - g) This
  - h) That
  - i) To the
  - j) tyhe

- 2) Open AutoCorrect dialog box and type the word organization in Replace list and organisation in With list and press OK. Now, in the document type the word organization and observe what happens.
- 

### **Summary**

- Spelling mistakes can be corrected both while typing as well as after the entire document is typed.
  - Automatic Spelling Correction feature corrects some of the misspelled words.
  - Auto Correct option is very useful for correcting the spelling of the commonly misspelled words.
-

## EXERCISES

### I. Fill in the blanks

- 1) The \_\_\_\_\_ key is pressed to select the Spelling correct dialog box.
- 2) The \_\_\_\_\_ button is used to skip the spelling change for the current word.
- 3) \_\_\_\_\_ option can be used to automatically replace the misspelled word with the correct spelling.

### II. State true or false

- 1) If the right spelling is not known StarOffice Writer can help the user with some choices.
- 2) StarOffice Writer puts a squiggly green line under some words to alert some possible spelling mistakes.
- 3) The StarOffice Writer will automatically correct some of the common misspellings.

### III. Answer the following

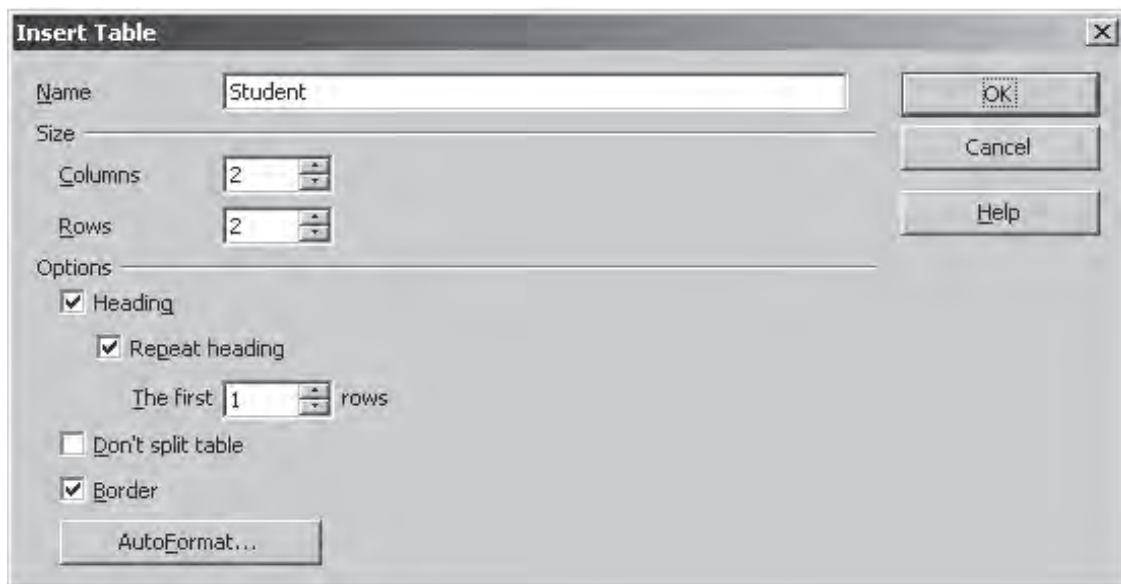
- 1) What does Automatic Spelling Correction mean ?
- 2) How would you carry out the spell check after the entire document is typed?
- 3) How would you add a word in the Auto Correct list of the StarOffice Writer?

## WORKING WITH TABLES

## 4.1 Creating a Simple Table

The following steps are used to create a simple, default - style table:

- 1) When **Table** → **Insert** → **Table** is selected from the menu bar, the **Insert Table** dialog box as shown in figure 4.1 appears on the screen.



**Fig 4.1 Insert Table dialog box**

- 2) In the **Name** textbox the name of the table is given. Using the spin arrows in the **Columns** and **Rows** text box the number of columns and rows are entered. StarOffice Writer displays the table as a grid with the specified number of columns and rows. By default each cell includes a border. The look of this border can be deleted or changed, which is described later in this chapter

**Note:** The floating toolbar as shown in figure 4.2 appears when **View** → **Insert** is selected from the menu bar.



**Fig 4.2 Floating Toolbar for Insertion function**

In this toolbar, the first icon is the **Insert Table** icon. Clicking on the arrow displays a grid. Move the mouse over the grid to specify the number of rows and columns. Alternatively click the icon to insert a display the Insert Table dialog box.

## 4.2 Entering Data in the Table

Once the table is created StarOffice Writer places the insertion point in the first cell of the table. A cell is the intersection of a column and a row. To type something in a particular cell first the insertion point is moved to the required cell and the required data is typed in it. To move to the desired cell first a click is made inside the cell or the arrow keys can be used. One can also press **Tab** key to move forward through the cells or **Shift + Tab** to move backward through the cells.



Any amount of text can be typed within the cell. If the entry is a long one, when the text or data reaches the cell border StarOffice Writer will wrap the text to the next line and expand the cell making it taller and not wider. The contents of the cell can also be formatted and edited using the features learnt in the previous chapters.

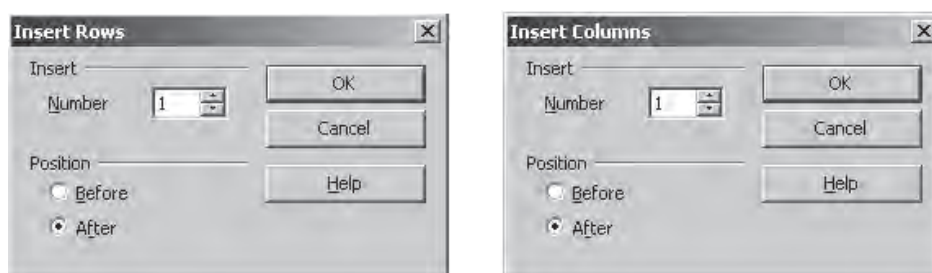
## 4.3 Adding or Deleting Rows and Columns

Many times it is difficult to tell in advance the number of rows and columns that the user needs in a table. After creating a table the required number of rows and columns can either be added or deleted very easily.

### 4.3.1 Adding Rows or Columns

The simplest way to add a row is to press **Tab** in the last row of the table. StarOffice Writer adds a new row. One can also add a row or a column in the table by following the steps given below:

1. To insert a row, select the  icon and press once. Now a new row is inserted below the present row.
2. To insert a Column, select the  icon and press once. Now a new column is inserted to the right of the present column.
3. To insert more than one row or column in the table, **Table → Insert → Rows or Table → Insert → Columns** command is used. Now a corresponding dialog box as shown in the figure 4.3 appears on the screen. Here one can select the rows or columns and the required quantity. If OK button is clicked the required number of columns or rows are added to the table.



### 4.3 Insert Columns / Row dialog box

### 4.3.2 Deleting Rows and Columns

Rows or columns can be deleted by selecting them by using the command shown below:

**Table → Delete → Rows** or **Table → Delete → Columns**.

### 4.3.3 Deleting the Entire Table

If the entire table is selected by dragging the mouse across it and the **Delete** key is pressed, only the entries are **deleted** and not the table. To delete the table the following steps are used:

1. The entire table is selected using **Edit → Select → All** command while keeping the insertion point inside the table.
2. The menu option **Table → Delete → Table** is used to delete the selected table.

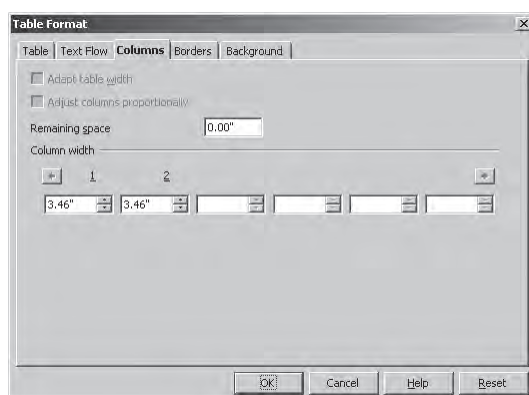
## 4.4 Changing the Rows/Column Width

To change the row/column width, follow these steps:

1. To resize a column, place the cursor in a table cell, hold down Alt, and then press the left or the right arrow. To resize the column without changing the width of the Table, hold down Alt+Ctrl, and then press the left or the right arrows.
2. To resize a row, place the cursor in the row, hold down Alt, and then press the up or the down arrows.

Alternatively if **Table → Table Properties** command is used one would get a **Table Format** dialog box as shown in the figure 4.4. Choose the **Columns** tab and type column size in the **Column Width** spin boxes and click **OK** button to change the column width.

This method can be used when precise measurements for each column are known or needed.



**Fig 4.4 Table Format dialog box**

#### 4.4.1 To Make Selected Rows/Columns the Same Size

Follow the steps to make the selected rows/columns the same size.

- Select the rows and columns that are to be resized.
- To make all columns even, right click inside the table and select **Column**→ **Space Equally** in the pop-up menu.
- To make all rows even, right click inside the table and select **Row**→ **Space Equally** in the pop-up menu.










## 4.5 Table Formatting Toolbar





Once a click is made inside a table a floating toolbar for tables appears at the top of the screen. The toolbar contains a number of icons as shown in the figure 4.5 which can be used for various functions related to the table.



### Fig 4.5 Table Formatting Toolbar

The function of each icon in the toolbar is indicated below:

-  - **Table Fixed**
-  - **Table Fixed, Proportional**
-  - **Table, Variable**
-  - **Merge Cells** - This icon is used to combine two or more cell into a single cell.
-  - **Split Cells** - This icon is used to split a cell into two or more cells.
-  - **Optimise** - Clicking on this icon displays a pop up menu with options like Space columns equally, Space rows equally, Optimum row height and Optimum column width.
-  - **Insert Row** - This icon is used to insert a row below the current row.
-  - **Insert Column** - This icon inserts a column to the right of the current column.
-  - **Delete Row** - This icon deletes the current row from the table.

-  - **Delete Column** - This icon deletes the current Column from the table.
-  - **Borders** - This icon displays a floating toolbar with different border option for the table.
-  - **Line Style** - This icon is used to choose the style of line to be used for the border.
-  - **Border Colour** - Clicking on this icon displays a palette of colours that can be used as a border colour for the table.

## Learn by solving

1. Create a simple table as shown below containing 3 rows and 6 columns and enter the following data as shown.

1.	Sunil	100	98	99	100
2.	Arvind	99	96	97	100
3.	Subish	100	90	100	95

2. Insert one row at the top and enter the following

Sl.No	Name	Maths	Physics	Chemistry	Computer Science
-------	------	-------	---------	-----------	------------------

3. Add two more columns in the table between the Name column and Maths subject column for inserting marks for two subjects viz., Tamil and English and enter some fictitious marks. Also insert two more rows before the last row and enter the details for two more students.
4. Change the borders, line style and background colour of the table to a new one.

## Summary

- Tables can be easily inserted at any point in the document.
- Rows and Columns can be added and deleted at either at the beginning, end or in the middle of the table.
- The row/column height/width can be easily resized to exact sizes.
- The table formatting tool bar contains a number of icons which can be used for various functions related to the tables.



## EXERCISES

### I. Fill in the blanks

1. A table is a grid with a specified no. of \_\_\_\_\_ and \_\_\_\_\_
2. To move backward through the cells of the table \_\_\_\_\_ key combination is used.
3. \_\_\_\_\_ dialog box can be used to set the width of the column to an exact size.
4. \_\_\_\_\_ tool bar is used to change the borders of the table.
5. \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_ command is used to delete the selected column.

### II. State true or false

1. Every table is identified by a specific name.
2. Rows and Columns can be inserted only at the end or beginning of the table.
3. The entire table can be deleted by selecting it and pressing the delete key.
4. There is a separate tool bar available for Table formatting.
5. All the columns in a table will be of uniform width.

### III. Answer the following

1. How will you create a table in the document?
2. How will you add a required number of rows and columns in a table?
3. How to change the width of a column in a table?
4. What are the various functions of the icons in the table formatting tool bar?
5. How to make the selected rows and columns of the same size?

## CHAPTER 5

# PAGE FORMATTING

### 5.1 Changing the Margin

Margins control how close StarOffice Writer prints to the edge of the page. If there is a big top margin for instance, much white space is left at the top of the document. For a small top margin the text is printed closer to the top of the page.

The default margins are 1inch top and bottom margins and 1.25 inches left and right margins. These margins work fine for most of the documents. But there may be occasions when there will be a need to change the size of the margins.

Using any one of the following methods, margins can be changed,

- Page Style Dialog Box
- Changing Margins Using Rulers

#### 5.1.1 Page Style Dialog Box

If the user knows the exact value for the margins then the page style dialog box can be used to make a change. The following steps are used:

- **Format** → **Page** command is selected. A dialog box as shown in the figure 5.1 appears on the screen.
- Click the **Page** tab, if necessary.
- In Margins group, type the new values in the spin boxes or use the spin arrows to change the value.
- Click the **OK** button.

Use the **Page Preview** option in the **File** menu can be used to see the change. This option gives an overall picture of the document and visually shows to the user, how the change has affected the document.

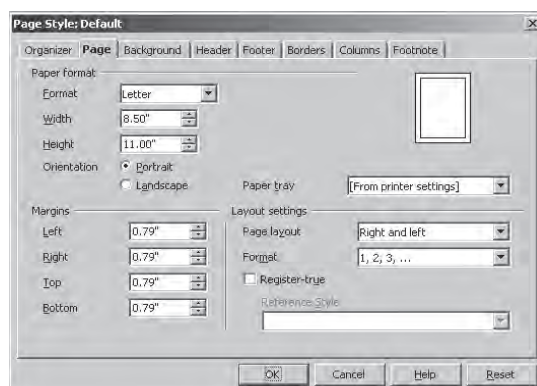


Fig 5.1 Page Style dialog box

### 5.1.2 Changing Margins Using Rulers

If the user is not having the exact value for the margins then the **Ruler** option on the **View** menu can be used to change the margins.

Following steps are used in this method:

- If the ruler is not displayed in the screen, **View** → **Ruler** option is clicked.
- The gray area of the ruler indicates the margin's top area.
- The mouse pointer is then moved in between the gray and white area of the ruler.
- When the pointer is in the right spot, it changes into a line with arrows on both sides
- The margin guide is dragged to a new location.

## 5.2 CHANGING PAGE ORIENTATION

Usually the length of a document will be more than the width. This orientation is called **portrait**. But in some of the documents the width will be more than the length. This type of orientation is called **landscape**. The default orientation is portrait. But, the user can change it to landscape if he wants. Most documents are printed in 8.5 inch by 11 inch paper. But the user has an option to change the paper size.

To change the orientation or paper size, the following steps are used:

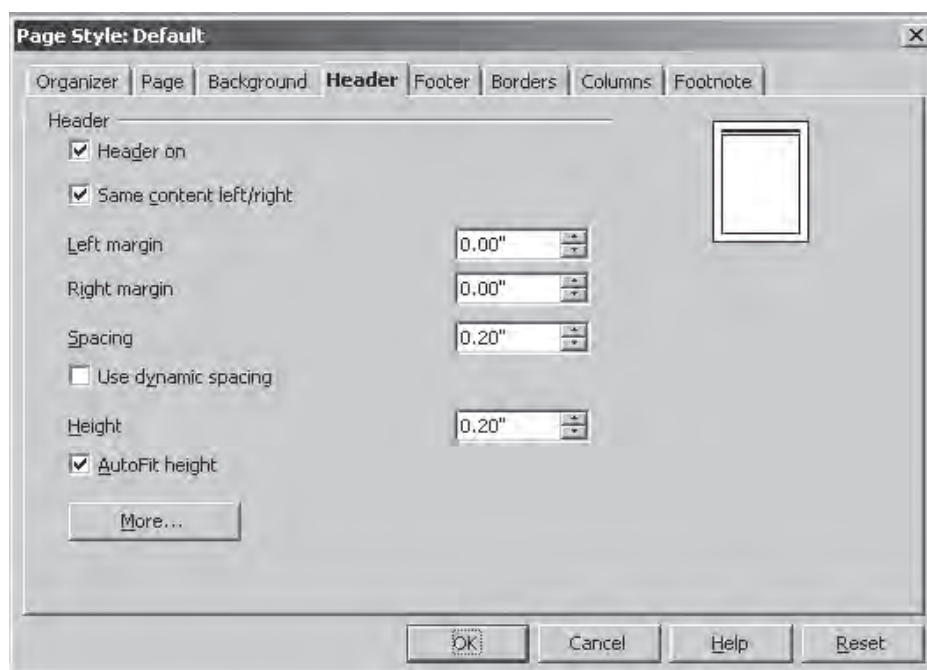
- The **Format** → **Page** option is clicked. The page style dialog box is displayed as shown in the figure 5.1.
- Click the **Page** tab, if necessary.
- Select the necessary paper format from the **Format** drop-down list in the **Page Format** section. Or enter the values in the **Width** and **Height** spin boxes.
- For changing the orientation **Portrait** or **Landscape** radio buttons are used.

## 5.3 CREATING A HEADER AND FOOTER

When the documents are longer than one page, normally some type of reference numbers are added on the page. For instance, page numbers are a must. Otherwise even if the pages are stapled or bound, a particular page cannot be easily referred. In addition, the user may want to include other text that helps the reader identify the document. For example, the document title or the author's name can be included in every page. Rather than typing this information on every page a Header and Footer can be created. **Header** is an area at the top of page and footer is an area at the bottom

of the page. The user can specify what he wants to display in this area and StarOffice Writer will insert the specified text automatically on each page.

To create a header, the **Header** tab on the **Page Style** dialog box can be used. A screen with various options for the header appears as shown in figure 5.2. In this dialog box, the **Header on** check box is clicked. Four spin boxes are also displayed. In those spin boxes the distance of the header from the text area, the header height, the distance from the left margin and the right margin are entered.



**Fig 5.2 Header dialog box**

The same procedure is followed for creating footer. The only difference is, to create footer, the Footer tab is clicked instead of the **Header** Tab.

A document page to which a header and footer has been added using the Page Style dialog box is shown in the figure 5.3. Note that the header and footer area is separated from the rest of the text by a thin line.

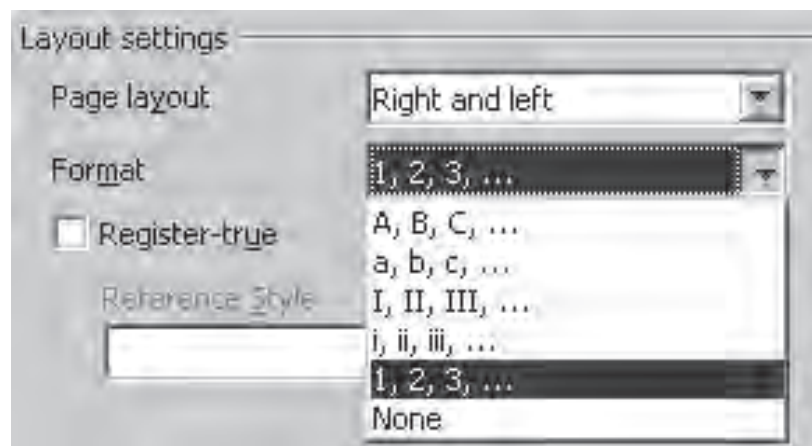
Header	
<p><b>6.4 Creating a Header and Footer</b></p> <p>When the documents are longer than one page normally some type of reference numbers are added on the page. For instance, page numbers are a must. In addition, the user may want to include other text that helps the reader identified the document. For example, the document title or the author's name can be included in every page. Rather than typing this</p>	
Footer	

**Fig 5.3 A Page with Header and Footer**

Once the header and footer are created, the text to be included can be specified. This can be done in the following steps.

1. Click in the header or footer area.
2. Click on **Insert** → **Fields**. A sub menu with a list of options appears. Clicking on an option in the list will insert the appropriate text in all the pages. For example, clicking on **Insert** → **Fields** → **Page Number** will insert page numbers on every page. Normal text can also be included in the header and footer. To do so, simply type the text where you want it.

Once the page numbers have been inserted, their format can be changed using the **Page Style** dialog box shown in figure 5.1. The required style can be selected from the list of styles displayed in the **Format** combo box of the **Page Style** dialog box shown in figure 5.4.



**Fig 5.4 Numbering Combo box**

---

### Learn by solving

1. Open the document Exercise 1. Change the margins and observe the changes.
2. Change the margins to their original settings. Using Print Preview ensure that this has been done.
3. Change the margins again using the Ruler option on the View menu and observe the changes.
4. Change the page orientation and observe the result.
5. Add a header and footer to the document. In the header area, include the date and the topic. In the footer, insert the page number.

## **Summary**

- The margins for a particular page can be set to an exact value using a Page Style dialog box or approximately using Rulers.
- There are two types of page orientation viz., portrait and landscape.
- The page numbers of different styles can be inserted to a StarOffice Writer document.
- Header and footer are some references remarks added at every page of the document of the top and bottom margins respectively.

## **EXERCISES**

### **I. Fill in the blanks**

1. The default margins are \_\_\_\_\_ top and bottom margins and \_\_\_\_\_ left and right margins.
2. \_\_\_\_\_, \_\_\_\_\_ can be used to see the page format change on the screen.
3. \_\_\_\_\_ dialog box is used to set the width of the paper.

### **II. State true or false**

1. Header is normally used to contain the page number.
2. The margin sizes can be set to an exact value.
3. The desired page orientation is selected using a combo box in the page dialog box.
4. Margins can be resized by using the ruler.
5. Portrait orientation is also known as landscape orientation.

### **III. Answer the following**

1. What does page formatting mean?
2. What are the two types of page orientations?
3. How can the ruler be used to change the margin?
4. What is meant by header and footer?
5. Explain the process of changing the margins using rulers.

## CHAPTER 6

# SPREADSHEET

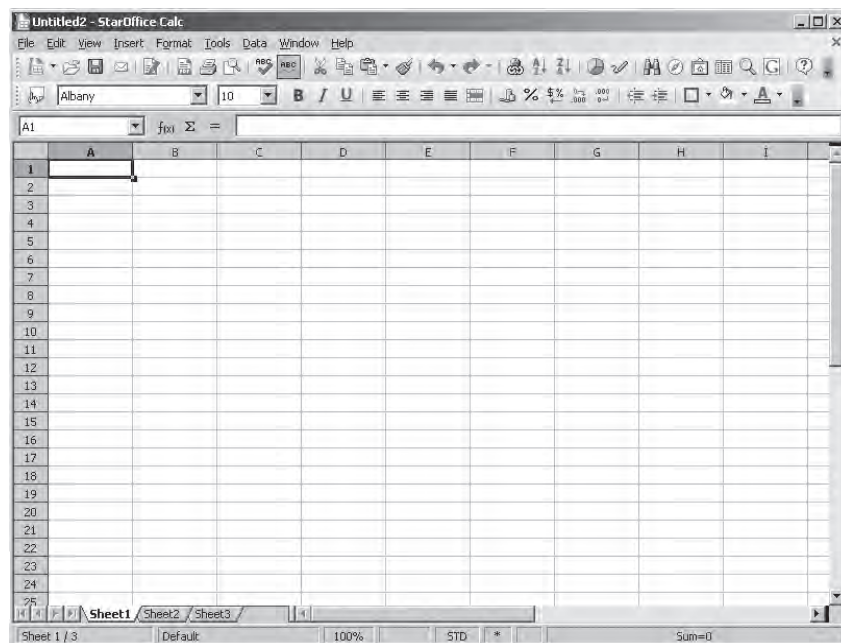
### 6.1 Introduction

The personal computer (PC) first appeared in 1975. But, initially, the PCs were of interest primarily to the electronics hobbyists. Later in 1977 when more usable PCs started appearing, it attracted a wider population. The PCs became more popular when Dan Bricklin and Bon Frankston invented VisiCalc for Apple II in 1979. VisiCalc, “The visible calculator” was the first electronic spreadsheet. This attracted accountants, book keepers, managers and all those who created budget, analysed statistics or collected numerical research data. This powerful computational tool could save time, help avoid endless and brain-numbing arithmetic, and eliminate mathematical errors. It was acknowledged by many that the invention of the spreadsheet, more than any other event, launched the personal computer revolution.

#### 6.1.1 What is an Electronic Spreadsheet?

An electronic spreadsheet is a worksheet used in a computer to create and quickly perform “What if” analysis of interrelated columnar data in workspaces.

Spreadsheets are made up of rows and columns as shown in figure 6.1. The intersection of rows and columns creates cells. The cells are addressed in terms of the row and column labels. Any data, like numbers, text or formulae can be typed into a cell. The power of the spreadsheet lies in the fact that the cells can contain formulae, which perform certain mathematical operations on the data in other cells and display the results in a new cell.



**Fig 6.1 Empty Spreadsheet**

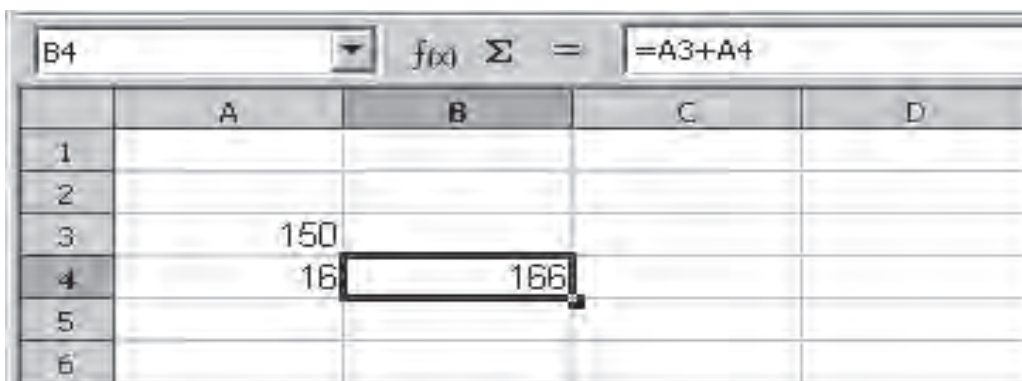


In a spreadsheet, the rows are numbered from 1 to some length (upto 32,000 in the case of StarOffice Calc) and the columns designated with the letters A through Z, AA through AZ, and so on, as provided by different spreadsheet packages.

The following simple example illustrates how to use spreadsheets:

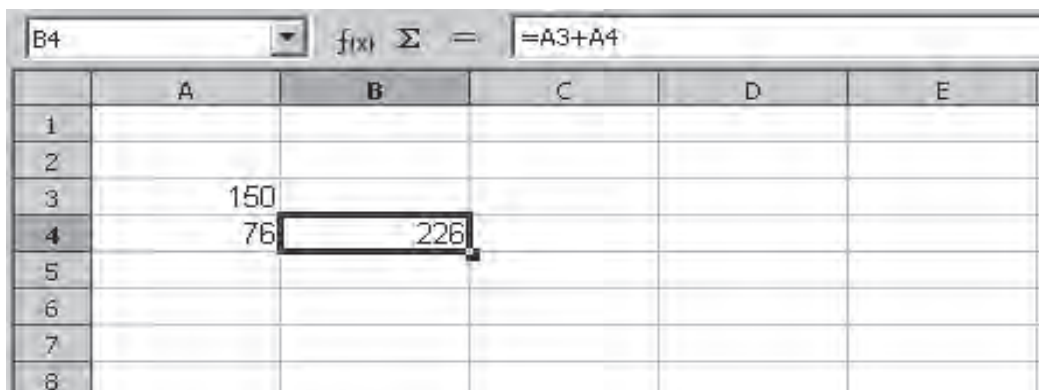
Consider an example of storing a number 150 in a cell and another number 16 in another cell. Add these two numbers electronically and store the result in a new cell. For this purpose, let us use the cells A3 and A4 for storing the numbers 150 and 16 respectively and the cell B4 for storing the sum of these two numbers.

To enter the value 150 in a cell A3, position the cursor in the cell A3, click the mouse to select it and type **150**. Now, you will find the number 150 appearing in cell A3. Similarly, enter the number **16** in cell A4. Let us now add the values in the cells A3 and A4 and store the result in B4. To perform this operation, select the cell B4 by taking the cursor to B4 and clicking the same. Type the formula as **= A3 + A4** or use the sum function as **= sum (A3:A4)**. The formula appears in the input line of the Formula bar (below the menu bars). Press the **Enter** key. You will now find the sum 166 appearing in the cell B4, as shown in figure 6.2.



	A	B	C	D
1				
2				
3	150			
4	16	166		
5				
6				

**Fig 6.2 Calculations in a spreadsheet**



	A	B	C	D	E
1					
2					
3	150				
4	76	226			
5					
6					
7					
8					

**Fig 6.3 Recalculations in a spreadsheet**



A spreadsheet is like a grid of cells with a programmable calculator attached to each cell. The computer can perform calculations at a blinding speed. It does not matter how many numbers and formulae you put in a spreadsheet, the computer can recalculate every formula every time you change any number. For instance, change the number in the cell **A4** in the above example. The spreadsheet recalculates the sum automatically and shows the result in the **Cell B4** (Figure 6.3).

Electronic spreadsheets can also be used for presenting the worksheet data in an impressive manner such as bar-charts, pie-charts, line graphs, three-dimensional charts and other visual forms.

The terms 'spreadsheet' and 'worksheet' mean one and the same. But now, over time, the term 'spreadsheet' has come to refer specifically to the software packages, while 'worksheet' refers to the files that you create with spreadsheet software.

### **6.1.2 Spreadsheet applications**

There are numerous applications possible using electronic spreadsheets. A few of the common applications are given below:

- Payment of bills
- Income tax calculations
- Invoices or bills
- Account Statements
- Inventory Control
- Cost-Benefits Analysis
- Financial Accounting
- Tender Evaluation
- Result analysis of students

### **6.1.3 Advantages of using Electronic spreadsheets**

The electronic spreadsheet offers several advantages over the manual one. The following are some of the main advantages of electronic spreadsheets:

- Calculations are automated through the built-in mathematical, financial and statistical functions.
- Accurate results to any desired level of decimal points are possible
- Worksheets can be quite big in size

- Any part of the worksheet can be viewed or edited.
- Worksheet can be saved and retrieved later.
- Any part or whole of an existing worksheet can be merged with any existing or new worksheet.
- Any part or whole of the worksheet can be printed in a desired format.
- Worksheet data can be viewed in the form of graphs or charts
- The worksheet information can be transferred to any database or word processing software.

#### 6.1.4 Popular Spreadsheet software

Commercial electronic spreadsheet packages are in use since late 1970s. VisiCalc (Visible Calculator) was the first commercial spreadsheet package developed for microcomputers in 1979. It contained 63 columns (A, B,.....BK) and 254 rows. VisiCalc was essentially a financial analysis program.

After seeing the power the success of VisiCalc, Several companies tried to develop spreadsheet packages. Lotus Development Corporation introduced Lotus 1-2-3 in 1982. This package became very popular because of its ability to combine database management and graphics features with its spreadsheet.

The other popular spreadsheet programs are 'Excel from Microsoft Corporation 'Quattro Pro from Borland International, 'Improve' from Lotus Corporation and 'StarOffice Calc' from Sun Microsystems.

### 6.2 Working with StarOffice Calc

StarOffice Calc is a powerful spreadsheet program included in StarOffice. It offers all the functions needed for business use, including various financial and statistical functions, StarOffice Calc database functions and much more.

Creating a worksheet is a process that involves several steps like organizing the data, entering the data, creating formulae, editing the worksheet, formatting values, labels, and cells, adding charts if required, analysing the data and printing the worksheet.

Let us learn how to create a worksheet using StarOffice Calc.

#### 6.2.1 Creating Your First Worksheet

To work with StarOffice Calc, open **StarOffice** and click on the **File** option in the main menu bar. Then click on **New→ Spreadsheet**. A new spreadsheet opens up as shown in figure 6.1

At the top of the window is the **Menu bar**. To use a menu, point to it with the mouse cursor and press the left mouse button. The menu will open displaying a list of options and you can select any option by clicking on it. Below the Menu bar is the **Main toolbar**. Below the Main toolbar are the **Function bar** and the **Object bar**. These bars have shortcut icons for frequently done tasks. Below these bars is the **Formula bar**. This bar is used to display the current cell and its contents. It also has a few more shortcut icons. Below the formula bar are the column headings of the worksheet. Next to it are the row headings. The data area is in the middle of the window. At the extreme bottom are the **Status Bars** and **Scroll bars**.

Before you can create your first worksheet, you have to first collect the data that you want to include in the worksheet. Suppose you want to create a worksheet containing the details of marks obtained by students. To do this, you can collect details such as 'Reg.No.', 'Name of the student' and 'Marks' of different subjects, obtained by each student. Then you can enter the data into the worksheet. Once the data has been entered into the worksheet, you can perform various calculations on this data and calculate things like 'Total marks' of all the subjects and 'Average marks'. You can then, format the data in the worksheet to make it look attractive. You can also draw a graph using this data.

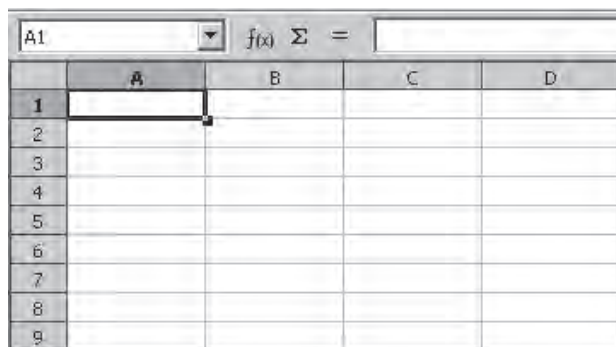
### 6.2.2 Entering Data in the Worksheet

After deciding and collecting the data to be entered in the worksheet, you are now ready to actually enter the data. The active cell in which you want to type the data is identified by the cell pointer which is a rectangular box covering that cell. To begin with, the cell pointer is always in cell A1. You can change the position of the cell pointer by clicking the mouse on the concerned cell or by using the arrow keys on the keyboard. The Tab, Home, End, PgUp and PgDown keys on the keyboard also allow you to move around the worksheet.

After selecting the cell, enter the data. If the data entered is a number, the program recognizes that as a number and allows you to perform calculations on it. If the data entered is a word, the program recognizes it as a label and does not permit you to perform calculations on it.

StarOffice Calc also allows you to enter dates and time in the worksheet. This is very useful because you can also perform various calculations using them. For example, you can find the difference between two dates, add a number to a date and find the new date and so on. You will learn more about this later. You can enter dates in the worksheet just like you enter numbers and labels. Place the cursor in the cell where you want to enter the date and type it in as **MM/DD/YY**. You can enter the time in the worksheet by typing it as **HH:MM:SS**.

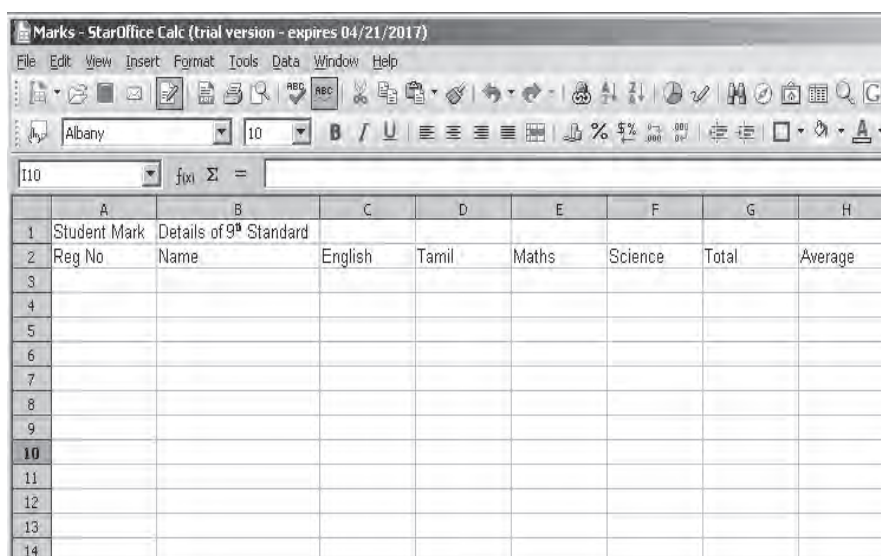
Listed below are the steps to create the worksheet for the student marks. Note that the cursor is in **Cell A1** to begin with.



**Fig 6.4 Blank Screen**

- In cell A1, type the title as 'Student Mark Details of 9th Standard'. Press the down arrow key to move to cell A2.
- In Cell A2, type the heading 'Reg.No'. Press the right arrow key to move to cell B2.
- In cell B2, type 'Name'. Move to cell C2.
- In cell C2, type the subject name as 'English'
- In cell D2, type 'Tamil'
- In cell E2, type 'Maths'
- In cell F2, type 'Science'
- In cell G2, type 'Total'
- In cell H2, type 'Average'

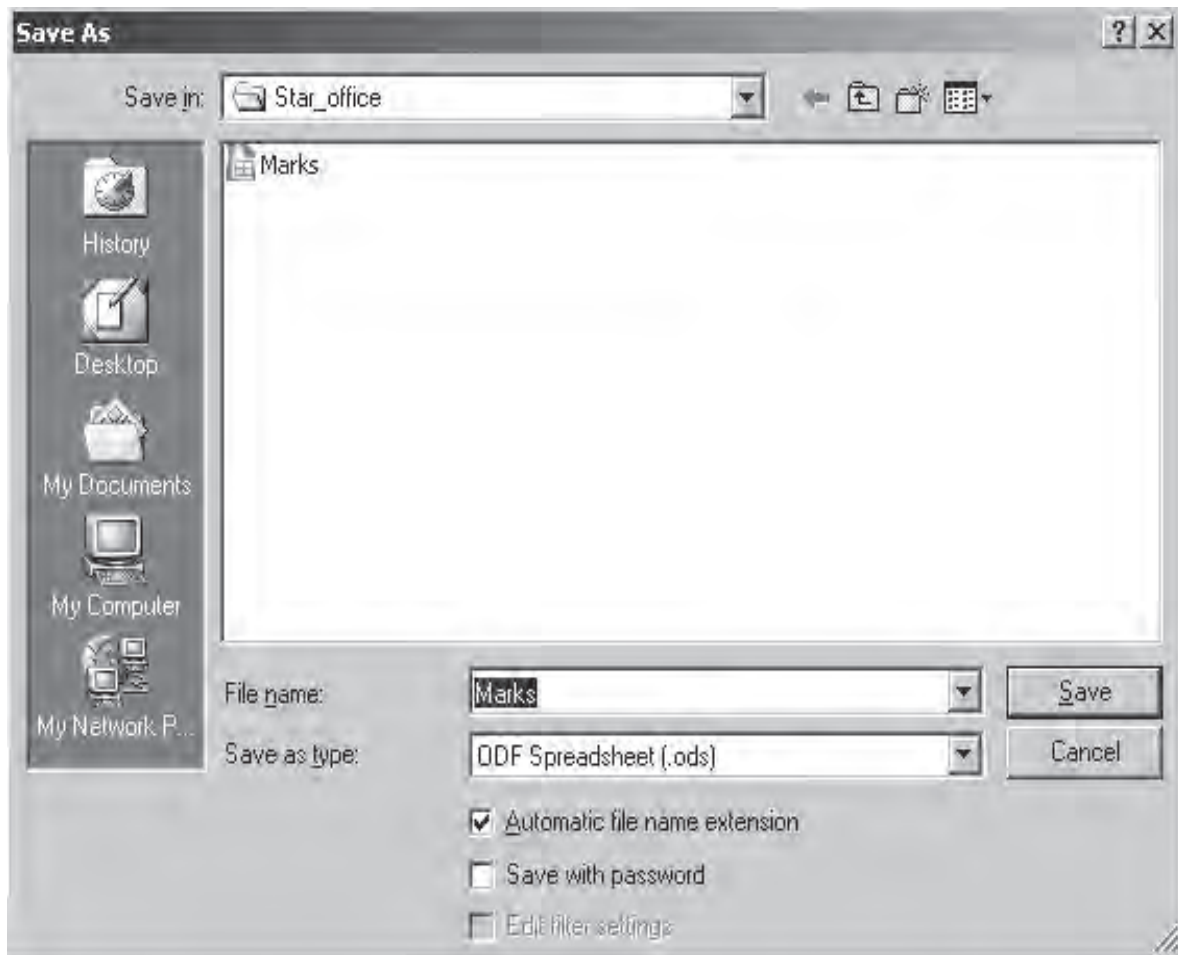
The worksheet thus created is shown in figure 6.5




**Fig 6.5 The student marks worksheet**

### 6.2.3 Saving the Worksheet

To save the worksheet created, go to the **File** menu and select the **Save** or the **Save As** option. A screen appears as shown in figure 6.6




**Fig. 6.6 Save As Dialog box**

Type in a file name and click on **Save**. For example, to save the student marks, type the name **Marks** in the **File name** box and click on the **Save** button. You can also click on the **Save**  icon on the Standard toolbar.

### 6.2.4 Closing the worksheet

In order to close the worksheet, go to **File** menu and select the **Close** option.

### 6.2.5 Opening a Worksheet

To open a worksheet that has been saved, select the **Open** option from the file menu. A dialog box with a list of files appears on the screen. Select the file that you want by clicking on it and then click on **Open**. You can also click the **Open**  icon on the Standard toolbar to open an existing file.

## 6.2.6 Quitting from StarOffice

The **Exit** option under the **File** menu can be used to quit from StarOffice.

### Learn by solving

Create the Marks worksheet created above with the following data:

Reg No.	Name	English	Tamil	Maths	Science
1000	Kumar A.	87	85	74	86
1001	Aravindan J.	63	86	62	94
1002	Govindan S.	63	76	73	75
1003	Velmurugan T.	75	72	63	85
1004	Thamizharasi G.	75	46	52	64

After you complete the data entry the worksheet will look be as shown below.

	A	B	C	D	E	F	G	H
1	Student Mark	Details of 9th Standard						
2	Reg No	Name	English	Tamil	Maths	Science	Total	Average
3		1000 Kumar A	87	85	74	86		
4		1001 Aravindan J	63	86	62	94		
5		1002 Govindan S	63	76	73	75		
6		1003 Velmurugan	75	72	63	85		
7		1004 Thamizharasi	75	46	52	64		
8								
9								
10								
11								
12								

**Fig. 6.7 Worksheet of the Student Database**

Note that in cells B6 and B7, the complete name is not displayed.

This is because the width of Column B is not enough; StarOffice Calc indicates this with small red triangles. Later in this chapter, you will learn how to increase the column width.

Save the worksheet as **Marks**

## 6.3 Editing the Data in the worksheet

To edit the data present in a worksheet, first open the worksheet by clicking on **File** → **open**. Next, move the cursor to the cell, which you want to edit. Note that the contents of the cell are displayed on the formula bar also. You can edit the contents in the following two ways:

1. Type in the new data. The new data will simply overwrite the old contents of the cell.
2. Click on the formula bar with the mouse, press the **F2** function key or simply double-click on the cell. A vertical cursor appears on the formula bar. Move the cursor to the left using the left arrow key or the backspace key and edit the data.

## 6.4 Creating Formulae

After entering data in the worksheet you can perform calculations on the data in the worksheet. This is done using formulae. In order to create formulae, you first need to know the syntax that describes the format of specifying a formula. The syntax of formula begins with an equal sign followed by a combination of values, operators and cell references. The various operators available for calculations, in StarOffice Calc are given below:

### 6.4.1 Arithmetic Operators

These operators return numerical results

Operator	Name	Example
+(plus)	Addition	1+1
-(minus)	Substraction	2-1
-(Minus)	Negation	-5
*(asterisk)	Multiplication	2*2
/(Slash)	Division	9/3
% (Percent)	Percent	15%
^ (Caret)	Exponentiation	3^2

### 6.4.2 Comparative operators

These operators return either true or false.

Operator	Name	Example
=	Equal	A1=B1
>	Greater than	A1>B1
<	Less than	A1<B1
> =	Greater than or equal to	A1>=B1
< =	Less than or equal to	A1<=B1
< >	Inequality	A1<>B1



### 6.4.3 Text operators

The operator combines sections of text to the entire text.

<u>Operator</u>	<u>Name</u>	<u>Example</u>
&(And)	Text operator: And	“Star” & “Office” yields StarOffice”

### 6.4.4 Reference Operators

These operators combine areas.

<u>Operator</u>	<u>Name</u>	<u>Example</u>
: (Colon)	Range	A1:C108
! (Exclamation point)	Intersection	SUM(A1:B6!B5:C12)

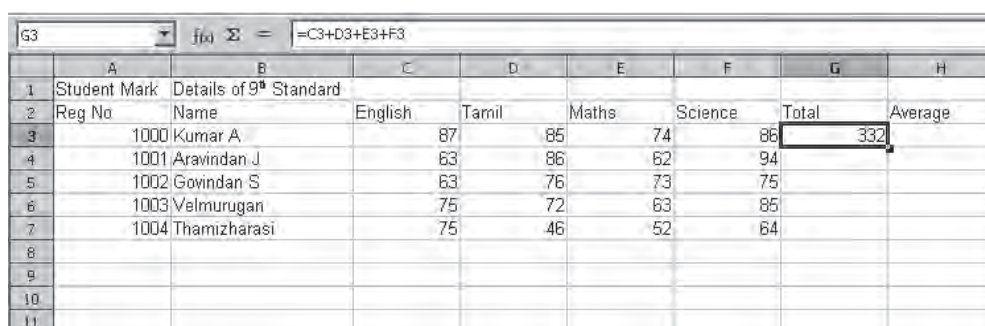
When arithmetic operators are used in formulae, StarOffice Calc calculates the results using the rules of precedence followed in Mathematics. The order is as follows:

1. Exponentiation (^)
2. Negation (-)
3. Multiplication and Division (\*, /)
4. Addition and Subtraction (+, -)

Here is an example to illustrate how to create formulae:

- Place the cell pointer in the cell where you want to enter the formula. In the Marks worksheet example, place the cursor in cell G3.
- Type the formula as =C3+D3+E3+F3 and press the **Enter** key. The total mark of the student Kumar A. appears in the Cell G3.

The marks worksheet with the total thus calculated is shown in figure 6.8



The screenshot shows a spreadsheet with columns A through H. Row 1 is the header for 'Details of 9<sup>th</sup> Standard'. Row 2 contains subject names: English, Tamil, Maths, Science, Total, and Average. Rows 3-7 contain student data. Row 3 (Kumar A) shows marks of 87, 85, 74, and 86, with a calculated total of 332 in cell G3. The formula bar at the top shows '=C3+D3+E3+F3'.

	A	B	C	D	E	F	G	H
1	Student Mark	Details of 9 <sup>th</sup> Standard						
2	Reg No	Name	English	Tamil	Maths	Science	Total	Average
3	1000	Kumar A	87	85	74	86	332	
4	1001	Aravindan J	63	86	62	94		
5	1002	Govindan S	63	76	73	75		
6	1003	Velmurugan	75	72	63	85		
7	1004	Thamizharasi	75	46	52	64		
8								
9								
10								
11								

**Fig. 6.8 The worksheet of the Student Database with Total**



## 6.5 Fill Command

You have learnt how to create and use formulae to calculate. But, in the above example, you have calculated only one total. To calculate the other totals, you can type the corresponding formulae in the cells G4, G5, G6 and G7. There is an easier way of entering these formulae. You can type the formula in cell G3 and then copy it to the remaining cells. This can be done using the **Copy** and **Paste** icons on the standard toolbar. Recall that you learnt how to use these icons in StarOffice Writer. You can also use the **Automatic Fill** feature of StarOffice Calc.

AutoFill automatically generates a data series based on a defined pattern.

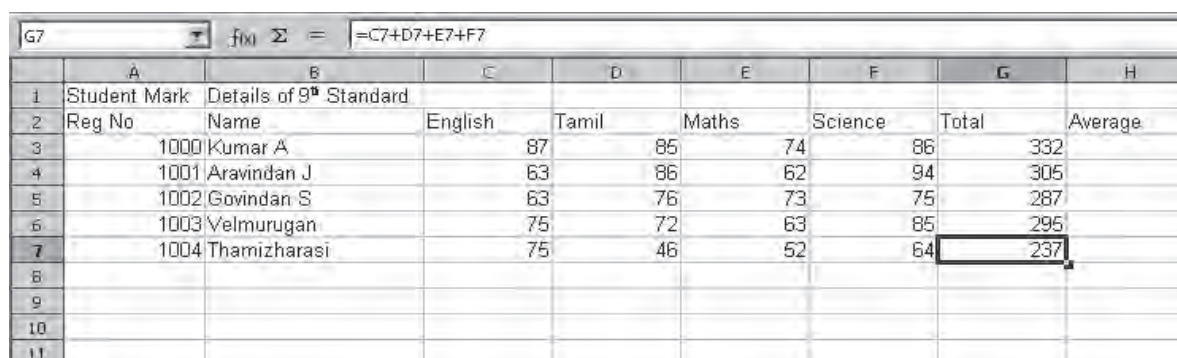
1. On a sheet, click in cell, and type a number.
2. Drag the fill handle in the bottom right corner of the cell across the cells that you want to fill, and release the mouse button.

The cells are filled with ascending numbers.

To copy the contents of a cell, click on the cell. Click and drag the mouse to highlight all the cells where you want to copy the contents. Now, select **Edit** → **Fill** → **Down (or Left)**. The content of first cell will be copied in all the highlighted cells.

**Note:** A continuous group of cells in a worksheet is called a **Range**. A range is referred to by the range address. A range address is the address of the first cell in the range, followed by a colon, followed by the address of the last cell in the range. For example, the cells, G1, G2, G3, G4 and G5 can be called G1:G5. The cells A1, B1, C1, D1, E1 and F1 can be called A1:F1 and the cells A4, A5, A6, B4, B5 and B6 can be referred to as A4:B6.

For example, to copy the contents of cell G3 in the cells G4, G5, G6 and G7, highlight the cells G3 to G7 (also referred to as G3:G7). Click on **Edit** → **Fill** → **Down**. The contents of the cell G3 will be copied to all the other cells, as shown in figure 6.9.

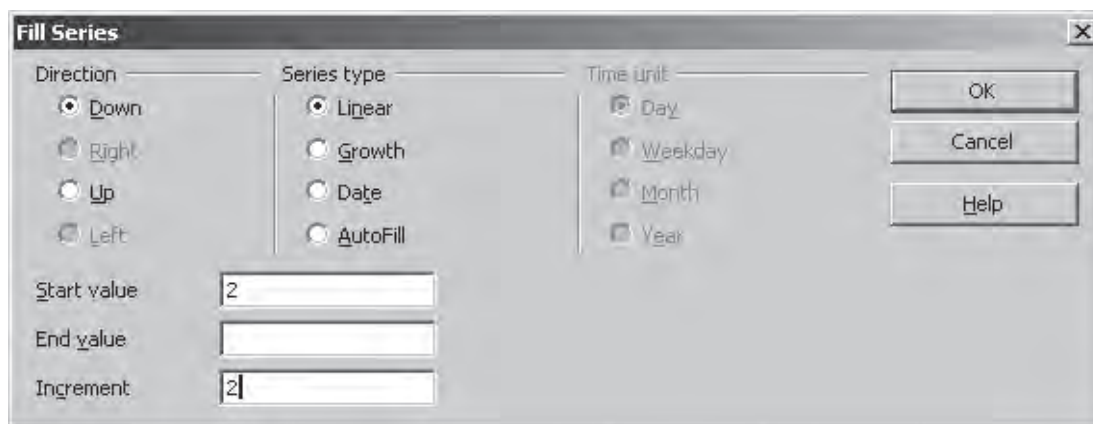


	A	B	C	D	E	F	G	H
1	Student Mark	Details of 9 <sup>th</sup> Standard						
2	Reg No	Name	English	Tamil	Maths	Science	Total	Average
3	1000	Kumar A	87	85	74	86	332	
4	1001	Aravindan J	63	86	62	94	305	
5	1002	Govindan S	63	76	73	75	287	
6	1003	Velmurugan	75	72	63	85	295	
7	1004	Thamizharasi	75	46	52	64	237	
8								
9								
10								
11								

**Fig. 6.9** The contents of cell G3 has been copied to G4:G7

You can also use the Fill command to generate a series of data directly from the values of the selected cells. First, select the cells of the worksheet that you want to fill. Choose the command **Edit** → **Fill** → **Series**. Select the type of series from the options that appear as shown in figure 6.10

For example, if you want to enter numbers in the cells of a column in an increasing order with a difference of 3 between subsequent cell values, you will have to just enter only the initial value in the first cell of the column and specify the direction, type and increment. In this case, select **Direction** as **Down**, **Type** as Linear and **Increment** as **3**. Then by clicking the **OK** button the values in the subsequent cells of the column will be automatically generated.



**Fig. 6.10 Fill series dialog box**

As another example, select the range A1:D6 in the worksheet. Click on **Edit** → **Fill** → **Series**. Choose

2 as your **Start value**

2 as your **Increment**

**Growth** as the **Type**, and

**Down** as the **Direction**.

Now, click on **OK** and you will find the worksheet filled as shown in figure 6.11

	A	B	C	D	E
1	2	4	8	16	
2	4	8	16	32	
3	8	16	32	64	
4	16	32	64	128	
5	32	64	128	256	
6	64	128	256	512	
7					
B					

**Fig. 6.11 Result for Fill command**

As you can see in the dialog box, you can also automatically fill in series of dates and times. For example, to list all Sundays in a given period, say March to May 2005, proceed as follows:

- Enter the date as 3/5/05 into a cell
- Select this cell and adequate number of cells depending upon the stop value (in this case, 13 cells since the period is March – May 2005).
- Select the command **Edit→Fill→Series**
- In the dialog box, select **Day** as the **Date Unit** and enter the **Increment** as 7. Click **OK**.

The Sundays of March, April and May 2005 automatically appear in the selected cell as shown in figure 6.12.

	A	B	C	D
1	Dates of Sundays in March, April and May 2005			
2				
3		03/06/2005		
4		03/13/2005		
5		03/20/2005		
6		03/27/2005		
7		04/03/2005		
8		04/10/2005		
9		04/17/2005		
10		04/24/2005		
11		05/01/2005		
12		05/08/2005		
13		05/15/2005		
14		05/22/2005		
15		05/29/2005		
16				
17				

**Fig 6.12 Auto filling Date and Month**

## 6.6 Cell Referencing

In the Marks worksheet example, you typed the formula = C3+D3+E3+F3 in cell G3 and then copied it to the cells G4:G7. Click on the cell G4. Note that the formula in this cell is =C4+D4+E4+F4. This is because spreadsheets refer the cell addresses in a formula not as absolutes but in a relative way. For example, consider the formula in cell G3. StarClac reads this formula as

- Add the value in the cell 4 columns to the left of current cell
- With the contents of the cell 3 columns to the left of current cell
- With the contents of the cell 2 columns to the left of the current cell
- With the contents of the cell 1 column to the left of the current cell.

So, when this formula is copied to the cell G4, it does not read as = C3+D3+E3+F3 but as

- Add the value in the cell 4 columns to the left of current cell
- with the contents of the cell 3 columns to the left of current cell
- with the contents of the cell 2 columns to the left of current cell
- with the contents of the cell 1 columns to the left of current cell

That is, = C4+D4+E4+F4.

This type of cell referencing is called Relative cell addressing. **Relative cell addressing** is the default type of cell addressing used by StarOffice Calc.

Relative cell addressing is also the reason why formulae are automatically recalculated every time the contents of the cells used in the formulae change.

The other type of referencing used in spreadsheets is **Absolute cell addressing**. A cell address can be made absolute by using the \$ (dollar) sign in front of row and column names. For example, the C4 becomes absolute when you enter it as \$C\$4.

**Absolute cell addresses do not change when copied.**

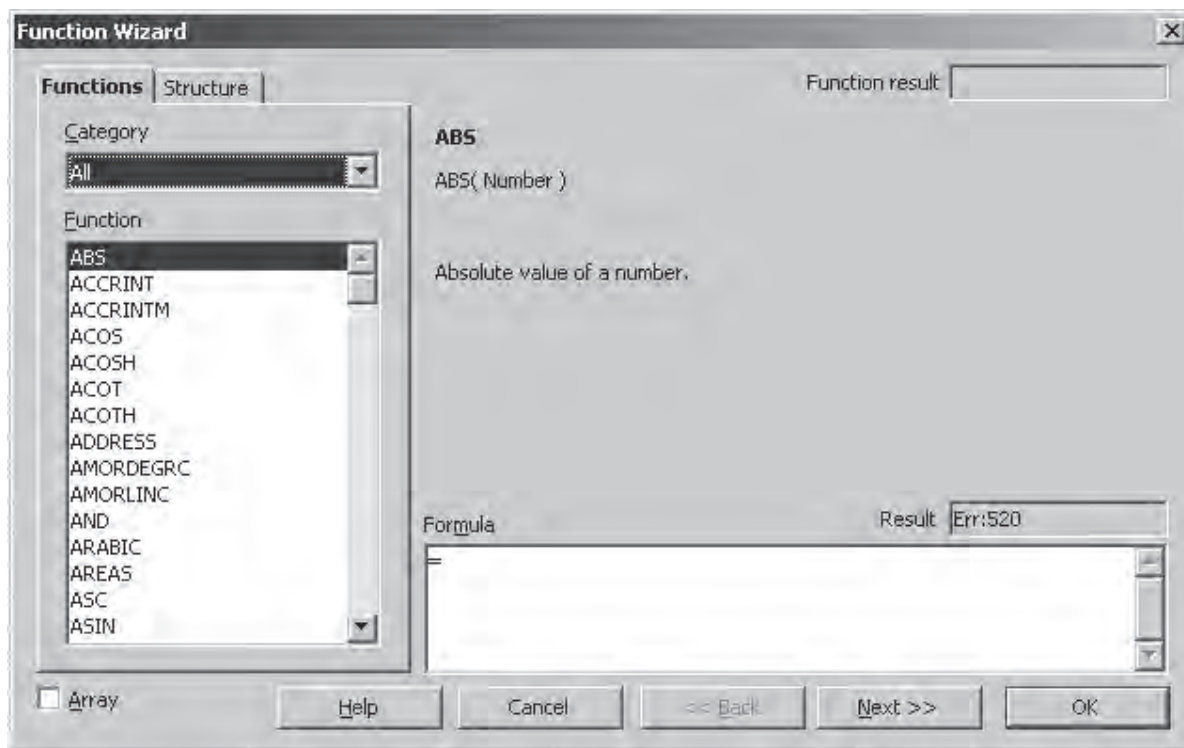
## 6.7 Using Functions

StarOffice Calc has a wide variety of functions that allow you to perform several frequently done calculations. Functions are predefined formulae that are available in StarOffice Calc.

These functions are available in StarOffice Calc in the pull down menu of **Function Wizard** window as shown in figure 6.13. The functions available are divided into different categories. The categories are listed in the category pull down menu.

To select a function, go to **Insert** menu and Select the **Function** option. The **Function Wizard** dialog box appears. A list of all **functions** is displayed in the Function box when **All** is selected in the **Category** box. If a category is selected (e.g. Mathematical) the functions related to that category alone will be displayed in the


function box.



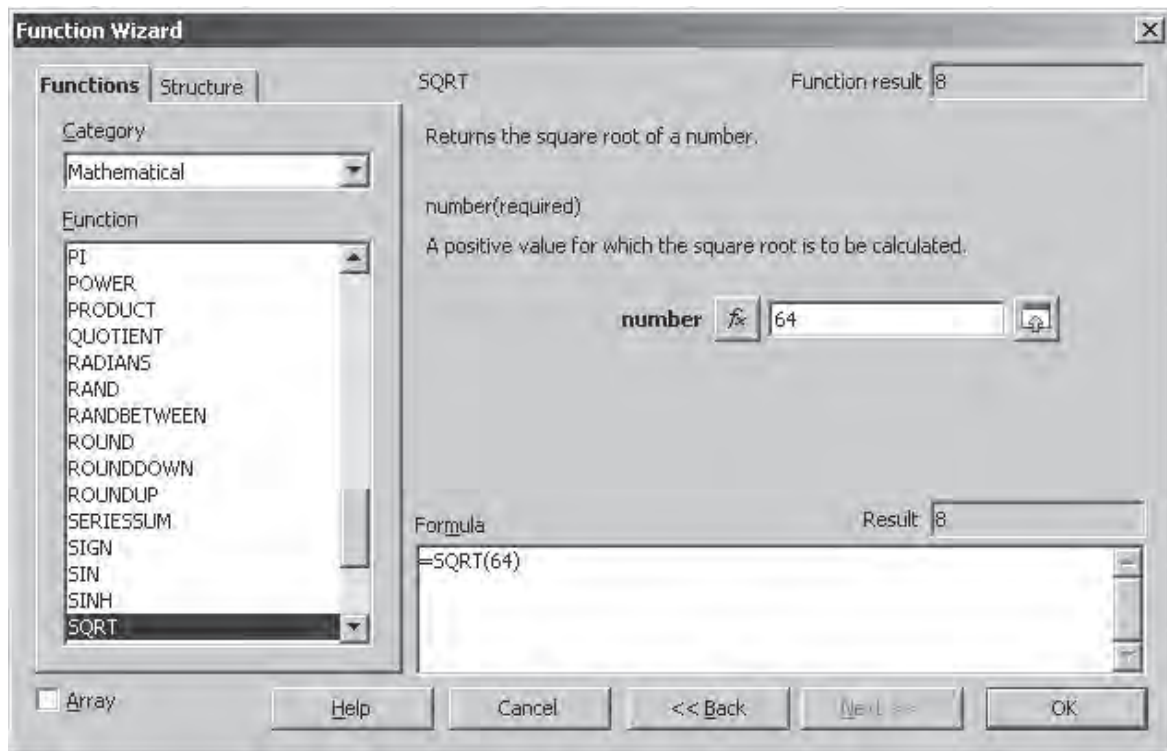
**Fig 6.13 Function Wizard window**

To select a function:

- Select the category in the **Category** box.
- Scroll down the list to find the function you want. Click once on the function name to see a short description of that function on the right side of the window. Double – click on it to insert it into the worksheet.

 The **Function Wizard** shortcut icon on the formula bar can also be used to select and insert functions.

For example, to insert the SQRT function (a function to find the square root of a number), place the cursor in the cell where you want to insert the function and click on the **Function Wizard** icon. Select **Mathematical** from **Category**. Select the **SQRT** function from the list of functions, which appears by double clicking on it. The Function Wizard displays a brief description of the function and prompts you to enter the number or the cell address on which the function should work. Enter the number **64**.



**Fig 6.14 The Function Wizard prompts you to enter a number or a cell address**

When you click on **OK**, the result is displayed in the cell in the worksheet.

**Σ** One of the most commonly used function is the **Sum** function. This function calculates the sum of a given set of numbers. To use this function, you can either click on the **Function Wizard** icon or then select **SUM** or you can just click on the **Sum** icon on the formula bar. StarOffice Calc suggests a range of cells for which sum is to be calculated. Press **Enter** to accept this suggested range or press the **Backspace** key and type in the range that you want.

For example, in the Marks worksheet, to calculate the total marks in cell G3, follow the steps given below.

- Click on the cell G3 to place the cursor there.
- Click on the **Sum** icon on the function bar. **= SUM (C3: F3)** appears in the cell.
- Press **Enter** to accept the suggested range. The result, that is, 332 is displayed in the cell G3.

### Learn by solving

1. Open the worksheet **Marks**.
2. Use a formula to calculate the total in cell G3.



3. Use the **Fill** command to copy the formula to the cell G4:G7.
  4. Close the worksheet without saving.
  5. Open the worksheet again and use the **Sum** function to calculate the total in cell G3.
  6. Use the **Fill** command to copy the formula to the cells G4:G7.
  7. Save the worksheet.
  8. Enter a formula in cell H3 to calculate the average marks.
  9. Use the **Fill** command to copy the formula to the cells H4:H7.
  10. Close the worksheet without saving.
  11. Open the worksheet again and use the **AVERAGE** function to calculate the average in cell H3.
  12. Use the **Fill** command to copy the formula to the cells H4:H7.
  13. Save and close the worksheet.
- 

## 6.8 Date Arithmetic

Manual date calculations can be tricky because you have to keep track of the number of days in a month. In spreadsheets, date calculations become very simple. Here you can add a number to a date and arrive at a new date, find the difference between two dates and use a wide variety of function and formats to get what you want.

For example, enter a date 03/04/05 in a cell, say A2. Remember that while entering dates the month always comes first. Suppose you want to calculate the date 79 days after this date. To do so, enter the formula, **= A2 + 79**, in another cell, say A4. The date 05/22/05 appears in the cell.

Now, suppose you want to calculate the difference between two dates, 05/10/05 and 12/8/70. To do so, enter the two dates in two different cells. In a third cell enter the formula **= first cell – second cell**. The result will be displayed as **12572**.

## 6.9 Formatting the Worksheet

In the earlier chapters, you learnt the various formatting options of StarOffice Writer. You can use several of those options in StarOffice Calc also. In addition,

StarOffice Calc provides you with several formatting options for formatting numbers. It is important to remember that by formatting the contents of a cell, only the display changes; the contents remain unchanged.

Before formatting the cells in a worksheet, you have to select the cells that you wish to format. You already know how to select one cell. Just click on the cell and it will be selected. To select a group of cells (a range), click on the first cell in the range and click and drag to the last cell in the range.

To select a complete row or a column, click on the respective row number or column name. To select more than one row or column, click on the first row number or column name and click and drag till all the rows and columns you want are selected.

Listed below are some of the formatting options available in StarOffice Calc.



This is the **Bold** icon and is used to display data in bold. To use this icon, highlight the cells and click on the icon.



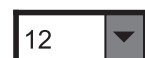
This icon is used to display the data in italics.



This is used to underline the data in highlighted cells.



This is the **Change Font** icon. This icon displays list of fonts that can be used. Select the font by clicking on it.



This icon is used to change the font size of the data. To do so, select the data and click on this icon.



This is the **Font Colour** icon. This can be used to change the font colour.



These are the **Align Left**, **Align Center**, **Align Right** and **Justify** icons. They are used to align the contents of cells.



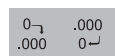
This is the **Number Format: Currency** icon. Clicking on this will display the contents of the selected cells in currency format, that is with a \$ in front and with two decimal digits.



This is the **Number Format: Percent** icon. Clicking on this icon will display the current contents in percentage format. Note that it multiplies the contents of the cell by 100 and displays the result with 2 decimals.



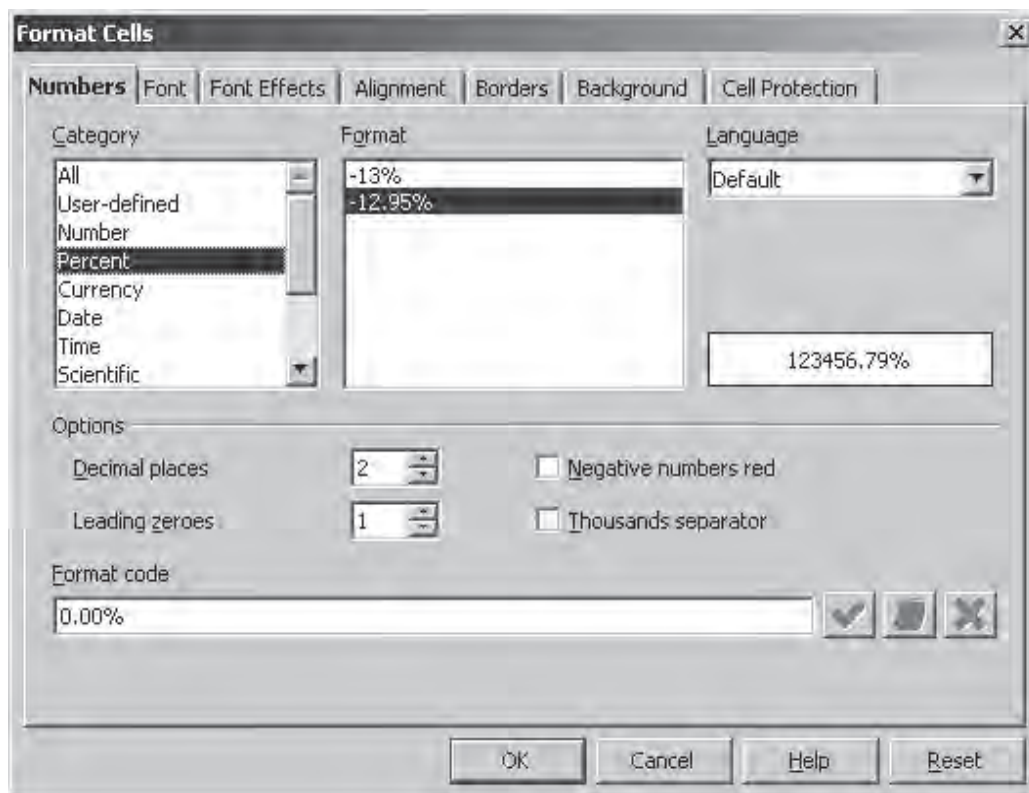
This is the **Number Format: Standard** icon. Clicking on this icon will display the contents of the selected cells in default format.



These are the **Number Format: Add Decimal** and **Number Format: Delete Decimal** icons. They are used to increase or decrease the number of decimal digits that are to be displayed in the selected cells.



The **Format** menu can also be used to format cells. To do so, select the cells you want to format and click on **Format** → **Cells**. The **Format Cells** dialog box appears as shown in figure 6.15.



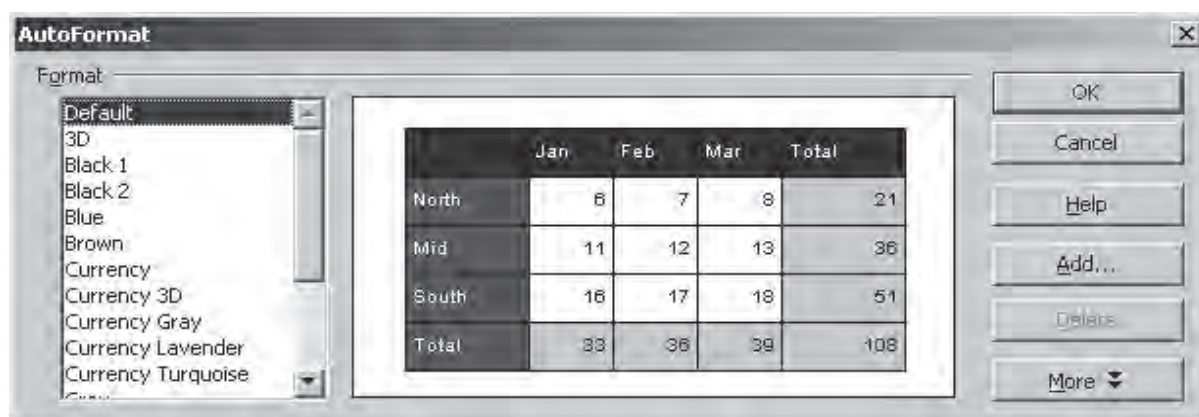
**Fig 6.15 Format Cells dialog box**

Tabs at the top of the dialog box can be used to choose the type of cell attribute you want to use. For example, the **Numbers** tab can be used for changing the format of numbers and the **Alignment** tab can be used to change the **alignment** of data in cells. Clicking on a tab will display all the formatting options available along with a preview of how the data will look if that format is used.

### 6.9.1 AutoFormat Sheet

The AutoFormat Sheet facility of StarOffice Calc helps to format the worksheet with different predefined styles and colours. For example, let us format the marks worksheet as detailed below:

- In the Marks worksheet, select the cell from A1 to H7.
- Click on the **AutoFormat** option on the **Format** menu.
- The **AutoFormat** dialog box appears, as shown in figure 6.16, displaying various predefined format styles. A preview for each style is also displayed in the box.



**Fig 6.16 AutoFormat dialog box**

- Select the **Default** format. The formatted worksheet is shown below.

	A	B	C	D	E	F	G	H
1	Student Mark	Details of 9 <sup>th</sup> Standard						
2	Reg No	Name	English	Tamil	Maths	Science	Total	Average
3	1000	Kumar A.	87	85	74	86	332	83
4	1001	Aravindan J	63	86	62	94	305	76.25
5	1002	Govindan S	63	76	73	75	287	71.75
6	1003	Velmurugan	75	72	63	85	295	73.75
7	1004	Thamizharasi	75	46	52	64	237	59.25
8								
9								

**Fig 6.17 Auto Formatted worksheet**

## 6.10 Changing Column Width and Row Height

Often while entering data into a worksheet you will realize that the width of the column is not enough. StarOffice Calc. allows you to change the width of a column and the height of a row. To change the column width, select the column whose width you want to change. Click on **Format** → **Column** → **Width** and type the new column width in the dialog box, which appears as shown in figure 6.18. Click on **OK**.



**Fig 6.18 Changing column width**

You can also change the column width in another way. Point to the line separating the column whose width you want to change from the next column. The mouse pointer becomes a double – headed arrow. Click and drag this pointer to the left to decrease the width and to the right to increase the width.

To change the row height, select the row whose height you want to change. Right click on the selected row and select **Height** from the menu that appears. Type the new height and click on **OK**.



**Fig 6.19 Changing row height**

You can also change the row height of a particular row by clicking and dragging the line separating that row from the next.

---

### Learn by solving

Format the Marks worksheet as follows,

1. Format all heading in Bold.
2. Change the font, size and color of the headings.
3. Change the format of the average column to display 2 decimal digits.
4. Using AutoFormat change the style and colors of the worksheet.
5. Change the row width and column width wherever necessary.

---

## 6.11 Inserting Cells, Rows and Columns

Often after creating a worksheet, you find the need to insert a row or column in the worksheet. For example, you may want to add another subject to the Marks worksheet. StarOffice Calc allows you to insert one or more cells, rows and column.

To insert an empty cell or an empty row or a column in a worksheet already created, follow the procedure given below:

- Click the **Insert Cell** from **View** → **Toolbar** menu. A floating toolbar with four icons appears. These icons are **Insert Cells Down**, **Insert Cells Right**, **Insert Rows** and **Insert Columns** icons.



- In order to insert an empty cell in a column and move the existing cells down, place the cursor in the cell where you want to insert the new cell and click on the **Insert Cells Down** icon. For example, in the Marks worksheet, to insert a cell in D4 and move the contents of the cell D4:D7 down, select D4 and click the icon. The output screen is shown in figure 6.20.

	A	B	C	D	E	F	G	H	I
1	Student Mark	Details of 9 <sup>th</sup> Standard							
2	Reg No	Name	English	Tamil	Maths	Science	Total	Average	
3	1000	Kumar A	87	85	74	86	332	83	
4	1001	Aravindan J	63		62	94	305	76.25	
5	1002	Govindan S	63	86	73	75	287	71.75	
6	1003	Velmurugan	75	76	63	85	295	73.75	
7	1004	Thamizharasi	75	72	52	64	237	59.25	
8				46					
9									

**Fig 6.20 Worksheet with cell inserted**

- In order to shift the content of a cell to the right and to create an empty cell, select the cell and click the **Insert Cells Right** icon. For example, suppose that the data in the fifth row of the Marks worksheet has to be shifted from C5-F5 to D5-G5. To do so, select the cell C5 and click the icon. The output screen is shown in the figure below.

	A	B	C	D	E	F	G	H	I
Student Mark	Details of 9 <sup>th</sup> Standard								
Reg No	Name	English	Tamil	Maths	Science	Total	Average		
1000	Kumar A	87	85	74	86	332	83		
1001	Aravindan J	63	86	62	94	305	76.25		
1002	Govindan S		63	76	73	75	287		
1003	Velmurugan	75	72	63	85	295	73.75		
1004	Thamizharasi	75	46	52	64	237	59.25		

**Fig 6.21 Worksheet with cell shift to right**

- In order to insert an empty row in a worksheet, select the row where you want to insert the new row and click the **Insert Rows** icon. For example, if you want to insert a new row between rows 4 and 5 in the Marks worksheet select the row 5 and click the icon. The output screen is shown below.

	A	B	C	D	E	F	G	H	I
1	Student Mark	Details of 9 <sup>th</sup> Standard							
2	Reg No	Name	English	Tamil	Maths	Science	Total	Average	
3	1000	Kumar A.	87	85	74	86	332	83	
4	1001	Aravindan J	63	86	62	94	305	76.25	
5									
6	1002	Govindan S	63	76	73	75	287	71.75	
7	1003	Velmurugan	75	72	63	85	295	73.75	
8	1004	Thamizharasi	75	46	52	64	237	59.25	
9									

**Fig 6.22 Worksheet with row inserted**



In order to insert a column, select the column where you want to insert and click the **Insert Columns** icon. For example, to insert a new column between E and F in the Marks worksheet, select the column F and click the icon. The output screen is shown in figure 6.23.

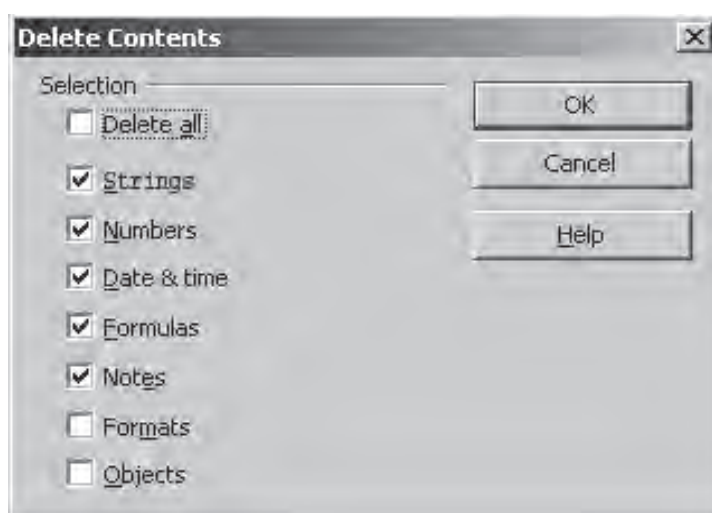
	A	B	C	D	E	F	G	H	I
1	Student Mark	Details of 9 <sup>th</sup> Standard							
2	Reg No	Name	English	Tamil	Maths		Science	Total	Average
3	1000	Kumar A.	87	85	74		86	332	83
4	1001	Aravindan J	63	86	62		94	305	76.25
5									
6	1002	Govindan S	63	76	73		75	287	71.75
7	1003	Velmurugan	75	72	63		85	295	73.75
8	1004	Thamizharasi	75	46	52		64	237	59.25
9									

**Fig 6.23 Worksheet with column inserted**

## 6.12 Deleting Cells, Rows and Columns

The procedure for deleting a cell (s), row or a column is the same.

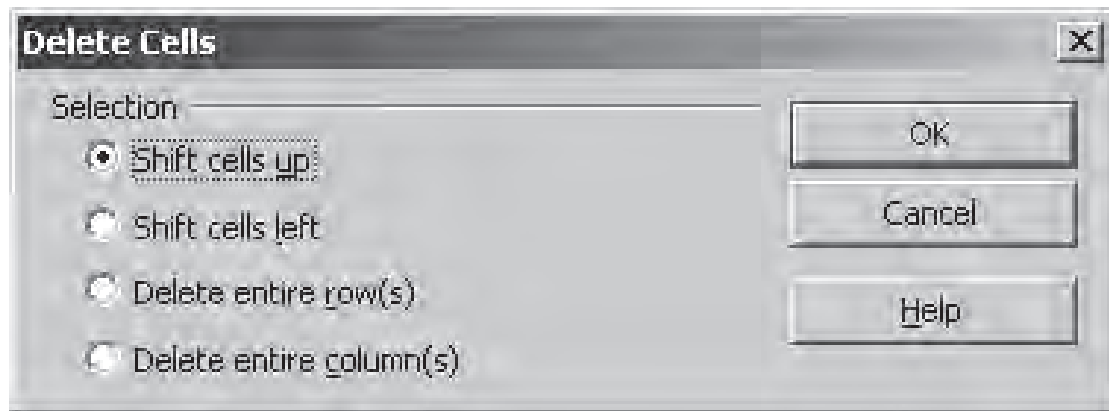
**Delete Contents** specifies the contents to be deleted from a cell or cell range. Before deleting, you must first select the cell or range. Contents are only deleted from the selected cells or active cell. Similarly, if several sheets are selected, only the active one will be affected. To access this command choose **Edit → Delete Contents**.



**Fig 6.24 Deleting contents**



Click on the **Delete all** check box and click **OK**. To delete all contents from the elected cell range, select **Delete Cells** from the same menu. A window appears as shown below. Select an option to specify how the sheets are displayed after deleting cells. **Shift cells up** fills the space produced by the deleted cells with the cells underneath it. **Shift cells left** fills the resulting space by the cells to the right of the deleted cells.



**Fig 6.25 Deleting cells**

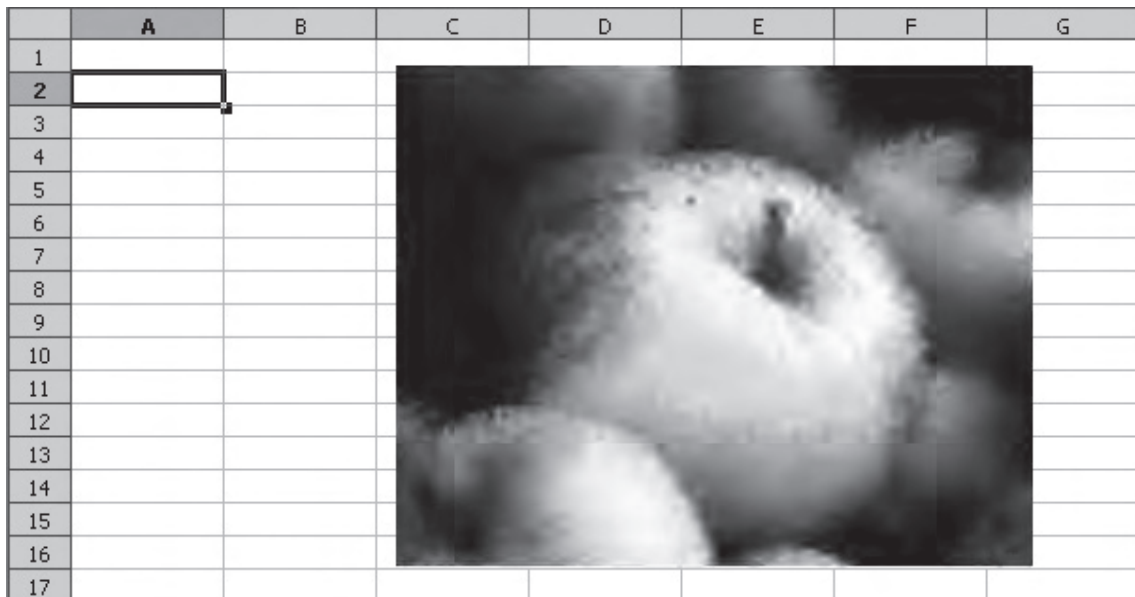
If you want to delete an entire row or column, choose **Edit → Delete Cells**. The Delete Cells dialog box will display **Delete entire Row(s)** or **Delete entire Column(s)**. Clicking **Ok** will delete the row or column without prompting.

### **6.13 Inserting Pictures and Special characters**

In the worksheet, StarOffice Calc also provides for inserting pictures and special characters like á, â. For inserting a picture or a special character in a worksheet follow the procedure given below:

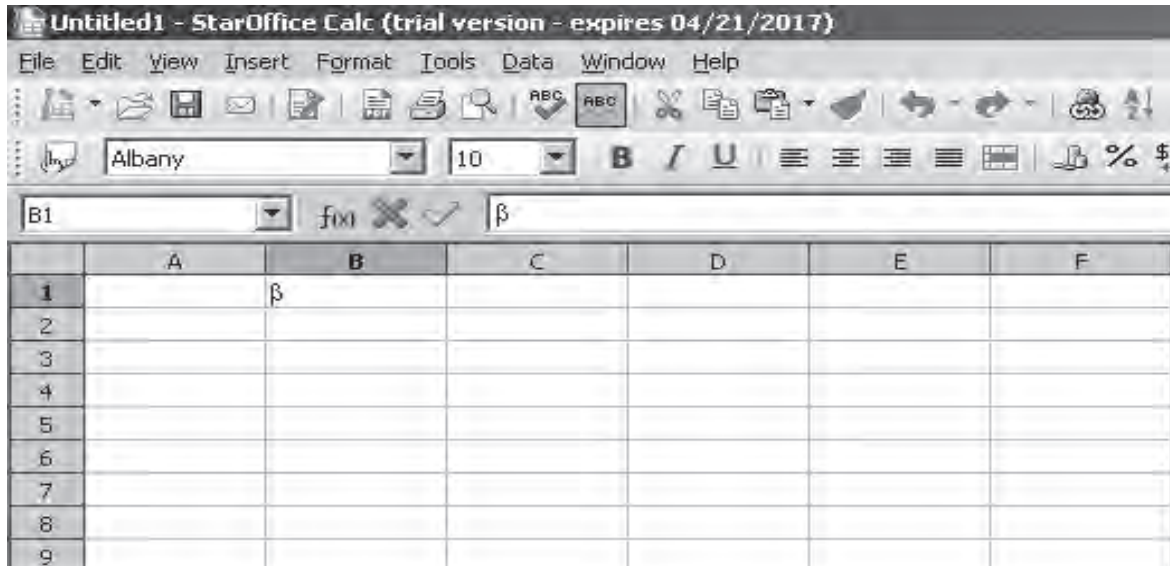
- Place the cell pointer in any cell, say B2.
- Choose **Insert → Picture → From File**. The **Insert Picture** dialog box appears. In the **File name** combo box, you can type the path of the file that contains the picture or you can directly select the desired file from the gallery directory of StarOffice. After selecting the picture file click **Open**. For example, select **Apple** file from the gallery directory and click **Open**.

Now you will find a picture of an apple appearing in your worksheet as shown in figure 6.26.




**Fig 6.26 Worksheet with Apple Picture**

- For inserting special characters, click the **Insert** → **Special characters** from the menu bar and select the desired special characters from the **Special Character** dialog box. For example, select <sup>2</sup> from the **Special Character** dialog box and click the **OK** button. The resultant output is shown in figure 6.27.



**Fig 6.27 Worksheet with Special character**

## 6.14 Drawing in a Spreadsheet

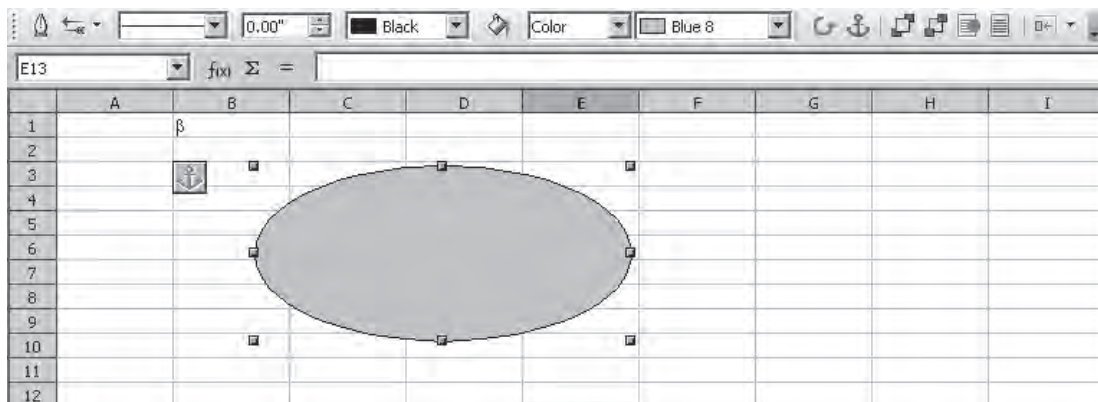
StarOffice Calc provides the facilities for drawing lines, circles, ellipse, square, rectangle, etc. within a worksheet. For this purpose, click the **Show Draw Functions icon** in the Standard toolbar. The **Draw Functions bar** appears as  shown below:



**Fig 6.28 Draw Functions toolbar**

You can select any tool from this toolbar according to your requirement, following the procedure given below:

- Click on a tool. For example, to draw an ellipse in the worksheet, click the **Ellipse** tool. Keep the pointer (cursor) in the worksheet area at the desired location and drag it till you get the desired shape and size. The resultant screen is shown in the following figure.



**Fig 6.29 Worksheet with Ellipse drawn**

## 6.15 Inserting Objects

StarOffice Calc provides tools for inserting objects like charts, images from image editor, formula, etc. in a worksheet. For this purpose, click on the **Insert Object** from **View** → **Toolbar**. A floating toolbar appears with the following icons in the order listed below:

1. Insert Chart Icon
2. Insert Formula Icon
3. Insert Floating Frame Icon
4. Insert Movie and Sound Icon
5. Insert OLE Object Icon
6. Insert Applet Icon



It is to be noted that some of the icons are used in very advanced applications that are beyond the scope of this book.

- Insert Chart Icon**



This icon is used for presenting the data in the worksheet in form of charts of different kinds such as Bar Chart, Pie Chart, Lines, XY plot, etc. More details on charting are given in a later section.

- **Insert Formula Icon**

This icon is used for inserting a formula in the worksheet for performing calculations.

- **Insert Floating Frame Icon**

This icon provides to generate a scrolling screen within a worksheet.

- **Insert Movie and Sound Icon**

This icon is used to insert sound or video files into the current worksheet.

- **Insert OLE Object Icon**

This icon is used to insert objects from other application into a worksheet.

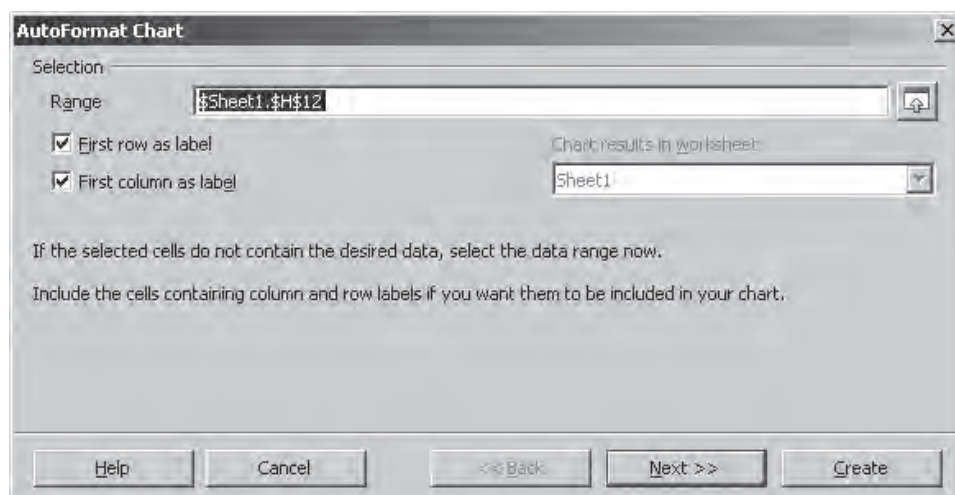
- **Insert Applet Icon**

This icon is used to import Applets written in Java programming language into the worksheet.

## 6.16 Working With Charts

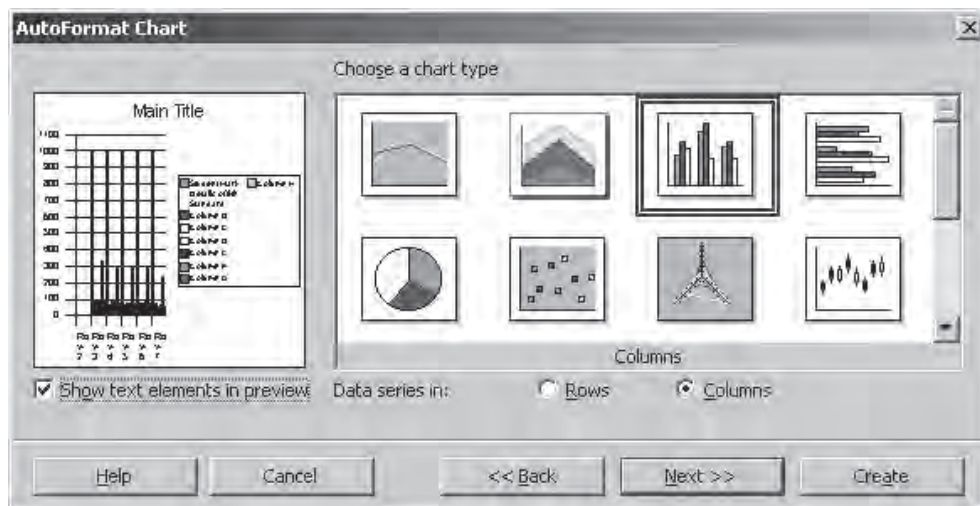
One of the most popular features of StarOffice Calc software is the ability to generate charts based on numeric data. The purpose of chart is to visually present the data for easy understanding. To draw a chart, follow the procedure given below:

- Select the data you want to chart.
- Click on **Insert** → **Chart** or click on the **Insert Chart** icon as discussed earlier.
- The cursor becomes a + sign with a small picture of the graph. Place this cursor where you want to insert the chart and click. The **Autoformat Chart** window appears as shown in figure 6.30.



**Fig 6.30 Autoformat Chart window**

- It prompts you to enter the area in the worksheet to be charted. Enter the range as **C3 : F7** and click **Next**.

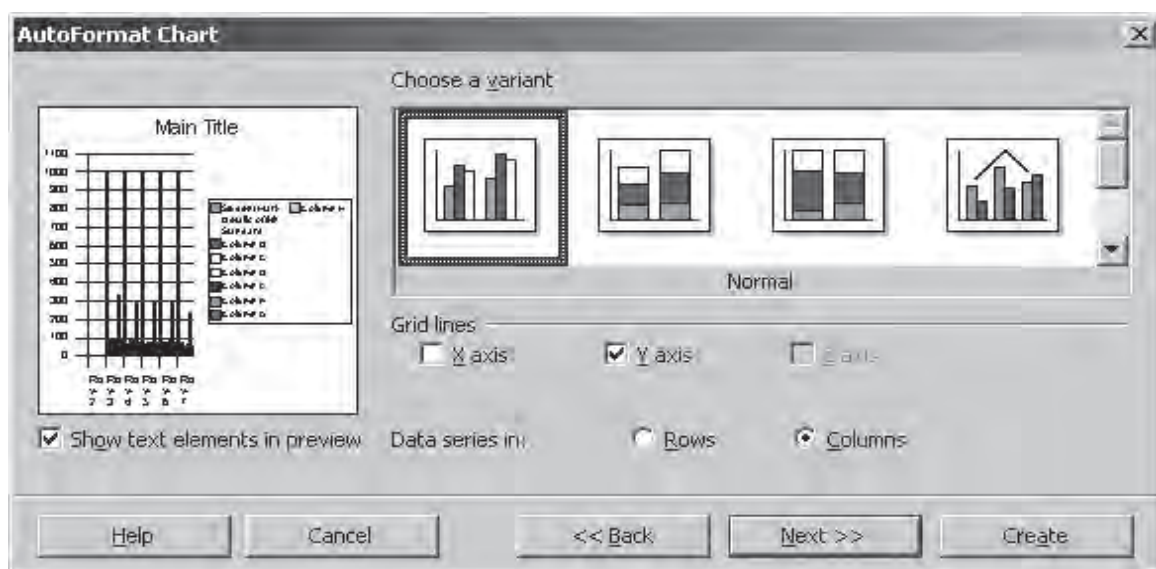


**Fig 6.31 Types of chart**

- The next window, which appears as shown in figure 6.31 displays the different types of charts that can be created along with a preview of each. Select the type of chart in which you want to present the data. The preview Window shows the chart. Click on **Next**.

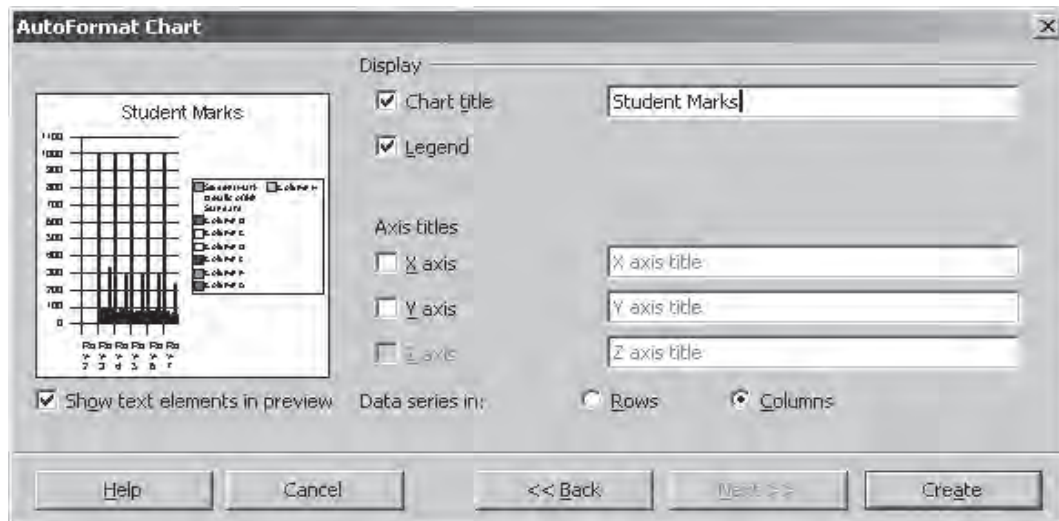
**Note:** Click the **Show text elements in preview** check box if it is not selected.

- For each type of the chart (say Pie, Bar, Column, etc.), there are different formats available. These formats are displayed in the next window (figure 6.32). a preview of the each format is also displayed. Here, select the desired format. Click on **Next**.



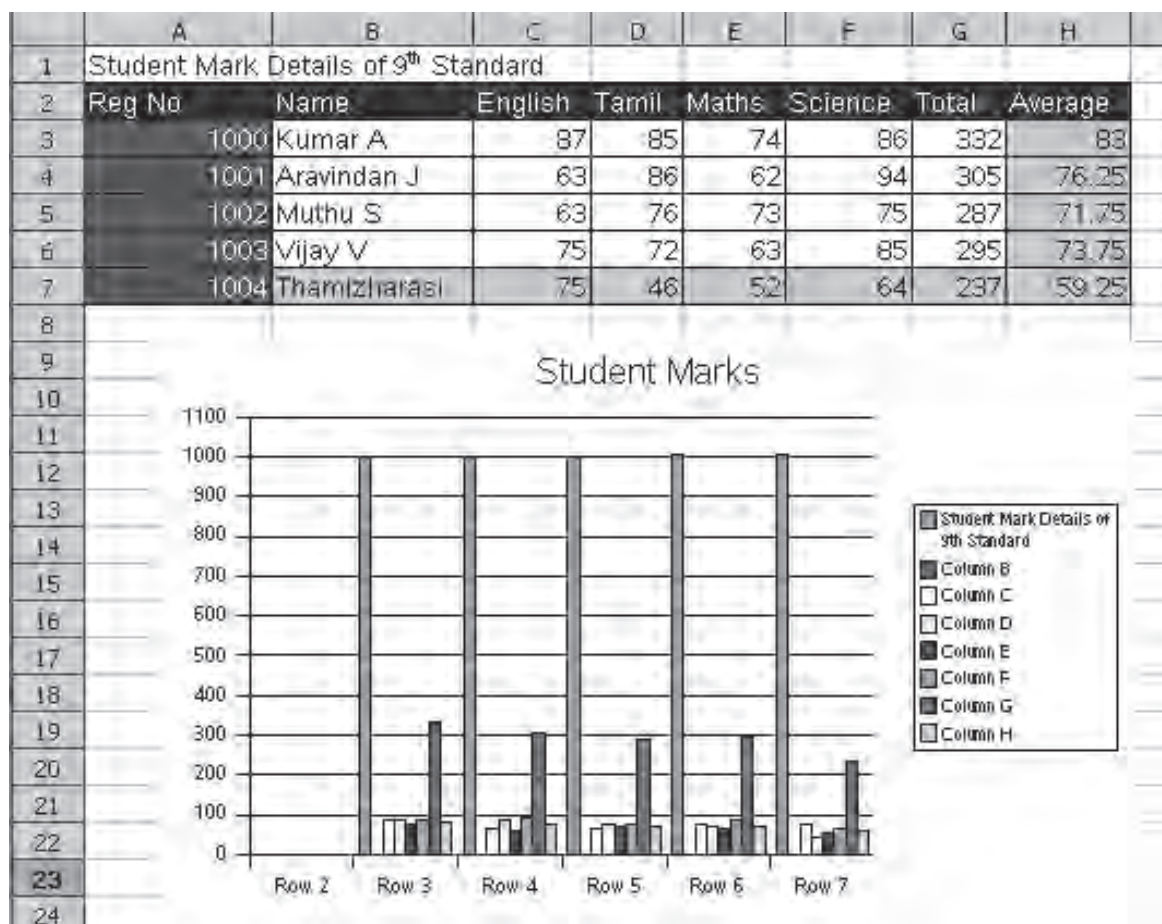
**Fig 6.32 The different format of a chart type**

- In the next window (figure 6.33), you have provisions to give a Title for the chart, Titles for X and Y axes and legends.



**Fig 6.33 Adding titles and legends**

- Click on **Create**. Now, your worksheet will look as shown below.



**Fig 6.34 The Worksheet with the chart**

---

## Learn by Solving

Create a line chart to show the variations of mark secured by each student in different subjects.

---

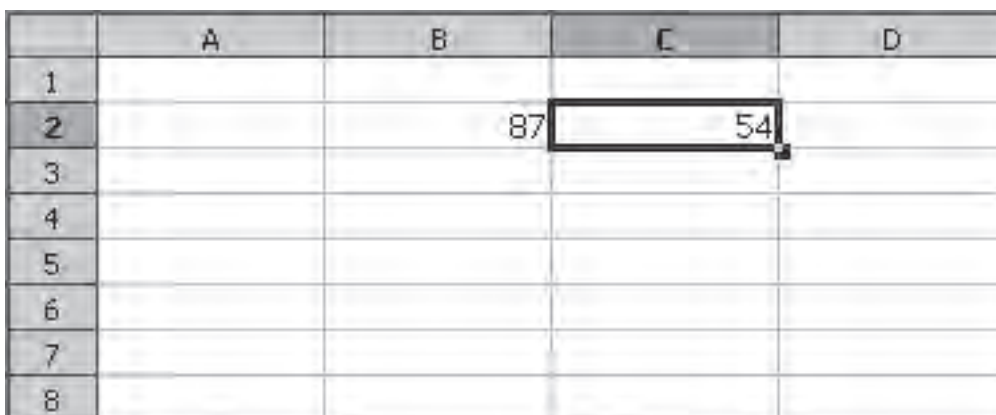
### 6.17 Working with Multiple Sheets

In StarOffice Calc, a spreadsheet contains multiple sheets. Each sheet has its own name and a list of sheets appears as tabs at the bottom of the window. To select a different sheet, click the tab with the sheet's name. The tab of the selected sheet appears in white.

Each sheet of a spreadsheet can be used entirely independently of the other sheets. You can also make them dependent on each other by referring to the data in another sheet or using the data from another sheet in calculations.

For example, you can enter data in Sheet 1 and Sheet 2 and can do the calculations in Sheet 2. You can calculate the sum of the numbers in the cell A1 of Sheet 1 and A1 of Sheet 2 and store the result in A3 of Sheet 2. To do this, type the 3-D formula as = **SUM (Sheet 1.A1; Sheet 2.A1)**

Here is another example. Let the cells B4 and C4 in Sheet 1 have numbers 87 and 54 respectively (figure 6.35). Let the cells B4 and C4 in Sheet 2 have numbers 45 and 34 respectively.



	A	B	C	D
1				
2				
3				
4		87	54	
5				
6				
7				
8				

**Fig 6.35 Worksheet with Values in Sheet 1**

Now, let us add cell values B4 and C4 in the two sheets and store the total in cell D5 of Sheet 2. For this purpose, enter the 3D formula = **SUM (Sheet 1. B4: C4; Sheet 2. B4:C4)**. In this formula, the: (colon) separating the cells of the same sheet adds up the values of B4 and C4 in that sheet and the; (Semi – colon) adds up the sums of B4 and C4 of Sheets 1 and 2.

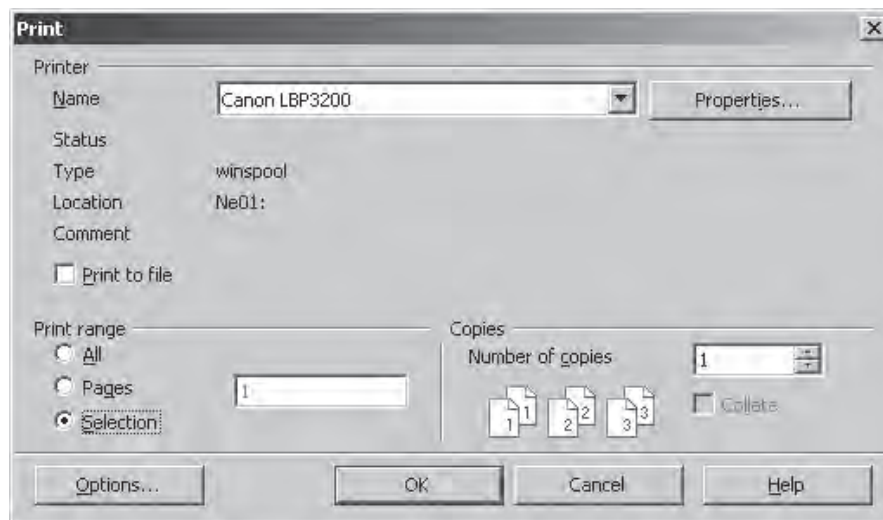
D5					
	A	B	C	D	E
1					
2					
3					
4		45	34		
5				220	
6					
7					

**Fig 6.36 Worksheet with Values and Results in Sheet 2**

## 6.18 Printing Worksheets

If you click the **Print** icon on the function bar (Standard toolbar), all the data in all the sheets of your document will be printed. You can also print a part of the worksheet. To do so, select the range to be printed and click on **Format** → **Print Ranges** → **Edit** and select the print range. Now, if you click on the **Print** icon, only the selected range will be printed. To remove the print range setting click on **Format** → **Print Ranges** → **Delete**.

The **Print** option under the **File** menu can be used to print the worksheet. The Print dialog box appears as shown in figure 6.37.



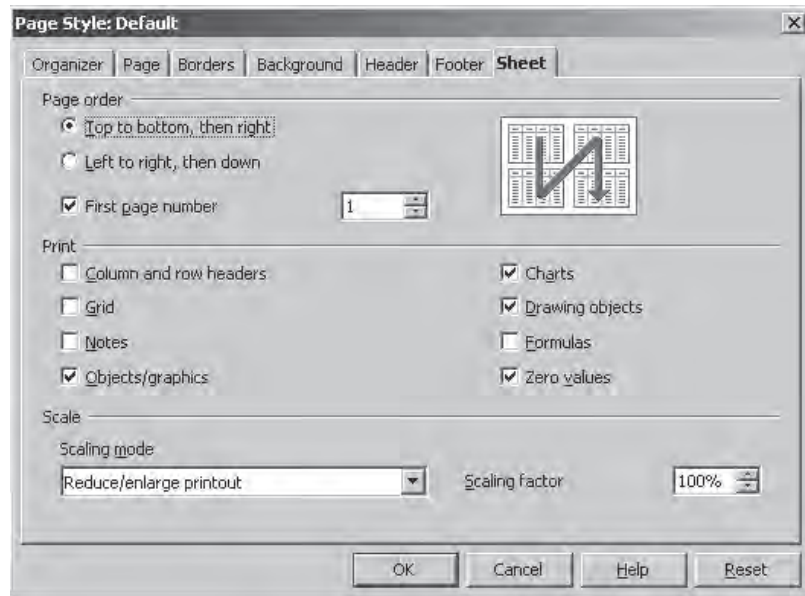
**Fig 6.37 Print dialog box**

In the **Print** dialog box, under **Print range** choose the option **All** to print all the sheets in the document. Select the option **Pages** to specify the pages, which are to be printed. The **Selection** option allows you to print only the selected area in a worksheet.



The **Page View** option on the **File** menu can be used to preview a worksheet before printing.

The above procedure will print the worksheet without grids. In order to print with the grids select the **Page** option from the **Format** menu. Click on the **Sheet** tab in the dialog box, which appears. Click the Grid check box to print the worksheet with gridlines.



**Fig 6.38 Page Styles Dialog Box**

## 6.19 Database Functions in StarOffice Calc

Spreadsheets in StarOffice can be used to manage large amounts of data. You can sort this data, search for specific information, group information based on some criteria, calculate totals, and much more.

The next chapter of this book gives more details about Databases.

### Summary

- A spreadsheet is a tool for working with numbers.
- A spreadsheet program is used to calculate and analyze sets of numbers.
  - A spreadsheet is divided into columns and rows. The intersection of row and a column is called a cell. A cell is referred to by its address – created by combining its column and row headings.
  - Cells can contain values, labels, or formulae.
  - A data file created with spreadsheet is called a worksheet.

## ■ Creating a Worksheet

- The process of creating a worksheet can involve organizing the data, entering data, creating formulae, editing, formatting, adding charts, analyzing data and printing the worksheet.
- A formula lets you create a value in one cell that is calculated based on the values in other cells.
- Formulae follow the order of mathematical operations: exponentiation, negation, multiplication and division, addition and subtraction.
- Functions are built –in formulae. StarOffice Calc offers a wide variety of functions.
- Creating a Worksheet References to cells can be relative or absolute.
- Data in the worksheet can be formatted in bold, italics and underlined. The alignment, font, size and color of the contents of the cells can also be changed. Numbers can be formatted as currency, percents and displayed with or without decimals.
- You can add pictures, special characters and your own drawings to the worksheet.
- You can insert movies, sounds, charts, floating frame, applets and OLE objects in worksheet.
- Charts are used to make data easier to understand.
- You can define which range of cells on a spreadsheet to print.
- Spreadsheets have database management capabilities including sorting, selecting and printing out reports.

## EXERCISES

### I. Fill in the blanks

1. You can use a \_\_\_\_\_ to calculate and analyze sets of numbers.
2. Graphic representations of numbers are known as \_\_\_\_\_.
3. A(n) \_\_\_\_\_ refers to the file you create with spreadsheet software.
4. Non – numerical entries are called \_\_\_\_\_.

5. The \_\_\_\_\_ identifies the active cell.
6. The \_\_\_\_\_ is the sequence of characters used in a formula.
7. You can automatically perform specialized calculations using \_\_\_\_\_.
8. The two types of cell addressing are \_\_\_\_\_ and \_\_\_\_\_ addressing.
9. The \_\_\_\_\_ command is used to generate a series.
10. The combined cell column and row headings are known as the cell's \_\_\_\_\_.
11. \_\_\_\_\_ is used to insert objects from other application into a worksheet.
12. The \_\_\_\_\_ option on the file menu can be used to preview a worksheet before printing.

## **II. Answer the following**

1. Define briefly a spreadsheet and describe its basic structure.
2. List and describe the other features available with spreadsheet software in addition to its ability to calculate numbers.
3. When entering a formula in a cell, does it matter in which order you enter the values and operators? Explain your answer.
4. Can we change the data present in a cell? If so, how?
5. How can you generate a series of values? Explain with an example.
6. What are functions? How can you use them in your worksheet? Explain with an example.
7. What must you include in a formula, to ensure that the formula will operate on a specific value, no matter where the formula might be moved or copied?
8. What is date arithmetic? Explain.
9. What spreadsheet feature allows you to represent data visually as a data – analysis tool?
10. Explain working with multiple sheets and printing worksheets?



# CHAPTER 7

## DATABASE

### 7.1 Introduction

We use computers for various applications. The applications may be scientific or data processing. Applications such as payroll, examination processing, banking, railway ticket reservation system, airlines reservation system, budget preparation, inventory control, etc., fall under the category of data processing. In this chapter, we will study about data processing in general and about working with a specific database package, namely, 'StarOffice Base'.

### 7.2 Data and Information

#### 7.2.1 Data

The term data comes from the word **datum**, which means a fact. The **data** is as fact about people, places or some entities. In computers, data is simply the value assigned to a variable. The term variable refers to the name of a memory location that can contain only one data at any point of time. For example, consider the following statements:

Vijay is 16 years old.

Vijay is in the 12th Std.

Vijay got 80% marks in Mathematics.

Let 'name', 'age', 'class', 'marks' and 'subject' be some defined variables. Now, let us assign a value to each of these variables from the above statements.

Name = Vijay

Class = 12

Age = 16

Marks = 80

Subject = Mathematics

In the above example the values assigned to the five different variables are called data.

Consider another example give below:

Coimbatore is 500 kilometres away from Chennai.

Coimbatore is 1500 feet above sea level.

These are two facts about a place called Coimbatore. In this example name of the city, i.e. 'Coimbatore', the distance in km, '500', the name of another place 'Chennai' and the height of the city in feet, i.e., '1500' are data for the variables 'city', 'distance', 'place' and 'height' respectively.

The data can be of different types. The **Data types** can be Character, Number, or Boolean. The numeric data type can be integer, float, double precision, date, time, etc. The programming languages and packages provide their own **built-in** (system defined) **primitive data types** such as **integer** and **float**. But for many applications, such as, 'supplier numbers', 'part names', 'city names', 'colours', etc., the system defined primitive data types alone are not adequate. For such purposes, Database Systems provide a facility by which users can define their own more sophisticated data types as referred above, which are known as **user-defined data types**.

### 7.2.2 Information

We will now see what is information with respect to computers. Information is defined as a set of processed data that convey the relationship between data considered. Processing means to do some operations or computations on the data of different variables to relate them so that these data, when related, convey some meaning. Thus, information is as group of related data conveying some meaning. In the examples above, when the data of the variables 'name' and 'age' are related, we get the following information:

Vijay is 16 years old.

In the same example, when the data of the variables 'name' and 'class' are related we get another information as

Vijay is in the 12th standard.

Further, when we relate the data of the variables, 'name', 'marks', and 'subject', we get more information that 'Vijay' got 80% marks in mathematics.

Similarly, when the data 'Coimbatore', 'Chennai' and '500' are related, it gives information that Coimbatore is 500 kms away from Chennai.

## 7.3 Data Processing

Let us consider the example of a mark list that a class teacher produces for his/her class. It contains the register numbers, student names, marks obtained in each subject by the students in the class and Pass/Fail status. This information is entered as data against the variable name 'REG.NO.', 'NAME', 'ENG', 'MATHS', 'PHY', 'TAMIL', 'CHEM', 'COMP' and 'RESULT' respectively in a table as shown in table 7.1.

**Table 7.1 Marklist Table**

<b>Mark list of XI STD – A Section</b>								
Reg. No.	Name	Tamil	Eng.	Maths	Phy.	Chem	Comp.	Pass/ Fail
1001	Anbu A	80	85	99	95	82	88	P
1002	Aruna S	85	90	90	92	90	98	P
1003	Balu S	35	56	95	75	70	80	F
1004	Xavier D.	80	99	95	96	97	99	P
1005	Banu M	75	80	56	50	60	70	P
1006	Chandran M	40	32	58	45	26	70	F
1007	Abdul Kader M	80	80	90	88	80	95	P

To process this mark list of a class, the class teacher first collects the details such as register number and name of the students; their marks in different subjects from the concerned teachers of that class and the passing regulations from the school authorities. Then he/she prepares a consolidated mark list, as in table 7.1. He / she first checks the mark list for any possible mistake that would have occurred while entering the data. Then the class teacher requests the concerned subject teachers to verify and validate the data entered in the mark list against their subject. After confirming that the marks entered in the mark list are correct, the class teacher applies the passing norms and determines the Pass/ Fail status of each student. Then he / she enters the result in the mark list against each student. These entries are verified and finally put up to the Headmaster for approval. After the Headmaster approves the results, a separate list is prepared containing the register number of the students along with their Pass/Fail status as shown in table 7.2 and the results are displayed on the notice board.

**Table 7.2 Result Table**

<b>Result of XI standard</b>		
<b>Reg. No.</b>	<b>Name of the Student</b>	<b>Result</b>
1001	Anbu A	Pass
1002	Aruna S	Pass
1003	Balu S	Fail
1004	Xavier D.	Pass
1005	Banu M	Pass
1006	Chandran M	Fail
1007	Abdul Kader M.	Pass

What we discussed so far relates to the result processing of a class of students. This process is otherwise known as Data Processing. Data Processing may relate to any field of application.

As discussed in the above example, data processing involves Data Collection, Verification and Validation of data, and Report Generation. Until digital computers came into use, data processing had been carried out manually. Even now, many organizations do data processing manually.

In the above case we see that throughout the process, right from the preparation of the mark lists by the individual teachers, processing the mark lists, etc., till the results are declared, the work has been carried out by long-hand effort (or otherwise known as manual). Hence, this type of processing is called **Manual data processing**.

Instead of the manual approach, if the school uses a computer to,

- Prepare the mark lists,
- Store the data,
- Process the stored data for analyzing the results and
- Print the results in a desired format,

Then we call this type of data processing as **Computerised data processing**.

### **7.3.1 Disadvantages of Manual Data Processing**

The manual data processing approach has the following disadvantages:

1. The efficiency and correctness of data processing are limited to the capabilities of the individual who is processing the data.
2. Manual methods take more time.
3. In general human beings are liable to make computational and parallax errors.
4. Manual data processing involves use of papers at each stage. Preservation and maintenance of large volume of paper records becomes difficult and unmanageable.
5. Implementation of corrections, changes and modifications are tedious in manual data processing.

### **7.3.2 Advantages of Computerised Data Processing**

The following are some of the advantages of the computerized data processing system:

1. Once we collect and enter the data into a computer system, We can perform other operations with less manual labour. So, manpower is considerable saved.
2. Though it takes some time to develop, test and put the required computer programs to use, the processing speed is fast, reducing the processing time, in certain cases, from man-years and man-months to minutes and seconds.

3. The chances of errors are less in computerized data processing.
4. We can store large amount of the data and information in the computer storage medium, which is compact. Hence, we need not store bundles of paper records, thus saving space.
5. Today computer networks are so common that we can share data and resources from one computer system to the other at a very fast speed and with very little effort, as in the case of railway and airline reservation systems.
6. It is easy to edit the data including correction, changes and modifications.
7. Computerized database is highly effective for searching, sorting and merging files and for other data manipulation activities.

#### 7.4 Database

A database is a repository of collections of related data or facts. It arranges them in a specific structure. For example, data such as “1001, Arul A., B.E. (CSE), Lecturer, Comp. Sc. & Engg. 27/01/97, 10000” in table 7.3 are all relevant to a staff database of an educational institution.

Similar to the Institution staff database, there can be different databases, like CSE Department Database, as in table 7.4, for different applications. For example, you may keep a list of addresses of your relatives and friends written down in an address book. This is a database of addresses.

Data in a database is most commonly viewed in one or more two-dimensional tables, each consisting of columns and rows. The entire collection or related data in one table is referred to as a **File** or a **Table**. Each row in a table represents a **Record**, which is a set of data for each database entry. Each table column represents a **Field**, which groups each piece or item of data among the records into specific categories or types of data.

**Table 7.3 Staff Table**

Institution Staff Database						
Emp. No.	Staff Name	Qualification	Designation	Department	Date of Joining	Monthly Salary (Rs.)
5001	Arul A.	B.E (CSE)	Lecturer	Comp. Sc. & Engg	23/01/96	10000
5002	Aruna B.	M.E. (CSE)	Sr. Lecturer	-do-	18/06/98	15500
5003	Ezhil M.	M.Tech.	Sr. Lecturer	Elec.Comm. Engg.	10/05/99	15000

5004	Thiagu N	M.B.A	Lecturer	Dept of Management	19/03/96	10000
5005	Laura I.	M.E. (CSE)	Sr. Lecturer	Comp.Sc. & Engg	01/07/99	15000

**Table 7.4 Staff Table**

<b>CSE Department</b>						
Emp. No.	Staff Name	Qualification	Designation	Department	Date of Joining	Monthly Salary (Rs.)
5001	Arul A.	B.E (CSE)	Lecturer	Comp. Sc. & Engg.	27/01/97	10000
5002	Aruna B.	M.E. (CSE)	Sr. Lecturer	-do-	18/06/98	15500
5005	Laura I.	M.E. (CSE)	Sr. Lecturer	Comp. Sc. & Engg.	01/07/99	15000

#### **7.4.1 Manipulation of a Database**

We may manipulate the database in one or more of the following ways:

- Searching
- Sorting
- Merging
- Performing Calculations on data
- Filtering
- Editing the database
- Report Generation

##### **❖ Searching**

Searching is a process to select a desired specific data from a database. For instance, you want to select the student ranking first in a class with respect to the total marks or in individual subjects or you want to see the performance of a particular student. Searching is done using database commands on the relevant fields.

##### **❖ Sorting**

Sorting is the process of arranging the data in a table in some order. In the above example, we may arrange the list of staff – members in an alphabetical order or according to the seniority in position or date of joining or in any other order we like.

In the case of the student database, you may sort the passed students and failed students separately or you may sort the students in the ascending or descending order of their total marks, and so on.

#### ❖ **Merging**

Merging is a process of joining data from two or more tables of the same or different databases. For example, in a student database, you may have mark lists for different classes. You want to prepare one table that shows all the first ranking students of each class. For this purpose, you can independently search in each table to select the first rank student from each class and merge all the names together in a single file/table.

#### ❖ **Performing Calculations on data**

You may do any kind of arithmetic calculations on the data stored in the database. For example, to obtain the total marks of all the subjects of a student, you may add the marks in the concerned fields of the database and store them in a separate field.

#### ❖ **Filtering**

There are times when viewing the entire table is unwieldy. Using a Filter is a way of limiting the information that appears on screen.

Filters are a feature for displaying and browsing a selected list or subset of records from a table. The visible records satisfy the condition that the user sets. Those that do not satisfy the condition are hidden.

#### ❖ **Editing the Database**

Editing is a process of performing corrections on the existing data, deleting the existing data, field(s), or record(s), adding new data, field(s), record(s) or changing the format of the database, and so on.

#### ❖ **Report Generation**

You may generate any desired report, from the data of the database. For example, in the case of student database, you may generate a report of all the students who have scored marks less than the minimum marks required for a pass, and another report that gives the list of passed students, and so on.

### **7.4.2 Database Types**

Based on the conceptual structures, the databases can be classified as follows:

1. Flat – File database
2. Relational database

3. Hierarchical database
4. Network database
5. Object-Oriented database

#### ❖ **Flat – File Database**

A database file that consists of a single data table is a **Flat-file database**. Flat-file database can be quite useful for certain singleuser or small-group situations, especially for maintaining lists such as address lists or inventories. Data that is stored, managed, and manipulated in spreadsheet is another example of a flat-file database.

Despite their simplicity, flat-file databases do have a significant limitation over databases consisting of multiple tables. They do not allow for more complex requests. This is fine if you merely need to locate a part or verify an address. However, very often you need to process more extensive information from multiple data tables which is not possible in flat-file databases.

#### ❖ **Relational database**

A relational structure represents a database made up of a set of related tables. In a relational database, one or more common fields existing in two or more tables create a relationship between these tables. The common field or fields are called the **Keys**.

A **primary key** is a key that uniquely identifies a record in a database table. In relational databases, a primary key can consist of one or more fields. For instance, in table 7.3, Emp.No. is a primary key, since it uniquely identifies an employee record.

The relational database structure is the most prevalent database in today's business organizations.

#### ❖ **Hierarchical database**

The hierarchical database structures were primarily used on Main Frame computers. In hierarchical databases, records are organized in a tree like structure by type. The relationship between record types is said to be a parent-child relationship, in which any child type relates only to a single parent type.

#### ❖ **Network database**

The Network database is very similar to the hierarchical structure except that any one record type can relate to any number of other record types.

#### ❖ **Object Oriented database**

An Object Oriented database is a newer structure that has been generating a great deal of interest in recent years. It represents very different approach to the



way data is treated by database developers and users. The object oriented structure groups data items and their associated characteristics, attributes, and procedures into complex items called objects. Physically an object can be anything: a product, or event, such as a house, an appliance, an art piece, a customer complaint, or even a purchase. An object is defined by its characteristics, attributes and procedures. An object's characteristics can be text, sound, graphics, and video. Examples of attributes might be colour, size, style, quantity, and price. A procedure refers to the processing or handling that can be associated to the object.

## 7.5 Basis Concepts of Database Management Systems (DBMS)

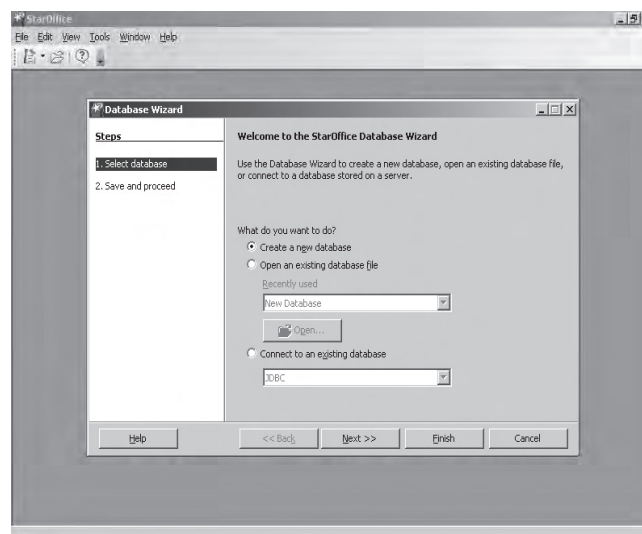
A DBMS is a program, or collection of programs that allows any number of users to access data, modify it (if necessary), and construct simple and complex requests to obtain and work with selected records. The biggest asset of the DBMS is its ability to provide extremely quick access and retrieval from large databases. A DBMS, especially when it is running on powerful hardware, can find any speck of data in an enormous database in minutes – sometimes even seconds or fractions of a second. The data management tasks in a DBMS fall into one of the following three general categories.

1. Entering data into the database.
2. Reordering records in the database.
3. Obtaining subsets of the data.

DBMS also provide the means for multiple for multiple users to access and share data in the same database by way of networked computer systems.

## 7.6 Working with StarOffice Base

To work with StarOffice Base, open StarOffice Base. A window appears as shown below.

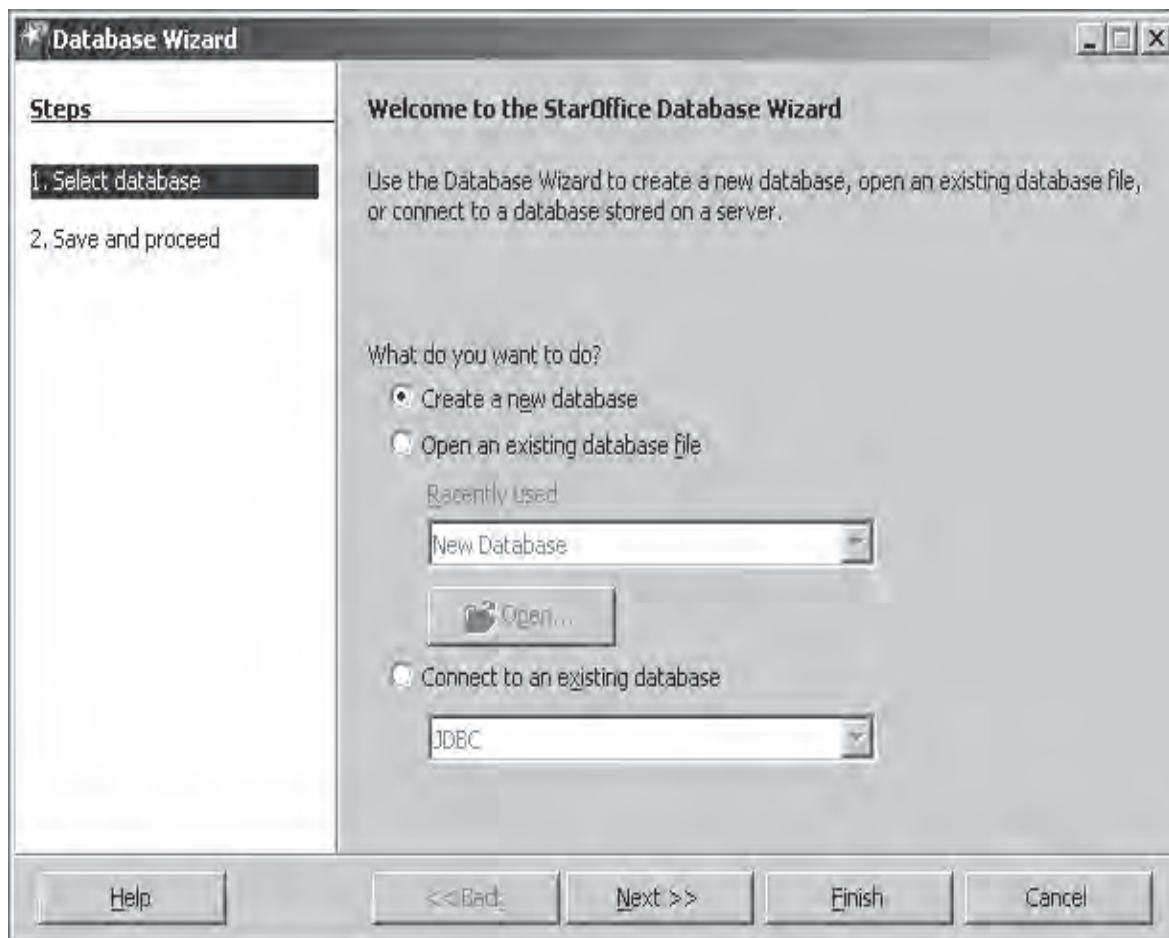


**Fig 7.1 Opening Screen**

**Note :**

- i) The StarOffice Base window on your computer may appear a little different if the StarOffice Base settings are different.

Now, let us see how to create database. To do this, open StarOffice Base or if a StarOffice window is open, click on **File** → **New** → **Database**. The **Database Wizard** dialog box appears on the screen as shown in figure 7.2.

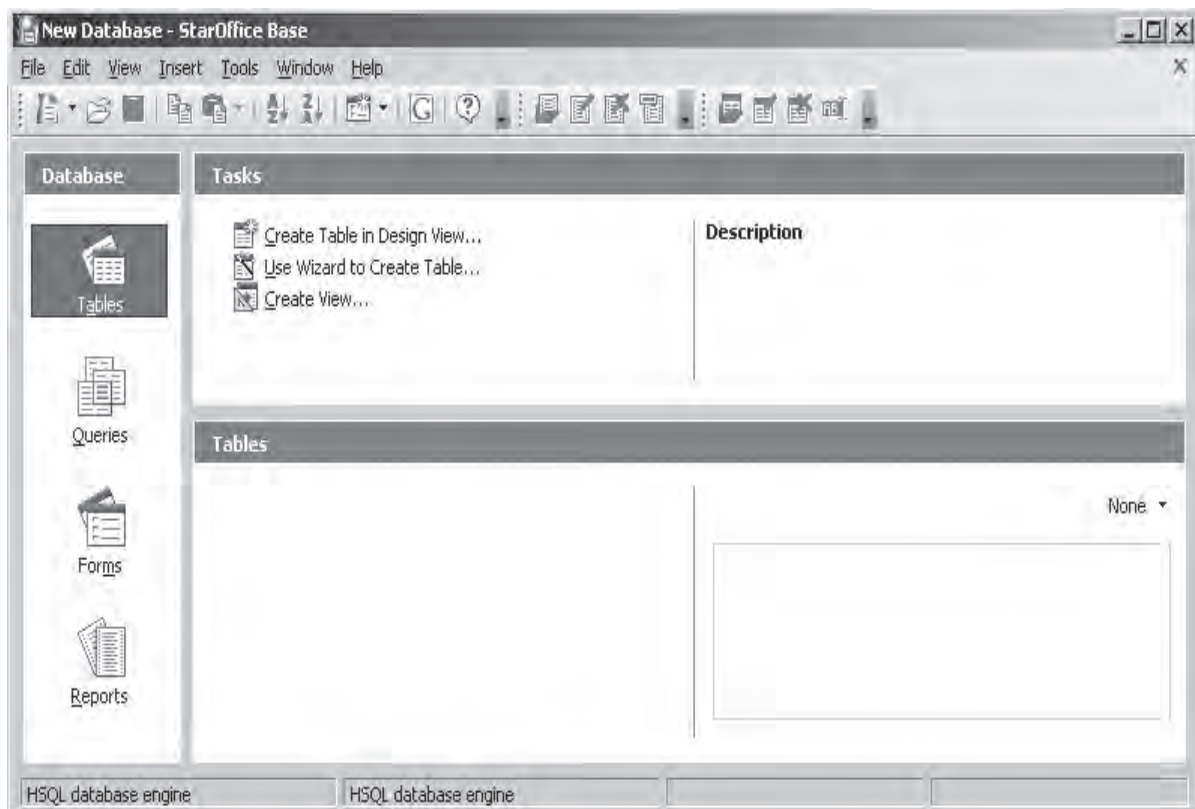


**Fig 7.2 Properties of <Database> Screen**

Select the '**Create a new Database**' option for creating a new database or select the '**Open an existing database file**' option for opening an existing database. Click **Finish**.

The **Save As** dialogue box appears prompting you to save the database created. Type the database file name in the **File name** text box. Click on **Save** to save the database file.

The screen shown in figure 7.3 appears.



**Fig 7.3** The Tables, Queries, Forms and Reports options can be used to work further with the database.

**Note:** that the left pane is the <Database> pane that displays **Tables, Queries, Forms** and **Reports**. And the pane on the right top is **Tasks** pane and the pane on the right bottom changes to **Tables, Queries, Forms** or **Reports** accordingly.

To work further with the database created, you can click on any of these options in the <Database> pane.

---

### Learn by solving

Using the above procedure, create a Database named 'Student'.

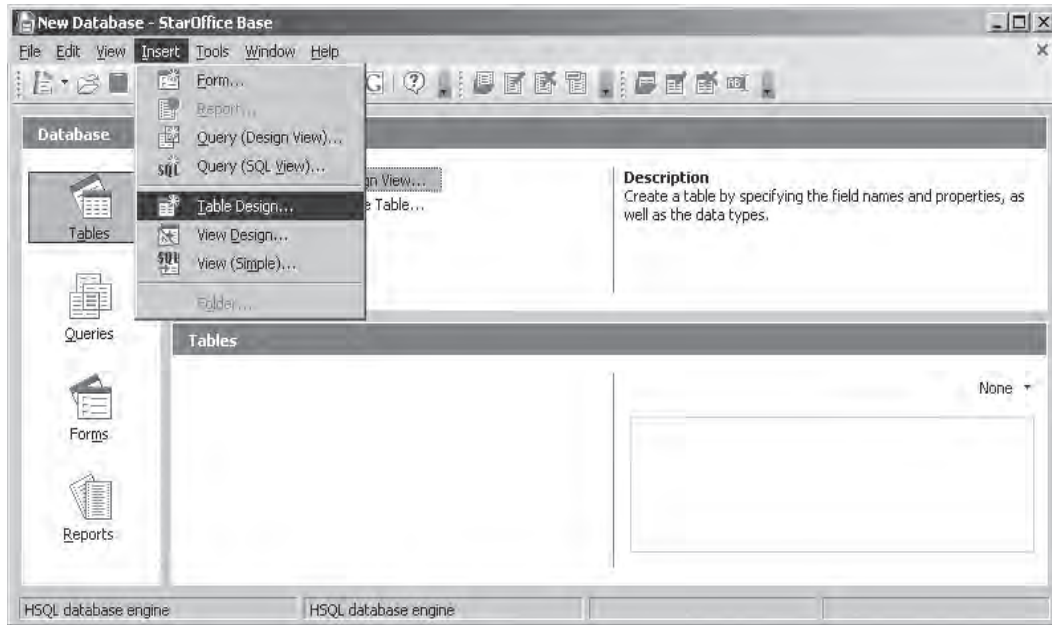
---

#### 7.6.1 Manipulation of the Database Created

As mentioned earlier, a database is a collection of related tables. Once the database is created, the next step would be to create the tables in that database and input data into them. Using these tables, StarOffice Base allows you to design forms, query the database and prepare reports. In the following sections, you will learn how to do all these.

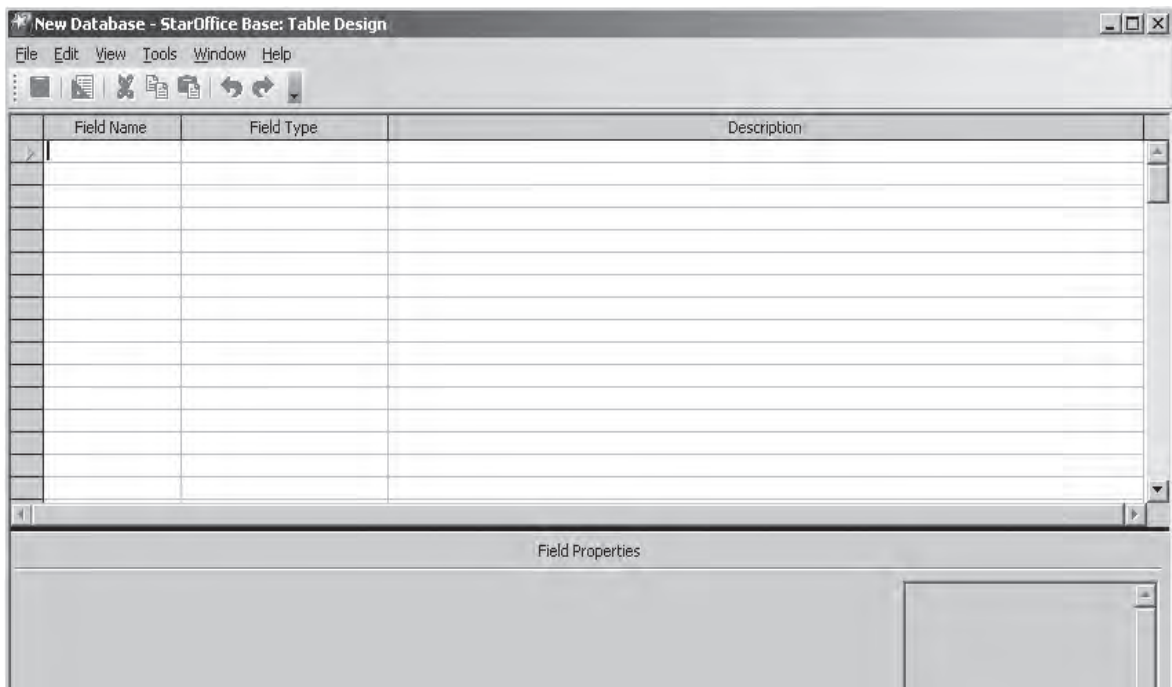
### 7.6.1.1 Creating Tables

To create a new table, click on the **Table** icon in the **<Database>** pane and click on the **'Create Table in Design view'** in the **Tasks** pane or select **Table Design** from Insert menu as shown in the figure 7.4.



**Fig 7.4 Creating a new table**

A Table Design window as shown in figure 7.5 appears.



**Fig 7.5 Table Design window**

The next step would be to design the structure of the table. Designing the table means identifying the different fields that are to be included in the table, assign a field name and type to each of them. You can also give a brief description of the contents of the field. StarOffice Base allows you to use twenty different field types as given below:

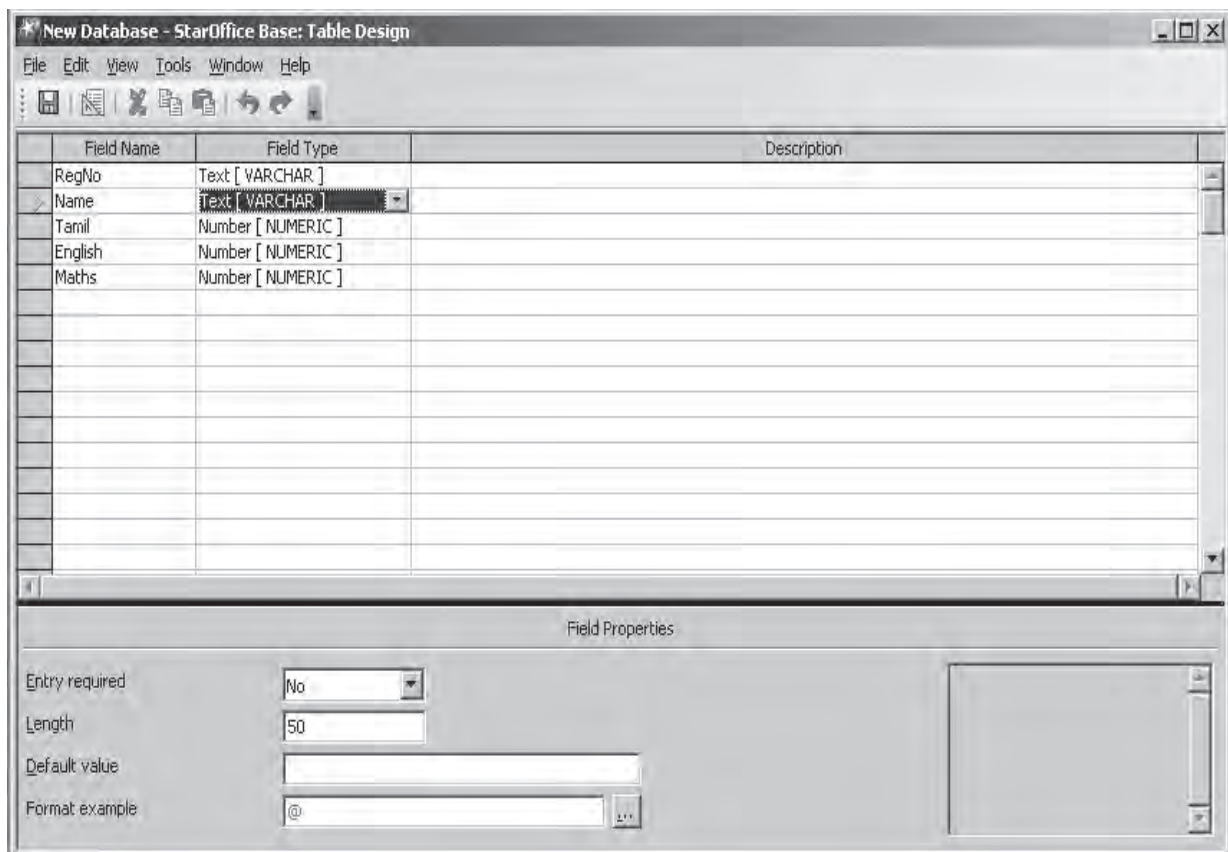
1. Text [VARCHAR]
2. Text [VARCHAR\_IGNORECASE]
3. Text (fix)
4. Number
5. Date / Time
6. Date
7. Time
8. Yes / No
9. Memo
10. Image
11. Decimal
12. Binary Field (Fixed)
13. Binary field
14. Integer
15. Tiny Integer
16. small Integer
17. Big Int
18. Float
19. Real
20. Double

**Integer** data type accepts only whole numbers. **Smallint** accepts small integer values up to a few thousands. **'Single'** and **'Double'** means floating point numbers with single and double precision. **Decimal** values are accurate up to a length of 7 places in the case of single precision and 14 places in the case of double precision.

To start with, the cursor is in the first column of the first field, the **Field Name**. Enter the field name. Press the **Tab** key to move to the next column, the **Field Type**. StarOffice Base displays a drop-down list box with the different field types. Select the type you want by clicking on it.

In the lower part of the window, the **Field Properties** pane is displayed. This pane allows you to further customize the field. It is important to note that different field properties are displayed for different field types.

For example, if the field type is **Text**, the following field properties will be displayed.



**Fig 7.6 Field Properties for a Text field**

The first option, **Entry required** is used to specify if the user should always enter a value for that field. The **Length** option is used to specify the maximum characters that can be entered in that field. The next option, **Default value** is used to specify a default value to be assigned to that field. The last option **Format Example** allows you to specify the number format and alignment of the text or number within the column.

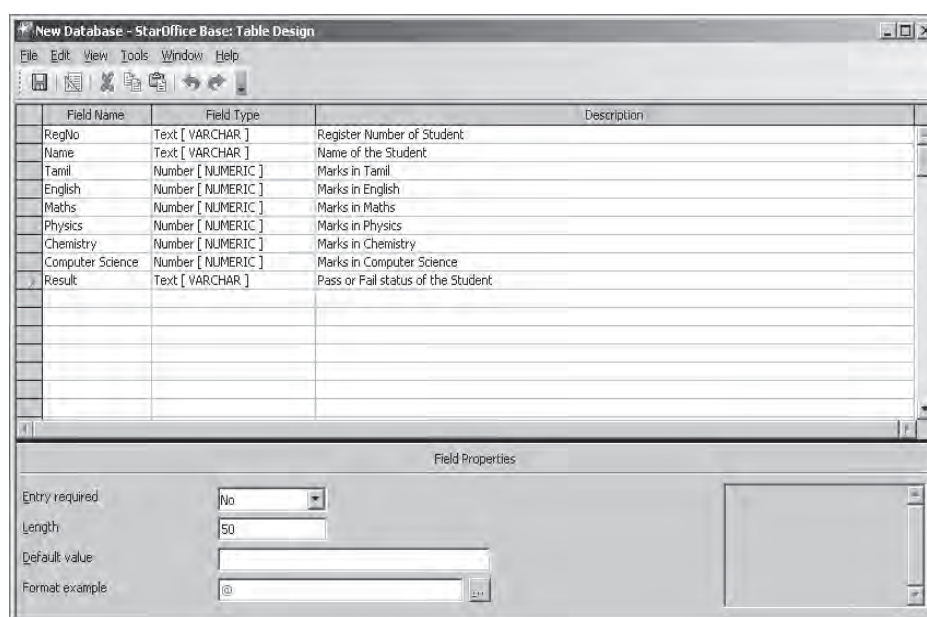
If the field type is **Number**, the **Length** option in **Field Properties** determines the length of this field. **Decimal places** option, specify the number of decimal places permitted in this field. Other options Default value and Format Example are same like **Text**.

After specifying the Field Properties, you can press the **Tab** key to move the next column, **Description**. Here a brief description of the field can be entered. Pressing **Tab** will take you to the second field. Enter the details of the remaining fields in the same way.

The next task is to create a Primary key. A Primary Key is a field (or a combination of fields) in a table that uniquely identifies every record in the table. For example, RegNo in the Mark list example. Each RegNo will uniquely identify one student and no two students can have the same Reg.No. The primary key becomes very important when there are multiple tables, with common fields. Every table in StarOffice Base must have a primary key. To assign a primary key, in this case, RegNo, right click on the small triangle to the left of the field RegNo. A shortcut menu appears. Click on **Primary key**. A small yellow key appears to the left of RegNo, to mark it as the primary key.

After creating the structure of the table, you have to save the table with a name. To do so, either press **Ctrl + S** keys or go to the **File** menu and click on **Save**. Enter the name of the table in the dialog box which appears and click on **Save**.

The figure below shows a table design for the Mark list example discussed in Section 7.3 (table 7.1). Let us save it with the name **Marklist**.



**Fig 7.7 Marklist table design**

## Learn by solving

Design an Employee Table for the data given below to experience the use of other field types such as, Date, Currency along with Text and Number.



Field Name	Type	Description
EmpNo	Number	Employee Number
StaffName	Text	Name
Qualification	Text	Qualification
Designation	Text	Designation
Department	Text	Department
D-of-Join	Date	Data of joining
MonSal	Currency	

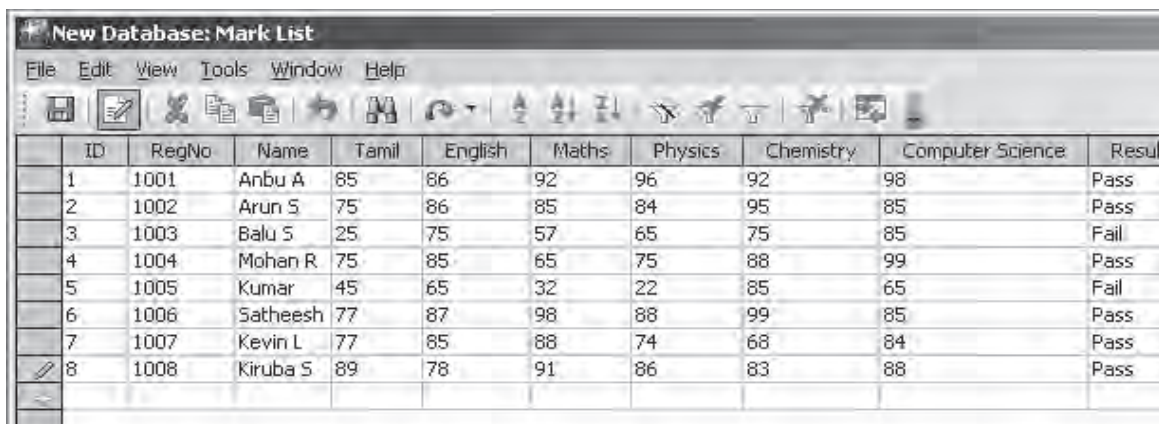
### 7.6.1.2 Entering the Data

Open the table **Marklist** on the **Table** pane by double clicking it or select the Marklist icon and then selecting the **Open Database Object** from the **Edit** menu. Now you can enter data into the table.

The cursor is on the first field of the first record. Now, simply type the data, pressing the **Tab** key to move from one field to the next.

**Each field can only accept data corresponding to the specified field type. For example, it is not possible to enter text in a number field.**

The **Marklist** table with some sample data is shown in figure 7.8.



ID	RegNo	Name	Tamil	English	Maths	Physics	Chemistry	Computer Science	Result
1	1001	Anbu A	85	86	92	96	92	98	Pass
2	1002	Arun S	75	86	85	84	95	85	Pass
3	1003	Balu S	25	75	57	65	75	85	Fail
4	1004	Mohan R	75	85	65	75	88	99	Pass
5	1005	Kumar	45	65	32	22	85	65	Fail
6	1006	Satheesh	77	87	98	88	99	85	Pass
7	1007	Kevin L	77	85	88	74	68	84	Pass
8	1008	Kiruba S	89	78	91	86	83	88	Pass

**Fig 7.8 Marklist table**

### 7.6.1.3 Editing the Data

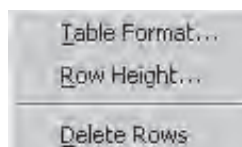
Editing may include changing a value of a field in a record, deleting a record or appending a record. In addition StarOffice Base also allows you to change the font type, font size and the height of the rows.

Editing the data that has been entered in a table is very simple. Just click on that field value and retype the new value. For example, to change the marks scored



by Mohan R in English from 85 to 75. Click on the field English in the fourth record and type in the new value, 75.

To edit a record, first select the record by clicking on the record pointer. The record pointer is a small triangle on the left most column of the table. Right click on the record pointer of the selected record. A shortcut menu appears as shown in figure 7.9.



**Fig 7.9 Submenu for editing**

This menu can be used to delete the selected record, change the height of the selected record and also format the table.

If you click on the **Delete Rows** option, the selected record will be deleted. For example, to delete the record that gives the particulars of 'Abdul Kader D', select the record, right click on the record pointer and click on **Delete Rows**.

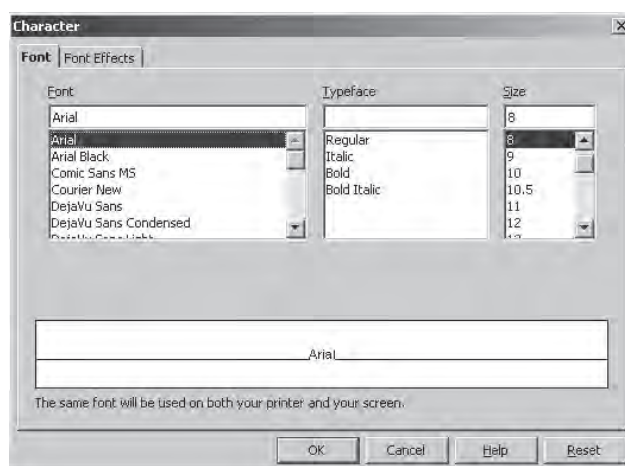
If you select the **Row Height** option, a dialogue box as shown below will appear.



**Fig 7.10 Dialogue box for setting height**

Specify the height and click on **OK** button. You will see that the change is effected.

If you click on the **Table Format**, a dialog box appears as shown in figure 7.11.



**Fig 7.11 Table Format window**

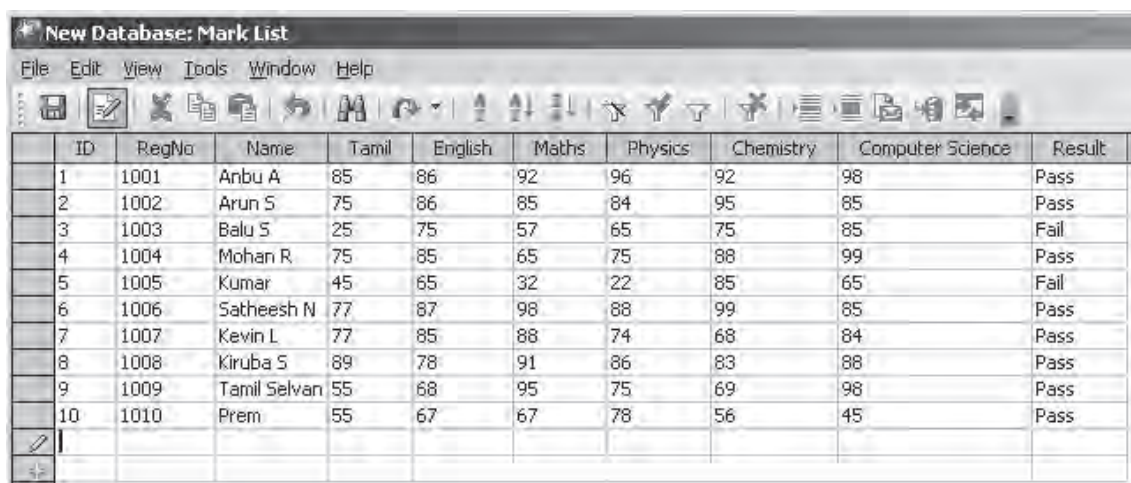
Using this dialog box, the font type, style, size, colour can be changed.

To append a record, enter the new record at the end of the table. For example, append two records,

- i. Tamil selvan S. with marks in Tamil, English, Maths, Physics, Chemistry, Computer Science and result as 55, 68, 95, 75, 69, 98 and Pass
- ii. Arun S. with marks in the respective subjects 55, 67, 67, 78, 56, 45 and Pass.

To do so, place the cursor at the last row of the table and type in the new values. When you enter data into the last row, a new blank row is automatically appended. You can use this row to enter the data of the second new record.

After you have finished all the editing save the table by pressing **Ctrl + S** keys or by clicking on the **Save** icon. The edited Marklist table is shown below.



ID	RegNo	Name	Tamil	English	Maths	Physics	Chemistry	Computer Science	Result
1	1001	Anbu A	85	86	92	96	92	98	Pass
2	1002	Arun S	75	86	85	84	95	85	Pass
3	1003	Balu S	25	75	57	65	75	85	Fail
4	1004	Mohan R	75	85	65	75	88	99	Pass
5	1005	Kumar	45	65	32	22	85	65	Fail
6	1006	Satheesh N	77	87	98	88	99	85	Pass
7	1007	Kevin L	77	85	88	74	68	84	Pass
8	1008	Kiruba S	89	78	91	86	83	88	Pass
9	1009	Tamil Selvan	55	68	95	75	69	98	Pass
10	1010	Prem	55	67	67	78	56	45	Pass

**Fig 7.12 Edited Marklist table**

#### 7.6.1.4 Viewing and Modifying the Table Design

To view or modify the design of an existing table, click the **Tables** from the <Database> pane. Select the table to be modified and right-click on it. Select **Edit** from the submenu that appears. A window with the existing table design appears. Here, you can change the field name, type, and properties, add or delete existing fields and so on. After completing the changes in the design, save the modified table to effect the changes.

#### Learn by solving

Modify the **Marklist** table with the following changes:

1. Change the marks of the subjects Tamil and English into a text field. Replace the existing data with grades according to the grading scheme given below:



Mark Range	Grade
96-100	H
90-95	A+
80-89	A
70-79	B+
60-69	B
50-59	C+
40-49	C
< 40	F

2. Add a new text field called **Comments** and enter comments such as 'outstanding', 'excellent', 'good', 'fair', and 'poor', for the grades 'A+ and H', 'B+' and A', 'C+ and B', 'C' and 'F' respectively.

---

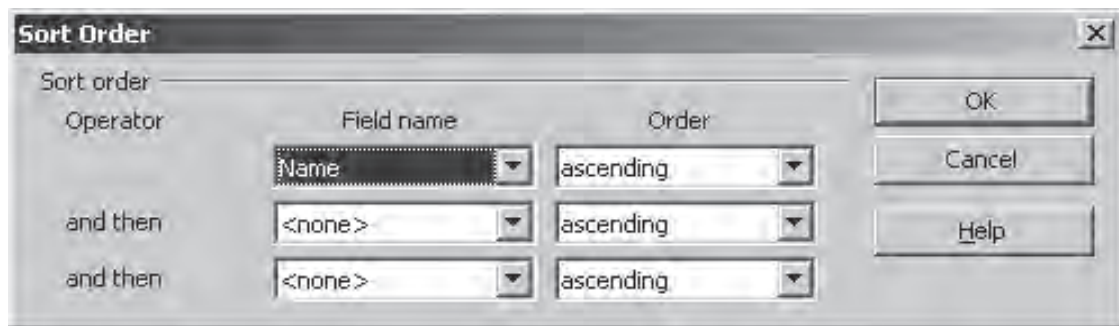
### 7.6.1.5 Sorting the Records

Once the records have been entered into the table, Star Base allows you to rearrange them by sorting them. You can sort the records in ascending or descending orders based on any field in the table. This can be done as follows.

1. Open the table that you want to sort.
2. Select the field you want to sort by clicking on the field name at the top of the table. Note that the entire column becomes highlighted. Then click on the **Sort Ascending** icon. The records in the table are displayed in the ascending order of the specified field. 
3. To sort the records in the descending order, select the field and click on the **Sort Descending** icon. 

Multiple sorting means sorting on more than one field of a table at the same time. For example, consider the Marklist table shown in figure. Let us assume that we would like to sort the records in the ascending order of names. This can be done as follows:

1. Click on the **Sort** icon. The dialog box as shown in figure 7.13 will appear. 



**Fig 7.13 Sort Order window**

2. Specify the fields you want to sort on by selecting them from the **Field name** drop-down list boxes. For each field specify the order in which you want to sort, in the **Order** drop-down list box. Click **OK** after you finish. The records will be displayed in the sorted order.

To display the records in the original order, click on the **Remove Filter/Sort** icon.




---

### Learn by solving

Sort the **Marklist** table on the field **Name** alphabetically and on **Reg No** in the ascending order of the register numbers.

---

### 7 .6.2 Querying a Database

Every DBMS supports a language that is similar to a programming language. This language, called the **Structured Query Language (SQL)**, is designed specifically for communicating with a database using statements that are closer to English than to programming languages. You can do the following using a query language:

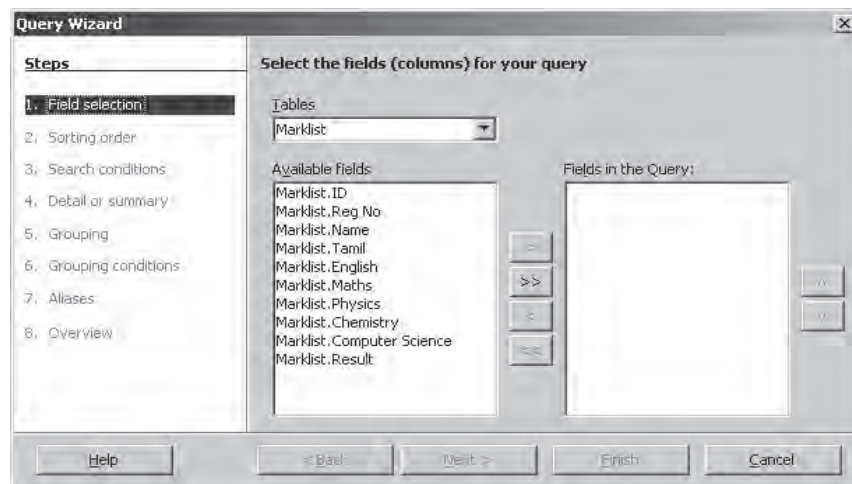
- Search the database to locate records
- Establish relationships or links between tables to update records.
- List a subset of records.
- Perform calculations.
- Delete obsolete records
- Perform other data management tasks.

Queries are special views of the data in a table. Unlike sort, the output from a query does not affect the original table. The result from a query is always stored separately and can be viewed at any time.

To create a query in StarOffice Base, right click on **Queries** in the **<Database>** pane.

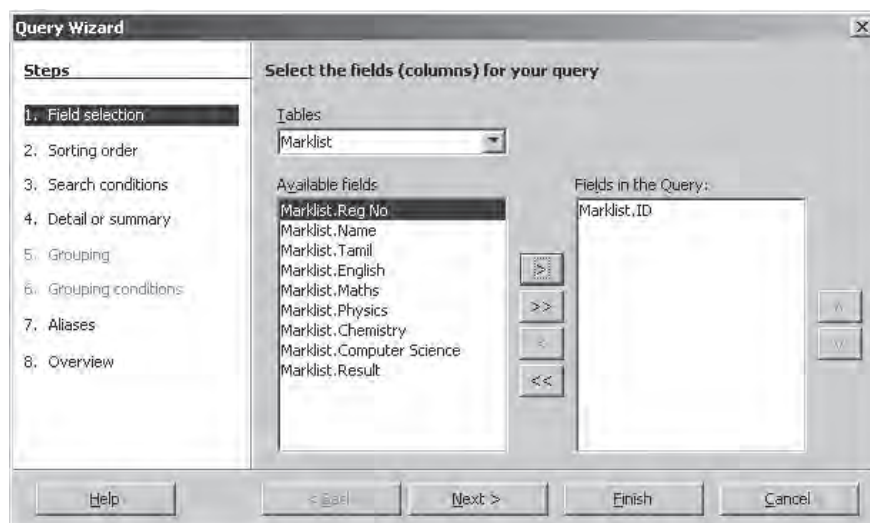
On the **Tasks** pane **'Create Query in Design view'**, **'Use wizard to Create Query'**, **'Create Query in SQL view'** options appears.

Click on **'Use wizard to Create Query'** option. The Query wizard appears as shown below.



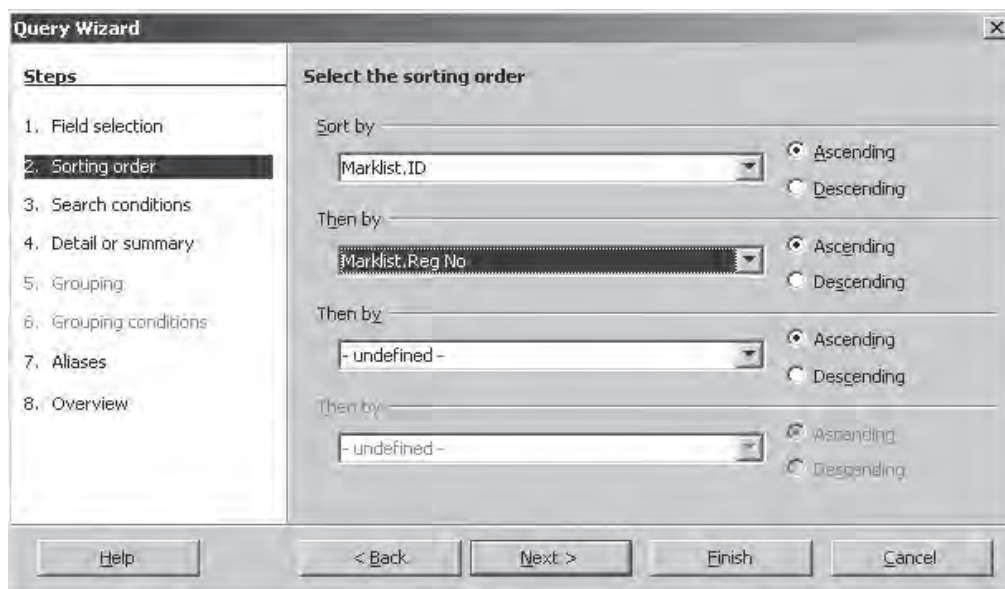
**Fig 7.14 Query Wizard**

From the **Tables** combo box, which contains the list of available tables, select the table on which you want to create the query. Automatically, the list of fields available in the selected table appears in the **Available fields** text area.



**Fig 7.15 Field Selection window**

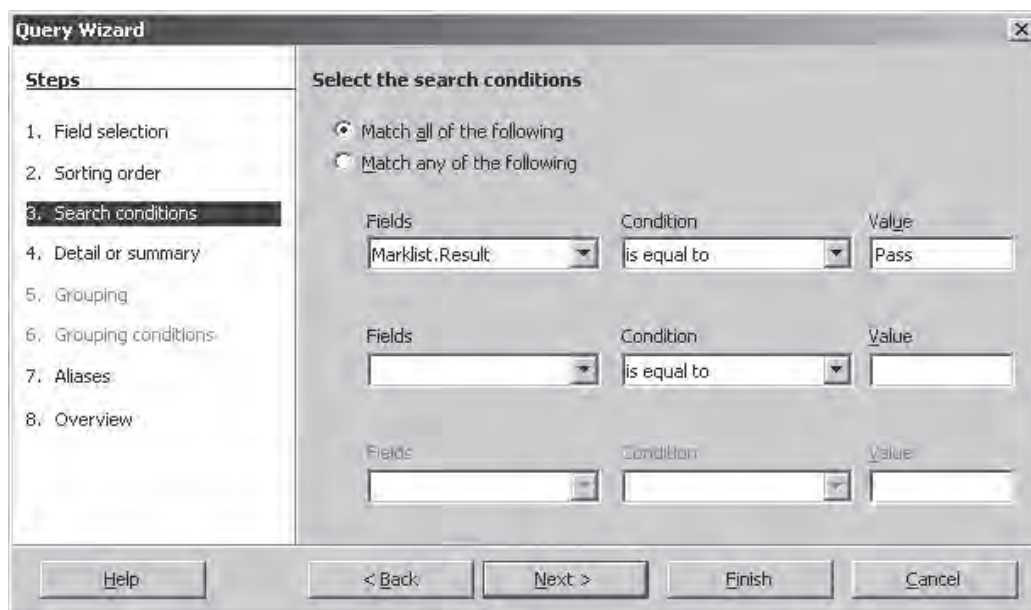
To select the fields to be included in the query, click on the field in the **Available fields** Text area and then click on the > button. To select all the fields at the same time, click on the >> button and click on the **Next** button. The following screen appears.



The 'Query Wizard' window is shown at the 'Select the sorting order' step. The 'Steps' list on the left includes: 1. Field selection, 2. Sorting order (selected), 3. Search conditions, 4. Detail or summary, 5. Grouping, 6. Grouping conditions, 7. Aliases, and 8. Overview. The main area has four 'Sort by' sections. The first two are populated: 'Sort by' is 'Marklist.ID' and 'Then by' is 'Marklist.Reg No.'. Each has radio buttons for 'Ascending' (selected) and 'Descending'. The last two sections are empty, with 'Then by' set to '- undefined -'. At the bottom are buttons for 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'.

**Fig 7.16 Sort Order window**

This window allows you to specify four fields on which the results should be sorted. The radio buttons on the right allow you to specify if the sort should be in the ascending order or in the descending order. It is not necessary to specify a sort order; you can simply click on the **Next** button if you do not wish to sort the result of the query, but the '**Ascending**' radio button is selected by default and click on the **Next** button. The following screen appears.



The 'Query Wizard' window is shown at the 'Select the search conditions' step. The 'Steps' list on the left includes: 1. Field selection, 2. Sorting order, 3. Search conditions (selected), 4. Detail or summary, 5. Grouping, 6. Grouping conditions, 7. Aliases, and 8. Overview. The main area has two radio buttons: 'Match all of the following' (selected) and 'Match any of the following'. Below are three rows of search criteria. The first row is populated: 'Fields' is 'Marklist.Result', 'Condition' is 'is equal to', and 'Value' is 'Pass'. The other two rows are empty. At the bottom are buttons for 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'.

**Fig 7.17 Search Conditions window**



This window is used to specify the fields and the conditions on which the query should be based.

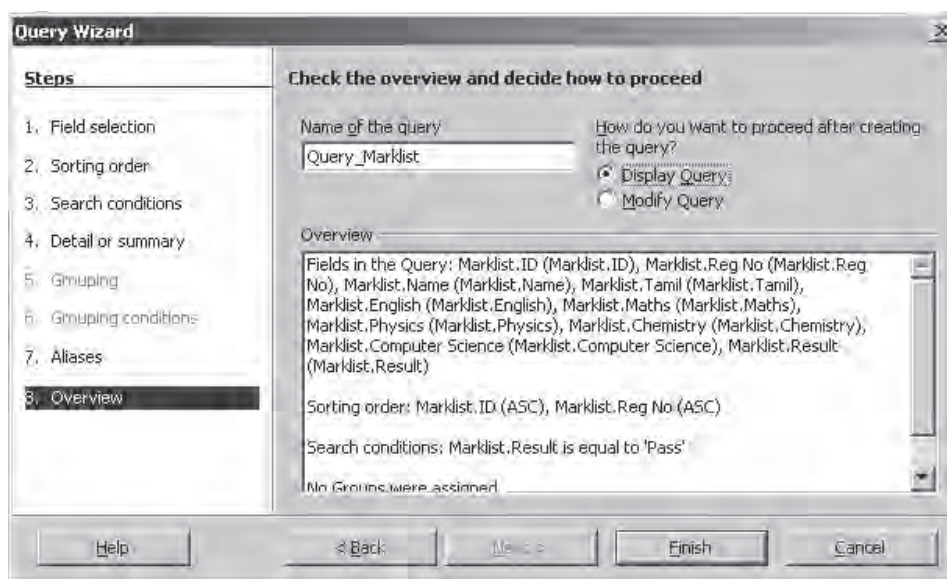
For example, suppose you want to list the students whose **Result** is 'Pass', you have to select the **Result** field in the first **Fields** combo box, then select '**is equal to**' sign in the **Condition** box and then type **Pass** in the **Value** box. Click **Finish** button. A new window appears with the records in which the field **Result** has the value **pass**.

If you want to display the records that satisfy any one of the search criterion then select '**Match any of the following**' or select '**Match all of the following**' to display records that satisfy all the search criterion.

This is an example with a very simple query. You can also create queries by using other relation operators (<>, <, <=, >, >=...) in the **Condition** box.

Click on the Next button, **Query wizard - Detail or Summary** appears. This specifies whether to display all records of the query, or only the results of aggregate functions. This screen is only displayed when there are numerical fields in the query that allow the use of aggregate functions.

Click on the Next button, the **Aliases** window of the **Query wizard** appears. This window allows you to provide the alias names for the fields. In the Aliases text boxes fill the alias names for the corresponding field names. Click on the **Next** button, the Overview window of the Query window appears as shown in the figure 7.18



**Fig 7.18 Create query window**


This window allows you to specify a name for the query. Selecting the **Display Query** radio button will execute the query immediately and the **Modify Query** option will allow you to modify the query, where you can change the sort order, Alias name, etc.

You can create any number of queries for a given table. Once the queries are created and saved, you can execute them whenever you want. To do so, double click one of the **Query** icons from the list of all the stored queries that appears in the **Queries** pane.


### 7.6.2.1 Filters

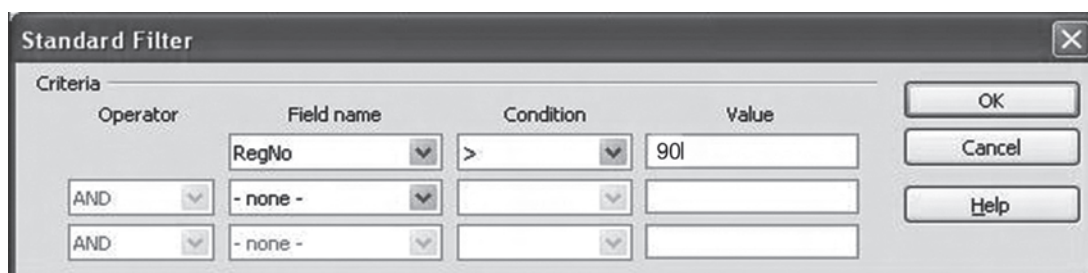
A filter is also a type of query. It is also used to select and display records, which match a certain condition. The remaining records are hidden from the user. The difference between query and filters is that queries can be saved for later use. StarOffice Base allows you to use two types of filters. They are **AutoFilter** and **StandardFilter**.

#### AutoFilter

The **AutoFilter**  icon is available on the toolbar. Click on this icon to display only the records, which match the value of the current field.

#### StandardFilter


Filter used with a condition called Default Filter. The condition can be specified by clicking on the **StandardFilter**  icon on the toolbar.



**Fig 7.19 Filter Window**

The **Filter** window is very similar to the one used for specifying conditions in a query.

Here is an example to help you understand better. Suppose you want to list all the records where the marks scored by the students in Maths is greater than 90. To do so, follow the steps given below:

1. Open an existing query from the **Queries** pane by double clicking on it. For example Query\_Marklist.
2. Click the **Standard Filter**  icon from the main toolbar. A **Filter** window will appear.
3. Select **Maths** in the Field name box.
4. Select > operator in the Condition box.



5. Type 90 in the Value box.
6. Click on OK.

The following figure shows the filtered records.

ID	RegNo	Name	Tamil	English	Maths	Physics	Chemistry	Computer Science	Result
1	1001	Anbu A	85	86	92	96	92	98	Pass
6	1006	Satheesh	77	87	98	88	99	85	Pass
8	1008	Kiruba S	89	78	91	86	83	88	Pass
9	1009	TamilSelva	55	68	95	75	69	98	Pass

**Fig 7.20 Filtered Records**

To remove the filter, click on the **Remove Filter / Sort**  icon. The original table, with all the records is displayed.

### 7.6.3 Form Designing

Although the display of data in columns and rows can be suitable for viewing records and performing small data-editing tasks, it is not as convenient for other data management tasks. For example, consider a table has so many fields that they cannot be displayed in a window. In such cases, while entering data, you have to use the scroll bars to see fields that are not displayed. This can be particularly inconvenient if a large amount of data has to be entered.

Such situations can be handled more conveniently with a customized form. A form in Star Base is just like the forms you fill up at the railway station, to get admission in a college and so on. A form is simply a screen that displays the fields of a record in a well-spaced out manner. The appearance of a form can be customized to your liking. A form can be associated with a single table or with multiple tables. A sample form based on the Marklist table is shown in figure 7.21.

Reg No: 1001

Name: Anbu A

Tamil: 85

English: 86

Maths: 92

Physics: 96

Chemistry: 92

Computer Science: 98

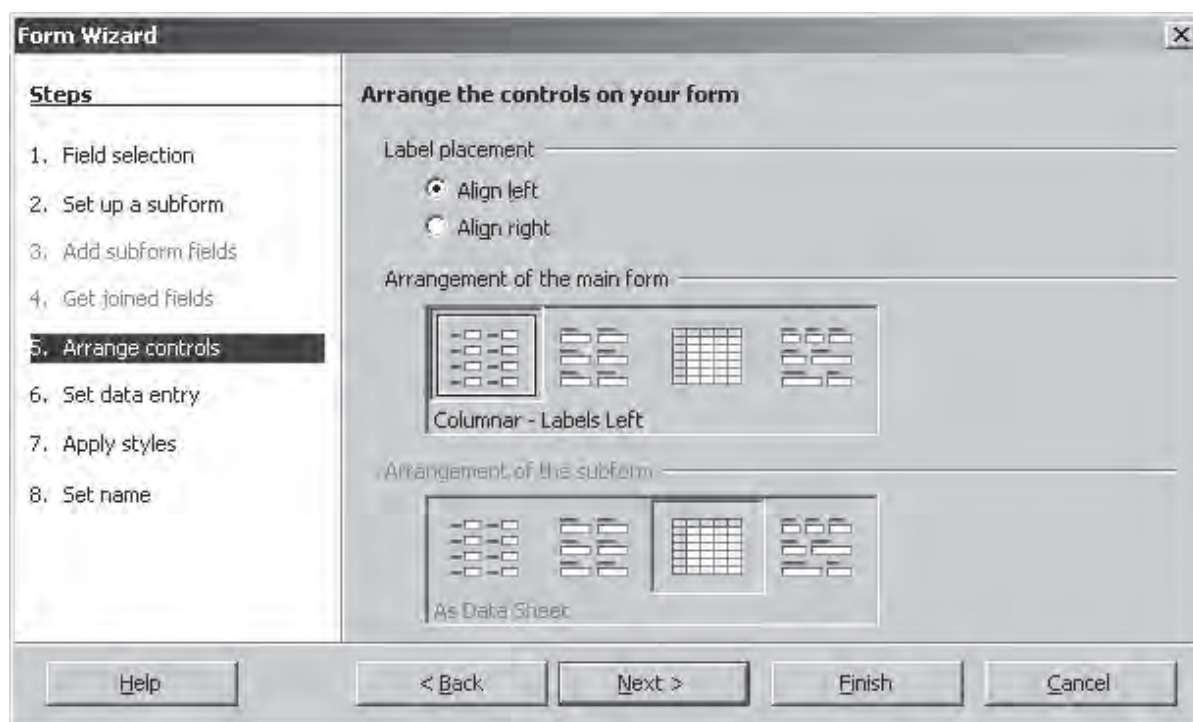
Result: Pass

Record 1 of 8

**Fig 7.21 Form based on the Marklist table**

Designing a form using the Auto Pilot option is very similar to creating a query using Autopilot. To design a form in Star Base, follow the procedure given below:

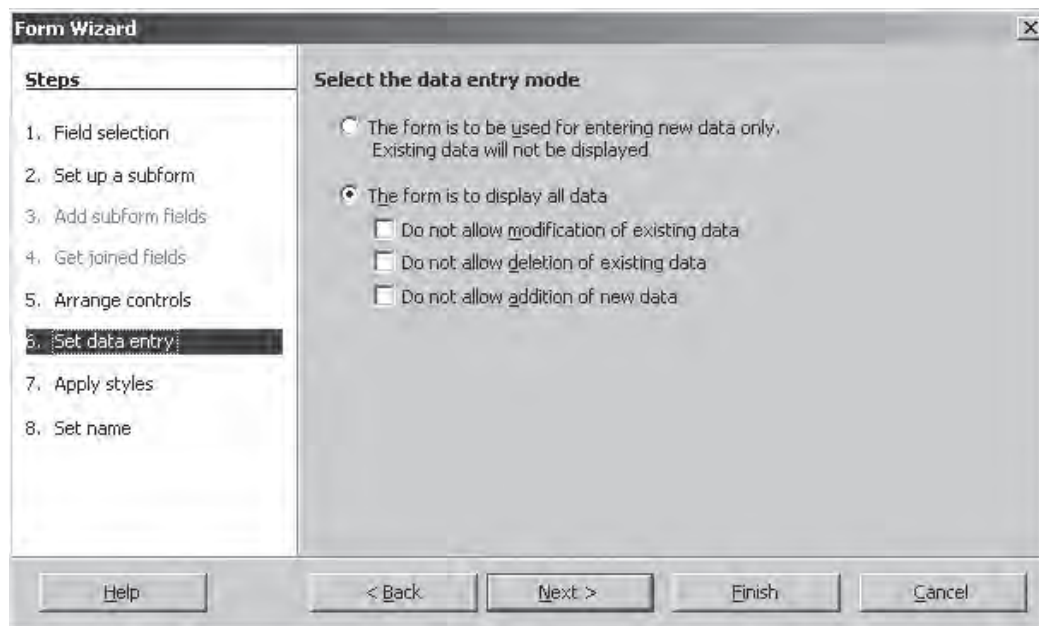
1. Select **Form** icon from the <Database> pane and then select '**Use Wizard to Create Form**'. A window, very similar to the one shown in figure 7.15, appears containing a list of the available tables and queries.
2. Select the table or query for which the form is to be designed. The field names of the selected table or query are displayed in the '**Available fields**' text area.
3. Select the field names that you would like to include in the form using the > or >> button and then click the **Next** button.
4. Next, StarOffice Base displays '**Set up a subform**' window. Select the '**Add Subform**' checkbox, if you want to insert another form within this form else click **Next**. The following window appears.



**Fig 7.22 Form Wizard window**

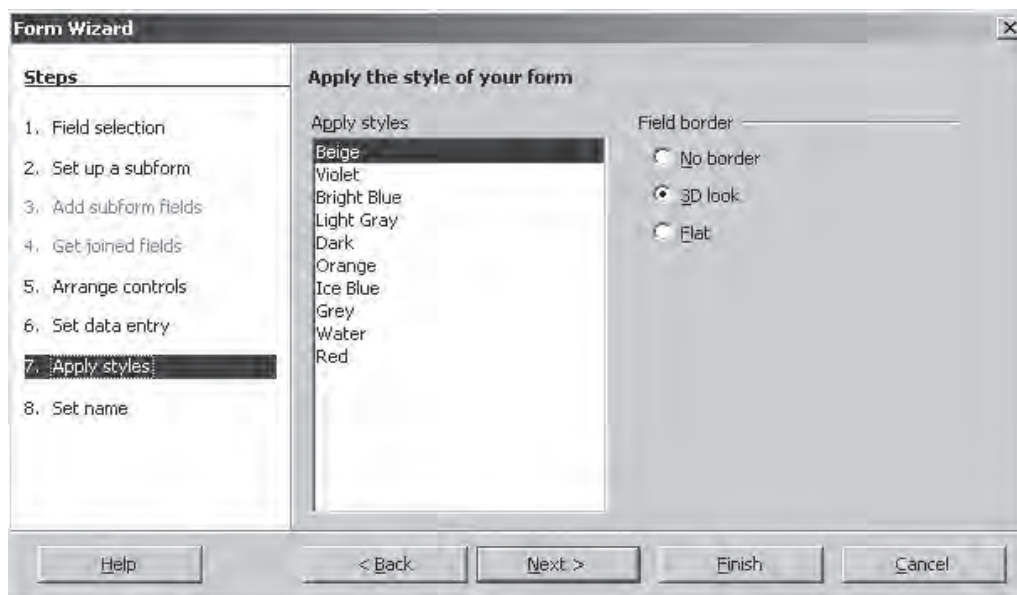
Here you can choose a style for displaying the fields in the form. The **Style** icons show you how the form will look if it is selected. Click on the **Style** icon and click on the **Next** button.

5. The next window (figure 7.23) is for selecting the data entry mode. Click on the radio button to choose whether the form is to be used only for entering new data or to be used for displaying all data.



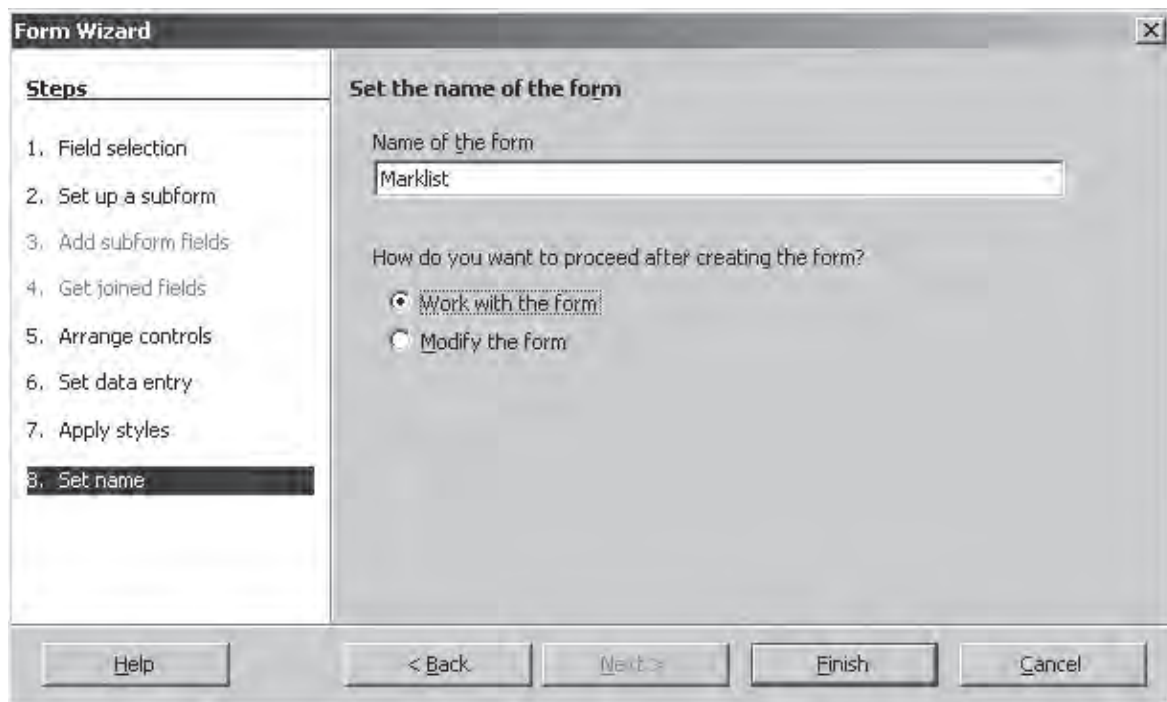
**Fig 7.23 Data Entry Mode window**

6. The next window (figure 7.24) **Form Wizard – Apply Styles** specifies the form style. The page style decides the font attributes like font type, size color and background of the text in the form. The **Field border** specifies the field border style for the form.



**Fig 7.24 Styles window**

7. Next, **Set the name of the form** window appears (figure 7.25) asking for a name for the form. The user can save the form, and open it as a form document to enter and display data using Work with the form option or opens it in edit mode to change the layout using Modify the form option. Type the name of the form and click the Finish button.



**Fig 7.25 Set the name of the form window**

Once a form is created, you can use to view, add, delete, and / or edit the records in the table. You can create any number of forms for a given table or a query. Once the form has been saved, you can use it by clicking on the **Form** icon in the **Forms** tab. Select the **form** you want to use by double clicking on it.

---

### Learn by solving

Using the Marklist table, design a form to view the marks scored by a student only in the science subjects such as, Maths, Physics and Chemistry.

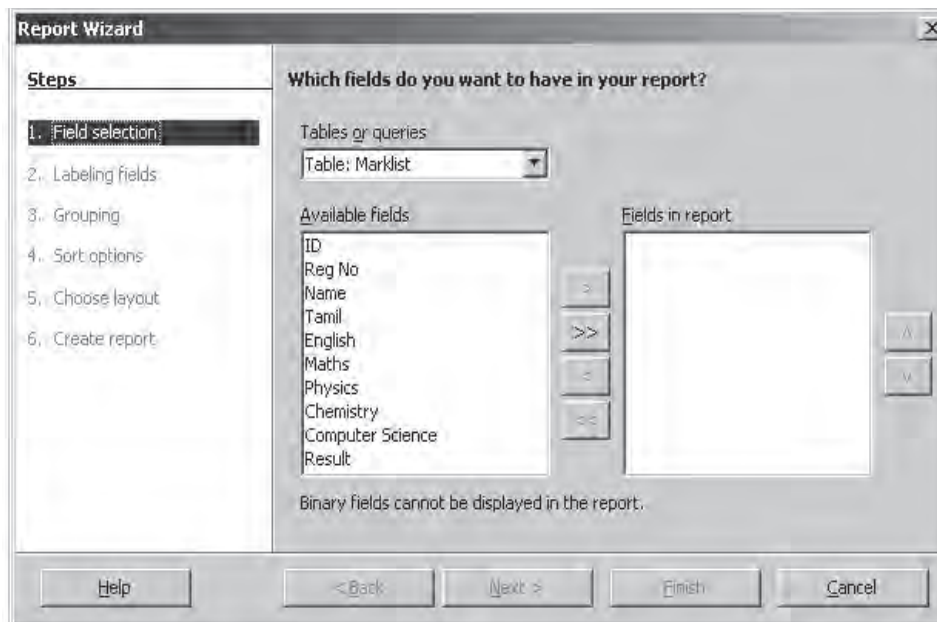
---

### Generating Reports

A report is printed information that is assembled by gathering data based on user supplied criteria.

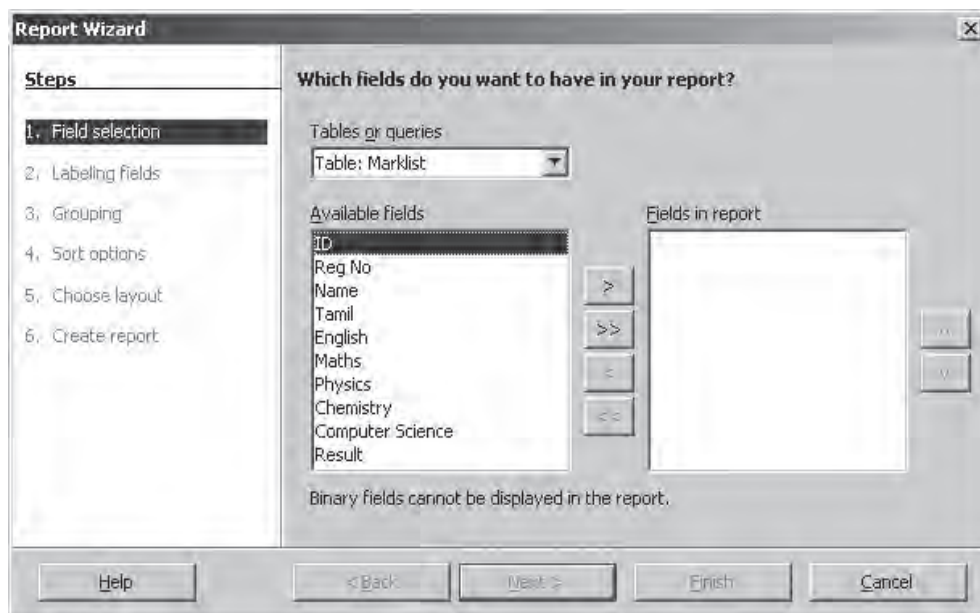
Reports can range from simple list of records to customized formats for specific purposes. Report generators can use selected data and criteria to carry out automated mathematical calculations as the output is being printed. In fact, report generators in most DBMSs create reports from queries.

Creating a report in StarOffice Base is very similar to creating a query or a form. To create a report, right click on **Reports** in the <Database> pane. Then click '**Use Wizard to Create Report**'. **Report wizard** appears as shown in figure 7.26.



**Fig 7.26 Report Wizard**

As in the case of form, this window displays a list of available tables and queries. Select the table or query you want. A list of fields from the selected table or query appears on the screen (figure 7.26). Select the fields to be included in the report by using the > or >> buttons and click on the **Next** button.

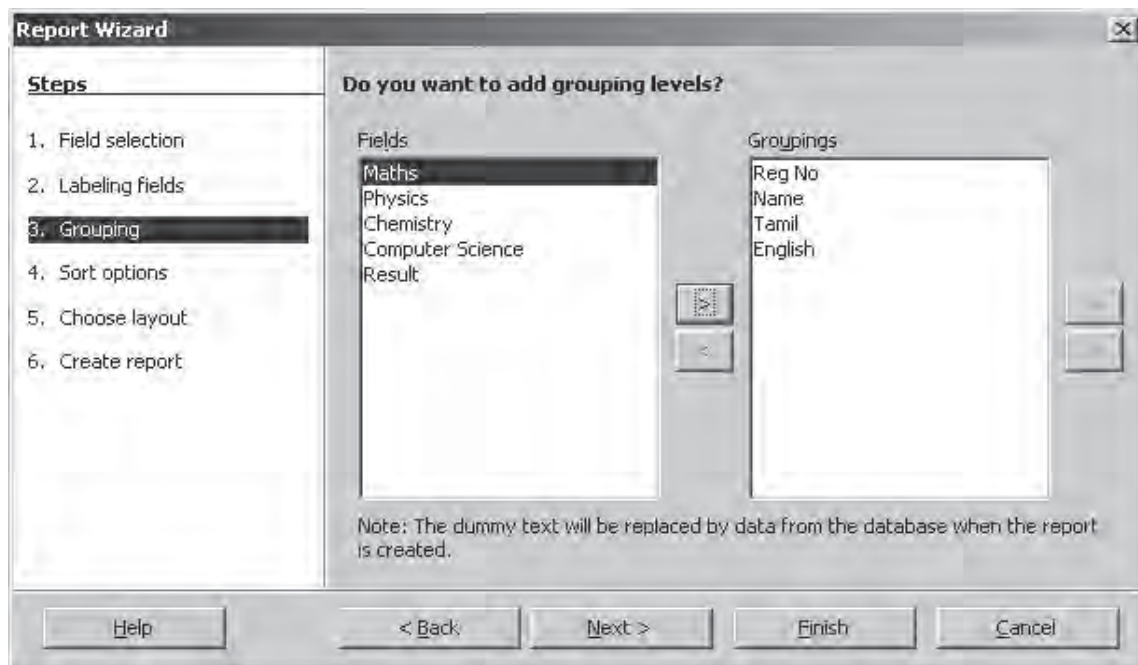


**Fig 7.27 Fields Selection Screen**

Next, a **Labeling fields** window appears with the field and Label. Modify the labels for the corresponding fields, if you want. Click on the **Next** button.

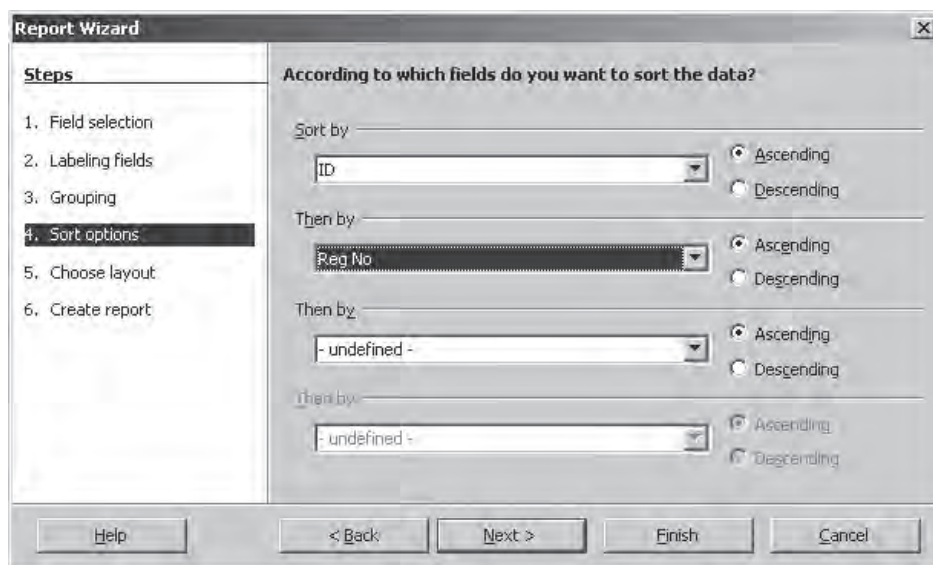
The **Grouping** window is displayed as shown in the figure 7.27. This window is used to specify the fields based on which the records can be grouped together.





**Fig 7.28 Report Wizard - Grouping window**

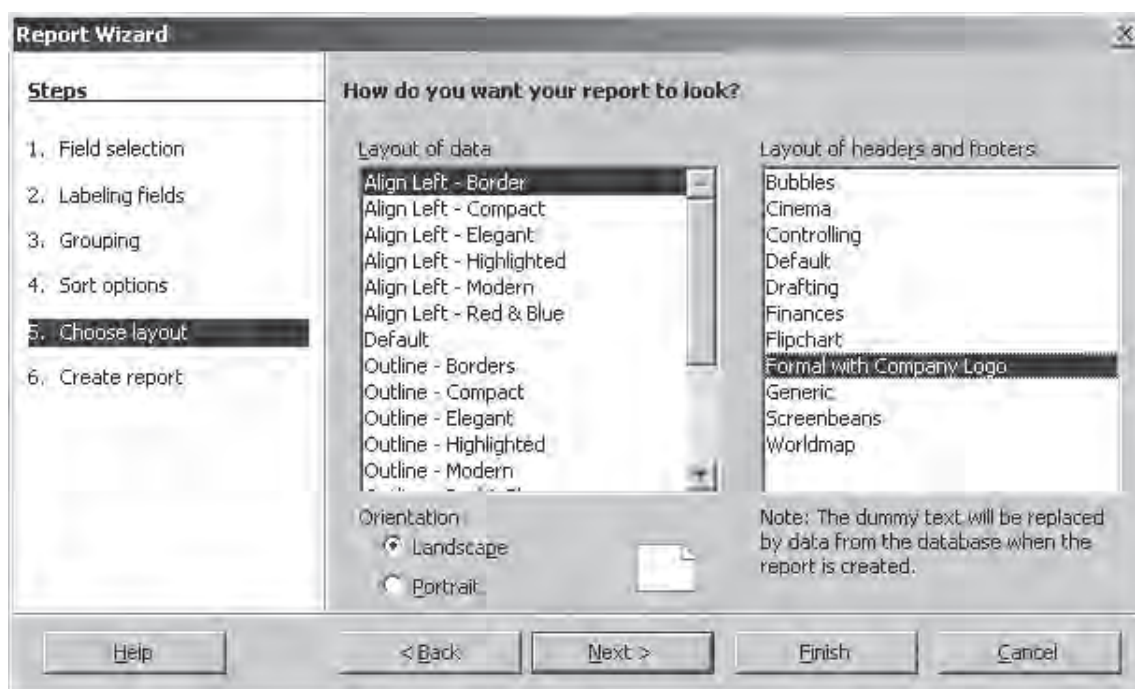
**Note:** It is not essential to specify a group field.



**Fig 7.29 Sort Options windows**

The **Grouping** window is followed by the **Sort Options** window (figure 7.29). Here the sort criteria, if any, can be specified.

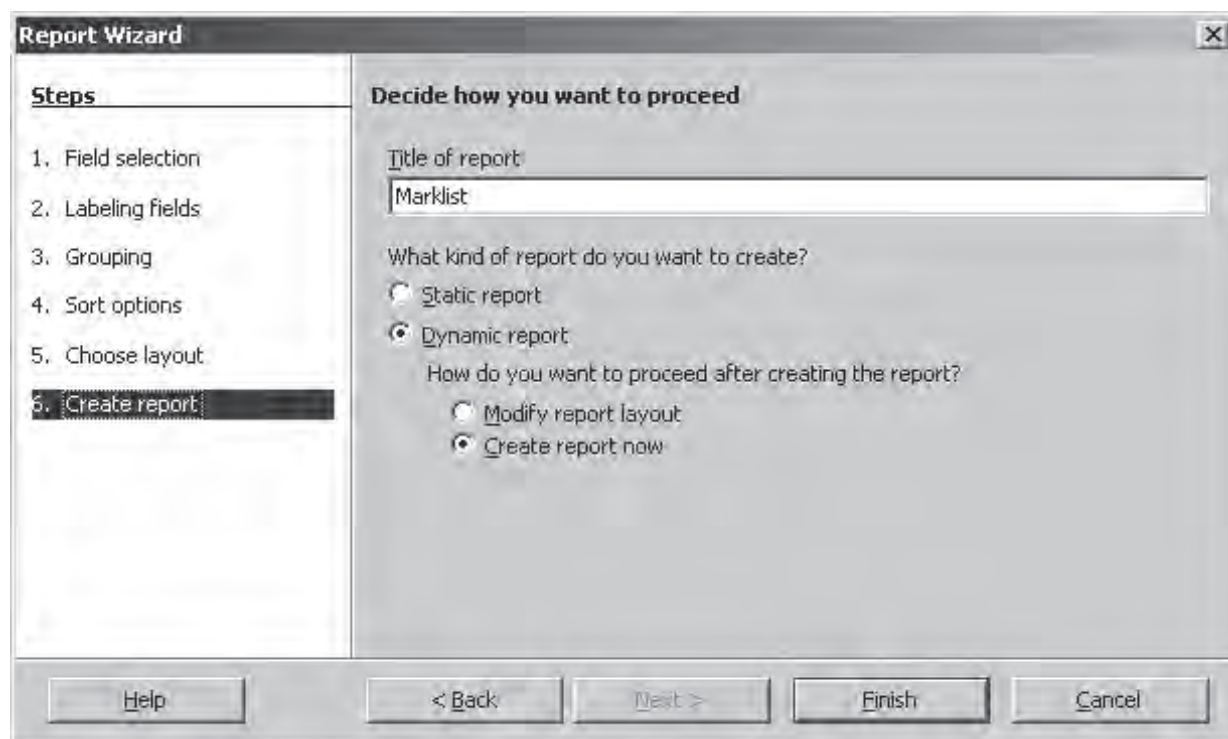
The **Choose Layout** window as shown in figure 7.30 is displayed, here you can customize the report by selecting the **Layout of Data**, **Layout of Headers and Footers** and **Orientation** options for the report.



**Fig 7.30 Choose Layout window**

The **Report Wizard** window (figure 7.30), which is displayed, next, allows you to choose **Static** or **Dynamic** Report.

The user also has the choice of either using the Report immediately or Modifying the Report Layout. Enter the title for the report and click the **Finish** button to view the report.



**Fig 7.31 Create Report Window**

You can create and store multiple reports for a table or a query. You can use a stored report in the same way as you used in stored form or query. Click on the **Reports** in the **Database Pane** and double click on the **Report Name** in the **Reports pane** of your choice.

## Learn by solving

1. Create a report showing all the records of the Marklist table.
2. Generates a report showing all those students who have scored grades A and above in Tamil. (Create a query first and then create the report)

## 7.7. Integrating with Office Automation Applications

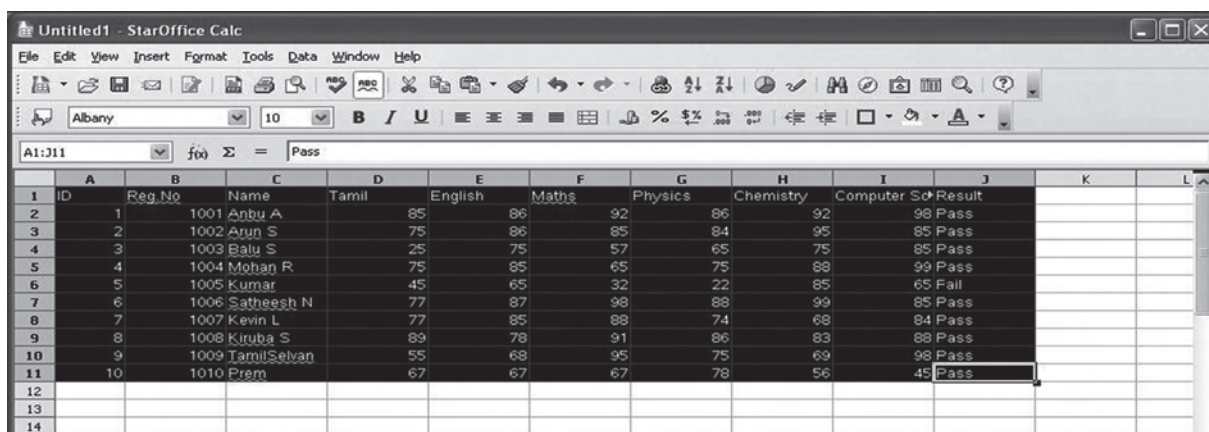
StarOffice applications are tightly integrated so that you can easily use the contents of one application in another application. The following sections provide a few examples of how you can take advantage of this integration.

### 7.7.1 Insert a Calc Cell Range into a Text Document

You can insert a range of Calc cells into a Writer document so that the data automatically updates when you modify the spreadsheet.

1. Open a StarOffice Writer (text) document
2. Open a StarOffice Calc (spreadsheet) that contains the data.
3. In the spreadsheet, select the cell range that you want to display as a table in the text document.

The selected cell range of spreadsheet is shown in figure 7.32

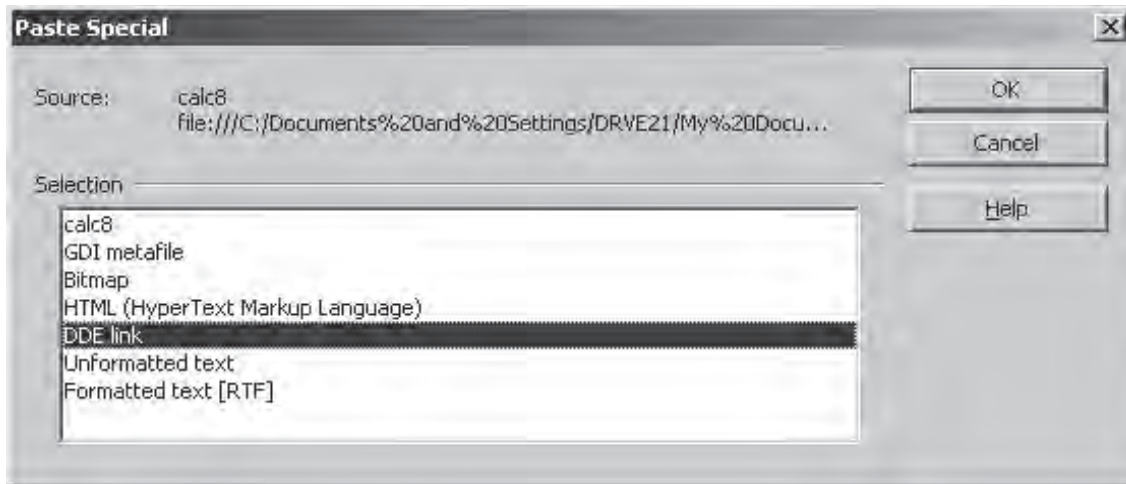


	A	B	C	D	E	F	G	H	I	J	K	L
1	ID	Reg. No	Name	Tamil	English	Maths	Physics	Chemistry	Computer Science	Result		
2	1	1001	Anbu A	85	86	92	86	92	98	Pass		
3	2	1002	Arun S	75	86	85	84	95	85	Pass		
4	3	1003	Balu S	25	75	57	65	75	85	Pass		
5	4	1004	Mohan R	75	85	65	75	88	99	Pass		
6	5	1005	Kumar	45	65	32	22	85	65	Fail		
7	6	1006	Satheesh N	77	87	98	88	99	85	Pass		
8	7	1007	Kevin L	77	85	88	74	68	84	Pass		
9	8	1008	Kiruba S	89	78	91	86	83	88	Pass		
10	9	1009	TamilSelvan	55	68	95	75	69	98	Pass		
11	10	1010	Prem	67	67	67	78	56	45	Pass		
12												
13												
14												

Fig 7.32 Selected cell range in spreadsheet

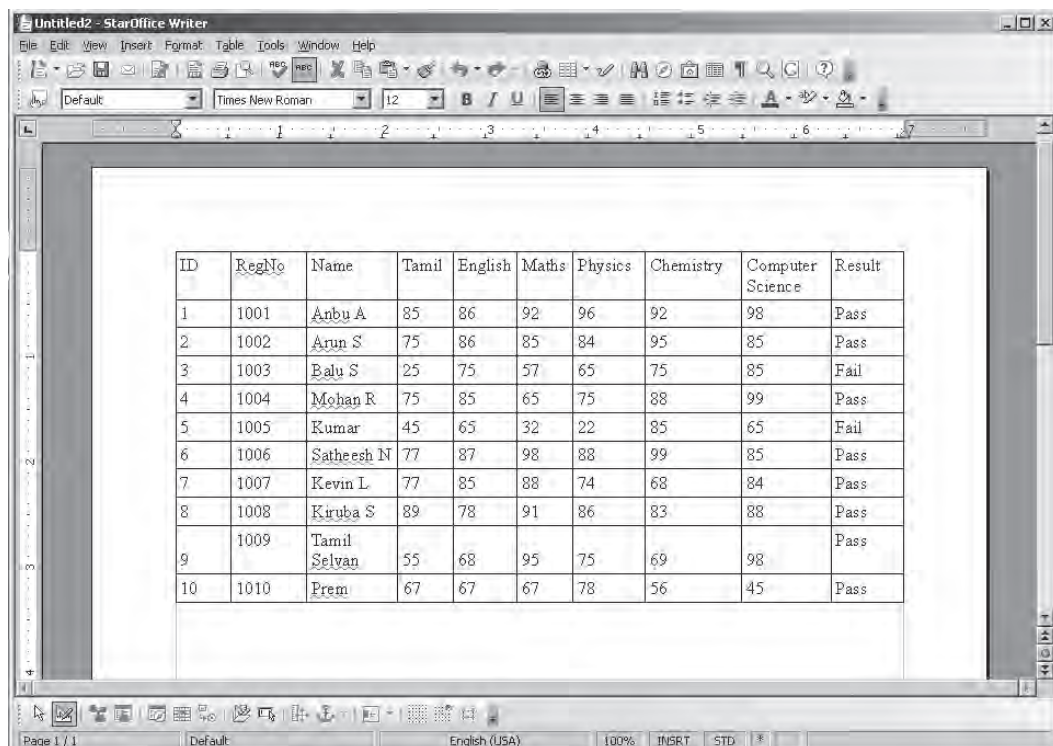


4. Choose **Edit** → **Copy**.
5. In the text document, choose **Edit** → **Paste special**.
6. In the Paste Special dialog, select **DDE link**, and then click **OK**.



**Fig 7.33 Paste Special dialog box**

7. Click Ok. The copied cell range will now be displayed in the text document as shown below.



**Fig 7.34 Text document with data copied from spreadsheet**

8. Now, modify the data in the spreadsheet. You will see the data automatically changes in the text document.

Untitled1 - StarOffice Calc

File Edit View Insert Format Tools Data Window Help

10 B / U = Vijay

	A	B	C	D	E	F	G	H	I	J	K
	ID	RegNo	Name	Tamil	English	Maths	Physics	Chemistry	Computer Science	Result	
1	1	1001	Anbu A	85	86	92	96	92	98	Pass	
2	2	1002	Arun S	75	86	85	84	95	85	Pass	
3	3	1003	Balu S	25	75	57	65	75	85	Fail	
4	4	1004	Mohan R	75	85	65	75	88	99	Pass	
5	5	1005	Vijay	45	65	32	22	85	65	Fail	
6	6	1006	Satheesh N	77	87	98	88	99	85	Pass	
7	7	1007	Kevin L	77	85	88	74	68	84	Pass	
8	8	1008	Kiruba S	89	78	91	86	83	88	Pass	
9	9	1009	Tamil Selvan	55	68	95	75	69	98	Pass	
10	10	1010	Prem	67	67	67	78	56	45	Pass	
11											
12											
13											

Fig 7.35 Data Modified in Spreadsheet

Untitled2 - StarOffice Writer

File Edit View Insert Format Table Tools Window Help

Table Heading Thorndale 12

ID	RegNo	Name	Tamil	English	Maths	Physics	Chemistry	Computer Science	Result
1	1001	Anbu A	85	86	92	96	92	98	Pass
2	1002	Arun S	75	86	85	84	95	85	Pass
3	1003	Balu S	25	75	57	65	75	85	Fail
4	1004	Mohan R	75	85	65	75	88	99	Pass
5	1005	Vijay	45	65	32	22	85	65	Fail
6	1006	Satheesh N	77	87	98	88	99	85	Pass
7	1007	Kevin L	77	85	88	74	68	84	Pass
8	1008	Kiruba S	89	78	91	86	83	88	Pass
9	1009	Tamil Selvan	55	68	95	75	69	98	Pass
10	1010	Prem	67	67	67	78	56	45	Pass

Page 1 / 1 Default English (USA) 100% INSERT STD

Fig 7.36 Data Automatically updates in Text document

## 7.7.2 Insert a Text Outline into a Presentation

Create an outline in a text document. An outline is text that uses one or more of the default heading paragraph styles, Heading 1, Heading 2, and so on. The text document with outline is shown in figure 7.37

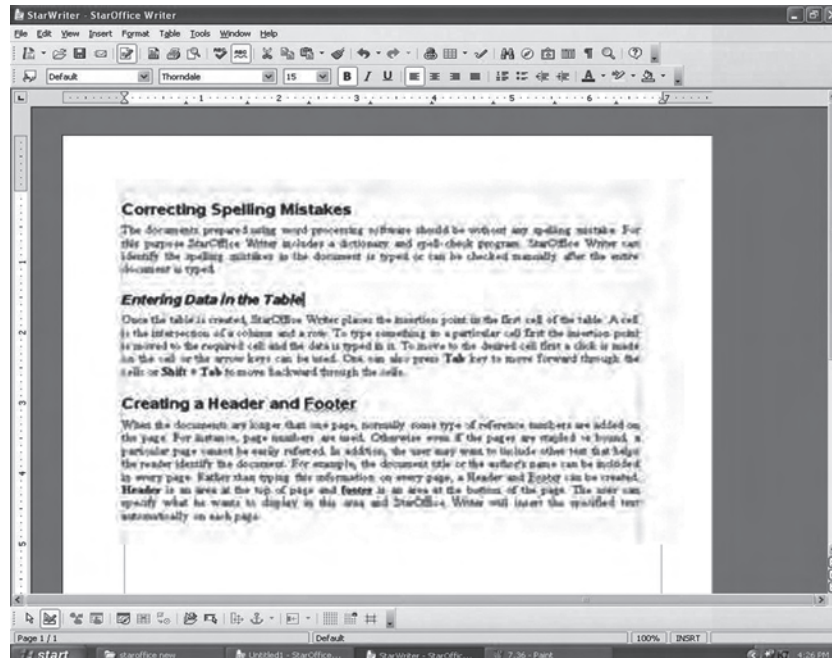


Fig 7.37 Text document with outline

1. In the text document that contains the outline, choose **File** → **Send** → **Outline to presentation**.

A new presentation document is created which has the outline applied as shown in figure 7.38. Each Heading 1 paragraph style corresponds to a new slide. The heading styles that occur following Heading 1 in the heading hierarchy are displayed as bullets on the slide.

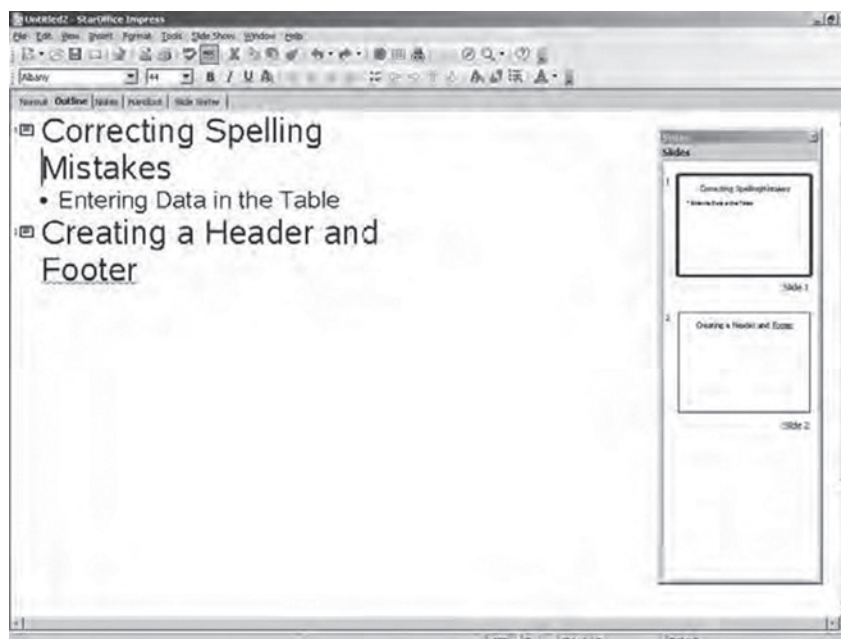


Fig 7.38 Presentation document with outline applied

2. Add more text to the outline or switch to Normal view to add objects.

If you want to transfer each heading together with its accompanying paragraphs, select the File → Send → AutoAbstract to Presentation command. You must have formatted the headings with a corresponding Paragraph Style to be able to see this command.

### 7.7.3 Copy Data by Drag-and-drop

You can use drag-and-drop to copy text and objects between StarOffice applications. For example, if you want to copy a cell range from a Calc sheet inside a presentation slide, proceed as follows:

1. Open StarOffice Impress presentation.
2. Open the Calc spreadsheet that contains the data that you want to copy.
2. In the spreadsheet, select the cell range that you want to copy.
3. Just drag-and-drop the selected range into the presentation (use Alt + Tab to open the presentation window).

The cell range is copied as a plug-in which is shown in figure 7.41.

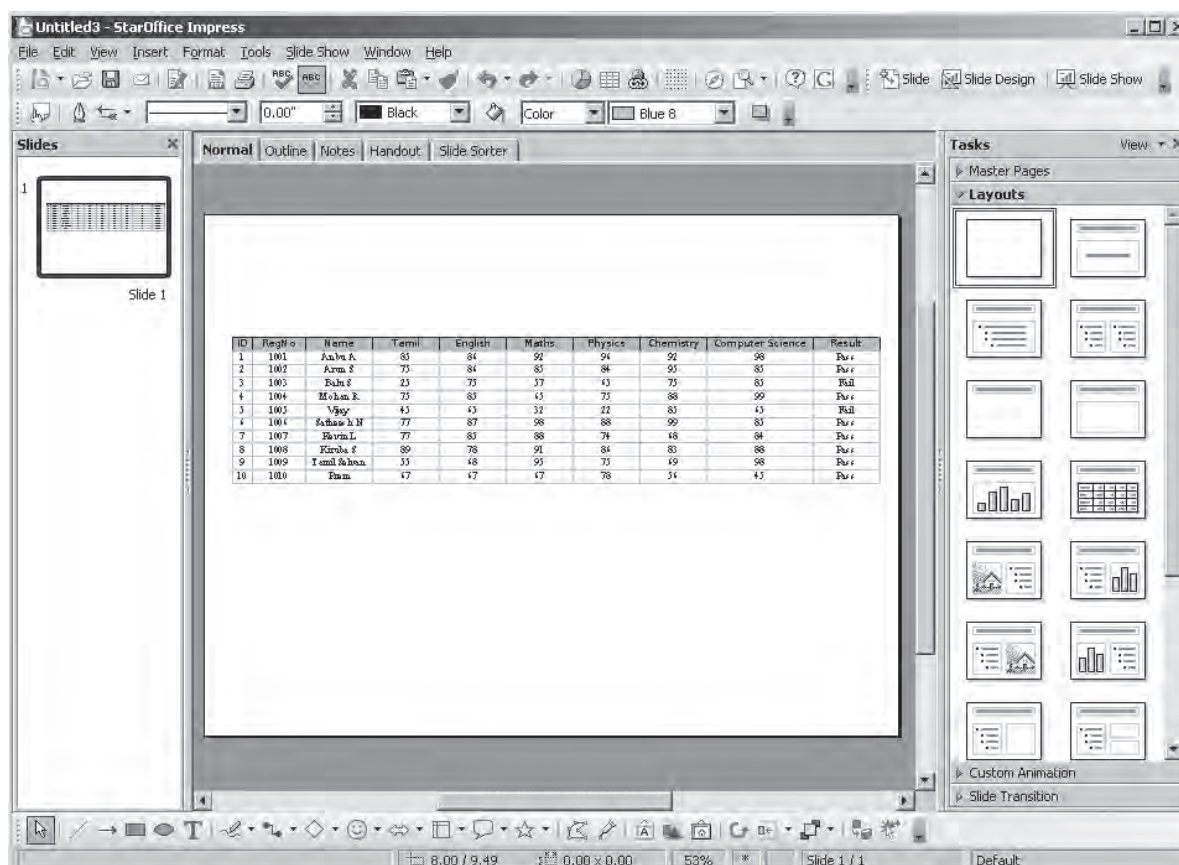
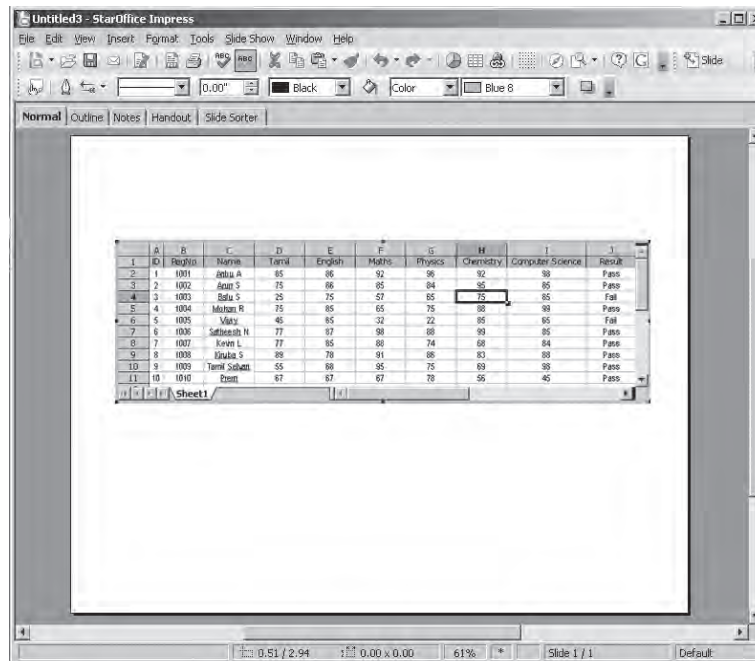


Fig 7.39 Cell range inserted as a plug-in in presentation

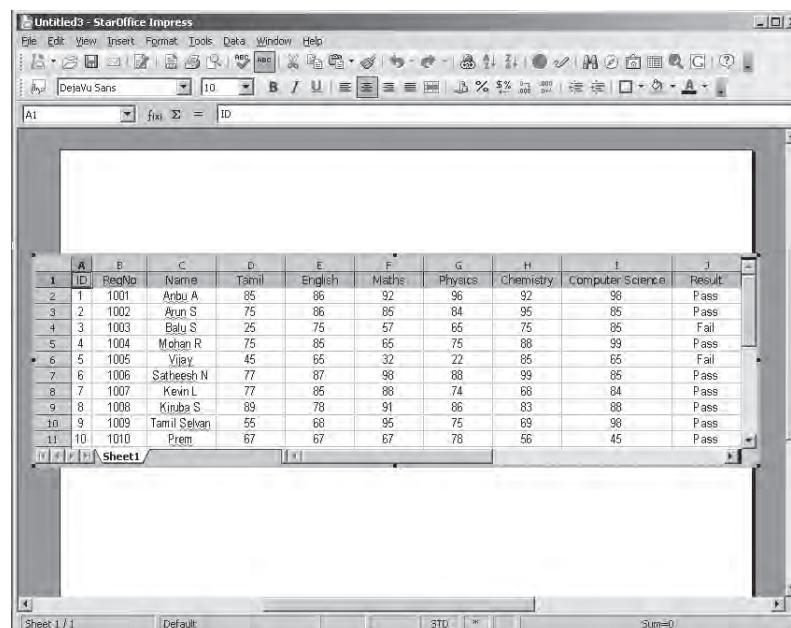


If you want to edit the contents of a copied cell in the presentation, double-click the cell. The Calc menus and toolbars are displayed when you are in this mode, even though you are in Impress presentation. To exit the edit mode, click outside the plug-in. The edit mode of the copied cell in the presentation is shown in figure 7.40.



**Fig 7.40 Edit mode of the copied cell in the presentation**

The same procedure is followed to copy a cell range from a Calc sheet inside a Writer text document. By double clicking the cell, you can edit the contents of the copied cell in the text document. The edit mode of the copied cell in the text document is shown in figure 7.41.



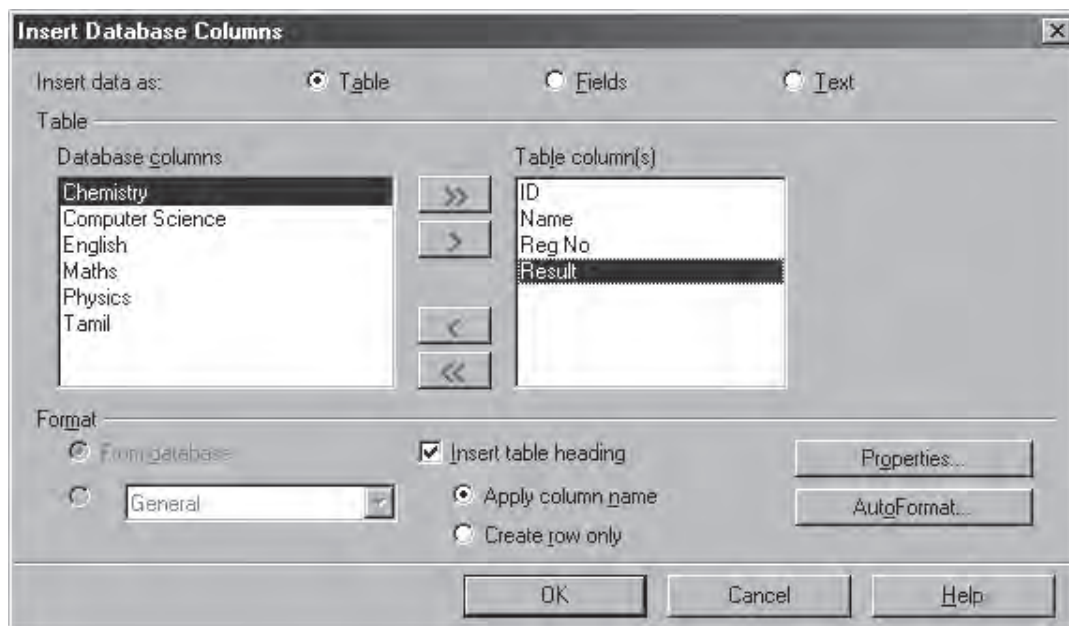
**Fig 7.41 Edit mode of the copied cell in the text document**

### 7.7.4 Inserting a Database table into a Text document

You can insert a Database table into a Writer document.

1. Open StarOffice Base, in the Database pane select Tables and in the Tables pane select the table which contains data.
2. Choose Edit → Copy or press Ctrl + C .
3. Open StarOffice Writer, select Edit → Paste.

The Insert Database Columns dialog box appears as shown in figure 7.42.



**Fig 7.42 Insert Database Columns dialog box**

4. Select whether the data should be inserted as a table, as fields or as text.

The preferences you set in the Insert Database Columns dialog are saved and will be active the next time the dialog is called. This save process is independent of the database and can record the preferences for a maximum of 5 databases.

If data is inserted into the document as a table, the table properties are not saved along with the data in the document. If you select the AutoFormat function for formatting the table, StarOffice will note the name of the format template. This template will then be used automatically if you insert data as a table again, unless the preferences have been changed.

5. Move the listed database fields into the table column(s) list box using > or >>  
>> - Moves all listed database fields into the Table column(s) list box. All fields listed in the Table column(s) list box are inserted into the document.

- > - Moves the selected database field into the Table column(s) list box. You can also double click an entry to move it to the Table column(s) list box. All fields listed in the Table column(s) list box are inserted into the document.
- 6. In the Table area, use the arrow keys to select the columns of the database table that you want to apply to the text table.
- 7. Insert table heading specifies whether to insert a heading line for the columns in the text table.

### Apply column name

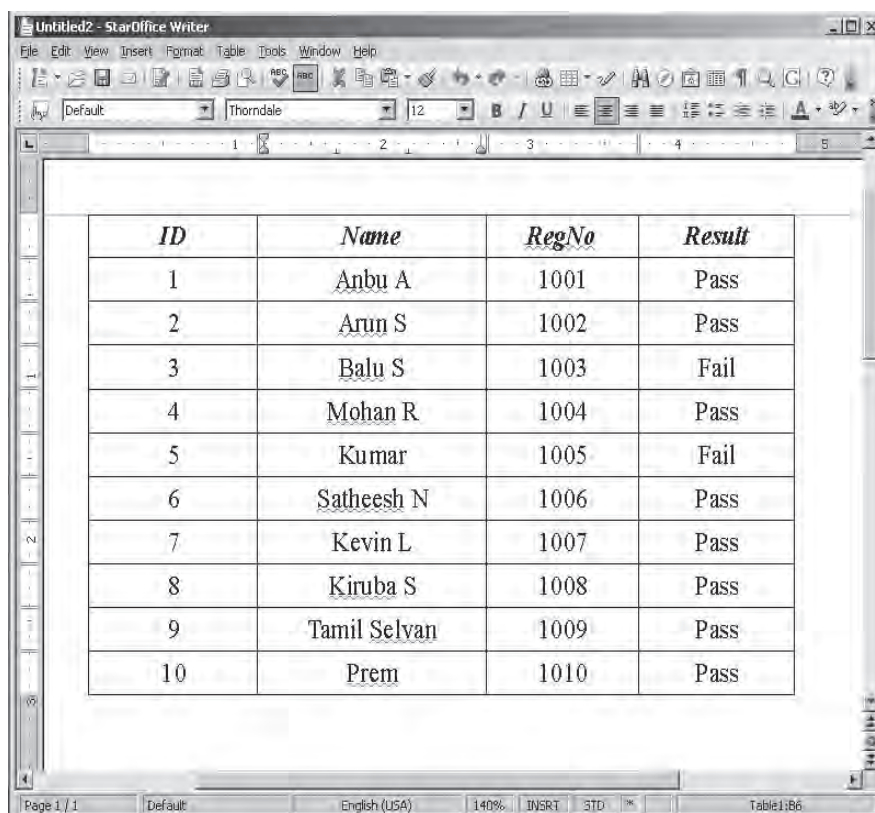
Uses the field names of the database table as headings for each of the text table columns.

### Create row only

Inserts an empty heading line into the text table. Using the Create row only option, you can define headings in the document, which do not correspond to the names of the database field.

- 8. Click Ok to view the inserted table in the text document.

Figure 7.43 displays a database table which is inserted into a text document.



<i>ID</i>	<i>Name</i>	<i>RegNo</i>	<i>Result</i>
1	Anbu A	1001	Pass
2	Arun S	1002	Pass
3	Balu S	1003	Fail
4	Mohan R	1004	Pass
5	Kumar	1005	Fail
6	Satheesh N	1006	Pass
7	Kevin L	1007	Pass
8	Kiruba S	1008	Pass
9	Tamil Selvan	1009	Pass
10	Prem	1010	Pass

**Fig 7.43 Database table inserted in text document**

## Summary:

- Data and information
  - Data are raw facts collected, such as those collected by an organization to record transactions.
  - Information consists of usable sets of data.
  - A DBMS is used to assemble information from data.
- Database
  - A database is repository for collections of related items or facts.
  - A DBMS is a software tool that allows users to create database tables and that provides access to multiple users.
  - A database table arranges data in columns representing fields and rows representing records.
  - Field types include text, numeric and logical
  - Flat-file databases cannot form relationships with other tables and are best suited for home or small business users.
  - Relational database are very powerful because they have the ability to form relationships between tables.
  - Hierarchical and Network Structures are conceptual model for older systems that define databases used on Main Frame Systems.
  - Object Oriented Databases define data, data characteristics, attributes, and procedures as complex objects that can combine with other objects through messages
- Working with StarOffice Base
  - Table or Datasheet view is used to create a table's structure, view lists of records, enter and edit data.
  - Filters let you browse through selected records that meet a set of criteria.
  - Sorting arranges records in a table according to specific criteria.
  - Queries are user-constructed statements that set conditions for selecting and manipulating data in one or more tables and assembling the criteria-matching data into information.



- SQL is a special query language for communicating with a database by describing data and sometimes instructing the DBMS to do something with the data.
- Forms are user defined screens that are used to make it easier to enter, view and edit the data in a table or a query.
- A report is printed information based on a query that gathers criteria-matching data and, in some cases, performs mathematical calculations.

## **EXERCISES**

### **I. Fill in the blanks**

1. A table column represents a(n) \_\_\_\_\_
2. Data such as names, addresses, and phone numbers are stored in a (n) \_\_\_\_\_
3. A row in a table represents a (n) \_\_\_\_\_
4. A screen that displays data for a single record is called a (n) \_\_\_\_\_
5. You can arrange records in alphabetical or numerical order by \_\_\_\_\_
6. \_\_\_\_\_ key field uniquely identifies every record in the table
7. \_\_\_\_\_ key combinations is used to save a table
8. The two types of filters are \_\_\_\_\_ and \_\_\_\_\_
9. \_\_\_\_\_ is used to insert a form into another form
10. The two types of reports are \_\_\_\_\_ and \_\_\_\_\_

### **II. Answer the following**

1. What is a Database?
2. What is a Database Management System?
3. What distinguishes information from data?
4. List and define the three components that make up a database.
5. What primary characteristics distinguish a Flat-files database from a relational database?

6. Which database structure is characterized by parent-child relationships among record types?
7. List and describe the elements that makeup an object in the object oriented database model.
8. List the various field types that can exist in a database.
9. What does it means to 'filter' database records?
10. Describe what a query is and what it is used for?
11. Explain form designing and describe how it is used?
12. Explain the process of report generation using a table or query?
13. How will you integrate a spreadsheet cell range into a text document?

## **CHAPTER 8**

### **INTRODUCTION TO MULTIMEDIA**

#### **8.1 What is Multimedia?**

Multimedia is a computer-based presentation technique that incorporates text, graphics, sound, animations, and video elements. A combination of these elements grabs the viewers' attention and retains it. The multi-sensory inputs address the different learning needs and styles of different users and enhance the entire experience for the user.

#### **8.2 Multimedia Applications**

Multimedia applications are being used in a variety of different fields. Some fields where multimedia is commonly used include entertainment, education, research, and business communications.

In the field of entertainment, multimedia is widely used to add special effects to movies. It is also used to create animated films and cartoon strips. Animated films such as Finding Nemo, Polar Express, Ice Age, Sindbad, Pandavas, Hanuman are popular with both adults and children. High-end graphics and animations are frequently used to make computer games thrilling and realistic. Such games are very popular and are found in most home and game arcades. A high-end extension of these games is the flight simulator that is used to train pilots.

The Multimedia Messaging System, or MMS, is an application that allows you to send and receive messages over cell phones. These are popularly being used to send and receive jokes, music, ringtones, pictures and sometimes even videos.

Multimedia also has an enormous impact on education. With the growing popularity of multimedia, sophisticated e-learning packages are used to train and educate people world over. The most common version of such packages are the Computer Based and Web Based Tutorials (CBT/WBT). These are self-paced learning aids that are either available on CD-ROMs or on the Internet. As mentioned, these tutorials are self-paced. So, the learner can learn by himself, at a pace and time that he is comfortable with. Plus there is no instructor; the learner does not have to travel to a location to attend the class. This eliminates the logistical problems of getting together people from different parts of the world. Such trainings are also cost-effective when the same training has to be imparted over and over again. CBTs/WBTs today are used to cover a wide range of topics including orienting new employees joining an organization, training employees on new software applications and systems used in an organization.

Multimedia based training is also being extensively used to train and educate children in schools and colleges. For example, a physics student can use an interactive

multimedia presentation to learn the basics of electronics and actually create and test a circuit board. Similarly, multimedia simulations are very effective in explaining difficult concepts and facts. Such media make learning more fun and effective.

The impact of information presented in a multimedia encyclopedia is several times greater than the same information presented in the printed version. For example, a video of Neil Armstrong landing on the moon with a sound recording of his first words has a much greater impact than anything that can be written in a printed book. The use of hyperlinks in such material makes it easier to search for and view related content. Such non-linear access to information definitely speeds-up the learning process and makes it more rewarding.

Multimedia is heavily used in the entertainment industry, especially to develop special effects in movies and animation for cartoon characters. Multimedia games are a popular pastime and these are software programs available either as CD-ROMs or online. Some video games also use multimedia features.

Multimedia applications that allow users to actively participate instead of just sitting by as passive recipients of information are called Interactive Multimedia. An example are interactive multimedia games. For instance, users can play a simulated multimedia soccer match without actually being on the field. The simulation is just an illusion, but it makes the users think that they are playing a real match. The environment is created by using input devices like joysticks and sensors and by using output devices like headphones and goggles. The various multimedia components are coordinated with a technique called virtual reality. They provide an environment which is experienced by users as similar to reality. This technique is used in some arcade games and also in flight simulators, to impart training to pilots, without having to go for a real flight.

Multimedia is also used for corporate communications and presentations. Context-relevant animations, images and charts help in grabbing the viewers attention and highlighting critical pieces of information in corporate presentations. Multimedia elements are also used in advertisements, product catalogs, and online magazines.

Multimedia applications are also widely used in the fields of engineering, medicine, and scientific research. For example, in engineering, multimedia tools are used for designing and testing new components and products. In medicine, doctors can get trained by viewing at a virtual surgery or by simulating a surgical procedure without endangering the life of a human being. Similarly, a scientist can look at the molecular model of a compound and manipulate it.

### 8.3 Multimedia Elements – Sound, Animation, and Video

It is true that there are many image formats. Some of them are GIF files, JPG files, Animated GIF files, MPEG files, Shockwave files and NxView files.

The two most common by far are GIF and JPG files. Both of these formats compress static (as opposed to animated) bitmap images. Photographs are typical examples of static images. Both the GIF and JPG formats compress the image in different ways. A JPG file uses a much more complex technique than GIF to compress images, like photographs, where the color of every pixel is different. A GIF file creates a perfect reproduction of the original, while a JPG does not.

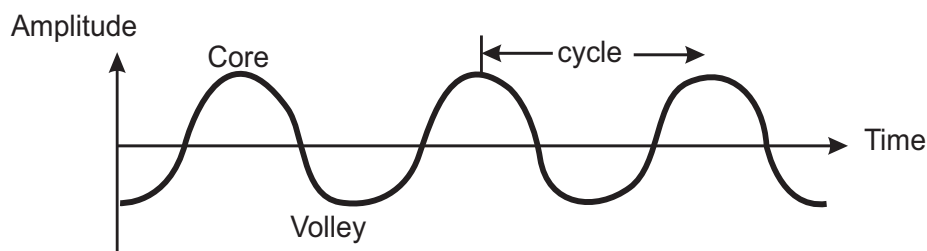
GIF (Graphic Interchange Format) is limited to an 8-bit palette. GIF is best suited for storing simple graphic images with relatively few colors. However, it is not well fitted for photographic works. JPG scores over GIF in this area. Despite this GIF is still used extensively on the Internet and Multimedia applications because of the great reservoir of GIF images available and support for animation.

The JPEG (Joint Photographic Experts Group) image files are a lossy format. Nearly all digital cameras have the option to save images in JPEG format. The JPEG format supports full color and produces relatively small file sizes. The two categories of image file compression are: lossy and lossless. The lossy compression algorithm takes advantage of the limitations of the human visual senses and discards information that would not be sensed by the eye. The loss of information is tolerable, and in many cases goes unnoticed. JPG is a lossy compression of the image.

Vector graphics help in rendering the image effectively on the screen. Many vendors provide special hardware called vector graphics cards to improve the efficiency in the display of images.

#### Sound

The sounds that you hear are analog wave patterns. Two attributes control the characteristics of sound - amplitude and frequency. The volume is the height of each crest in the wave. The frequency, also called as the pitch, is the distance between the crests of the wave. The greater the distance, the lower is the sound.



**Fig 8.1 An Analog wave pattern**

Before including sounds in an application, you have to convert the analog sound waves into a digital format. This conversion of analog sound waves to a digital format is called Sampling. You can further enhance the quality of the converted sound waves and add special effects, such as echo, fade in, and fade out effects, by using sound editing programs such as Sound Forge.

Sound can be stored in several different formats. You will learn more about these formats a little later in the chapter.

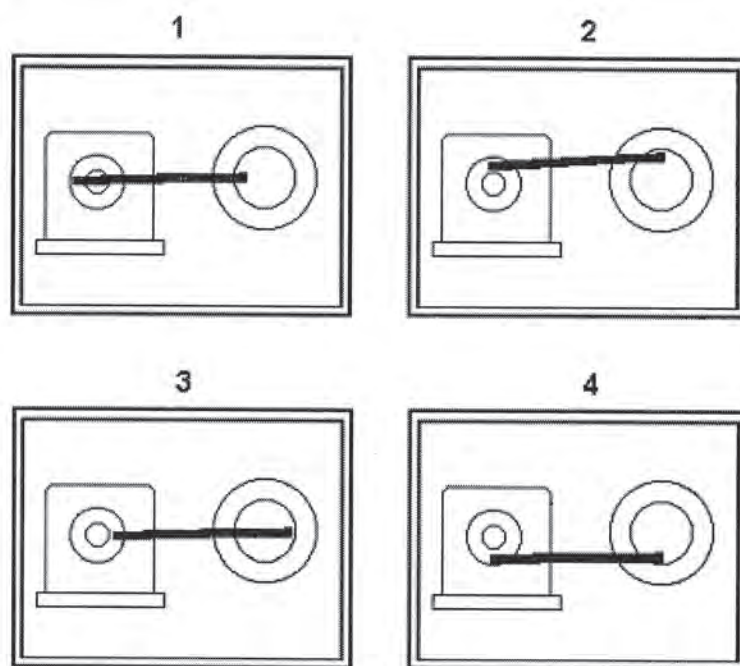
## Animations

Animations are primarily used to illustrate or demonstrate an idea or a concept. Unlike videos that are usually taken from life, animations are based on drawings.

Animations can be two- or three-dimensional. Based on how 2-D animations are created, they can be broadly classified into the following two categories:

- Cel-based animations
- Object-based animations

Cel-based animations consist of multiple drawings, each one a little different from the others. When displayed in rapid sequence, these drawing appear to move.



**Fig 8.2 Cel-based Animations**

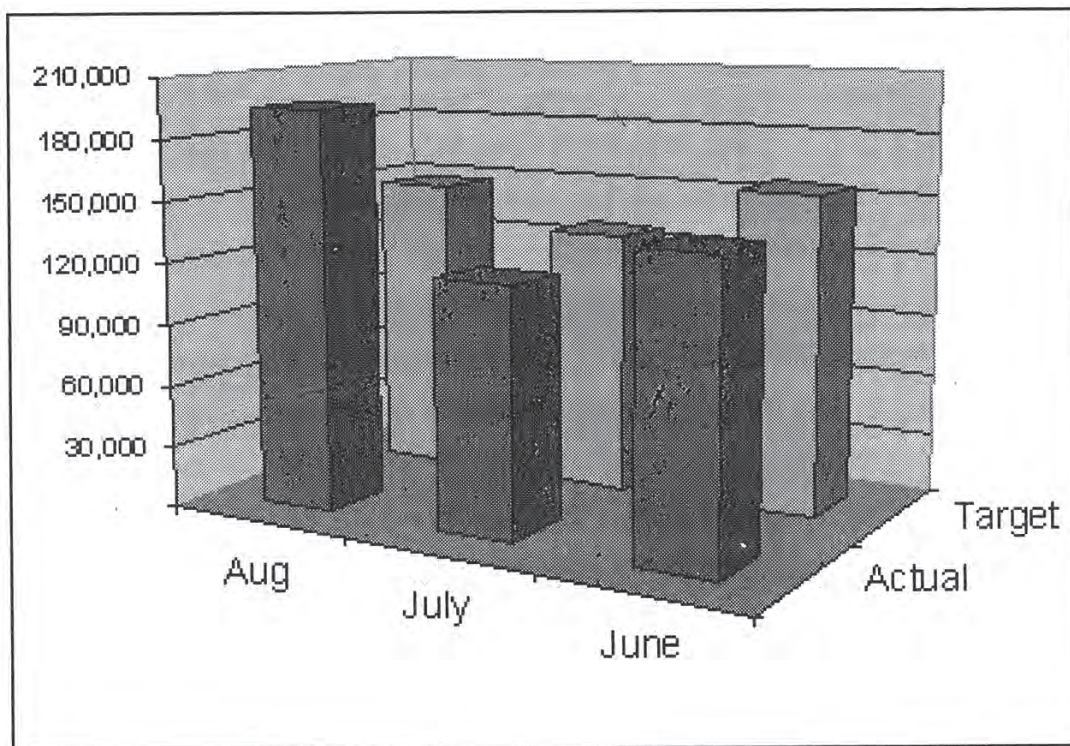
Object-based animations, also referred to as slide or path animations, are created by moving an object across a screen. This type of animations are usually seen in computer games. For example, a ball moving across the screen.



### Creating a 3-D animation is a 3-step process:

1. Modeling
2. Animating
3. Rendering

To create a 3-D animation, you have to first create the broad shapes and structures of the 3-D objects to be included in the animation. Then, you have to animate these models to define their motion. Finally, you have to render them by giving attributes, such as colors, and textures, to the objects.



**Fig 8.3 A 3-D Image**

Once the animation is created, you can further enhance it by adding special effects such as morphing and warping. Morphing is a technique by which you can blend two or more images to form a new image. Warping is the technique of distorting a single image to represent something else. Most modern multimedia applications, particularly games, combine these techniques with virtual reality to create an environment in which gives the viewer the feeling that he is part of that environment.

### Video

Like sound, video is also recorded and played back as an analog signal. So, you have to convert the video signal into a digital format before including it in a multimedia application.



Video files tend to be very heavy. Several elements such as frame rate, image size, and color depth determine the size of these files. You can vary these elements to reduce the file size to acceptable levels. However, reducing the color depth to less than 256 colours results in the image looking murky. Similarly, reducing the frame count to less than 15 frames per second causes the video to look jerky.

You can also reduce the size of videos by compressing them. Compression is a technique by which a recurring set of information is identified and replaced by a single piece of information. There are two types of video compressions:

- Lossless compression – Retains the exact image throughout the compression.
- Lossy compression – Provides a comparatively higher ratio of compression but results in some loss of quality.

## **Popular Multimedia Formats**

Some popular multimedia formats are as follows:

### **The MIDI Format**

The Musical Instrument Digital Interface or the MIDI format is one that is commonly used for transferring music information between electronic music devices like synthesizers and sound cards in computers.

It was developed in 1982 and is a very flexible format that can be used for a wide range of musical applications. The MIDI format cannot contain sounds- it contains only digital notes. As a result, these files are relatively light. For example, a MIDI file that plays for approximately 5 minutes may be only 25 KB. This format is supported by most of the popular softwares in the market including all popular browsers.

Sound files in the MIDI format have the extension .mid or .midi. The RealAudio/RealVideo Format The RealAudio format was developed by RealNetworks in 1995 and supports both sound and videos. This format is more popular for transfer of data over the Internet and allows you to stream files even over Internet connections with low bandwidths. However, the quality of these transmissions is often poor. Files in this format have the extension .rm or .ram.

### **The AU Format**

The AU format is another sound format that is supported by most of the popular softwares in the marker across different platforms. Files in the AU format have the extension .au.

### **The AIFF Format**

The Audio Interchange File Format or AIFF format was developed by Apple. This is not a very popular format as it is neither cross-platform nor is it supported by all web browsers. Files in the AIFF format have the extension .aif or .aiff.

## **The SND Format**

The Sound or SND format was also developed by Apple. Like the AIFF format, the SND files neither cross-platform nor supported by popular web browsers. Files in the SND format have the extension .snd.

## **The WAVE Format**

The WAVE format was developed by IBM and Microsoft. It is one of the more popular formats as it is not only supported by all computers running the Windows operating system, but also by all popular web browsers. Files in the WAVE format have the extension .wav.

## **The MP3/MPEG Format**

The MPEG format was developed by the Moving Pictures Experts Group. This MPEG format developed for video files while the MP3 format is used to store sounds (music). However, these formats have become one of the most popular format on the Internet in recent years. The main reason for the popularity of these formats is that they offer good compression and high quality. They are also cross-platform and supported by most popular web browsers.

Sound files stored in the MP3 format have the extension .mp3, or. mpga (for MPG Audio). Video files stored in the MPEG format have the extension .mpg or .mpeg.

## **The AVI Format**

The Audio Video Interleave or AVI format was developed by Microsoft in 1992. It is supported by all computers running the Windows operating systems and by most of the popular browsers. It is a very common format on the Internet. Videos files stored in the AVI format have the extension .avi.

## **The Windows Media Format**

The Windows Media format was developed by Microsoft. It is also one of the very popular formats on the Internet and on computers with the Windows operating system. It should be noted that this format requires the installation of an additional component in non-Windows computers. Files in the Windows Media format have the extension .wmv.

## **The QuickTime Format**

The QuickTime format was developed by Apple primarily to store videos. It is a popular format on the Internet but requires the installation of an additional component in non-Windows computers. Files in the QuickTime format have the extension .mov.

## **The Shockwave Format**

The Shockwave format was developed by Macromedia and is used to store multimedia components created using Flash. This format requires an extra component to play. However, this additional component comes preinstalled with the latest versions of Netscape and Internet Explorer. Files in the Shockwave format have the extension .swf.

## **Multimedia Hardware and Software**

Most common personal computers are adequate to work with multimedia. However, some additional hardware and high resolution monitors would greatly enhance the effectiveness of using multimedia. Also special software such as Windows Media Player would help in viewing the multimedia presentations.

Creating multimedia content requires special software. Some commercial multimedia content development software are Flash, Dreamweaver and Maya. More advanced multimedia hardware and software are available.

## **Inline Sound and Video**

Sounds and video in multimedia applications can be played “inline” or by using a “helper”. How the sound or video is played depends on the HTML tags that you use. When sound or video is included as part of a web page, then it is called inline sound or video. The disadvantage of using inline sound or video is that it plays automatically whenever the web page loads without giving the viewer any control. You can add inline sound to a web page by using the <bgsound> and the <img> tags. A helper application, also called as a Plug-in, is a program that can be launched by the browser to “help” play sound or video. The advantage of using helper applications is that you can allow the viewer control some or all of the player settings. For example, the viewer can control the volume settings, in addition to letting him rewind, play, pause, and stop the sound or the video.

Helper applications can be launched using the <embed>, <applet>, or the <object> tags.

Note: These tags are not standard HTML or XHTML tags. Some are supported by both Netscape and Internet Explorer while others are supported only by Internet Explorer. You can also include a hyperlink to a media file. When you do so, the browser will load a helper application, such as Window Media Player, to play the file.

## **8.4 Using Multimedia Elements in Content**

Indisputably, including multimedia elements in content adds immense appeal to it. However, before deciding to include multimedia elements, it is essential to consider

one major factor. This factor is the file size. Files with multimedia elements tend to be very heavy. For example, a 5-second audio file can be as heavy as 1MB. This, combined with the limitations of the medium of presentation can limit if not totally nullify the impact of the presentation. For example, a heavy video or animation file in a WBT may not load because of poor bandwidth. Or it may take so long to load that the learner loses interest and decides to move on to something else.

Here are a few tips that you can keep in mind while making such decisions.

Consider the appropriateness of using the element. Ensure that there is a justifiable need for every element that you include in your content and do not include any element without a compelling benefit from it.

Consider using alternate multimedia elements to provide the same impact. For example, consider replacing a heavy video file with a sound file and a static image.

Streaming the heavy audio and video files. During a regular HTTP transfer, the entire file is downloaded to the user's computer. When you stream the file, the user's computer is in constant contact with the server with the file.

Provide the viewer control over the multimedia element. For example, it is a good idea to include a skip button that allows the viewer to skip the introductory animation. Such a feature would be very popular with viewers viewing the content for a second time or third time.

Provide the view feedback about the status of download. Including a status bar that displays the percentage of download completed and the approximate time required to complete the task, will help in keeping the viewer engaged.

Display the multimedia element, particularly videos, in smaller windows.

## **Summary**

- Multimedia applications are now becoming very popular
- These applications include text, sound and video to produce high quality products
- Popular uses of multimedia are in the areas of education, entertainment and presentations
- Special hardware and software for multimedia are now available in the market
- There are a number of formats in which audio and video are used in multimedia applications.
- These formats are useful in including audio, video and other special effects in creating attractive web pages applications.
- Developing multimedia content is very challenging.

## **EXERCISES**

1. Create a HTML page with text, audio, images and video.
2. Note the formats of the audio, images and video files provided in the accompanying CD.

# CHAPTER 9

## PRESENTATION

### 9.1 Introduction

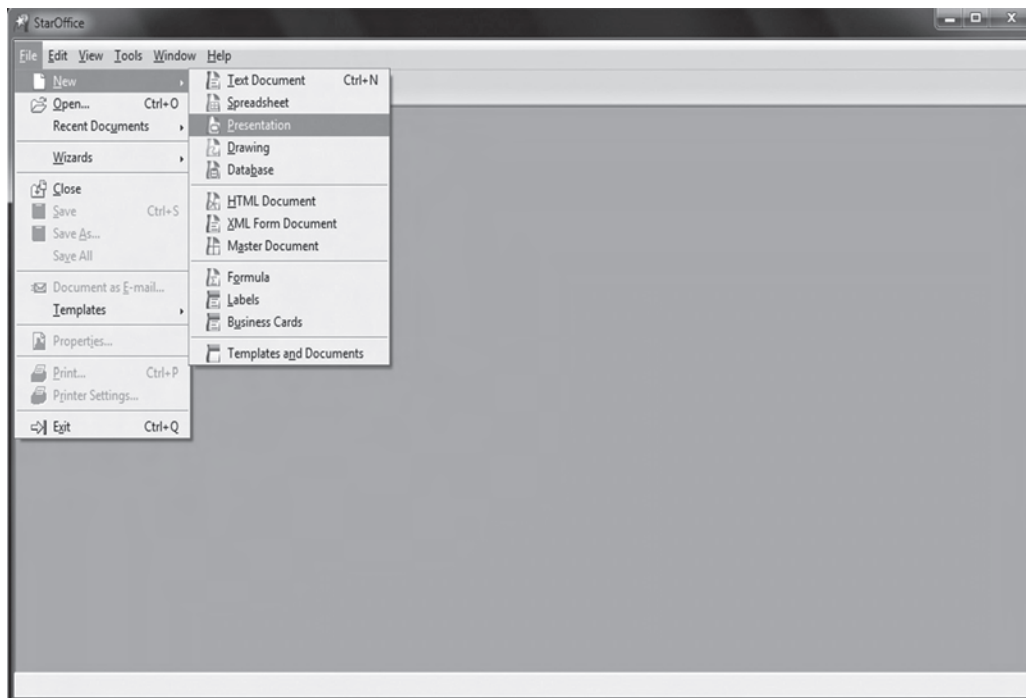
Among the many tools that are used to communicate to an audience, presentations are among the most powerful and effective ones. In a presentation, you can include a variety of items such as text, audio, hand-drawn images and videos. You can further enhance the appearance of the presentation by inserting charts and tables, adding background images, slide animation and transition effects.

StarOffice Impress is the StarOffice application that allows you create attractive and professional-looking presentations. The application provides various tools that allow you to save and modify presentations, print them as handouts, and view presentations as slide shows or HTML pages. It also allows you to import and modify Microsoft PowerPoint presentations.

### 9.2 A Basic Presentation

You can create a presentation in StarOffice using any of the following several different ways.

You can create a new Impress presentation from within any StarOffice application, by choosing **File** → **New** → **Presentations** as shown in figure 9.1.

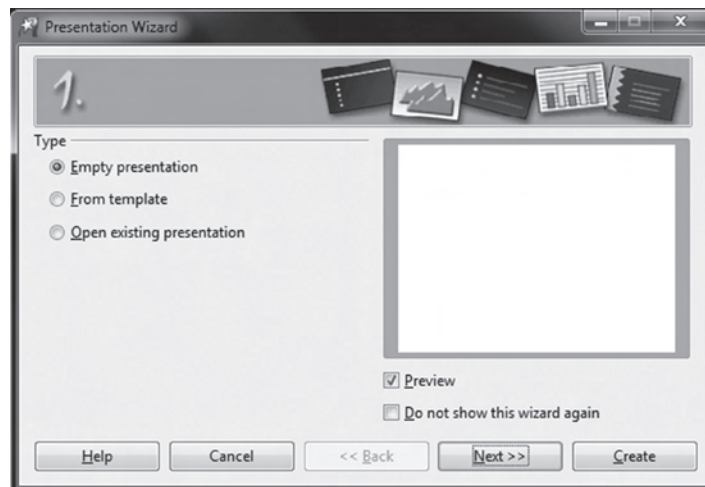


**Fig 9.1 Creating a Presentation from any StarOffice Application**

You can also create a new presentation by launching Star Office Impress. You can do so by choosing **Start** → **All Programs** → **StarOffice 8** → **StarOffice Impress**.

### 9.2.1 Creating a Presentation using the Presentation Wizard

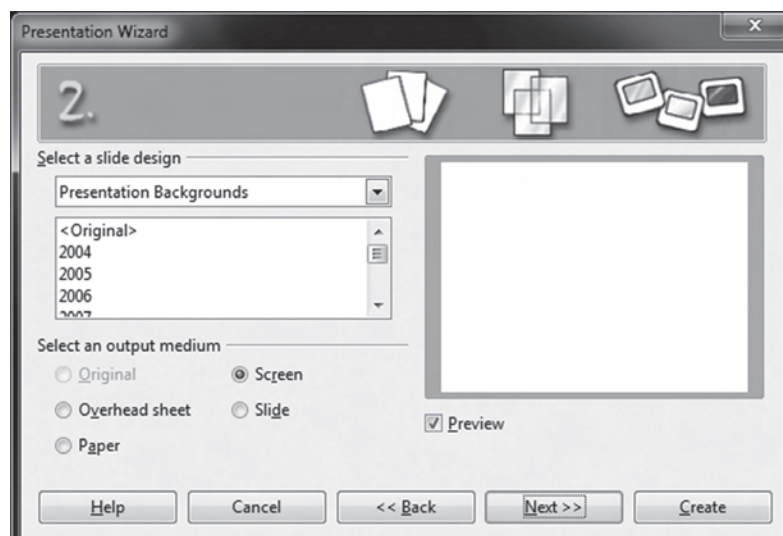
When you create a new presentation, the Presentation Wizard automatically starts as shown in figure. 9.2. This wizard will guide you through the steps of creating a presentation.



**Fig 9.2 Choosing the Type of the Presentation**

The first page of the Presentation Wizard displays three options that allow you to create an empty presentation, use a pre-created template to create the presentation, or open an existing presentation. For the purpose of this activity, let's select **Empty presentation** and click **Next** to proceed to the second page of the wizard.

The second page of the Presentation wizard appears as shown in figure. 9.3.



**Fig 9.3 Choosing the Background and Output Medium of the Presentation**

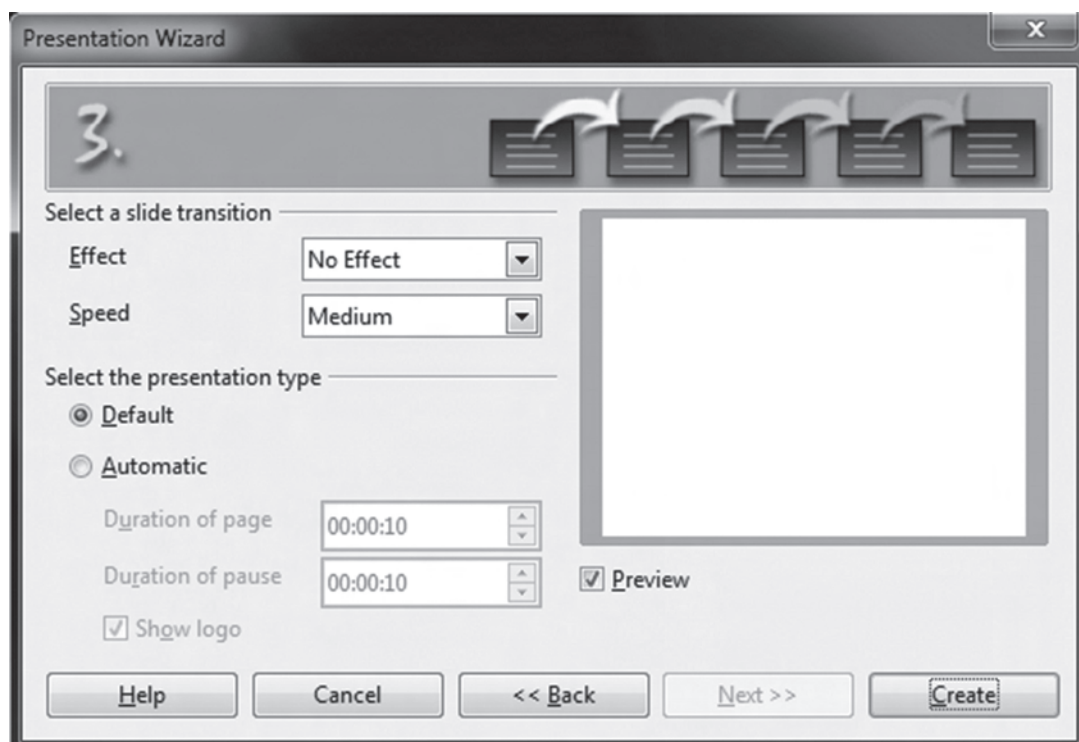


This page allows you to select a background for all the slides in the presentation. A drop-down list on the left provides three categories of presentation backgrounds for you to choose from. A list box below the background category displays different backgrounds in that category. A Preview box on the right allows you to view a preview of the slide with the selected background. If you do not see a preview, make sure the **Preview** check box is checked.

As the name indicates, the **Select an output medium** section in the lower half of the page allows you to specify the final output medium for the presentation.

Choose a suitable background for the presentation and click the **Next** button to proceed. Note that the background chosen can be changed later if necessary.

The third page of the Presentation wizard appears as shown in figure. 9.4.



**Fig 9.4 Choosing the Slide Transition and Type**

This page of the wizard allows you to specify the transition effects to be used in the presentation. The **Select a slide transition** section at the top of the page allows you to choose the transition effect and speed in the slides.

The **Select the presentation type** section of the page allows you to specify if the slide transition should be manual or automatic. The **Default** option allows you to manually control the transition of slides using options such as mouse clicks. The **Automatic** option allows you to specify the duration for each slide and will automatically flip through the slides at the specified speed.

The **Duration of page** spin box allows you to specify the duration for the slides and the **Duration of pause** spin box allows you to specify the duration for which the presentation will pause before it starts again.

Checking the **Show Logo** check box will display the message, “Created with StarOffice” during the pause between each presentation.

As on the previous page of the presentation, the preview box on the right allows you to view a preview the selected options.

Now, click **Create** to create the presentation. The StarOffice 8 Impress environment with one slide appears as shown in figure.9.5.

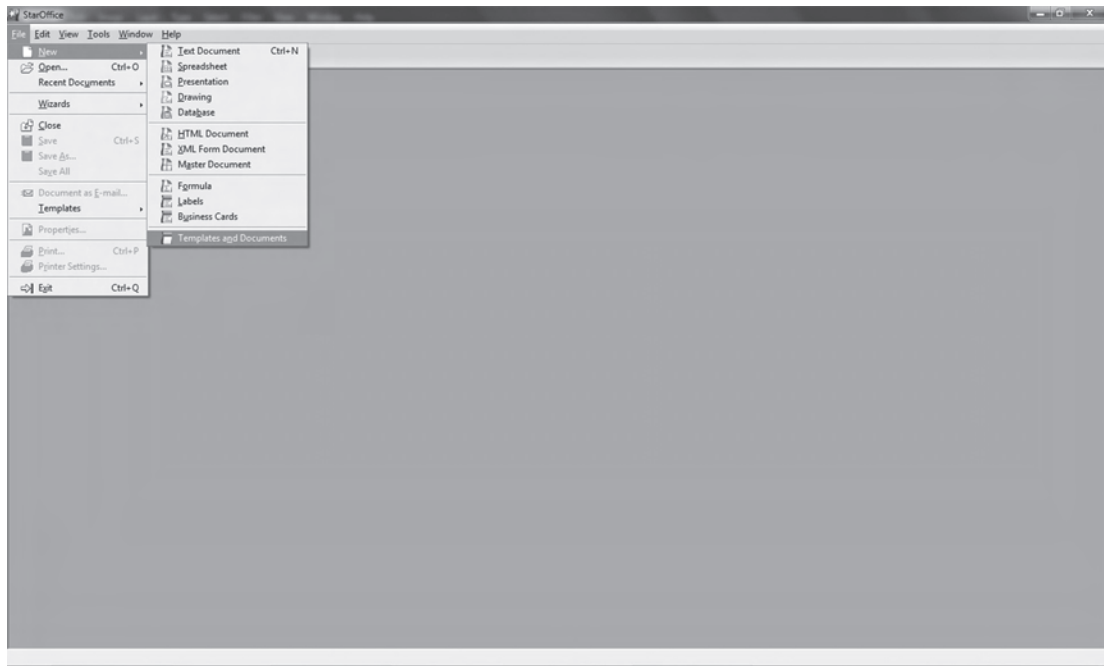


**Fig 9.5 StarOffice Impress Presentation Window**

### **9.2.2 Creating a Presentation without using Presentation Wizard**

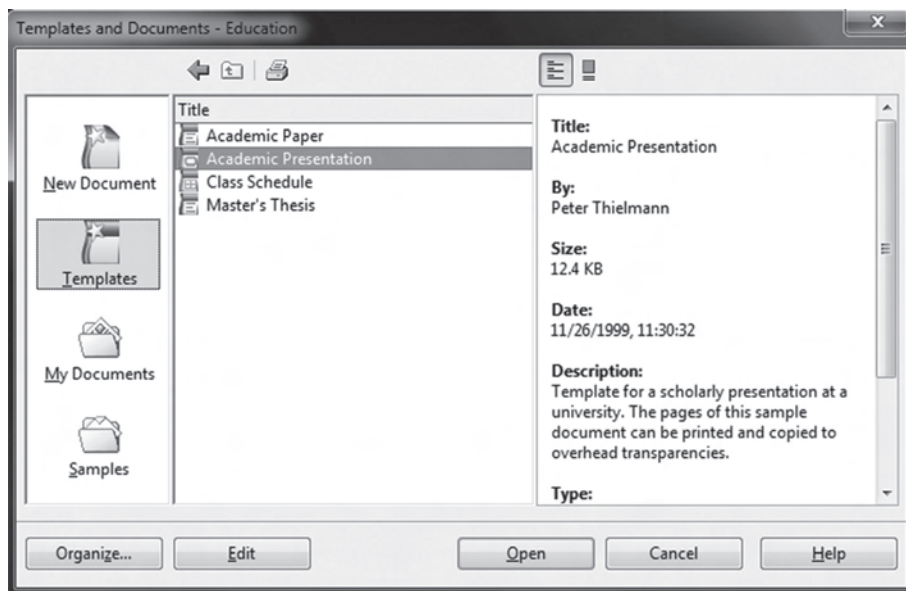
StarOffice Impress provides several default presentation formats. The easiest way of creating a presentation is to use one of these formats and then modify it to suit your requirements. The advantage of using these formats is that they already contain slides with content outlines that you can populate to quickly build presentations.

To create a new presentation using a template, choose **File → New → Templates and Documents**, or press **Shift+Ctrl+N**. The window appears as shown in the figure. 9.6.



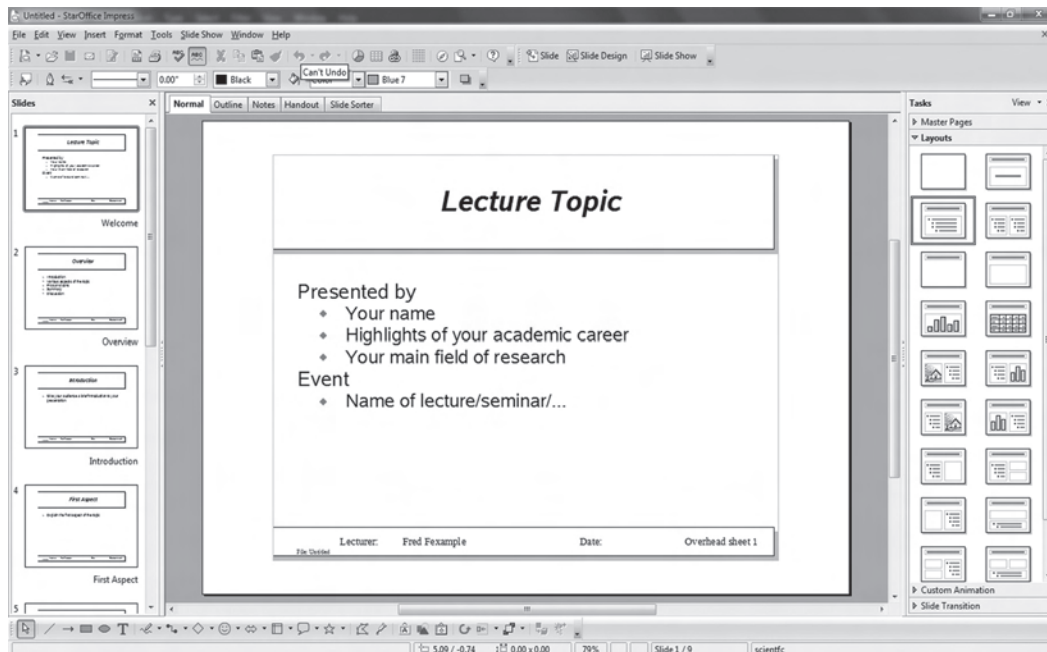
**Fig 9.6 Choosing Templates and Documents Window**

- In the **Templates and Documents** dialog that now appears, click the **Templates** icon on the left and then double-click **Education**. Double-click one of the templates, such as **Academic Presentation**. This is shown in figure 9.7.



**Fig 9.7 Choosing Academic Presentation from Templates Pane**

Click **Open** to open the presentation. The selected presentation format will open in the Impress window as shown in figure. 9.8.



**Fig 9.8 Selecting Academic Presentation from Templates Pane**

### 9.2.3 The StarOffice Impress Presentation Window

Let's now take a closer look at the StarOffice Impress presentation window displayed in figure. This window is very similar to the other application windows of StarOffice. It has a menu bar at the top of the window. Below the menu bar, one or more toolbars display shortcut icons for most of the frequently used options of StarOffice Impress. The status bar at the bottom of the screen displays status information such as the number of slides.

The middle pane displays the slides in the presentation. As you can see, this section contains three panes:

The Slides pane on the left displays a thumbnail image of the slide and allows you to rename, delete, or rearrange them.

The pane in the center allows you to view the slides in the presentation. This pane displays five tabs at the top that allow you to view the presentation in different formats.

**Normal view:** Allows you to create and edit slides

**Outline view:** Allows you to reorder slides, edit slide titles and heading.

**Notes view:** Allows you to add notes to the slides or view any existing notes for the slide. Notes are typically used by a presenter to add additional information to a slide.

**Handouts view:** Allows you to scale the slides so that several slides can fit into a page. This view is typically used when the presentation is to be printed and distributed as a handout.

**Slide Sorter view:** Allows you to view miniature images of all the slides in the presentation. This view is also used to rearrange slides.

The Tasks pane on the right displays four pages that allow you to specify the master slide, layout, transitions, and animation effects for the objects on your slides.


**Master Page:** This page can be used to specify basic background information that needs to be included in all the slides. For example, you can insert a company logo to the master slide and it will appear in all the slides.

**Layouts:** This page displays the various layouts provided by StarOffice Impress. You can choose a layout while creating a new slide.

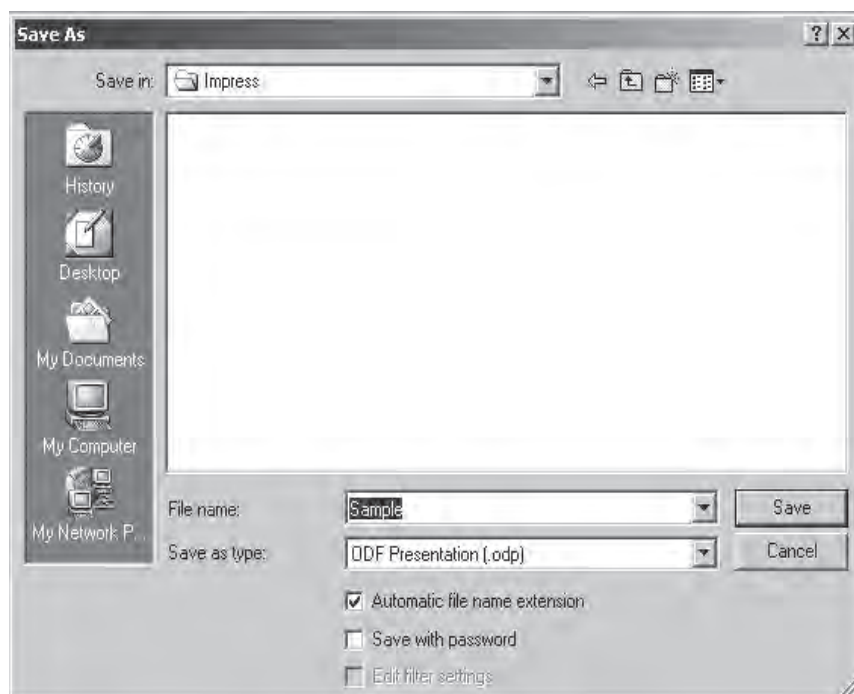
**Custom Animation:** This page displays various options that allow you to add or modify animation effects to elements of a slide.

**Slide Transition:** This page displays various transition effects that can be attached to a slide along with other options that allow you to control the transition of the slides. Note that you can have a different transition for each slide in the presentation.

#### 9.2.4 Saving a Presentation

To Save a Presentation choose **File→ Save** or click  icon. If you have not saved the presentation before, the **Save As** dialog box opens as shown in the figure.9.9. In the File name text box, enter the name of the presentation. Click **Save**.


Tip: You can also press Ctrl + S to save your work.



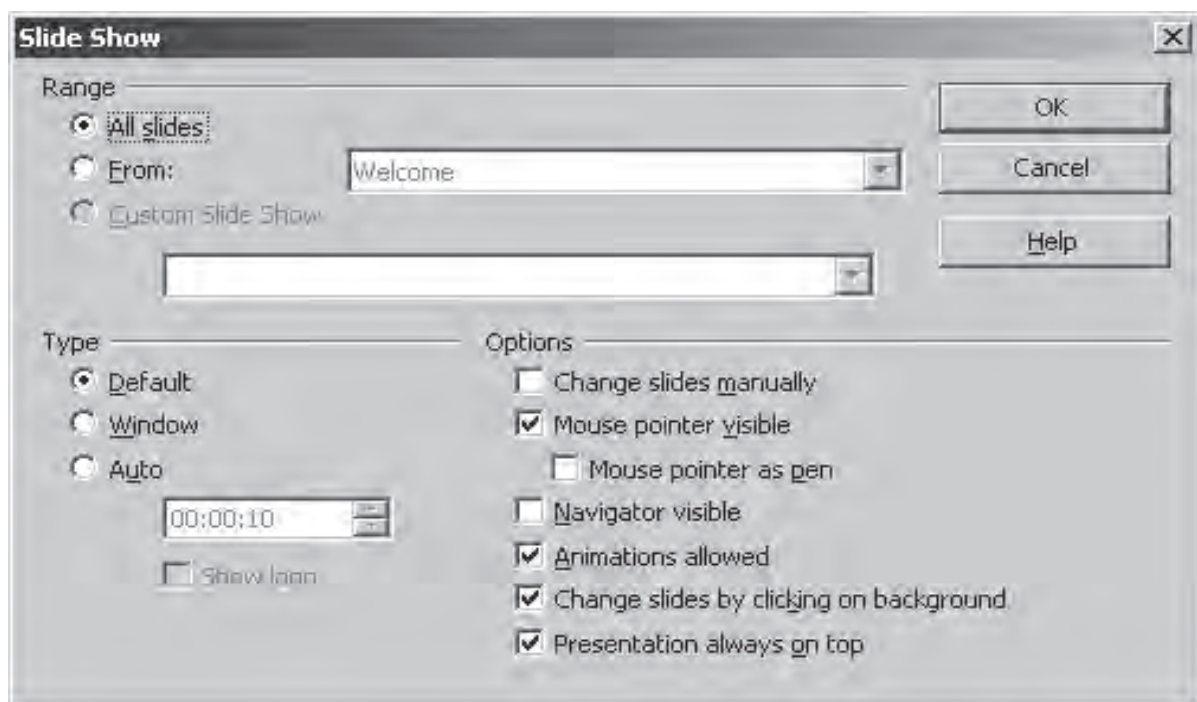
**Fig 9.9 Save as dialog box**

## 9.3 Managing a Presentation

### 9.3.1 Starting Presentation

To start a presentation, click the **Slide Show** icon  on the **Presentation** Toolbar or choose **Slide Show → Slide Show** or press **F5**. The on-screen presentation starts automatically in the full-screen mode. If you click once, the second slide will be displayed (preceded by the defined transition effect). After the last slide, you will see a black slide. End the presentation by pressing the Escape key (also used to stop the presentation before the end).

Presentation settings can be adjusted under **Slide Show → Slide Show Settings**. A dialog box appears as shown in the Fig 9.10.



**Fig 9.10 Slide Show Settings**

Select **All slides**, if you want to display all the slides else select **From** option and select the page from which you want to display.

Select **Custom Slide Show**, to run a custom slide show in the order that you defined in **Slide show → Custom Slide show**.

### 9.3.2 Inserting, Deleting and Renaming Slides

To insert a slide, click the **Slide** button in the Presentation toolbar or choose **Insert → Slide** from the menu bar. The new slide uses the page layout of the previous slide.



If you want to delete a slide, select the slide, which you want to delete in the Slides Pane and press **Delete** key or right click on the slide, and choose **Delete Slide**.

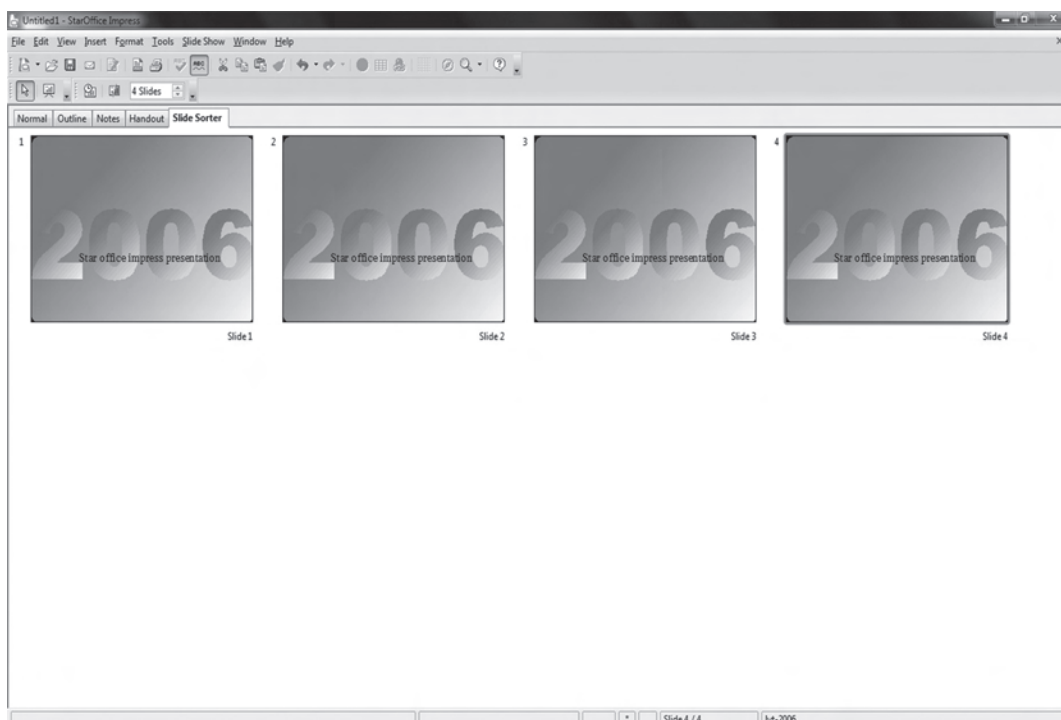
To rename a slide, select the slide, which you want to rename and choose **Slide** → **Rename** Slide or right click on the slide and choose **Rename Slide**. A dialog box appears as shown in the figure 9.11.



**Fig 9.11 Renaming a Slide**

### 9.3.3 Changing Slide Order

To change the Slide order, click the **Slide Sorter** in the **Switching Presentation view** tab. A slide sorter window opens as shown in figure 9.12. In this mode, all slides in the presentation document are displayed in reduced size. Click the slide you want to move and drag it with the mouse to the desired position. A vertical black bar indicates the position where the slide will be inserted.



**Fig 9.12 Slide Sorter window**



## 9.3.4 Inserting Pictures, Objects, Sound and Video

### 9.3.4.1 Inserting Pictures



To Insert a Picture in a slide, choose **Insert → Picture → From File** or click the Insert Picture  icon from the Insert toolbar. An open dialog box appears. Choose the desired picture to be inserted from the open dialog box. figure.9.13 shows a picture of a flower that has been inserted in a slide.



Fig 9.13 Inserting a Picture

### 9.3.4.2 Sound and Video

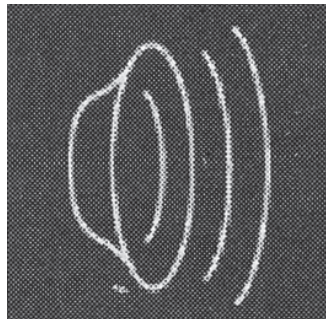
To Insert a Movie in a slide, choose **Insert → Movie and Sound** or else click the Insert Movie and Sound icon  from the Insert toolbar.

The Insert Movie and Sound dialog box appears as shown in figure.9.14



Fig 9.14 Insert Movie and Sound dialog box

The Inserted sound file will be displayed in your slide as shown in figure 9.15



**Fig 9.15 Sound file display in slide**

To play a movie or sound file in an Impress presentation









1. Open the slide that contains the movie or sound file.
2. Click the object icon for the movie or sound file on the slide.
3. Click **Play** on the **Media Playback** toolbar.

The Media Playback toolbar is shown in figure 9.16.



**Fig 9.16 Media Playback Toolbar**

You can also use the Media Playback Bar to pause, stop, loop, as well as to adjust the volume or to mute the playback of the file. The current playback position in the file is indicated on the left slider. Use the right slider to adjust the playback volume. For movie files, the bar also contains a list box where you can select the zoom factor for the playback.

	Play icon	- Plays the current file.
	Pause icon	- Pauses or resumes the playback of the current file.
	Stop icon	- Stops the playback of the current file.
	Repeat icon	- Plays the file repeatedly.
	Position slider	- Moves to a different position in the file.
	Mute icon	- Turns sound off and on.
	Volume slider	- Adjusts the volume.
	Zoom	- Adjusts the size of the movie playback.


StarOffice Impress enables Media Player where you can preview movie and sound files as well as insert these files into the current presentation slide. The Media Player supports many different media formats.

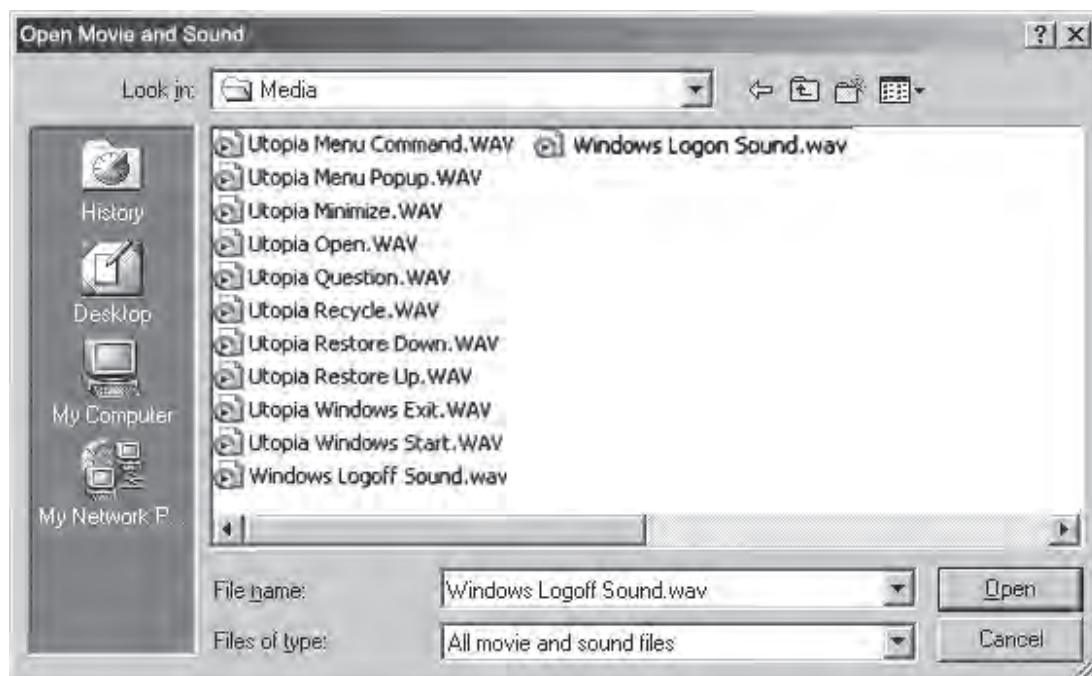
To open Media Player window Choose **Tools → Media Player**

The Media Player window is now displayed as shown in figure 9.17.




**Fig 9.17 Media Player**

Click **Open** icon  which is displayed at the lower left corner of the media player window. The Open Movie and Sound dialog box appears as shown in figure 9.18



**Fig 9.18 Open Movie and Sound dialog box**

Select a video or sound file and view the preview.

Click the **Apply** icon  to apply the selected movie or video file into your presentation.

Close the media player window

Start slide show by choosing **Slide Show → Slide Show** or press the F5 key.

Figure 9.19 displays a sample movie file played during a slide show.



**Fig 9.19 Movie file played during slide show**

#### **9.3.4.3 Inserting Objects**

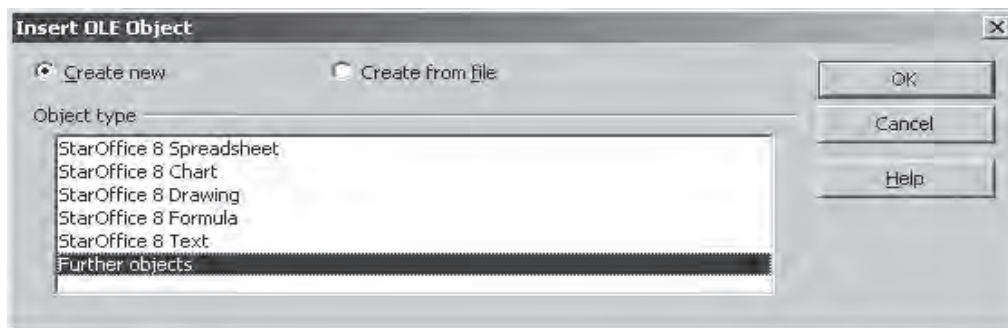
StarOffice Impress provides tools for inserting objects like charts, formula, etc. in a presentation. To insert an object in a slide, choose an object from **Insert → Object**.

#### **9.3.4.4 Insert Formula**

This is used for inserting a formula into the presentation for performing calculations.

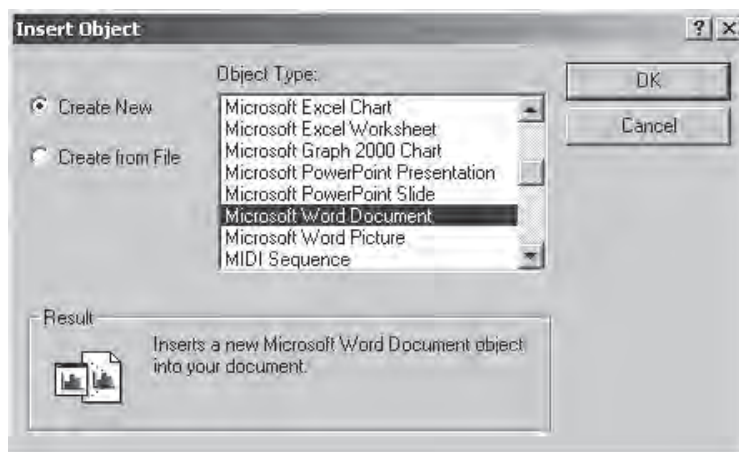
#### **9.3.4.5 Insert OLE Object**

This is used to import objects from other application into a presentation. For example, StarOffice spreadsheet, formulae, text and even Microsoft application objects can be inserted. Figures 9.20 and 9.21 show how to insert a Microsoft Word Document.



**Fig 9.20 Inserting OLE Object**

If you want to insert a Microsoft Word Document, choose **Further** objects and **Create New** option in the **Insert OLE Object** dialog box and click OK. In the **Insert Object** dialog box, select **Microsoft Word Document** as object type and click **Ok**.



**Fig 9.21 Object Type Selection window**

Microsoft Word Application is opened automatically. Type your information in MS Word and close it. StarOffice Impress automatically saves the document and inserts it into the slide. figure.9.22 shows the inserted MS Word document.



**Fig 9.22 OLE Object in Presentation**



### 9.3.4.6 Insert Applet

This is used to import Applets into the presentation.

## 9.3.5 Slide Transitions, Effects and Animation

### 9.3.5.1 Automatic Slide Transition

The easiest way to assign slide transition effects to slides is in **Slide View**. Choose **View → Toolbars → Slide View**. Select the number of slides to be listed in the **Slides Per Row** spin box. The number of slides in the Slides view changes according to the number specified in the **Slides Per Row** spin box as shown in the figure.9.23. The Slides view displays the slides in certain numbers specified in the **Slides Per Row** spin box as shown in the figure.9.24.



Fig 9.23 Slide View Floating window

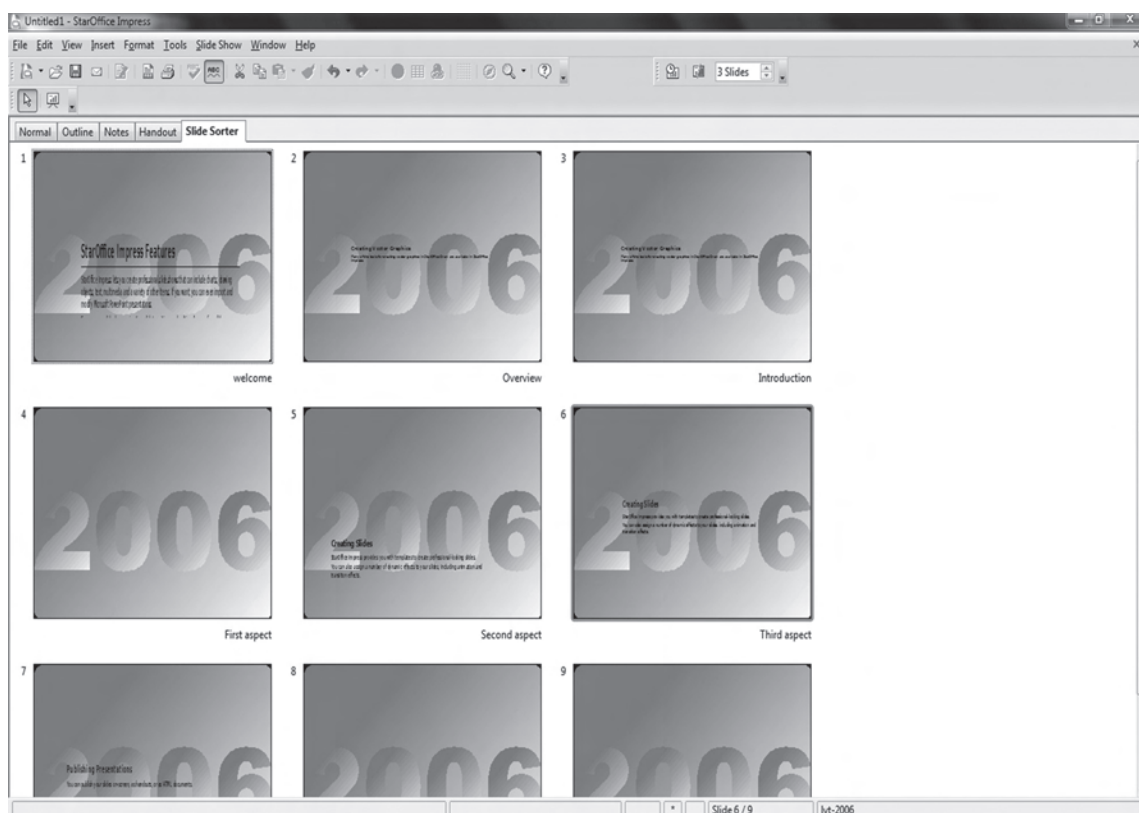


Fig 9.24 Slide View window

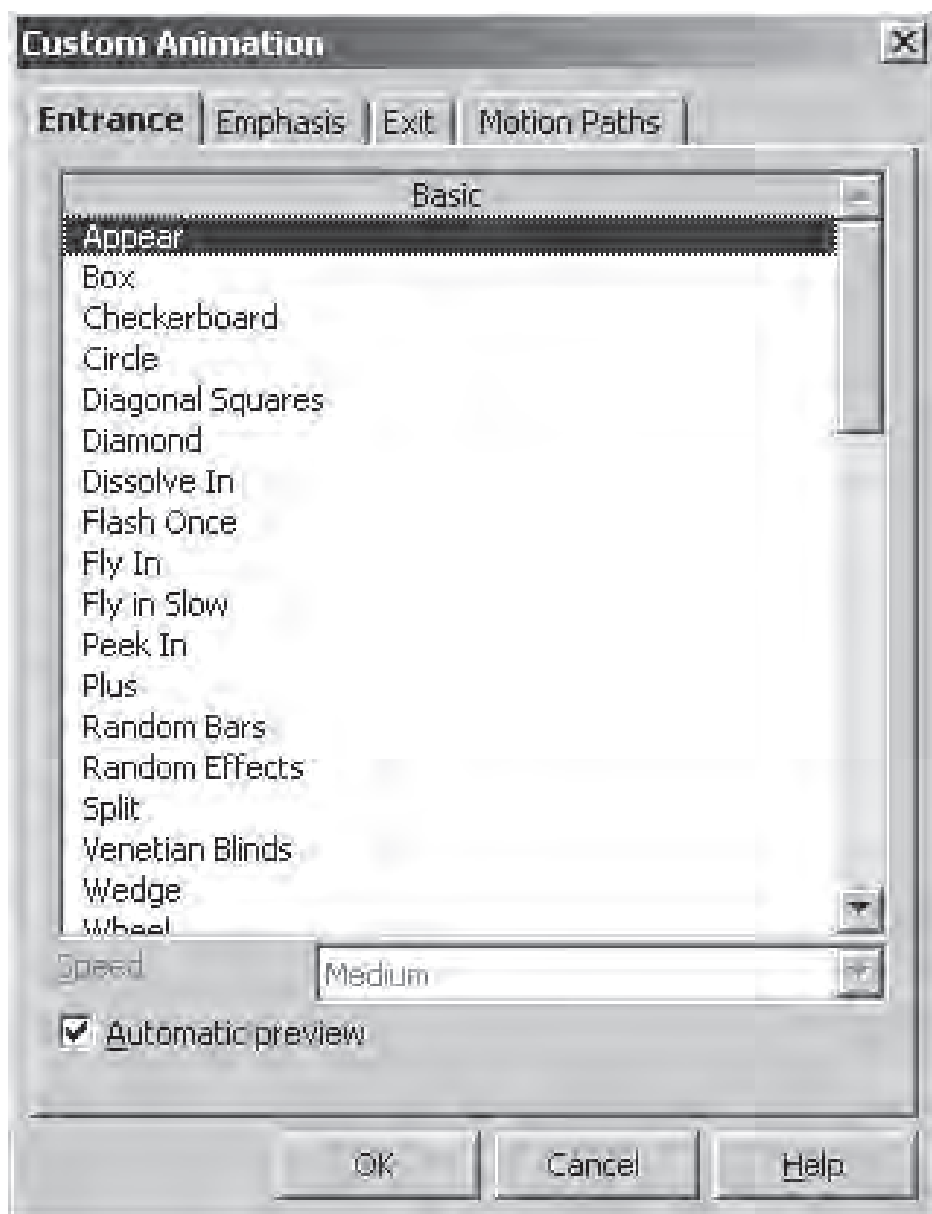
### 9.3.5.2 Effects for Objects

The objects in your slide can have various effects, for example, they can roll into your presentation from the left side, the text can be slowly drawn onscreen, and so on.

All these effects can be assigned using the **Normal View (View → Normal View)**.

Choose **Custom Animation** from **Slide Show → Custom Animation**. First you must select the object to which you want to apply the effect.

Click **Add** in **Modify Effect**. The Custom Animation window as show in the figure.9.25 appears.



**Fig 9.25 Custom Animation window**



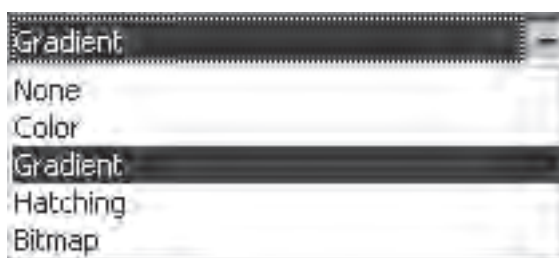
Then select the desired effect. Click the **Ok** button. You can also have your object **fade in** (in the **Custom Animation** window, on the **Entrance** tab under **Special** category) and additionally, fade out (in the **Custom Animation** window, on the Exit tab under **Special** category) per mouse-click.

### 9.3.5.3 Changing Slide Background

You can change the background color or the background fill of the current slide or all of the slides in your document. For a background fill, you can use hatching, gradient, or bitmap image as shown in the figure 9.26.

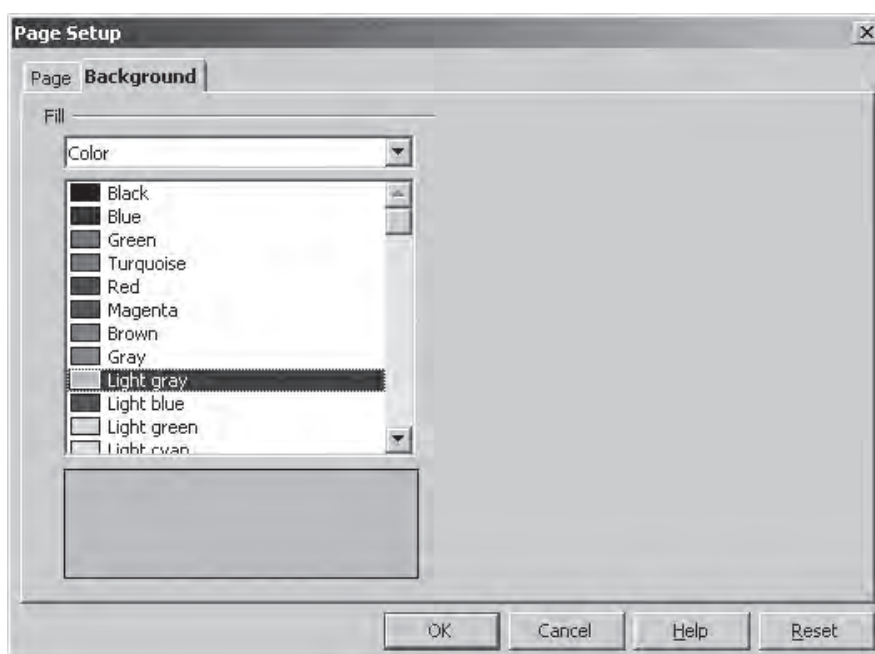
Choose **Format** → **Page** → **Background**. Select the background fill from the following options.

Color  
Gradient  
Hatching  
Bitmap Image



**Fig 9.26 Background Fill combo box**

In this example, Colour is selected as the **Fill** type. Note that the list box below the **Fill** combo box changes to a list of different colours as shown in the figure 9.27.



**Fig 9.27 Selecting Background Color**

In the Fill area, do one of the following:

- Select **Color**, and then click a color in the list.
- Select **Gradient**, and then click a gradient style in the list.
- Select **Hatching**, and then click a hatching style in the list.
- Select **Bitmap**, and then click bitmap image in the list.

Then, Click **Ok. Page Settings** dialog box appears as shown in figure 9.28. If you want to change the background fill for all of the slides, click **YES** when the 'Background setting for all pages' dialog box appears else click **NO** to change the background fill for a single slide.



**Fig 9.28 Background Fill Dialog Box**

#### **9.3.5.4 Applying a Slide Design to a Master Slide**

A master slide determines the text formatting style for title, outline and the background design for individual slides, or for all of the slides in a presentation. You can change the appearance of a master slide by applying a new slide design.

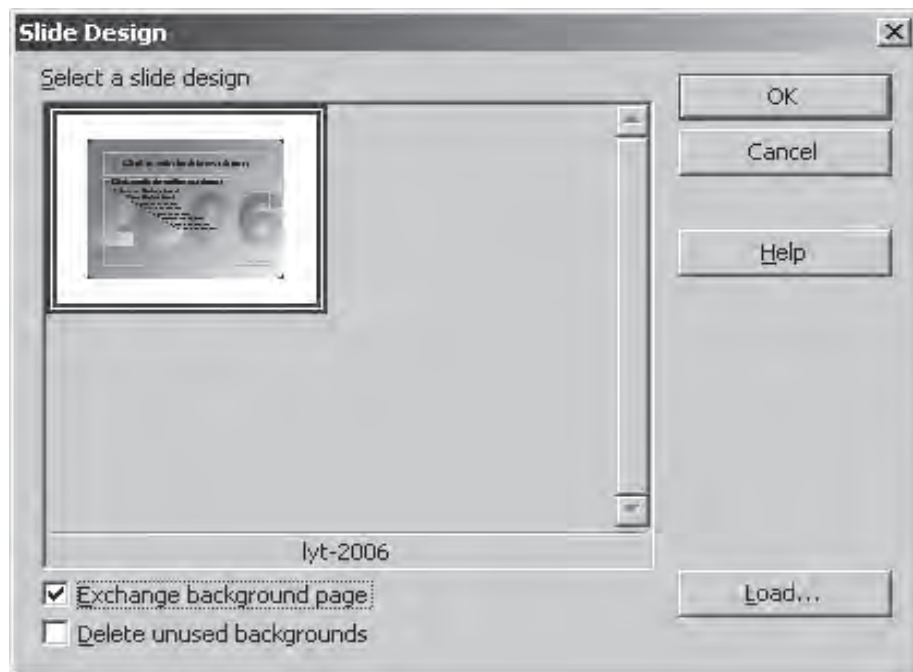
**To apply a new slide design:**

Select **Format → Slide Design**. A dialog box appears as shown in the figure 9.29.

**Do one of the following:**

To apply the slide design to all of the slides in your presentation, select the **Exchange background page** check box.

To apply the slide design to the current slide only, clear the **Exchange background page** check box.

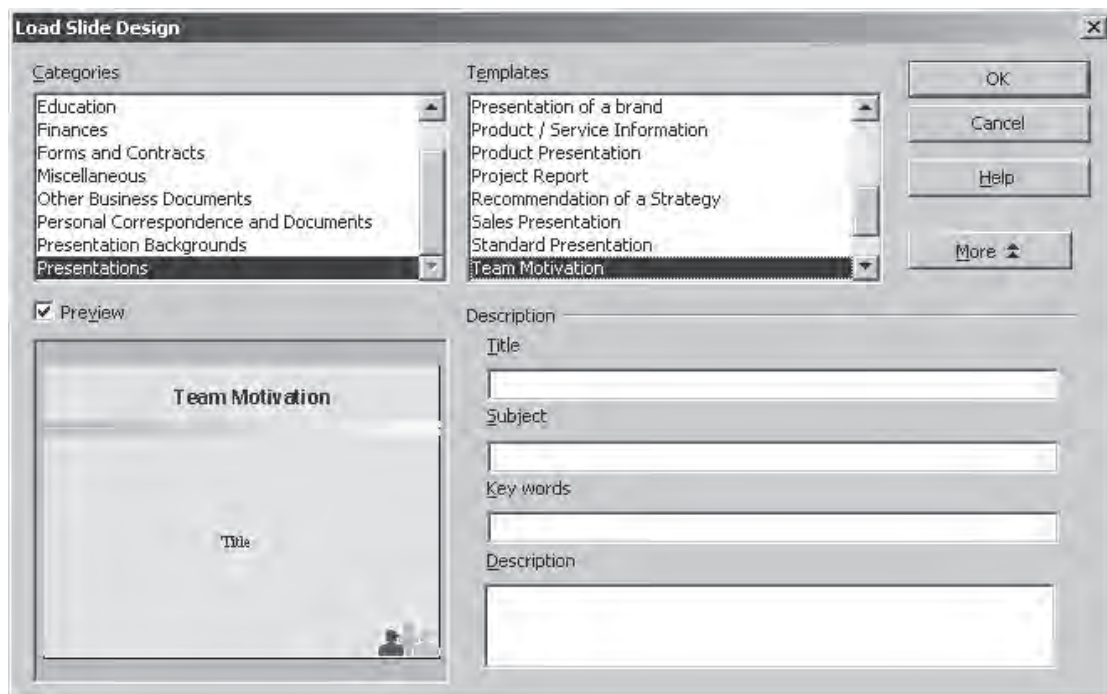


**Fig 9.29 Slide Design window**

Click **Load**. A window as shown in the figure.9.30 appears.

Under **Categories**, select a slide design.

Under **Templates**, select a template with the design that you want to apply. To preview the template, click **More**, and then select the **Preview** check box.



**Fig 9.30 Load Style Design window**

Click **OK** to see the changes in the master slide.

### 9.3.5.5 Presentation Styles

Along with the slide design you can also assign a whole set of Presentation Styles to your slides. Open the Style list to see a list of the pre-defined Styles. You can modify the existing Styles and you can create new Styles. The Styles Outline 1 through Outline 9 enables you to give the outlined headings and topics on your slides a uniform look.

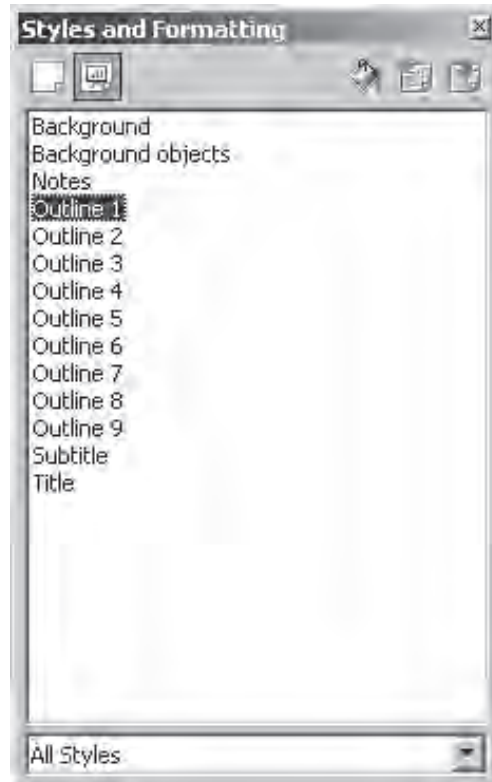
Open a new, empty presentation with a slide layout that allows outline levels. The slide layout called "Title", "Text" is suitable for this purpose.

**Note:** The slide layout you choose determines the number of outline levels. If you choose a slide layout, which does not enable any outline, you will have no access to outlines.

Activate the outline view mode by choosing menu **View → Outline View**.

Open the Style list either by choosing **Format → Styles and Formatting** or with the **F11** key.

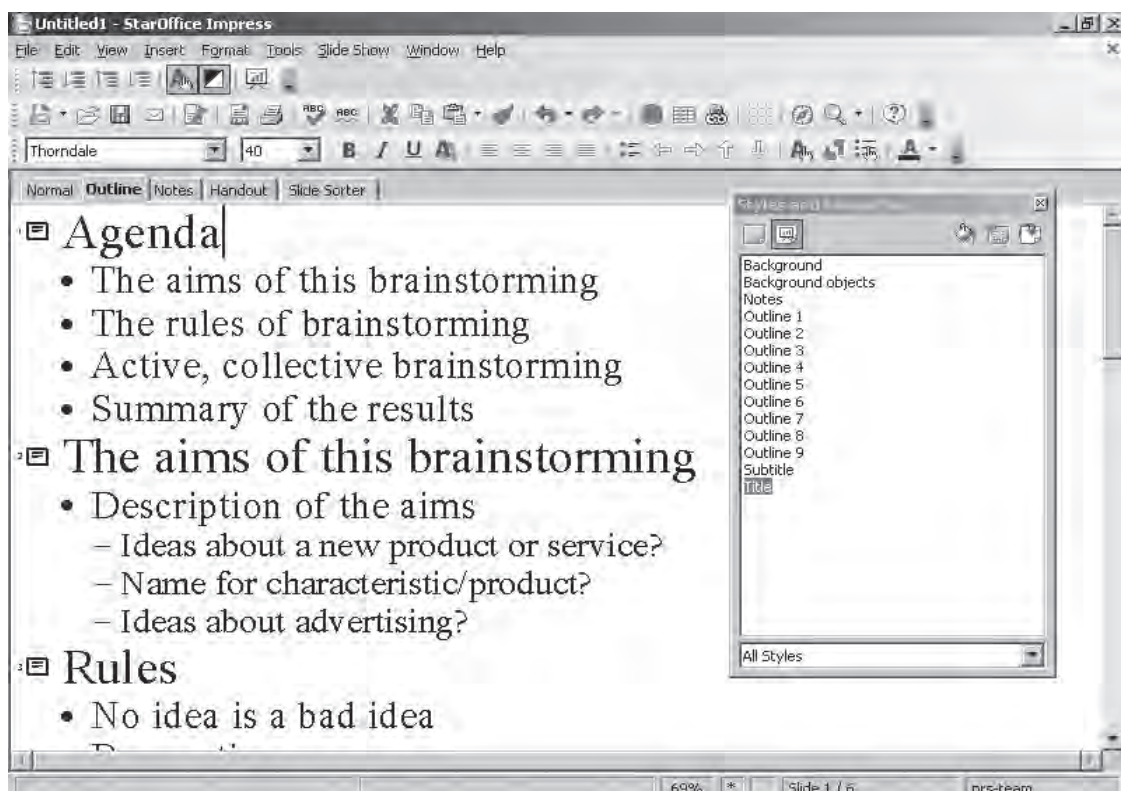
A window appears as shown in the figure 9.31.



**Fig 9.31 Styles and Formatting window**

- The cursor is now blinking next to the small icon for slide 1. Enter the desired text, for example First Page. This will be the title for slide 1.
- If you press the **Enter** key, a small icon for slide 2 appears in the next line of the Outline view. Enter a title for the second slide, for example, **Second Page**.
- Press the **Enter** key again. Now press the Tab key before you enter any text. By doing that, you ensure that the text you then enter here will be a subtitle on slide 2.
- You can also first start entering the text for the subtitle and then press the Tab key. In case, if you have already entered text in the line, you will see a message, which warns you that with this action, you will delete the existing text. To delete click Ok.
- Continue to enter more subtitles of the first level on page two. By pressing the Tab key, you move the line down one level and with Shift + Tab one level up. This enables you to even make a new slide out of a sub-topic.

The levels you create with these steps automatically contain the Presentation Styles title, outline 1, outline 2, and so forth. These Styles can be modified by having a new slide design assigned to them, for example, a new font, font size and font color. You can, of course, edit these designs (for example, with the Style list's context menu). The outline view is shown in figure 9.32.



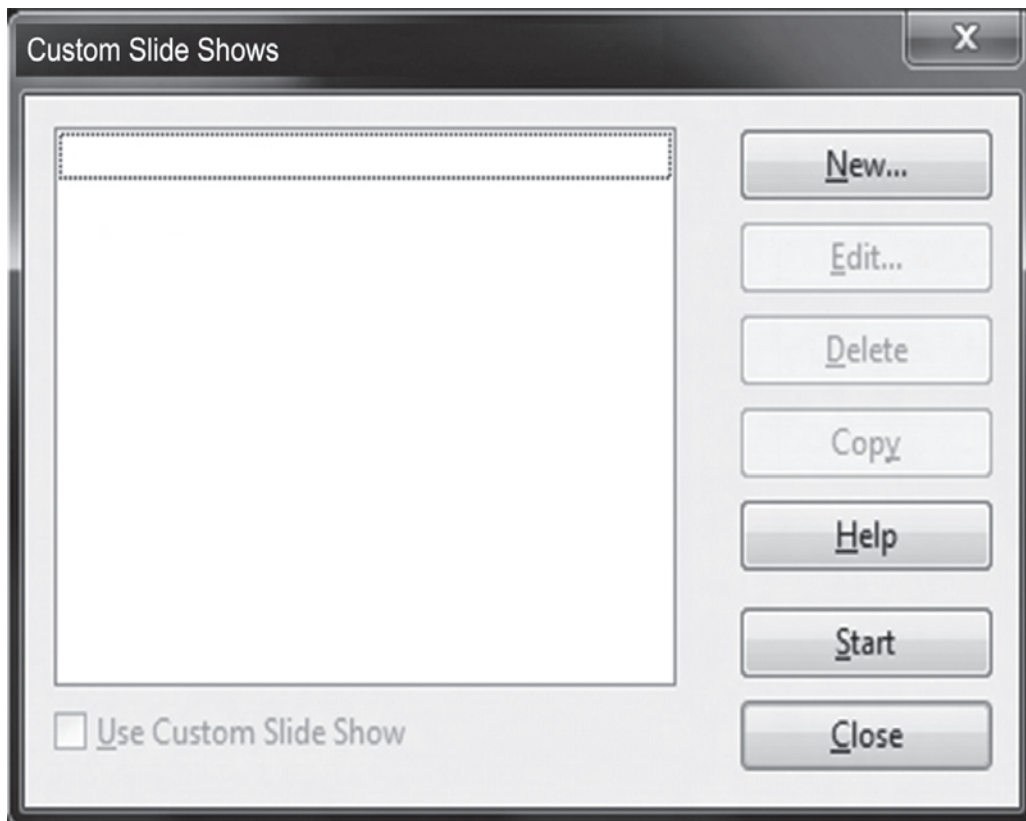
**Fig 9.32 Outline View of Presentation**

## 9.4 Customizing a Presentation

You can create as many custom slides shows as you want. StarOffice Impress also allows you to start slide shows from the current slide as well as hide slides during a slide show.

**To create a custom slide show:**

- Choose **Slide Show → Custom Slide Shows**, and then click **New**.
- Figures 9.33 and 9.34 show the **Custom Slide shows** dialog box.
- Enter a name for your slide show in the **Name** box.
- Under **Existing Slides**, select the slides you want to add to your slide show, and click the >> button. Hold down Shift to select a range of slides, or Ctrl to select multiple slides.
- You can change the order of the slides in your custom slide show, by dragging and dropping the slides under **Selected Slides**.
- Click **Ok**.



**Fig 9.33 Creating New Custom Slide show**





**Fig 9.34 Define Custom Slide show**

**To start a custom slide show:**

- Choose **Slide Show → Custom Slide Show**.
- Select the slide show you want to start from the list.
- Click **Start**.

**Note:** To start the selected custom slide show when you click the **Start** button check whether the **Use Custom Slide Show** check box is selected. This is clearly shown in figure.9.35.



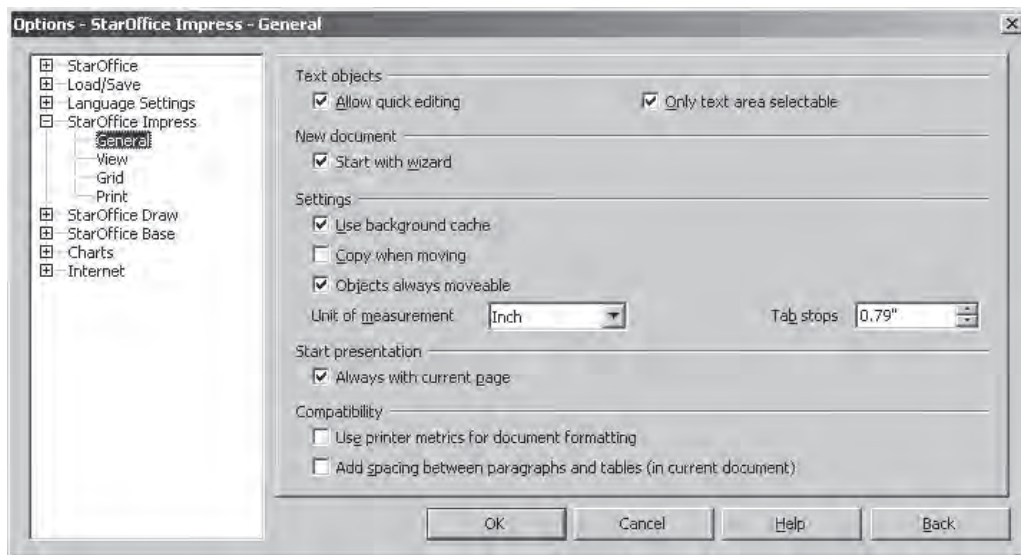
**Fig 9.35 Custom Slide show**



### 9.4.1 Options for Running a Slide Show

To start a slide show from the current slide:

1. Choose **Tools** → **Options** → **StarOffice Impress** → **General**. A window as shown in the figure 9.36 appears.
2. In the Start presentation area, select **Always with current page** check box.



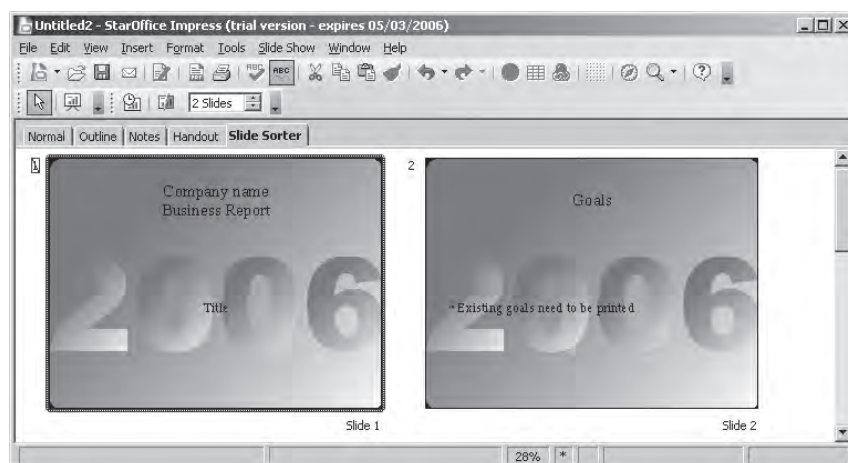
**Fig 9.36 General Settings in StarOffice Impress**

**Note:** Do not select this option if you want to run a custom slide show.

### 9.4.2 To hide a slide

- Select the slide(s) that you want to hide in the slide show.
- Choose **Slide Show** → **Show/Hide Slide**.

The Slide number is struck out, as shown in the figure 9.37 but it is not removed from your presentation.



**Fig 9.37 Hidden Slide in Slide View**

### 9.4.3 To show a hidden slide

Select the slide(s) that you want to hide from the **Slides Pane**. Choose **Slide Show → Show/Hide Slide** to show the slide in the slide show. This is shown in figure 9.38.

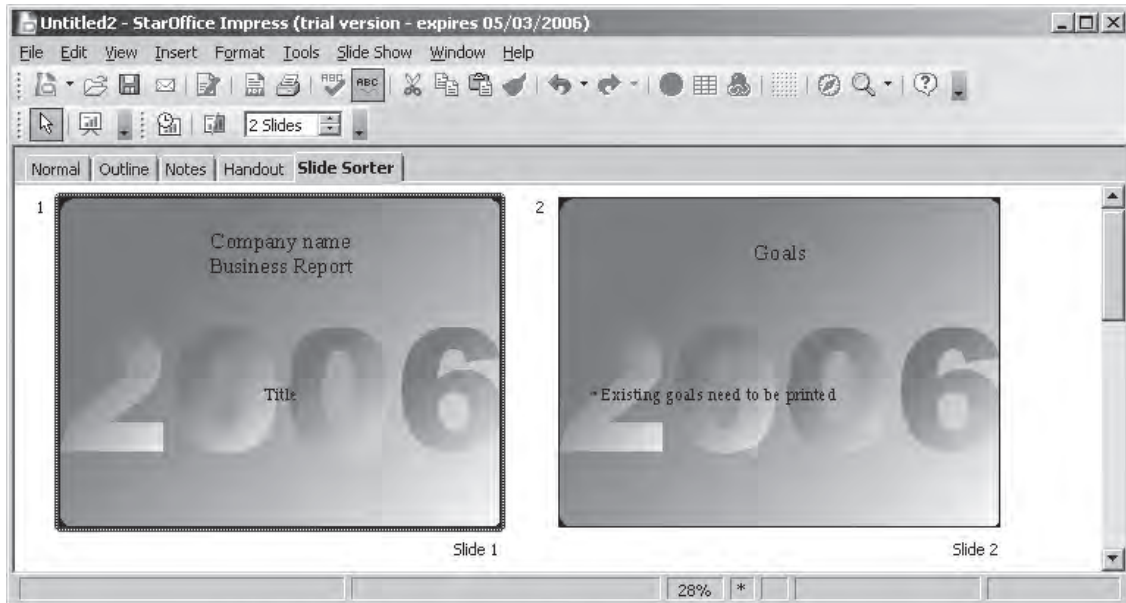


Fig 9.38 Shown Slide in Slide view

### 9.4.4 Rehearse Timings of Slide Changes

StarOffice assists you in defining the right rehearse timings for automatic slide changes. StarOffice records the display time for each slide, so the next time you play the show with automatic slide changes, the timing will be as recorded.

To record a show with rehearse timings:

- Open a presentation, and switch to **Normal View**.
- Start the show with the **Rehearse Timings** from **Slide Show** menu. You see the first slide and a timer in the bottom left corner as shown in the figure 9.39.
- When it is time to advance to the next slide, click the timer. Continue for all slides in your presentation.
- StarOffice has recorded the display time for each slide.
- If you want the whole presentation to auto-repeat, open the menu **Slide Show → Slide Show Settings**. A window appears as shown in the figure 9.40.
- Select **Auto** and then click **OK**.

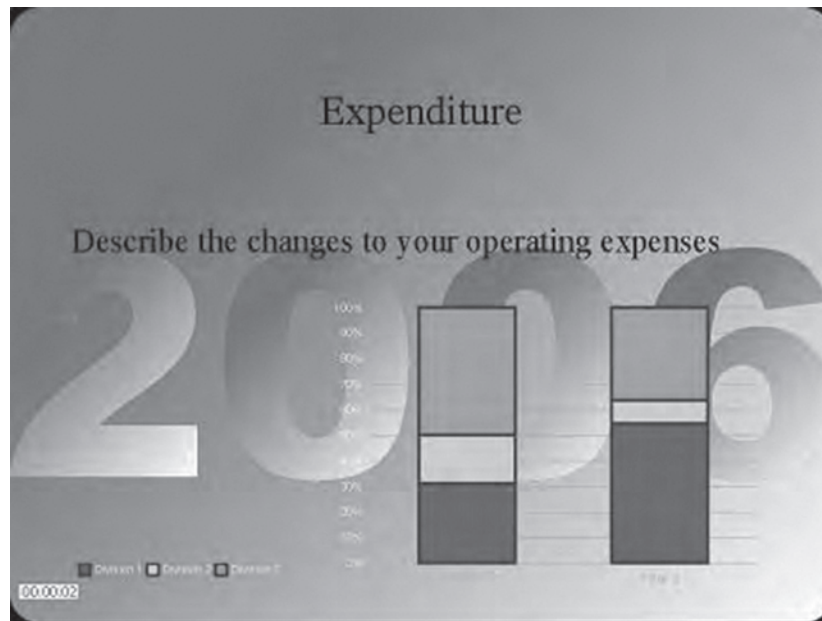


Fig 9.39 Rehearse Timings in Slide Show

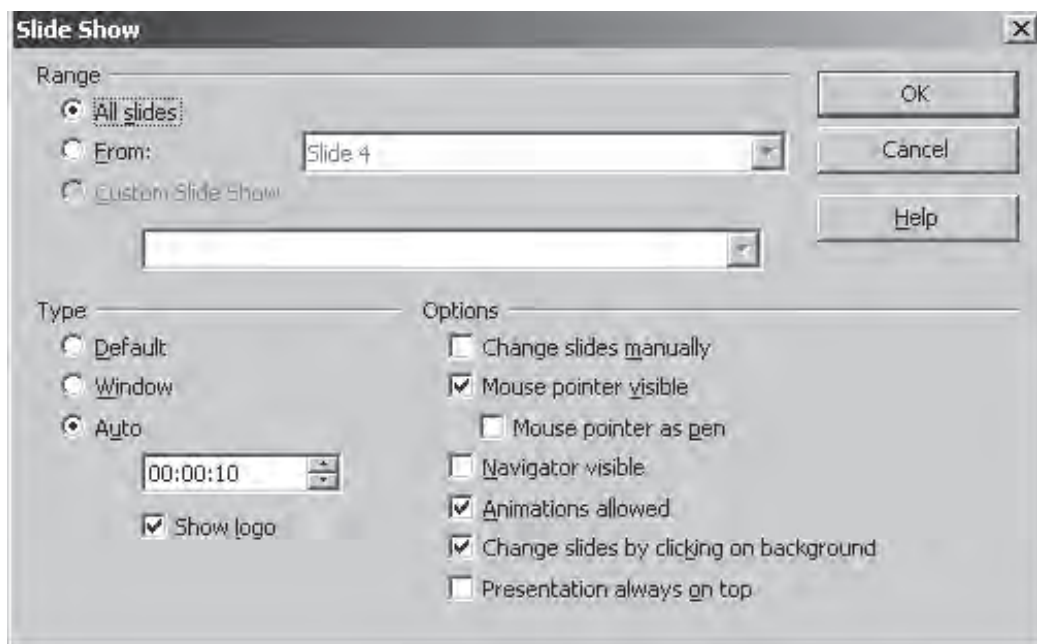
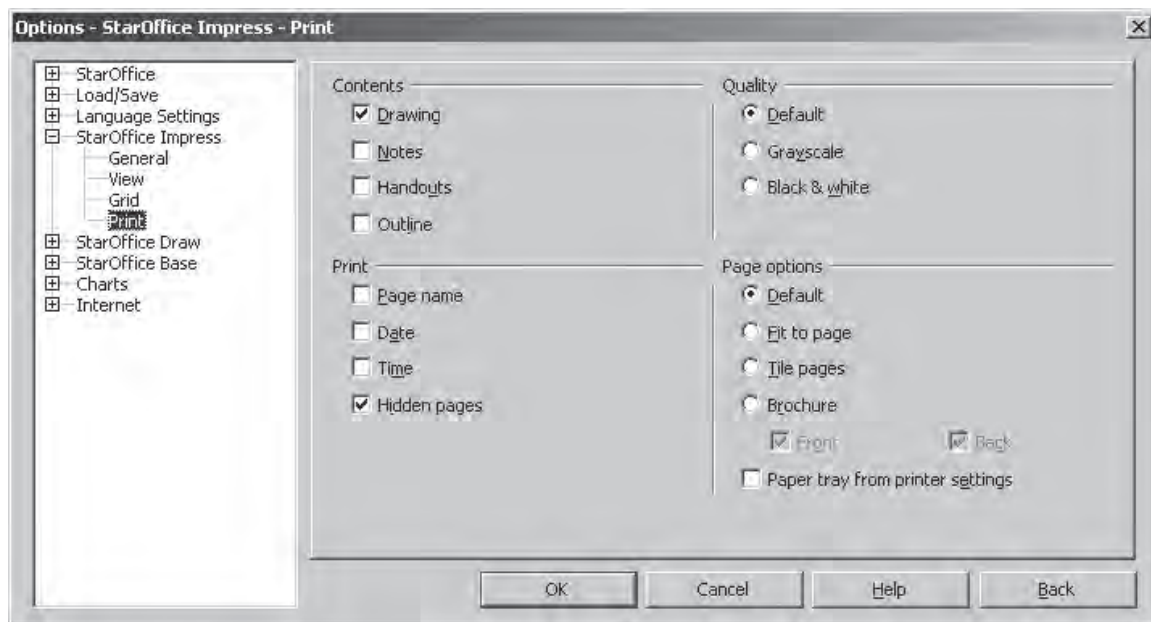


Fig 9.40 Slide Show Type

## 9.5 Printing Presentations

### 9.5.1 Default Printer Settings

To set the default printing options for StarOffice Impress, choose **Tools → Options → StarOffice Impress → Print**. A window as shown in the figure 9.41 appears. Any change made in this window is set as the default setting for the printer.

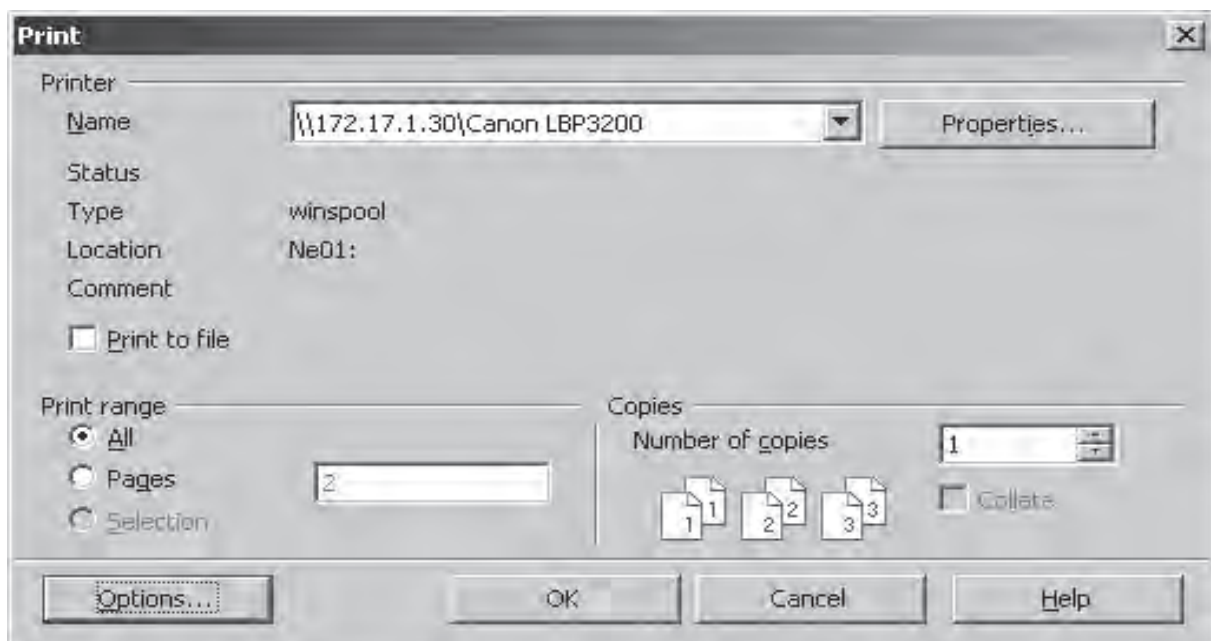


**Fig 9.41 Printer Options window**

#### 9.5.1.1 Setting printer options for the current presentation

- ★ Choose **File** → **Print**. (A window as shown in the figure 9.42 appears.)
- ★ Click **Options**, and then select the printer options.

These settings override the default printer options in **Tools** → **Options** → **StarOffice Impress** → **Print** for the current print job only.



**Fig 9.42 Printer Options window for Current Presentation**

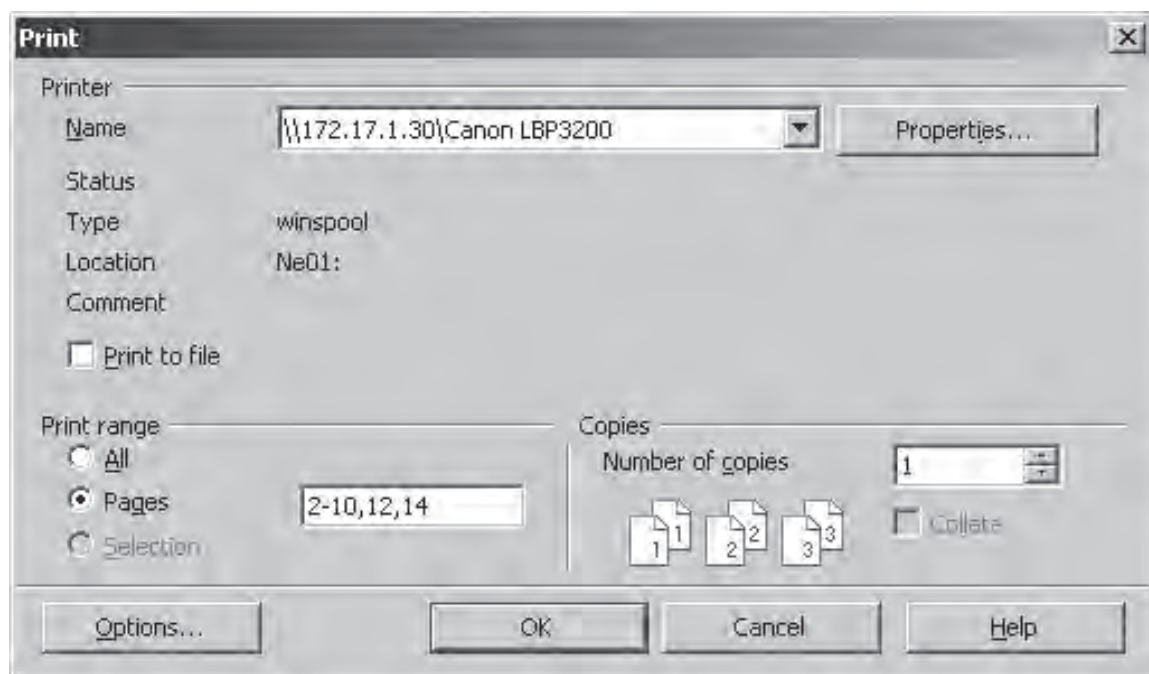
### 9.5.1.2 Printing a range of slides

- ★ Choose **File** → **Print**.
- ★ In the **Print range** area, click **Pages**.
- ★ Enter the numbers of slides to print in the **Pages** box, and then click **OK**.

For example to print the 3rd, 4th and 5th slide enter 3, 4, 5 or 3- 5 in the pages box as shown in the figure 9.43.

In StarOffice, you can print the current document by clicking the **Print File Directly** icon in the **Function Bar**. The document will be printed immediately according to the printer default settings, without a dialog appearing.

To print in StarOffice, choose **File** → **Print** or press **Ctrl + P**. The Print dialog opens. Use this to set your printing options, for example, whether to print a selected range of pages, individual pages or all of the pages in the document.



**Fig 9.43 Printing Options for printing a range of slides**

### 9.5.1.3 Printing a Slide to Fit a Paper Size

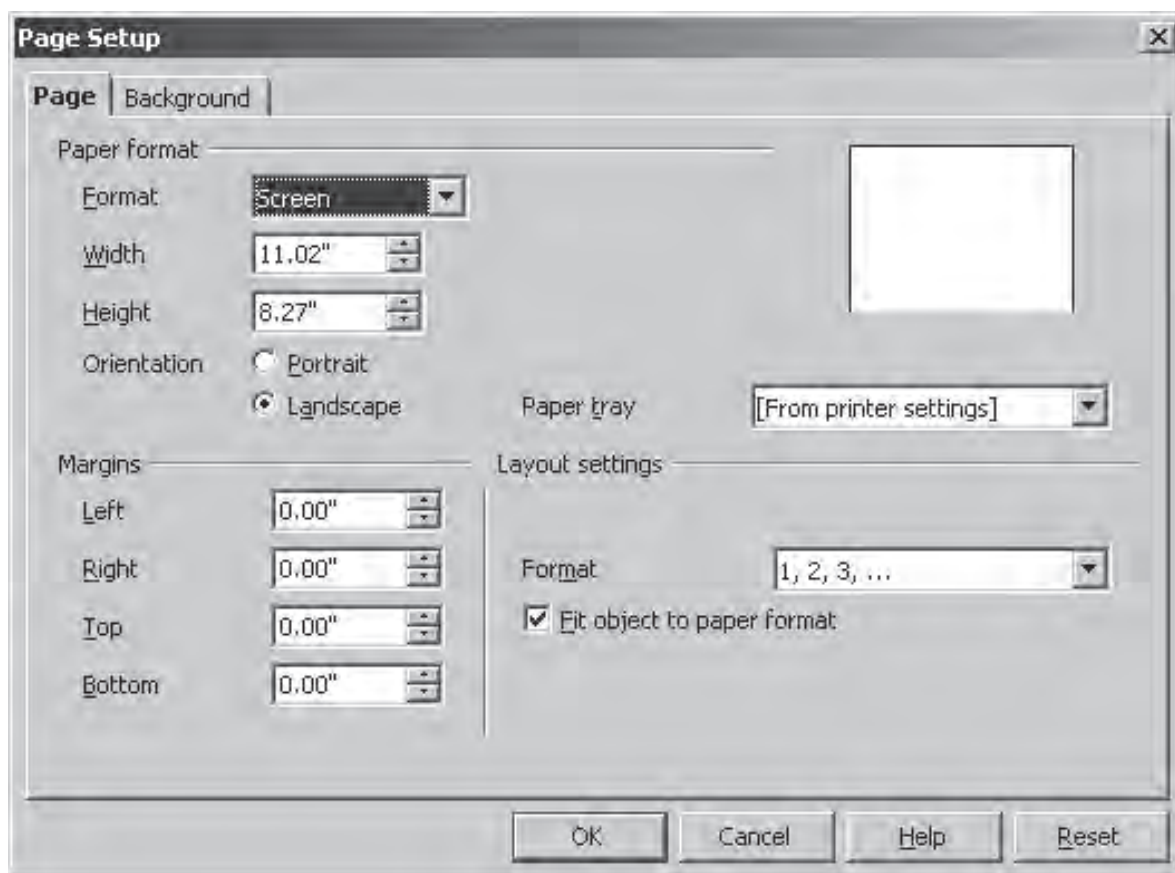
You can reduce the size of a slide when you print, so that the slide can fit on a printed page.

- ★ Open the document that you want to print.
- ★ In Normal View, choose **Format** → **Page**, and then **Page** tab. (figure 9.44)



- ★ In **Layout settings** area, select the **Fit object to paper format** check box.
- ★ In the Paper format area, select a Format.
- ★ Click **OK**.

The slide is resized to fit the printed page, while maintaining the relative positions of the objects on the slide.

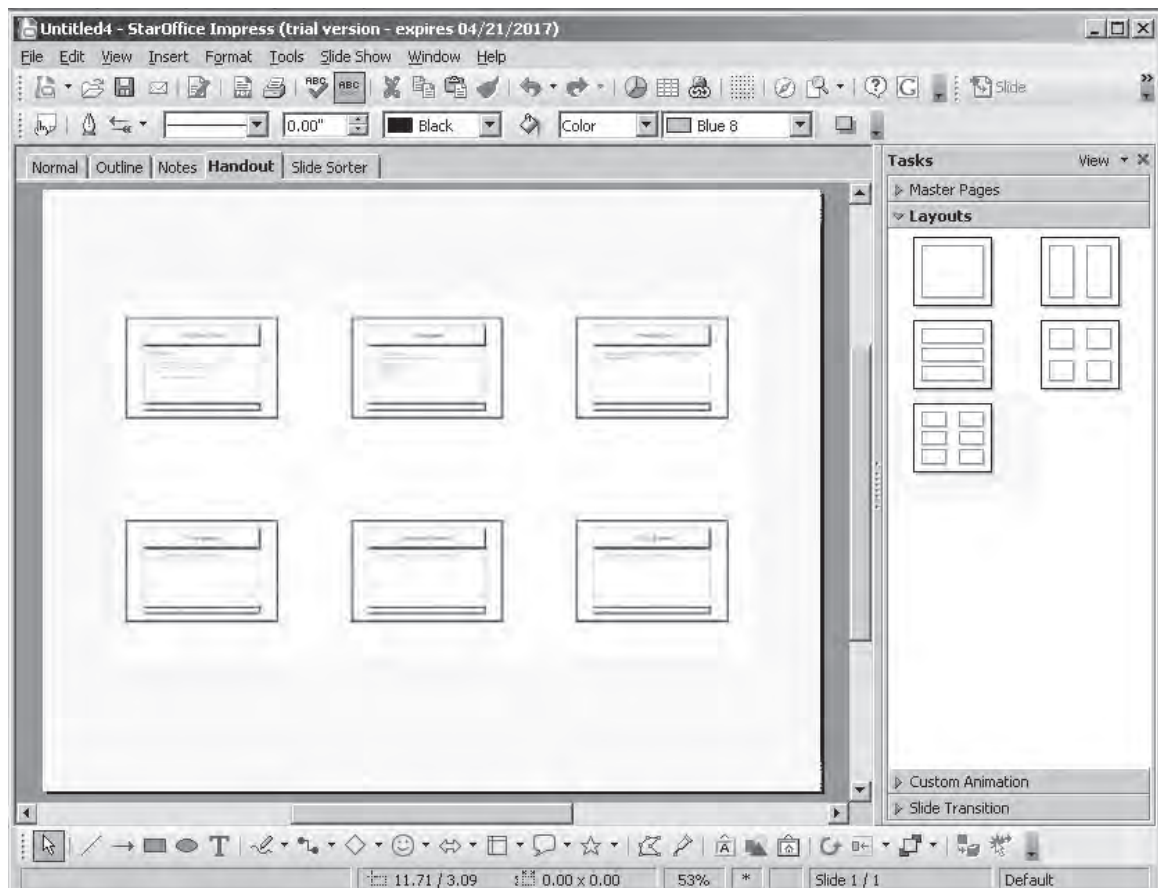


**Fig 9.44 Page Setup for Printing a Slide to fit paper size**

## 9.5.2 Creating and Printing Handouts

If you want to provide handouts you can use the **Handout View** mode. Since the handouts contain both the individual slides and space for notes, they can be a very helpful presentation aid.

Click the **Handout View** tab in the view bar. Open the **Layout Pane** in **Tasks Pane**. Choose a layout to specify how many slides are printed on a single sheet of paper as shown in the figure 9.45.



**Fig 9.45 Printing Handouts**

To print them, choose **File → Print** and click **Options** to define the print settings.

### 9.5.3 Organizing and Printing Notes

Enter notes for individual slides in the **Notes** view. The notes view is shown in the figure 9.46.

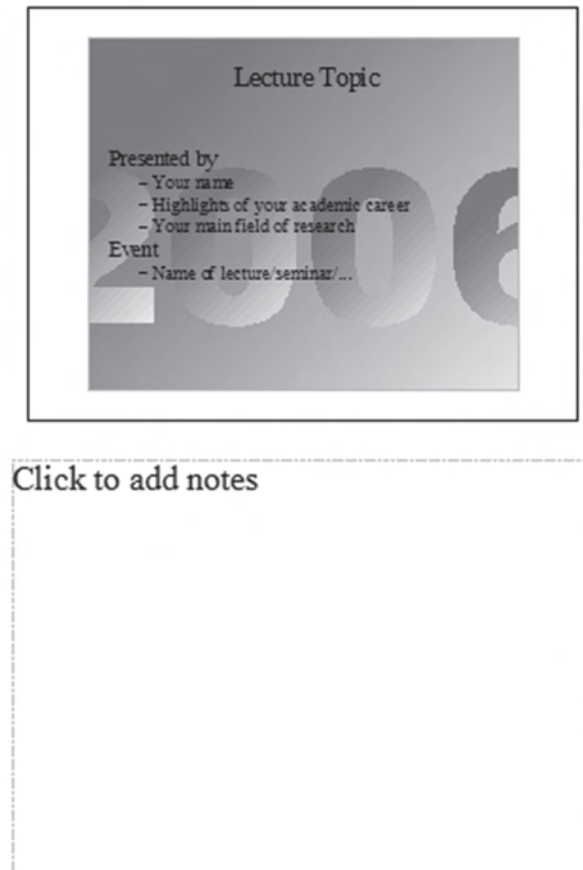
You can access the **Notes** view in the view bar.

#### To print slide notes:


1. Choose **File → Print** and click **Options**.
2. In the **Printer Options** dialog, select **Notes** in the Contents area and click **OK**.
3. In the **Print** dialog, select the slides that you want to print and click **OK**.

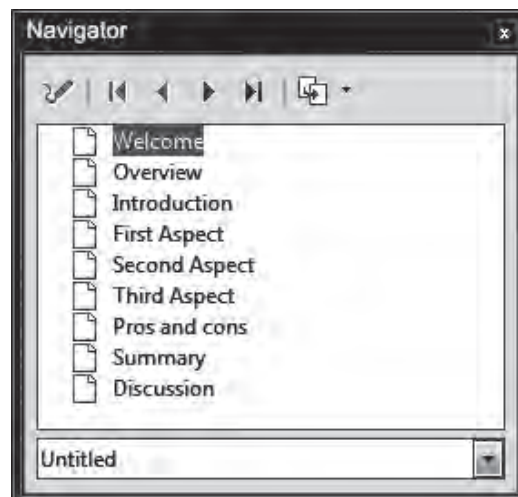
Remember to clear the **Notes** check box in the **Printer Options** dialog if you do not want to include notes in the print out. The settings in Printer Options apply only to the current document. If you want specific settings for all presentations, choose **Tools → options → StarOffice Impress → Print**.





**Fig 9.46 Entering Notes for Printing**

Using the Navigator, you can move from slide to slide quickly. Open Navigator by choosing **Edit → Navigator** or by clicking the navigator icon  in the function bar. The Navigator window as shown in figure 9.47 appears. Double-click a slide title to jump to that slide.



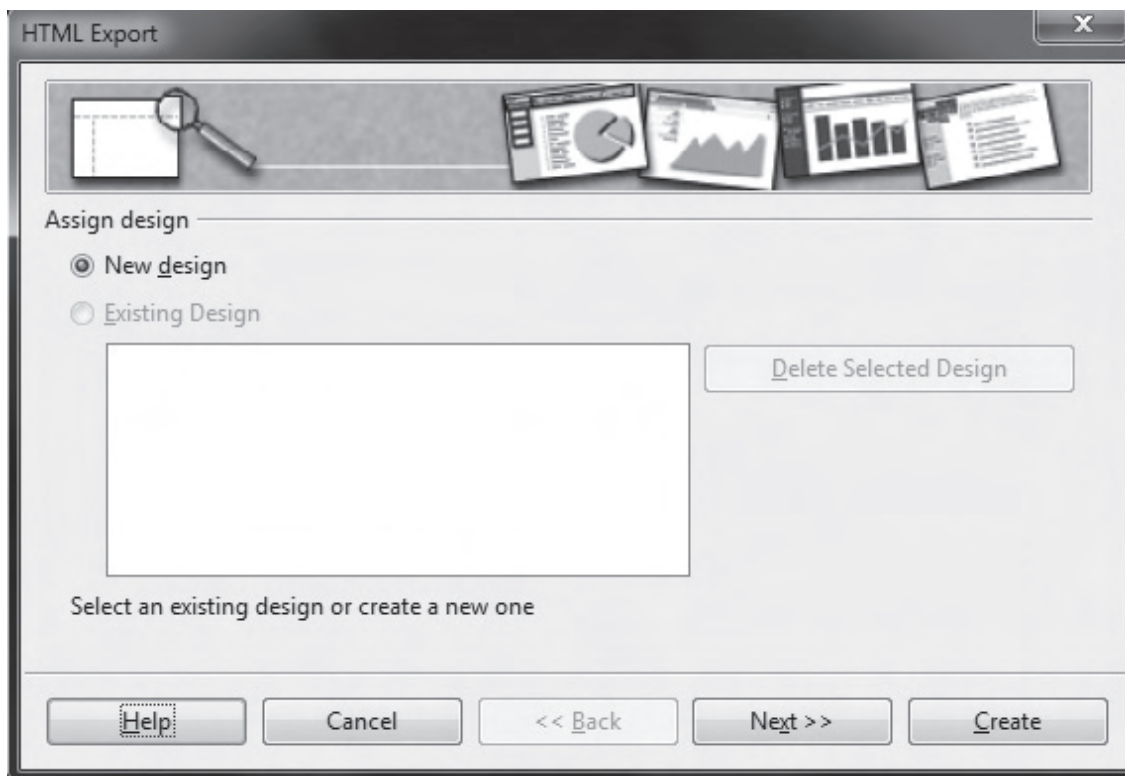
**Fig 9.47 Navigator window**

## 9.5.4 Exporting Presentations

StarOffice Impress automatically starts a wizard to help you to produce an attractive HTML presentation. A number of **HTML** pages are created that are connected to one another by hyperlinks and in which the graphics are saved as GIF or JPEG images. You can work on these HTML pages in the text module of StarOffice to give them headings and additional hyperlinks, for example.

- ★ Choose **File→ Export**.
- ★ Select the file format as **HTML Document (StarOffice Impress)**.
- ★ Enter a file name and click **Save**.

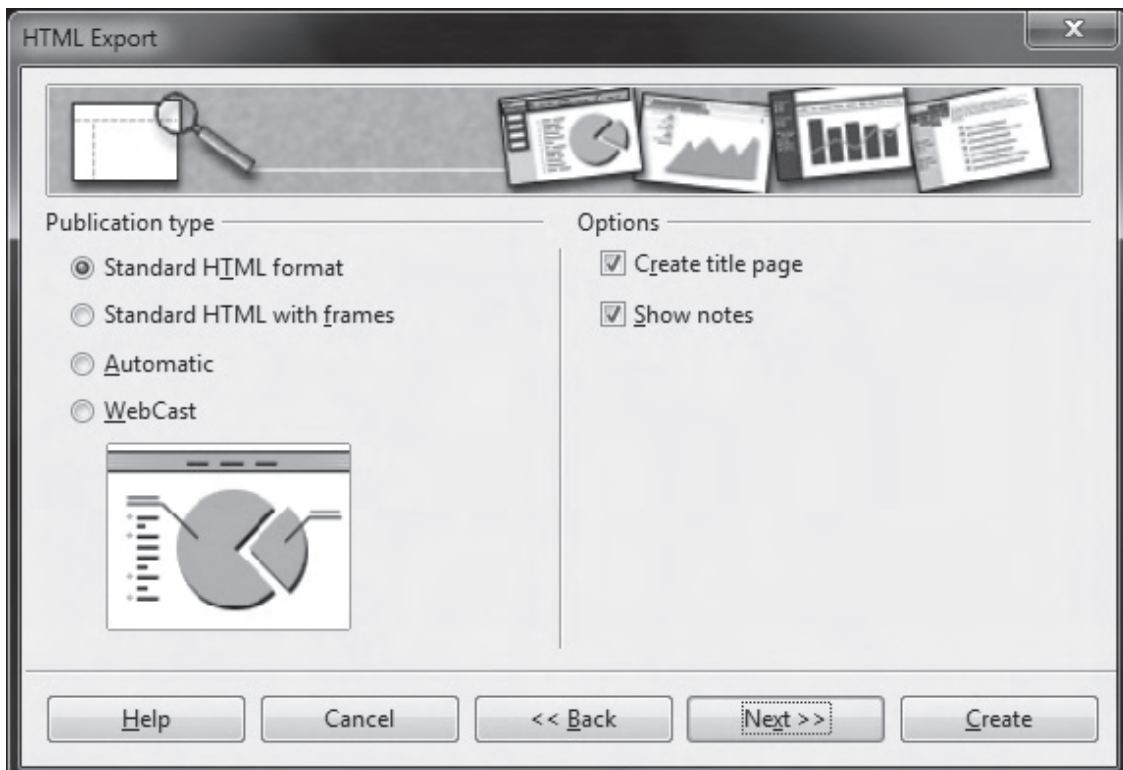
This opens the **HTML Export** wizard. Figure 9.48 shows the opening screen of the HTML Export wizard.



**Fig 9.48 HTML Export Wizard**

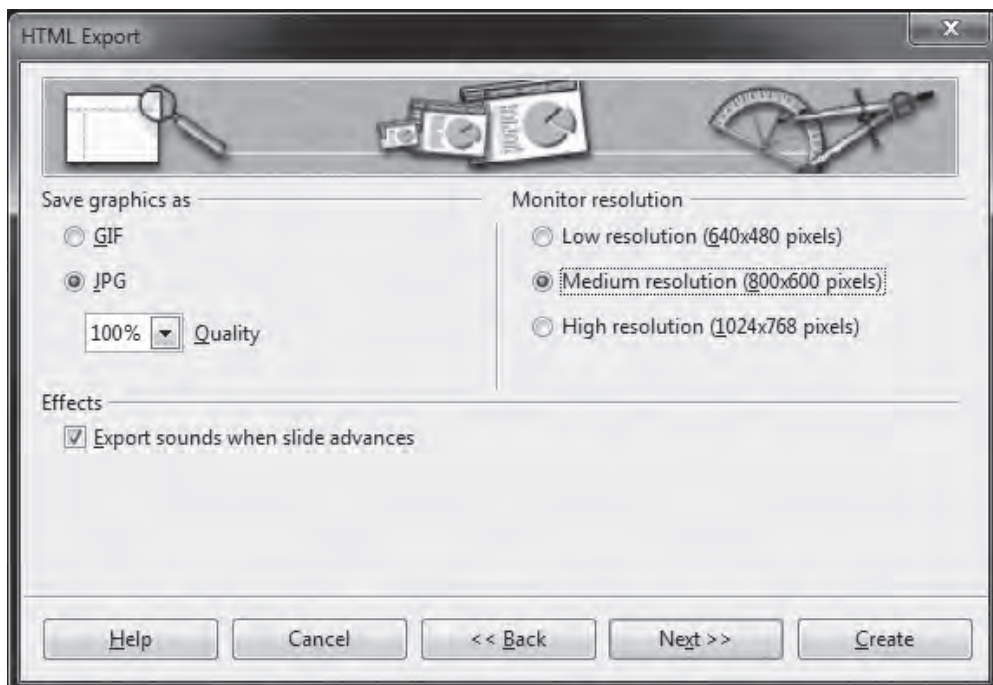
Select the **New Design** option, and click Next to see the screen as shown in the figure 9.49.

In the **Publication Type option**, select **Standard HTML format** to create html pages without frames. Check **Options** checkbox to have a title page.



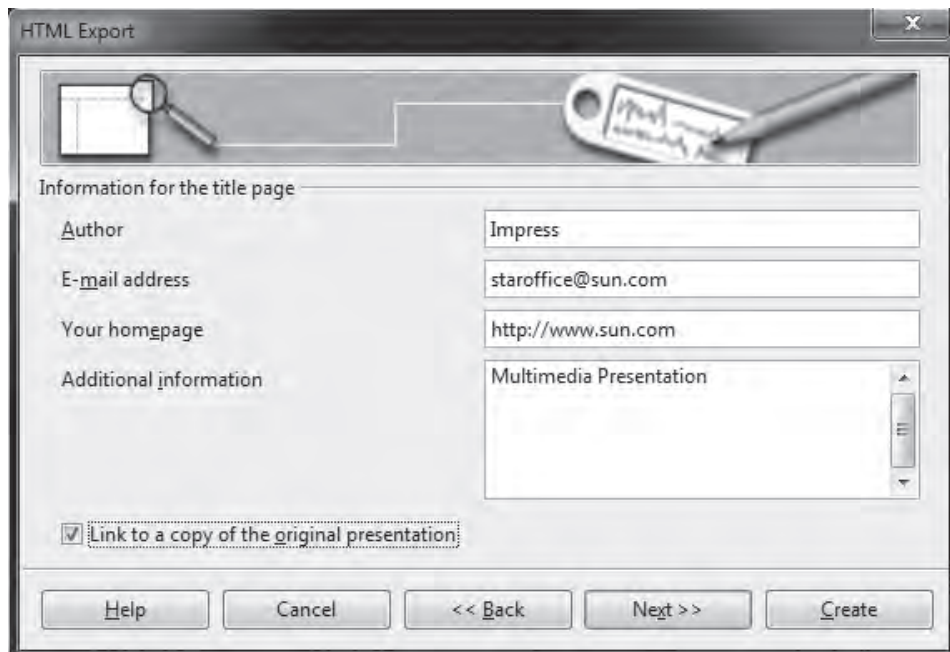
**Fig 9.49 Choosing Publication Type**

The next screen (figure 9.50) allows you to set the size of the display. Select GIF or JPEG format to save the pictures in your presentation. The medium resolution setting is the standard resolution, so select it and click the **Next** button.



**Fig 9.50 Setting Display Size**

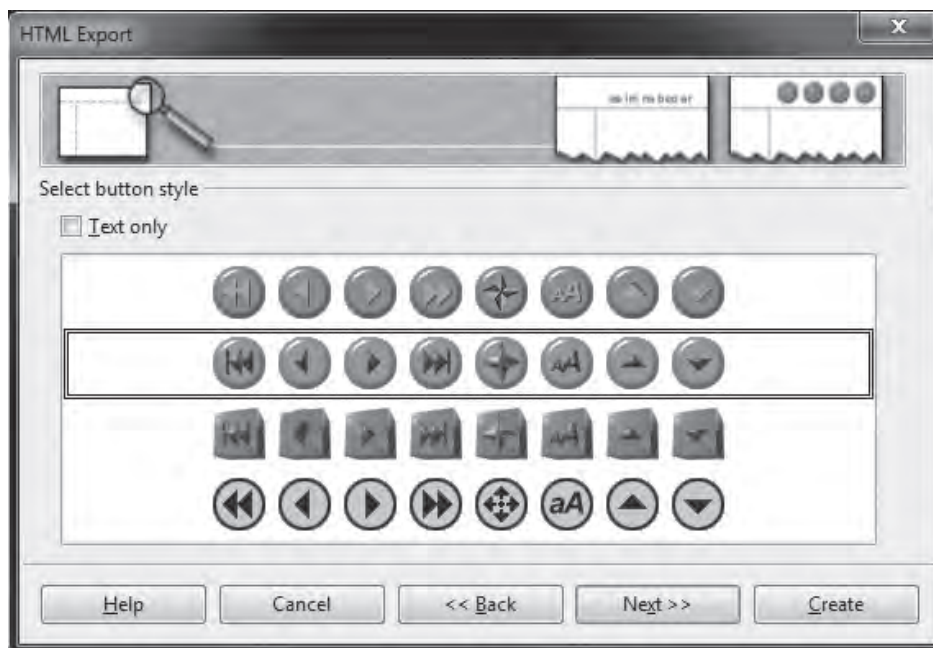
Your next screen choice allows you to enter contact information. Enter it if you wish. Click the **Next** button to see the screen as shown in the figure 9.51.



The 'HTML Export' dialog box features a header with a magnifying glass icon and a tag icon. Below the header, the 'Information for the title page' section contains four input fields: 'Author' (filled with 'Impress'), 'E-mail address' (filled with 'staroffice@sun.com'), 'Your homepage' (filled with 'http://www.sun.com'), and 'Additional information' (filled with 'Multimedia Presentation'). A checkbox labeled 'Link to a copy of the original presentation' is checked. At the bottom, there are five buttons: 'Help', 'Cancel', '<< Back', 'Next >>', and 'Create'.

**Fig 9.51 Entering Contact Information**

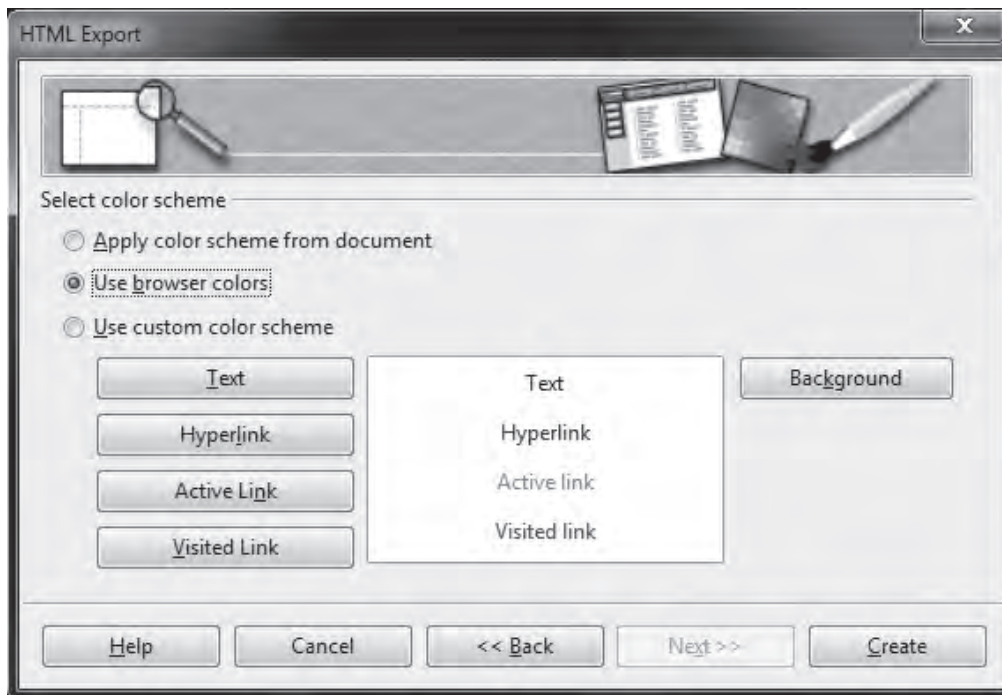
Now, the screen allows you to pick the graphics to be used in the Web site navigation bar. Pick the buttons that best match your presentation's background. Click **Next** to see the screen as shown in the figure 9.52.



The 'HTML Export' dialog box shows the 'Select button style' section. A checkbox labeled 'Text only' is unchecked. Below it, there are four rows of navigation button icons. The second row is highlighted with a rectangular selection box. At the bottom, there are five buttons: 'Help', 'Cancel', '<< Back', 'Next >>', and 'Create'.

**Fig 9.52 Selecting Button Style**

Select **Use browser colors**, to apply the colours supported by browsers (figure 9.53) to the html pages and click **Create** button. You will be prompted to save all of your answers in the wizard as an HTML design. Name the design and click **Save** button. Html pages will be created in the specified directory. You can open the starting page and navigate to other pages.



**Fig 9.53 Selecting Color Scheme**

## Summary

- A slide show is a series of slides, or pages, that present information on a specific topic.
- A presentation is used while speaking to a group, with a presentation to support and organize your information.
- Some presentations, or slide shows, can be used without a speaker, like a collection of photographs.
- Creating a Presentation.
- A presentation can be created using a template or can be created from scratch.
- An automatic presentation flips through the slides at a defined duration until the Escape key is pressed.
- Normal, Outline, Notes, Handout, Slide Sorter views helps us to work with slides in ease to different needs.

- A presentation can contain different type of objects like Picture, Movies, Sound, Chart, Spreadsheet and other OLE objects.
- A presentation can be exported in different format like Web page, PDF, SWF, JPEG and so on.
- Slides in Impress can be designed according to different needs, by using Color, Gradient, Hatching and Bitmap Image styles.
- A master slide determines the text formatting style for the title and outline and the background design for individual slides.
- Like slides, Objects in your slide can have various effects, like rolling into your presentation from the left side, the text can be slowly drawn onscreen, and so on using Custom Animation Effects.
- Custom Slide Show allows you to display important slides during a presentation.
- StarOffice Impress allows you to start slide shows from the current slide as well as hide slides during a slide show.
- Slides in a presentation can be printed as a handout, so that several slides can be printed on a single page of paper.
- StarOffice Impress enables Media Player where you can preview movie and sound files as well as insert these files into the current presentation slide
- The Media Player embedded in StarOffice Impress supports different audio and video file formats.

## EXERCISES

### Fill in the blanks

1. A 'slide show' is a series of pages that are \_\_\_\_\_
2. A presentation can be created from scratch or from \_\_\_\_\_
3. Five types of views are Normal, Outline, Notes, \_\_\_\_\_ and \_\_\_\_\_
4. \_\_\_\_\_ gives effect to an object in a slide.
5. In \_\_\_\_\_ view mode, all the slide titles will appear in a list along with the headings and sub-topics.
6. By pressing \_\_\_\_\_ key in the keyboard the view of the slide can be enlarged.
7. To select an object that is covered by another object, hold down \_\_\_\_\_ key and click the object.
8. \_\_\_\_\_ slide determines the text formatting style for the title and outline.
9. \_\_\_\_\_ window allows us to quickly jump from one slide to other slide or move between open files.
10. \_\_\_\_\_ is used preview movie or sound files in a slide

### Answer the following

1. Define briefly about Impress and describe how to create a presentation.
2. List and describe the features of Impress.
3. Explain the steps of Incorporating Slide Show Effects.
4. Explain the steps of exporting a presentation as web pages.
5. What is custom Animation? Explain the process of animating objects in a slide.
6. What is slide transition? Explain the process of applying transition to slides.
7. How to include pictures, movies and other OLE objects in a presentation?
8. What is 'Rehearse Timings'? Explain in detail.
9. Explain about 'Custom Slide Show' in detail.
10. Explain how to print a range of slides and also explain how to print a slide to fit a paper size?



## APPENDIX

### FURTHER READING

#### StarOffice

The **docs.sun.com** Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for StarOffice specific book title.

The URL is **<http://docs.sun.com>**.

## **CHAPTER 10**

### **OBJECT ORIENTED CONCEPTS USING C++**

#### **10.1 Object Oriented Programming Language**

A computer is a tool to solve a wide range of problems. The solutions to the problems are in the form of computer programs or application software. These programs are written using a chosen programming language.

A computer program operates on a set of known input data items. The program transforms this input data into a set of expected data items. Only this set of expected data items must be the output of the computer program.

In the early programming languages the input and output data items were represented as variables. Data types categorized these input data items. Control statements provided a way of instructing the computer on the operations that need to be performed on the data items.

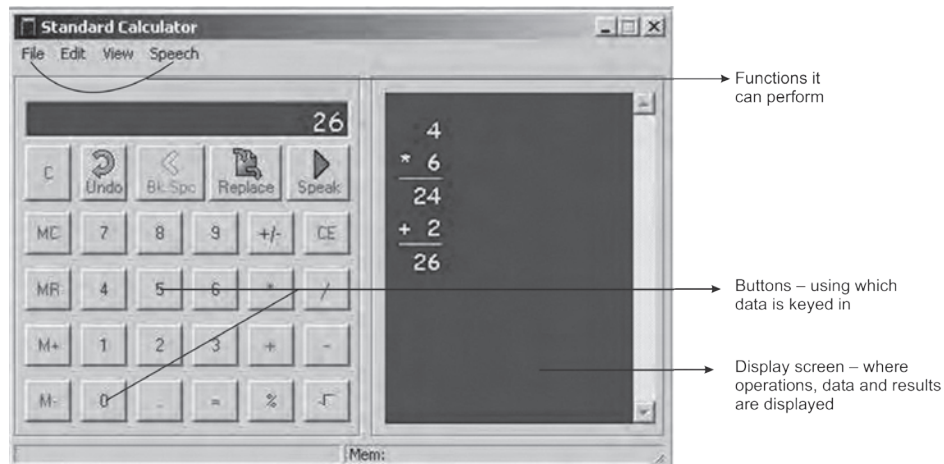
Programming languages have another use. They help us in organizing our ideas about the solution of the problem. As the problems being solved or the applications being developed became complex, this aspect of programming languages became very important. Many programming languages emerged to address this issue along with the ease of instructing the computer.

It was realized that viewing the solution of a problem as two separate segments 'data' and 'operations' does not resemble the way human beings solve the real life problems.

Object oriented programming languages such as C++ are based on the way human beings normally deal with the complex aspects of real life. It has been observed that human beings normally solve real life problems by identifying distinct objects needed for the solution. Human beings then recognize the relationships amongst these objects. The relationships are like 'part of the whole' or are 'a type of'. Simple abilities such as recognizing that one object is a part of the bigger object and one object is a type of another object are proving to be very important in solving problems in real life. Object Oriented programming facilitates this way of problem solving by combining 'data' and 'operations' that are to be performed on the data.

In other words, the set of data items is split into smaller groups such that a set of operations can be performed on this group without calling any other function. This group of data and the operations together are termed - 'object'. The operations represent the behavior of the object. An object attempts to capture a real world object in a program.

For example, take a look at any calculator, it has both state and behaviour. Its state refers to its physical features like its dimensions, buttons, display screen, operators and the like. Its behaviour refers to the kind of functions it can perform like addition, subtraction, storing in memory, erasing memory and the like.



**Fig. 10.1 Standard Calculator**

In an object oriented programming language, a calculator is viewed as follows :

**Object – calculator**

**Data :**

Number1,result, operator, Number\_backup

**Functions :**

Additon()

Subtraction()

Erase\_Memory()

Display\_Result()

- ✓ **An object is a group of related functions and data that serves those functions.**
- ✓ **An object is a kind of a self-sufficient “subprogram” with a specific functional area.**

The process of grouping data and its related functions into units called as objects paves way for **encapsulation**.

**The mechanism by which the data and functions are bound together within an object definition is called as ENCAPSULATION.**

It is easy to see how a bank-account, a student, a bird, a car , a chair etc., embodies both state and behaviour. It is this resemblance to real things that gives objects much of their power and appeal. Objects make it easy to represent real systems in software programs.

## Examples of objects – BANK ACCOUNT & STUDENT

<b>BANK ACCOUNT</b>	<b>STUDENT</b>
<b>Data :</b> Account number – long int Name – char[15] Opening balance –float; Account type – char	<b>Data :</b> Date_of_birth – char[10] Name – char[15]; Class, sec char[4]; Marks float
<b>Functions :</b> Accept_Details() Display_Details() Update_opening_balance() Withdrawals() Deposit()	<b>Functions :</b> Accept_Details() Display_Details() Total() Average() Grading()

### 10.2 Polymorphism

Now let us consider the job of drawing different shapes like a rectangle, square, circle and an arc. We tend to define different functions to draw these different shapes. The definitions may be like this :

Draw_Square() Read side Draw required lines	Draw_Rectangle() Read length,breadth Draw required lines	Draw_Circle() Read radius Draw	Draw_Arc() Read Start_angle, End_angle,radius draw
---	--	--------------------------------------	--

Now look at the following function :

Draw( side) – is defined to draw a square

Draw (length, breadth) - is defined to draw a rectangle

Draw(radius) - is defined to draw a circle

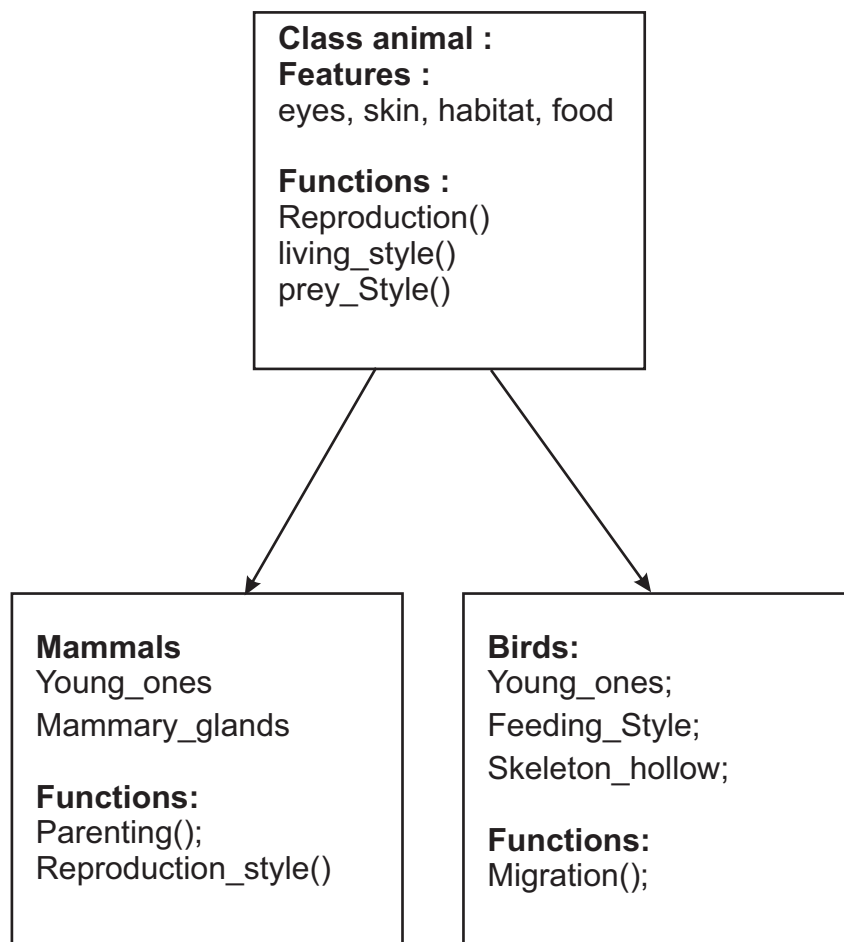
Draw(radius,start\_angle,end\_angle) – to draw an arc

The function draw() accepts different inputs and performs different functions accordingly. As far as the user is concerned, he will use the function **draw()** to draw different objects with different inputs. This differential response of the function draw() based on different inputs is what is called as **polymorphism**.

The ability of an object to respond differently to different messages is called as **polymorphism**.

### 10.3 Inheritance

The data type **Class** conventionally represents an object in the real world. Class is a template for entities that have common behaviour. For example animals form a group of living beings, or in other words **animals** is a class. We know that animals are divided into mammals, reptiles, amphibians, insects, birds and so on. All animals share common behaviour and common attributes. Eyes, skin, habitat, food refer to the features or attributes of the animals, while reproduction, living\_style, prey\_style etc refers to the behaviour of the animals. Every sub group of animals has its own unique features or styles apart from the common behaviour and features. The sub groups do share the properties of the parent class – “ANIMALS” apart from its own sub classes viz ., mammals, reptiles, amphibians, insects, birds. This may be pictorially depicted as follows :



**Fig. 10.1 Inheritance**

Animals is called the base class, and Mammals and Birds are called derived classes. The derived classes are power packed, as they include the functionality of the base class along with their own unique features. This process of acquiring the Base class properties is what is called **inheritance** .

Inheritance increases the functionality of a derived class and also promotes reusability of code (of the base class).

#### Advantages of Object Oriented Programming –

- ✓ Class data type allows programs to organize as objects that contain both data and functions .
- ✓ Data hiding or Abstraction of data provides security to data, as unrelated member functions(functions defined outside the class) cannot access its data, or rather it reveals only the essential features of an object while curtailing the access of data
- ✓ Polymorphism reduces software complexity, as multiple definitions are permitted to an operator or function
- ✓ Inheritance allows a class to be derived from an existing class , thus promoting reusability of code, and also promote insertion of updated modules to meet the requirements of the dynamic world

#### 10.4 A Practical Example : Domestic Waterusage



**Fig. 10.2 Domestic Waterusage**

For example, let us consider developing a program that modeled home water usage. The objective of this program is to compute the water consumed by each outlet in a building and also total consumption. All that we require for this program is the number of taps installed in the building, amount of water that flowed through each tap, and finally the amount of water consumed. Each tap may be viewed as an object. The

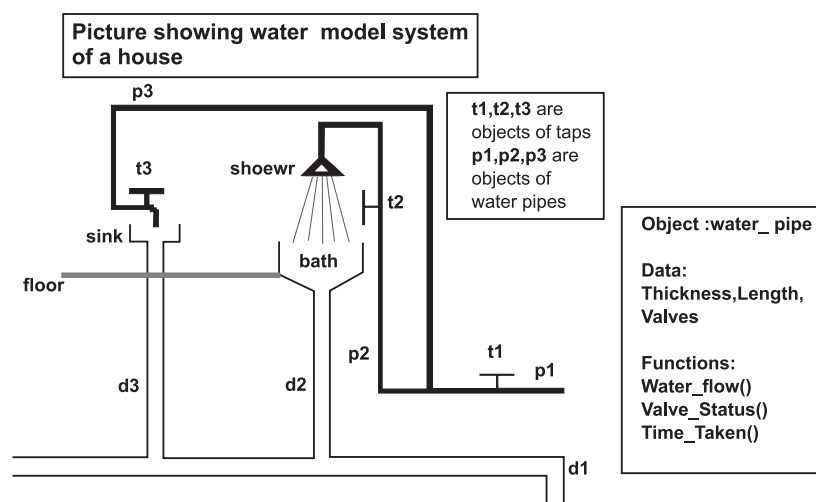
functions associated would be to start and stop the flow of water, return the amount of water consumed in a given period, and so on. To do this work, the tap object would need instance variables to keep track of whether the tap is open or shut, how much water is being used, and where the water is coming from. The object may be visualised as:



**Fig. 10.3 Tap as an Object**

The program that models water usage will also have WaterPipe objects that delivers water to the taps . There could be a Building object to coordinate a set of WaterPipes and taps. When a Building object is asked as to how much water is being used, it might call upon each tap and pipe to report its current state. The project may be visualised as shown in Fig.1.4.

Now the total\_amount of water consumed would be calculated as  $t1.water\_consumed() + t2.water\_consumed + t3.water\_consumed()$  and water consumption by each outlet would be given away individually by  $t1.water\_consumed$ ,  $t2.water\_consumed()$  and so on.



**Fig. 10.4 Water Distribution System in a House**



t1.water\_consumed() would in turn communicate with p1 to get the amount of water flowed through that pipe, as tap-1s(t1) water consumption is determined by pipe1(p1). Thus a program consists of objects that call each other to compute. Each object has a specific role to play, and the co-ordinated working of all the modules produces the end result of a program. Objects communicate with one another by sending data as inputs.

Everyday programming terminology is filled with analogies to real-world objects like tables, students, managers, bank accounts, and the like. Using such entities as programming objects merely extends the comparison in a natural way. This line of thinking about functions and object behaviour is the key factor of object-oriented programming.

## EXERCISES

### I. Fill in the blanks

- a. \_\_\_\_\_model entities in the real world
- b. Binding of data and member functions together is called as\_\_\_\_\_
- c. The ability of an object to respond differently to different messages is called as \_\_\_\_\_
- d. The process of creating new data types from existing data type is called as \_\_\_\_\_

### II. Answer the following briefly

1. What is the significance of an object ?
2. What is Encapsulation?
3. How is polymorphism different from inheritance?

### III. Design data type for the following project

A company wishes to prepare a data model for its activities. The company stores information of all its employees. The common details of all employees are : Name, date\_of\_birth,language and nativity.

Additional details of employees based on their placement are stored as :

- a. Stores – date of joining, dept, salary
- b. Scientist – area of specialisation, current project details, paper\_presentations
- c. Technician – Height, Weight, ailments, risk factor, department, wages.

# CHAPTER 11

## OVERVIEW OF C++

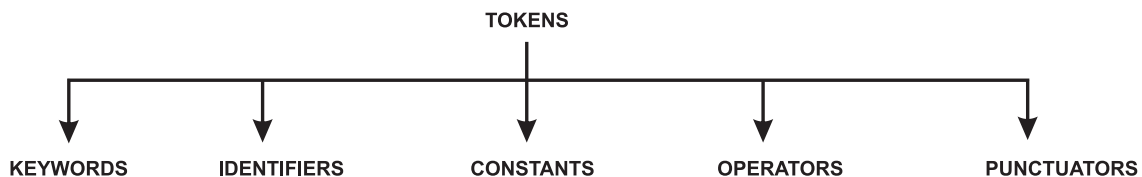
### 11.1 Introduction

C++ was developed at AT & T Bell laboratories in the early 1980s by **Bjarne Stroustrup**. The name C++ (pronounced as C plus plus) was coined by Rick Mascitti where “++” is the C increment operator.

### 11.2 C++ character set

Like the C language, C++ also comprises a character set from which the tokens (basic types of elements essential for programming coding ) are constructed. The character set comprises of “A” .. “Z” , “a” .. “z”, 0 .. 9, +, -, /, \*, \, (, ), [, ], {, }, =, !=, <, >, ., ' " ; : %, !, &, ?, \_, #, <=, >=, @, white space, horizontal tab, carriage return and other characters.

A quick recap of the basic types : The basic types are collectively called as **TOKENS**. A token is the smallest individual unit in a program. Tokens are classified as shown in Fig 11.1.



**Fig. 11.1 Classification of Tokens**

#### 11.2.1 Keywords

Keywords have special meaning to the language compiler. These are reserved words for special purpose. These words cannot be used as normal identifiers. Table 11.1 shows the list of keywords used in C++.

auto	break	case	const	class	continue
default	delete	do	else	enum	for
friend	goto	if	inline	new	operator
private	protected	public	return	signed	sizeof
static	struct	switch	this	unsigned	virtual
while					

**Table 11.1 Keywords**

### 11.2.2 Identifiers

Identifiers are also called as variables. Variables are memory boxes that hold values or constants. A variable name must begin with an alphabet or underscore followed by alphabets or numbers. For example `_test` ; `test` ; `sum12` are some valid identifiers. We shall see more about variables after dealing with data types

### 11.2.3 Constants

Constants are data items whose values cannot be changed. A constant is of numeric or non-numeric type. Numeric constants consist of only numbers, either whole numbers or decimal numbers. Integer, floating-point are numeric constants.

### 11.2.4 Integer Constant

- ✱ Integer Constant must have at least one digit and must not contain any fractional part.
- ✱ May be prefixed with a + or – sign
- ✱ A sequence of digits starting with **0** (zero) is treated as Octal constant Ex. `010`  
 $= 8 \text{ ( } [8]_{10} = [10]_8 \text{ )}$
- ✱ A sequence of digits starting with **0x** is treated as hexadecimal integer. Ex. `0xF`  
 $= 15 \text{ ( } [15]_{10} = [F]_{16} \text{ )}$

### 11.2.5 Floating Point Constant

Floating Point Constant is a signed real number. It includes an integer portion, a decimal point, a fractional portion and an exponent. While representing a floating point constant the integer portion or the decimal portion can be omitted but never both. For example `58.64` is a valid floating point (Real) constant. It can be represented in exponent form as follows :

- $5.864E1 \Rightarrow 5.864 \times 10^1 \Rightarrow 58.64$
- $5864E-2 \Rightarrow 5864 \times 10^{-2} \Rightarrow 58.64$
- $0.5864E2 \Rightarrow 0.5864 \times 10^2 \Rightarrow 58.64$

The letter E or e is used to represent the floating-point constant exponent form

### 11.2.6 Character Constant

Character constant is a constant that contains a single character enclosed within single quotes. It can be any character as defined in the character set of C++ language (alphabet, numeral, mathematical, relational or any other special character as part of the ASCII set). Certain special characters like tab, backspace, line feed, null, backslash

are called as non-graphic character constants. These characters are represented using escape sequences. Escape sequences are represented using characters prefixed with a backslash. Table 11.2 shows the escape sequences.

Escape Sequence	Nongraphic Character
\a	Bell
\b	Back space
\n	New line/ line feed
\t	Horizontal tab
\v	Vertical tab
\\	Back slash
\' or \"	Single / double quotes
\o	Octal number
\x	Hexadecimal number
\0	Null

**Table 11.2 Escape Sequences**

### 11.2.7 String Literal

String Literal is a sequence of characters surrounded by double quotes. String literals are treated as array of characters. Each string literal is by default added with a special character '\0' which marks the end of a string. For example “testing”

### 11.2.8 Operator

Operator specifies an operation to be performed that yields a value. An operand is an entity on which an operator acts. For example :

**RESULT = NUM1 + NUM2**

NUM1 and NUM2 are operands. + is the additional operator, that performs the addition of the numbers. The result (value) generated is stored in the variable RESULT by virtue of “=” (Assignment) operator. Table 11.3 shows the operators in C++.

[]	*	%	==	=	>=
()	+	<<	!=	*=	&=
.	-	>>	^	/=	^=
->	~	<		+=	=
++	!	>	&&	-=	, --
&	size of	<=		%=	#
	/	>=	?:	<<=	# #

**Table 11.3 Operators in C++**

The following operators are specific to C++.

::      .\*      ->\*

The operators # and ## are used only by the preprocessor.

Operators are classified as

- Arithmetic
- Assignment
- Component Selection
- Conditional
- Logical
- Manipulator
- Member dereferencing
- Memory Management
- Preprocessor
- Relational
- Scope Resolution
- Shift
- Type Cast

Based on operand requirements, operators are also classified as unary, binary and ternary operators.

For example :

**Unary operators require one operand**  
**Binary operator requires two operands**  
**Ternary operator requires three operands.**

**Table 11.4a Unary Operators**

&	Address of
!	Logical Not
*	Indirection
++	Increment
~	Bitwise
--	Decrement
-	Unary minus
+	Unary plus

**Table 11.4b Binary Operators**

Additive	+	Binary Plus
	-	Binary minus
Multiplicative	*	Multiply
	/	Divide
	%	Remainder (Modulus)
Shift	<<	Shift Left
	>>	Shift Right
Bitwise	&	AND
		OR
	^	XOR
Logical	&&	Logical AND
		Logical OR
Assignment	=	Assignment
	/=	Assign quotient
	+=	Assign sum
	*=	Assign product
	%=	Assign remainder
	-=	Assign difference
	<<=	Assign left shift
	>>=	Assign right shift
	&=	Assign bitwise AND
	^=	Assign bitwise XOR
	=	Assign bitwise OR
Relational	<	Less Than
	>	Greater than
	<=	Less than or Equal to
	>=	Greater that or Equal to
Equality	==	Equal to
	!=	Not Equal to
Component Selection	•	Direct Component Selection
	->	Indirct Component Selection
Class Member	::	Scope access / Resolution
	.*	Dereference Operator
Conditional	->*	Dereference pointer to class member
	?:	Ternary operator
Comma	,	Evalute

### 11.2.7.1 Arithmetic Operators

Arithmetic Operators are used to perform mathematical operations. The list of arithmetic operators are :

- +
- -
- \* multiplication operator
- / division operator
- % modulus operator - gives the remainder of an integer division
- += , -=, \*= , /= , %=

Arithmetic expressions are formed using arithmetic operators, numerical constants/variables, function call connected by arithmetic operators.

#### Examples :

- $a = -5;$
- $a = +b;$
- $a /= 5; (a = a/5)$
- $a++;$  (Post increment operator . Equivalent to  $a = a+1$ )
- $a--;$  (Post decrement operator. Equivalent to  $a = a-1$ )
- $++a;$  (Pre increment operator. Equivalent to  $a = a+1$ )
- $--a;$  (Pre decrement operator. Equivalent to  $a = a - 1$ )
- $a *= 2 (a = a * 2)$
- $a \% = 5 (a = a/5)$
- $a = b + \text{pow}(x,y) (a = b + x^y)$

Operators are executed in the order of precedence. The operands and the operators are grouped in a specific logical way for evaluation. This logical grouping is called as association. Table 11.5 indicates the Mathematical Operators, its Type, and Association.

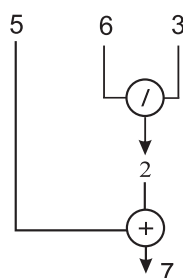


Operator Precedence	Type	Associativity
() []	Mathematical	Left to right
Postfix ++, -- ,	- Unary	Left to right
prefix ++, --		Right to left
+unary , - unary	mathematical	Right to left
* / %	Mathematical –binary	Right to left Left to right
+ -	Mathematical– binary	Left to right

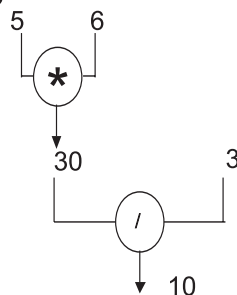
**Table 11.5 Mathematical Operator Precedence**

The following examples demonstrate the order of evaluation in arithmetic expressions :

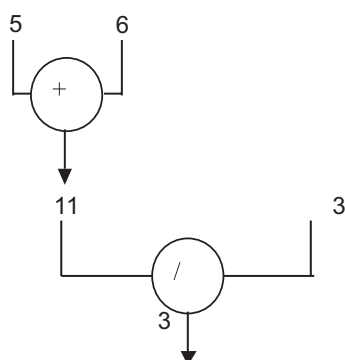
$$5 + 6/3 = 7$$



$$5 * 6 / 3 = 10$$



$$(5 + 6) / 3$$

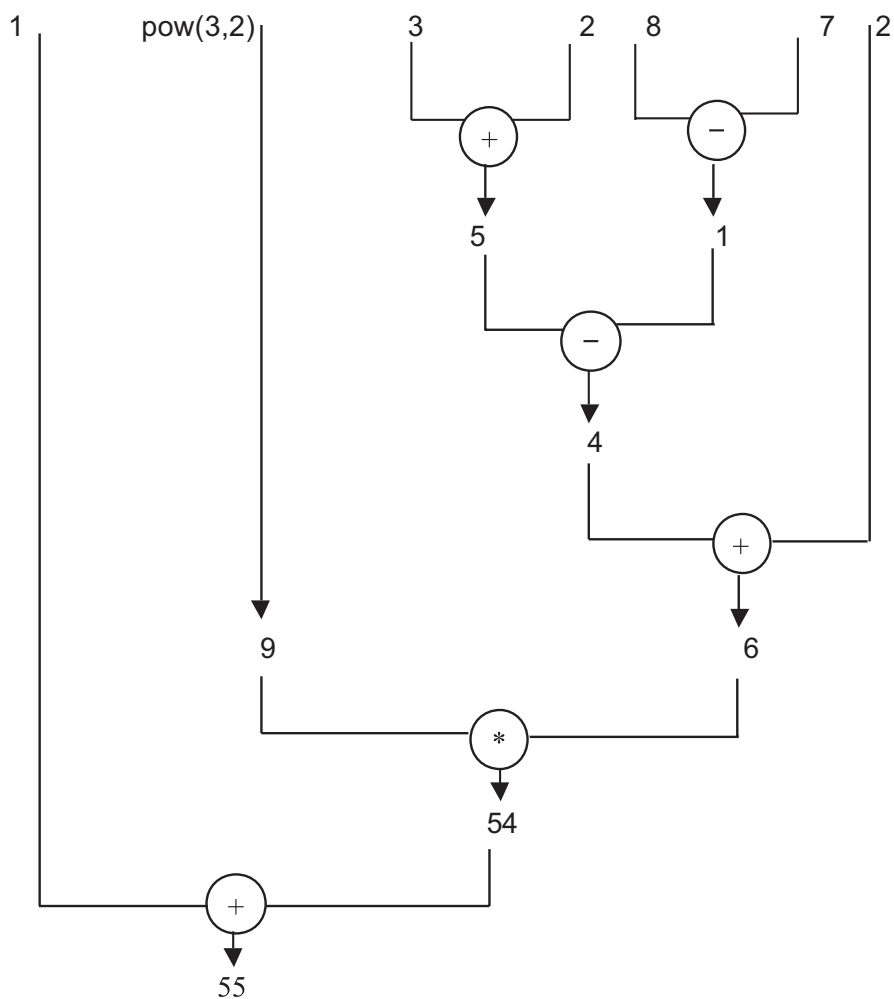
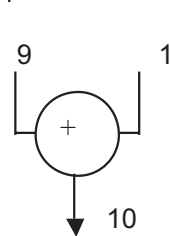


The result is 3, as all the inputs are of integer type.

The result will be 3.66 if any of the inputs are of float type.

`1 + pow ( 3 , 2 )`

`pow( 3 , 2 )`



Increment and Decrement operators are unique to C++. Evaluation of expressions using these operators are indicated in the Table 11.6 .

Expression	Operation	Example
a++	Get the value of a, then increment the value of the variable by 1	a = 5; c = a++; Execution : c = a; a = a + 1; Hence the value stored in the variable c is 5.
++a	Increment the value of the variable a by 1, and then get the value	a = 5; c = ++a; Execution: a=a+1 c = a; Hence the value of c will be 6
a--	Get the value of a , then decrement it by 1	a = 5; c = a--; Execution : c = a; a = a - 1 What will be the value of c ?
--a	Decrement the value of a by 1, then get the value of a	a = 5 c= --a; Execution : a = a - 1 c = a; What will be the value of c ?

**Table 11.6 Increment and Decrement Operator**

What will be the values stored in the variables of the following snippets as shown in Table 11.7

1. a = 5 b = 5 a = a + b++ Value stored in the variable a is _____	2. x = 10 f = 20 c = x++ + ++f Value stored in the variable c is _____ x is _____ f is _____	3. fun = 1 sim = 2; final = --fun + ++sim – fun Value stored in the variable fun is _____ sim is _____ final is _____
--	--	---

**Table 11.7 Simple Problems**

### 11.2.7.2 Relational Operators

Relational Operators are used to compare values. The list of relational operators are :

- `==` equal to
- `>` greater than
- `<` lesser than
- `>=`, `<=` greater than or equal to , lesser than or equal to
- `!=` not equal to

Relational operators are used to compare numeric values. A relational expression is constructed using any two operands connected by a relational operator. For example the relational operators are used to construct conditions such as

- `10 > 20`
- `500.45 <= 1005`
- `99 != 99.5`
- `9 == 9`

The result of a relational operation is returned as true or false. The numeric constant zero (0) represents False value, and any non- zero constant represents true value. The above expressions output will be

- `0` (`10 > 20` is false ) ;
- `1` ( `500.45 < 1005` is evaluated to True hence any non zero constant) ;
- `1` (`99 != 99.5` will be evaluated to True hence non zero constant)
- `1` ( `9 == 9` will be evaluated to true, hence the output will be non zero constant)

What will be the value of the following expression ?

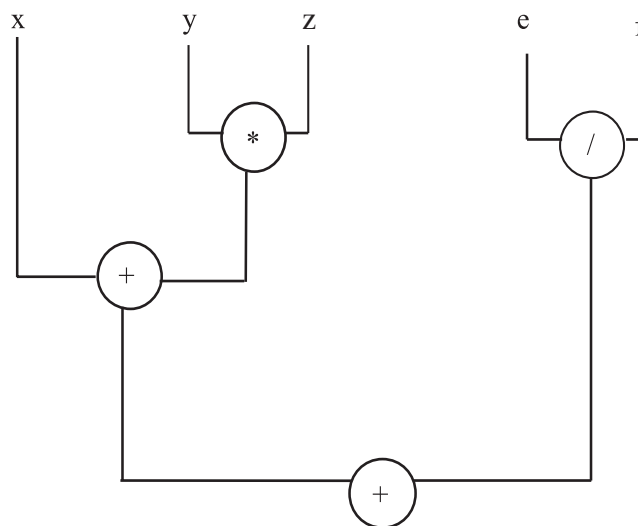
`( num1 + num2 – num3 )/5 * 2 < ( num1 % 10)`  
where `num1 = 99` , `num2 = 20`, `num3 = 10`

Evaluate the relational expressions shown in the following Table 11.8.

Operator	Expression	Result
==	5 == 6	0
!=	'a' != 'a'	
>	5 > 6	
	'a' > 'A'	
<	5 < 6	
	'a' < 'A'	
>=	'a' >= 'z'	
	5 >= 5	
<=	'a' <= 'z'	
	5 <= 6	

**Table 11.8 Evaluate Relational Expressions**

Relational operators have lower precedence than the arithmetic operators. For example the expression  $x + y * z < e / f$  will be evaluated as follows :



### 11.2.7.3.Logical Operators (Boolean Operators)

Logical operators combines the results of one or more conditions. The various logical operators are && (AND) , || (OR) , ! (NOT)

Example : c = 5 , d = 6 , choice = 'y', term = '2' (Assume True is indicated as 1 and False as 0)

Result\_1 = (c == d) && (choice != term)

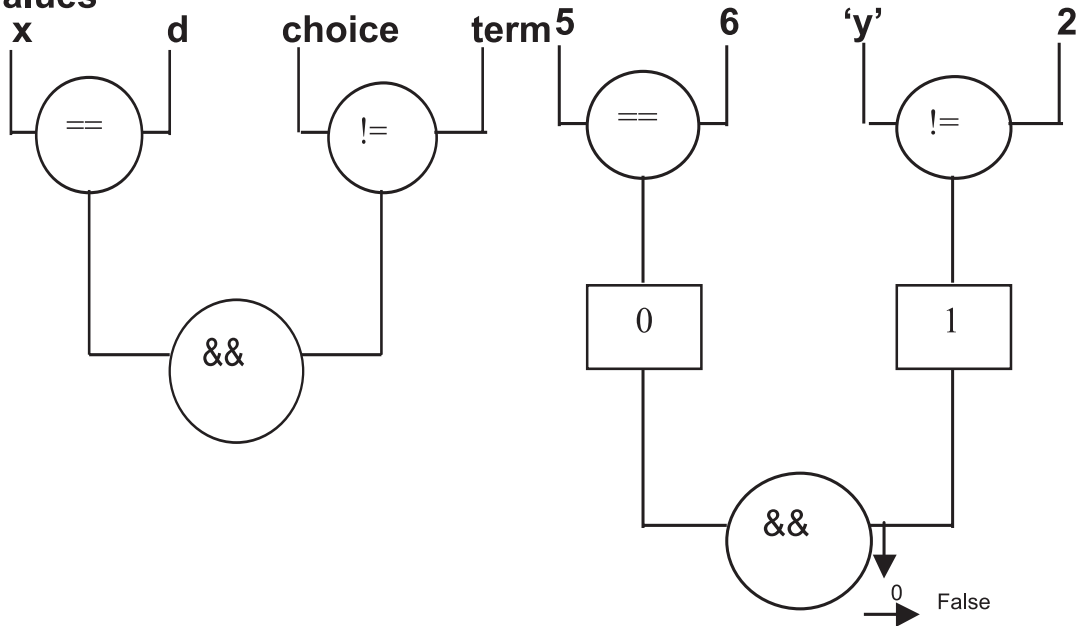
Result\_2 = ( 'y' == 'Y') || (term != '0')

Result\_3 = (c == d) && ('y' == 'Y') || choice != term

Result\_4 = (c == d) || ('y' == 'Y') && choice != term

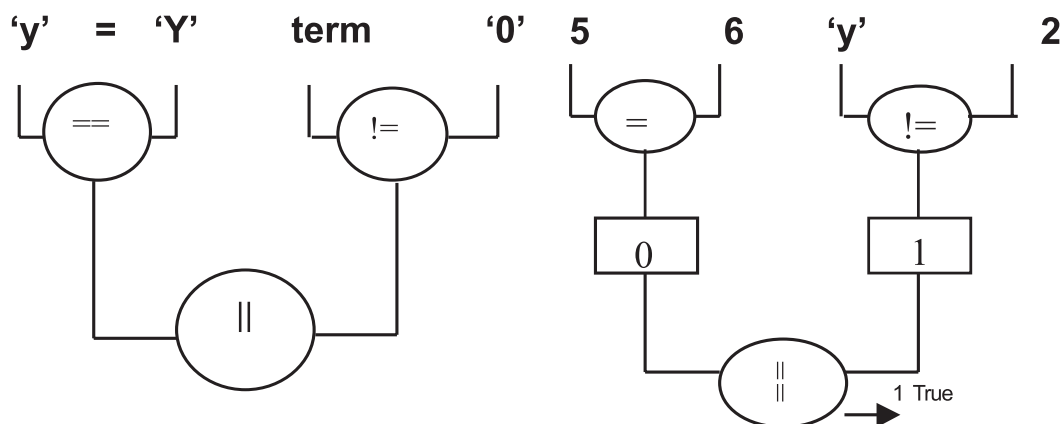
What will be the values stored in Result\_1 and Result\_2 ?? The values stored in Result\_1 is 0 (False) ; Result\_2 is 1 (True) , Result\_3 is 1 (True) and Result\_4 is 0 (False).

**Result\_1 = (c==d) && (choice != term)**  
values

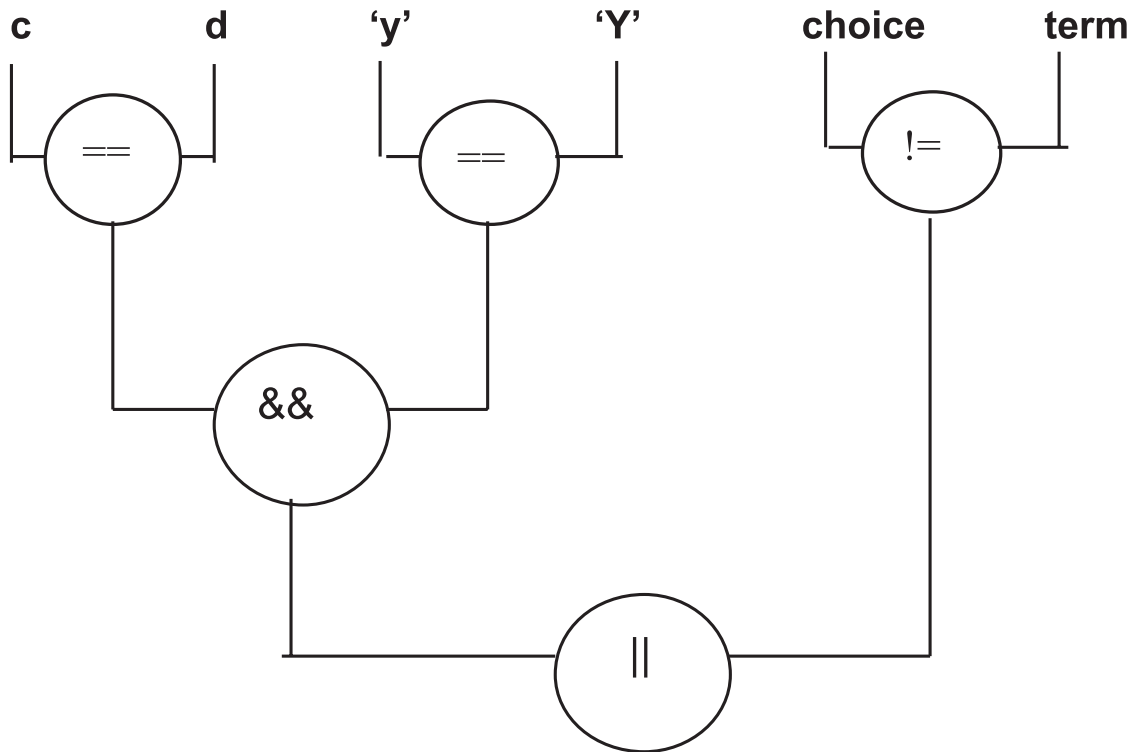


**Result\_2 = ('y'=='Y') || (term != 0)**

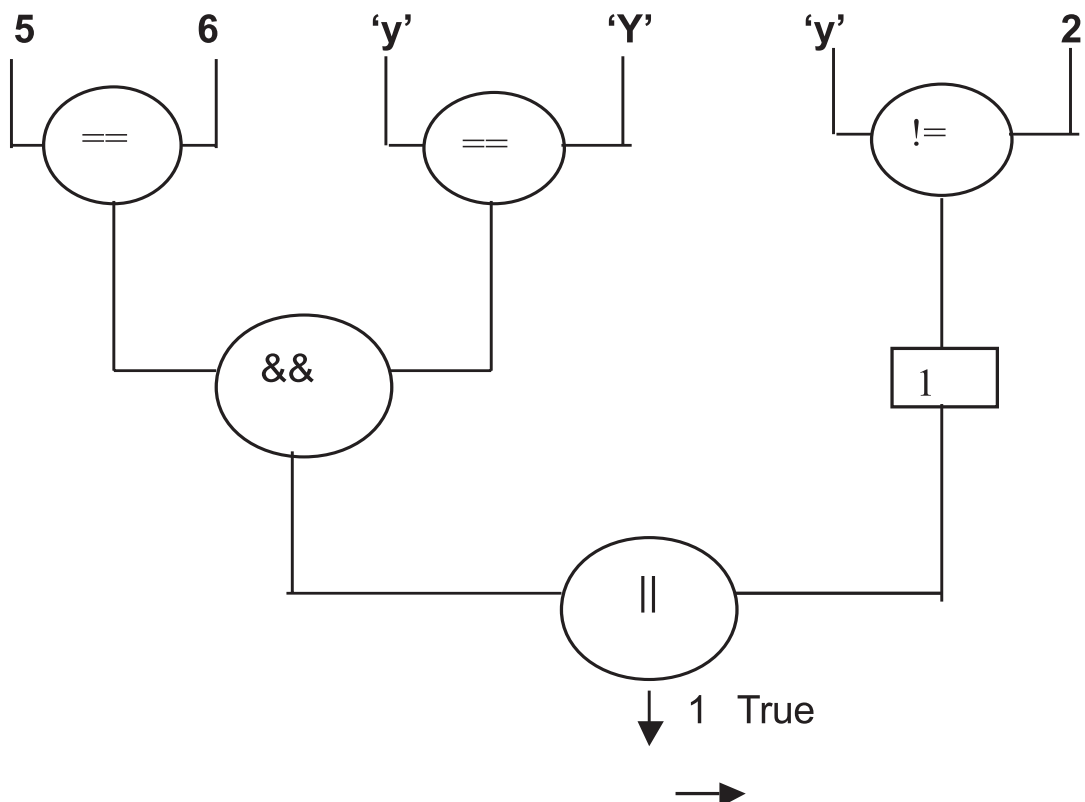
**By substituting values**



**Result\_3 = (c==d) && ('y' == 'Y') || (choice != term)**

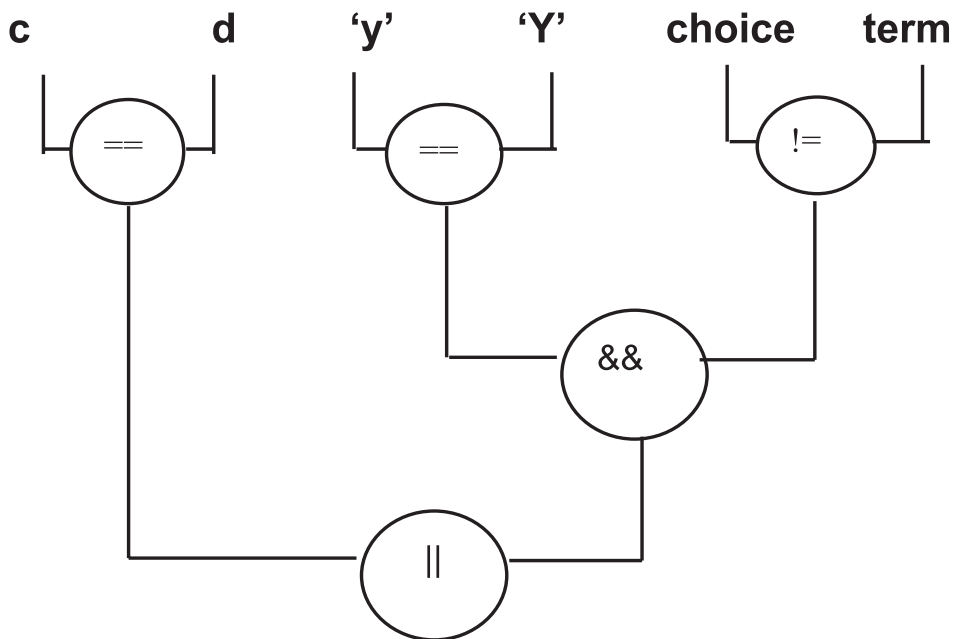


**Result\_3 By substituting values**

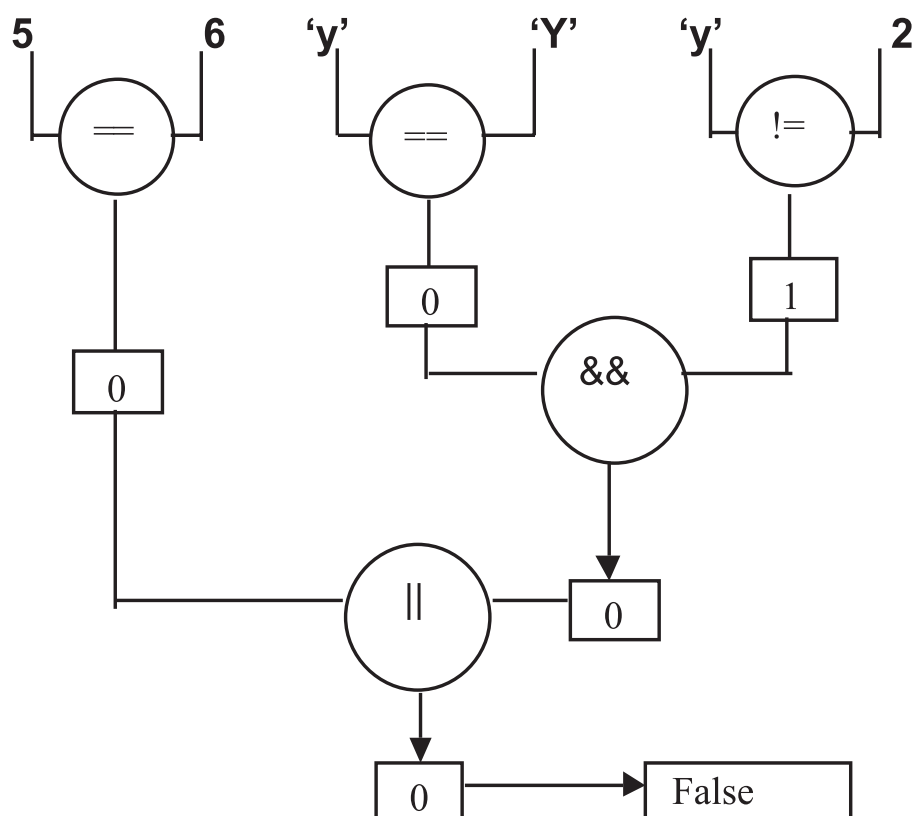




```
Result_4 = (c == d) || ('y' == 'Y') && choice != term
```

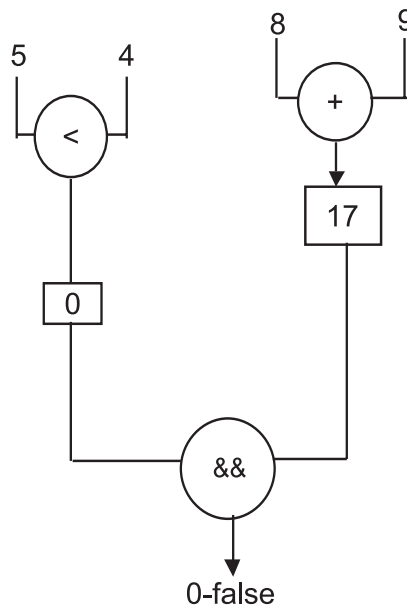


### Result\_4 By substituting values



The Logical operators have lower precedence to relational and arithmetic operators. Can you evaluate the value of the following expression ?

**5 < 4 && 8 + 9**



#### 11.2.7.4. Conditional Operator (?:)

(num1 > num2) ? "true" : "else" - ?: Is a ternary operator – (num1 > num2, "true", "false" are the operands. A ternary operator ( ? :) is also called as conditional operator. The general syntax is E1 ? E2 : E3 where E1, E2, E3 are operands. E1 should essentially be of scalar type, E2 and E3 are values or statements. For example to assign the maximum value of the two values one can express it as :

**max = (num1 > num2) ? num1 : num2;** The variable **max** will take the value of num1 if num1 is greater than num2, otherwise max will be assigned with the value of num2.

Can you write out what will be the value stored in **x** of the following snippet ?

```

a = 10
b = 10
x = ( a < b ) ? a*a : b % a;
  
```

#### 11.2.7.5. Assignment Operators

= is the simple assignment operator. It is used to assign the result of an expression (on the right hand side) to the variable (on the left hand side of the operator). In addition to the simple assignment operator, there are 10 'shorthand' assignment operators . Refer to the Table 11.9 for all assignment operators.

Expression	Working	Result
A = 5	The value 5 is assigned to the variable A.	The variable takes the value 5.
A += 2	A += 2 is interpreted as A = A + 2	The value stored in A is 7
A *= 4	A = A * 4	The value stored in A is 20
A /= 2	A = A / 2	The value stored in A is 2
A -= 2	A = A - 2	The value stored in A is 3
A %= 2	A = A % 2	The value stored in A is 1
Evaluate the following expressions where a = 5, b = 6, c = 7		
A += b*c C *= a + a / b B += a % 2 * c		

**Table 11.9 Assignment Operators**

Table 11.10 gives the complete Operator precedence of all operators used in C++

Operator Precedence	Type	Associativity
() []		
Postfix ++, --, prefix ++, -- ! –logical not + unary, - unary	Mathematical-Unary Logical – unarymathematical	Left to right Left to right Right to left Right to left Right to left Left to right
* / %	Mathematical –binary	Left to right
+ -	Mathematical– binary	Left to right
< <= > >=	Relational-binary	Left to right
= = !=	Relational-binary	Left to right
&& (AND)	Logical – binary	Left to right
(OR)	Logical – binary	Left to right
?:	Logical – ternary	Left to right
= *= /= %= += -= <<= >>= &= ^=  =	Assignment	Right to left

**Table 11.10 Operator Precedence**

(note : operators specific to C++ will be dealt with in their relevant topics)

### 11.2.8 Punctuators

Punctuators are characters with a specific function. Refer to the Table 11.11 for Punctuators and their Purpose.

Punctuators	Purpose
;	Terminates a C++ statement
//	Treats statements prefixed with this as comments
/* */	Blocks enclosed within these characters are treated as comment
{ }	Used to group a set of c++ statements. Coding for a function is also enclosed within these symbols
[ ]	Index value for an element in an array is indicated within these brackets
' '	Is used to enclose a single character
" "	Is used to enclose a set of characters

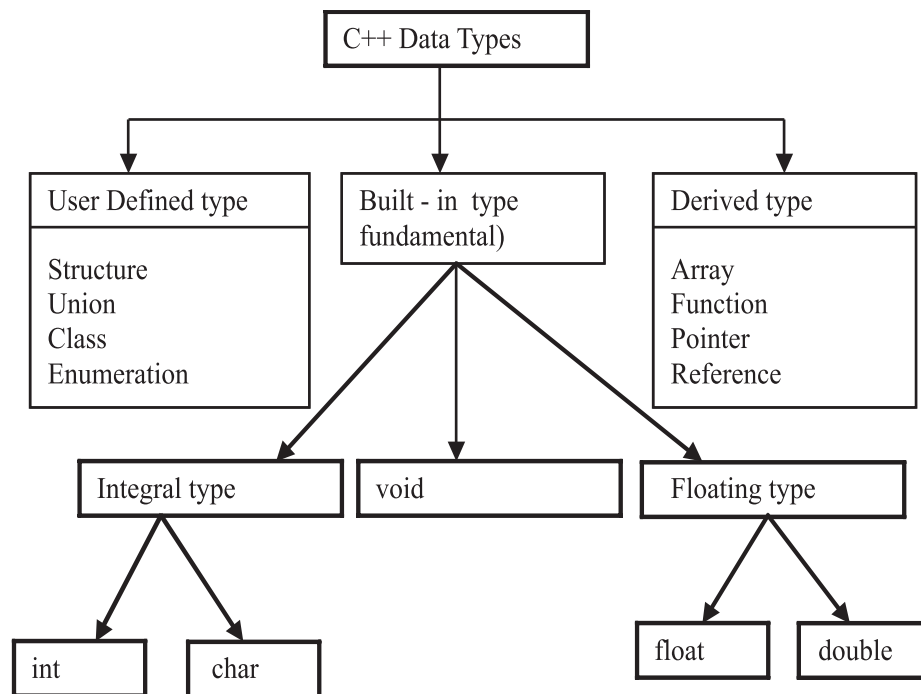
**Table 11.11 Punctuators and their Purpose**

### 11.3 Data Types

Data Types are the kind of data that variables hold in a programming language. The ability to divide data into different types in C++ enables one to work with complex objects. Data is grouped into different categories for the following two reasons :

- The compiler may use the proper internal representation for each data type
- The programmer designing the programs may use appropriate operators for each data type. They can be broadly classified into the following three categories.
  - User defined type
  - Built-in type
  - Derived type

The broader classification is indicated in the Fig. 11.2

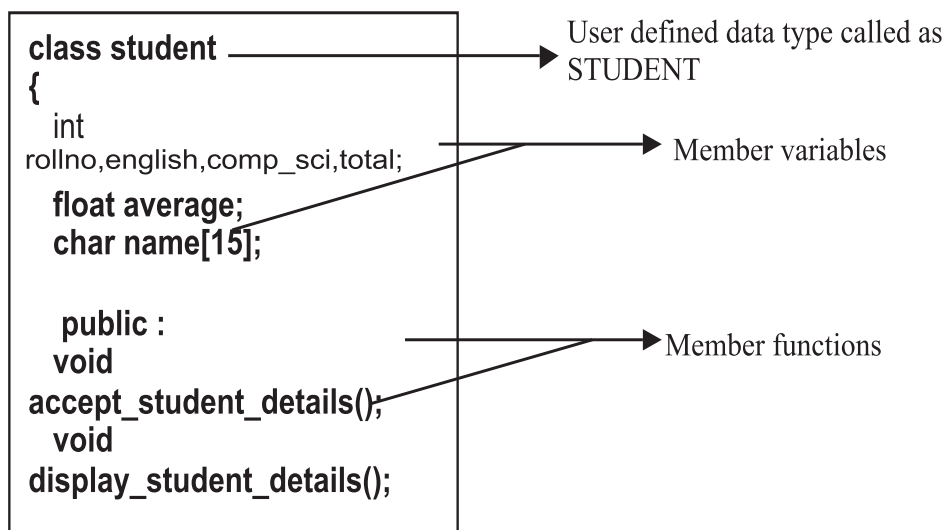


**Fig 11.2 C++ Data Types**

### 11.3.1 User Defined Data Type

User Defined Data Type enables a programmer to invent his/ her own data type and define values it can assume. This helps in improving readability of the program.

For example consider the following user defined data type



**Fig. 11. 3 User Defined Data Type**

**student** is a user defined data type of **class**. This data type defines the features of a student in terms of member variables, and the associated functions like accepting data for a student, displaying details, and also calculating their respective totals and averages. Thus the **class student improves the credibility and readability of the program by combining** the data requirements and its associated functions in the form of a data type for a student.

Users can define a variable that would represent an existing data type. “**Type definition**” allow users to define such user defined data type identifier. The syntax :

```
typedef    data_type    user_defined_data_type_identifier;
```

For example:

```
typedef int marks;
```

```
typedef char grade;
```

The data type identifiers **marks** and **grade** are user defined identifiers for int and char respectively. Users can define variables of int and char as follows:

```
marks eng_marks, math_marks;
```

```
grade eng_grade, math_grade ;
```

**typedef** helps in creating meaningful data type identifiers, that would increase the readability of the program.

Another user defined data type is they enumerated data type. As the name suggests, enumerated data type helps users in creating a list of identifiers, also called as symbolic numeric constants of the type int.

The syntax :

```
enum data type identifier (value 1, value 2, ... value n);
```

Examples :

```
enum working_days (Monday, Tuesday, Wednesday, Thursday, Friday);
```

```
enum holidays (Sunday, Saturday);
```

The identifiers **working\_days** , **holidays** are user defined data type. Monday, Tuesday ... is the list of values also called as **enumeration constants** or **numeric constants**.

Users can declare variables of this enumerated data type using the syntax :

```
enum identifier variable1, variable2 ...,variable n;
```

For example the variables `first_workingday` and `last_workingday` of the type `working_days` may be declared as follows:

```
working_days first_workingday, last_workingday;
```

These variables can take only one of the values defined for `working_days`.

```
first_workingday = Monday ;
```

```
last_workingday = Friday;
```

The enumeration constants (Monday, Tuesday, Wednesday...) are given integer constants starting with 0 (zero) by the compiler. The above assignment statements can also be rewritten as:

```
first_workingday = 0 ;
```

```
last_workingday = 4;
```

Users can also redefine these integer constants by assigning explicit values to the enumeration constants as

```
enum working_days (Monday = 1, Tuesday, Wednesday, Thursday, Friday);
```

Here, the constant Monday is assigned the value of 1. The remaining constants are assigned successive integer constants.

### 11.3.2. Storage Class

Storage Class is another qualifier (like long or unsigned) that can be added to a variable declaration. The four storage specifiers are **auto**, **static**, **extern** and **register**. static and register variables are automatically initialized to zero when they are declared. Auto variables are not initialized with appropriate values based on their data type. These variables get undefined values known as garbage. The following Table 11-12 gives the meaning and relevant examples.



Storage	Meaning	Example
<b>auto</b>	Defines local variable known to the block in which they are defined. By default the local variables are auto hence rarely used.	<pre>void main() { <b>auto</b>float ratio; int kount; }</pre> <p>The variables ratio and kount defined within the function main() have the storage specifier as auto.</p>
<b>static</b>	Variables defined within a function or a block cease to exist , the moment the function or the block loses its scope. <b>Static</b> modifier allows the variable to exist in the memory of the computer, even if its function or block within which it is declared loses its scope. Hence the variable also retains the last assigned value.	<pre>void fun(){ <b>static</b> int x; x++; }</pre>
<b>extern</b>	Global variable known to all functions in the current program. These variables are defined in another program.	<pre>extern int filemode;extern void factorial();</pre>
<b>register</b>	The modifier register instructs the compiler to store the variable in the CPU register to optimize access.	<pre>void fun(){ register int l;}</pre>

**Table 11.12 Storage Classes**

#### 11.3.4 Built in Data Types

Built in Data Types are also called as Fundamental or Basic data types. They are predefined in the compiler. Integral, Float and Void are the three fundamental data types.

Integral type is further divided into **int** and **char**. Int is the Integer data type. It cannot hold fractional values. **char** is character data type that can hold both the character data and the integer data. For example consider the declaration and initialization of the

variable **ch** - `char ch = 'A'`. The statement `char ch = 65` would also yield the same result of storing the value 'A' in the variable `ch` as character data type can hold both character and integer values.

Floating type is further divided into **float** and **double**. Floating type can store values with fractional part (Refer to floating point constants representation)

Void type has two important purposes :

- To indicate the a function does not return a value
- To declare a generic pointer

For example consider the following functions defined in C++ (Program `void.cpp` & `fun.cpp`).

```
Program void.cpp
#include<iostream.h>
#include<conio.h>
void fun(void)
{
    int a,b;
    cin >> a >> b;
    cout << a+b;
}

void main()
{
    fun();
}
```

```
Program fun.cpp
#include<iostream.h>
#include<conio.h>
int fun(int a, int b)
{
    return a+b;
}

void main()
{
    int a = 5 , b = 6, sum = 0;
    sum = fun(a,b);
    cout << sum;
}
```

In the example `void.cpp` the prototype `void fun(void)` indicates that the function does not return any value, nor does it receives values(in the form of parameters). Hence the call statement in the `main()` function is given as '**fun()**'. In the example `fun.cpp`, the prototype **`int fun(int a, int b)`** , instructs the compiler that the function `fun()` returns an integer value. Hence the call statement in the `main()` function is given as '**sum = fun(a,b)**'. The variable `sum` receives the value from the return statement (`return a+b`)

- ✓ **void** data type indicates the compiler that the function does not return a value, or in a larger context void indicates that it holds nothing.

Basic data types have several modifiers. These modifiers have a profound effect in the internal representation of data. signed, unsigned, long and short are some of the modifiers. Table 11.13 gives a list of the data types, memory allocation and range of values.

Type	Byte	Range
char	1	-128 to 127
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2	-32768 to 32767
unsigned int, unsigned short int	2	0 to 65535
signed int,short int, signed short int	2	-32768 to 32767
long int,signed long int	4	-2147483648 to 2147483647
unsigned long int	4	0 to 4294967295
float	4	3.4e-38 to 3.4e+38
double	8	1.7e-308 to 1.7e+308
long double	10	3.4e-4932 to 1.1e+4932

**Table 11.13 Data Types Size & Range of Values**

#### 11.3.4. Derived Data Type

These are built from the basic integer and floating type (built in type) or user defined data types. For example

```
int num_array[5]      =    {1,2,3,4,5};
```

```
chardayname[7][3]    =    {Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
```

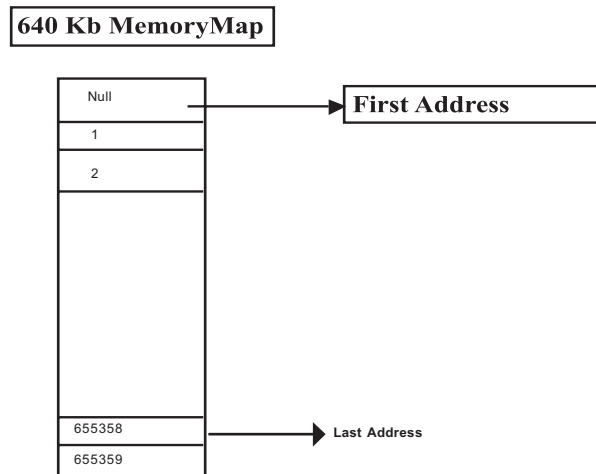
num\_array stores 5 values. Each element is accessed using the positional value of the element in the array. The position numbering commences from zero. num\_array[0] stores value 1 and num\_array[4] stores value 5.

Can you write as to what is stored in dayname [0], dayname [5] and dayname [3] [2] ?

##### 11.3.4.1 Pointers

A pointer is a variable that holds a memory address. Pointers provide the means through which the memory locations of a variable can be directly accessed. Every byte in the computer's memory has an address. Addresses are numbers just as our house numbers. The address number starts at NULL and goes up from there.. 1, 2 , 3.....

For example a memory size of 640 KB will have addresses commencing from NULL and goes up to 655, 358 as shown in Fig. 11.4.

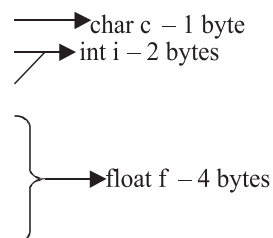


**Fig. 11.4 640 Kb Memory Map**

When a program is compiled, some memory is allocated to the variables by the compiler. The amount of memory allocated to each variable depends on the data type of the variable.

For example consider the declarations :

`char c ; int i; float f;`



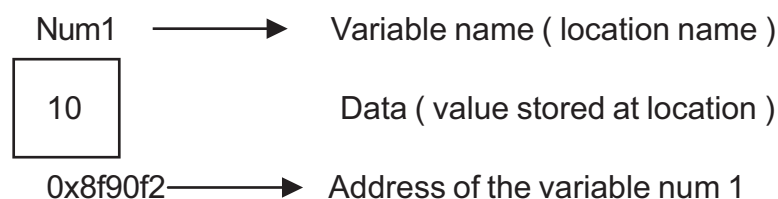
Every variable will be referred by its address. From our example the address of the variables c,i and f will be 1,2 and 4 respectively. Addressing is done using the hexadecimal system.

When dealing with pointer data type one needs to know about the **address of (&) operator** and the **value at operator (\*)**.

**The ‘ &’ operator :** When we type `int num1=10;`

the C++ compiler performs the following operations / actions : zz

- 1) Reserves space in the memory to hold the integer value
- 2) Associates the variable name num1 with a memory location
- 3) Stores the value 2 at this location in the memory



```

// Program – 11.1
// to demonstrate use of & operator
#include<iostream.h>
#include<conio.h>
void main()
{
    clrscr();
    int i = 10;

    cout << "\n Address of the variable... " <<&i;
    cout << "\nValue stored in the variable .." << i;

    getch();
}

```

Now consider this

```

int *x, num 1;
num1 = 10;
x=&num1;
cout<<*X;

```

Note :

The asterix ( \* ) is

- 1) Used to declare a pointer variable
- 2) Used to display the contents stored at a location ( value at the address operator )
- 3) It is a unary operator

## 11.4 Variables

The name assigned to a data field that can assume any of a given set of values is defined as the variable. For example consider the following group of statements

```

int num;
num = 5;

```

The statement **int num;** may be interpreted as “ num is a variable of the type integer “. The assignment statement **num = 5** may be interpreted as the value 5 is stored in the variable num.

**Variables** are user defined named entities of memory locations that can store data.

Variable names may contain letters, numbers and the underscore character(\_). Names must begin with a letter or underscore. (However names beginning with an underscore are reserved for internal system variables). Names are case sensitive, which means that it differentiates between lower case and upper case letters.

Complete the Table – 11.14.

Variable	Valid/Invalid	Reasons if invalid
A_b	Valid	
1a_b	Invalid	Variables must begin with an alphabet or an underscore only.
_Test		
Balance\$		
#One		
Include		

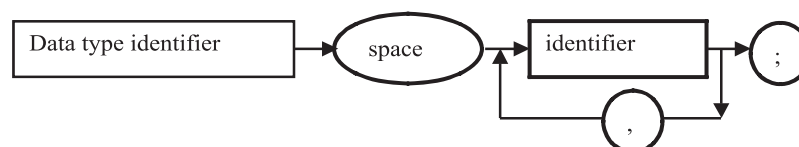
**Table 11.14 Validity of Variable Names**

#### 11.4.1 Declaration of Variables

Variables are allocated memory to store data. Compiler allocates memory, based on the data type of the variable. Hence variables must be declared before they are used.

Example :   int a;  
              float f1,f2;  
              char name[10],choice;

Syntax :



Consider the declaration **int side, float hypotenuse , area ;** This is an erroneous declaration because the compiler interprets this statement as follows :

- The variables side, float, hypotenuse and area will be treated as instances of the data type **int**. Hence it throws an error stating that “ **comma is expected after float**”

- The intention was to declare the variable side of int data type and the variables hypotenuse & area of the data type float.
- Hence the above declaration statement should be rewritten as follows :

```
int side ;
float hypotenuse , area ;
```

```
int side ; float hypotenuse,area ;
```

✓ **More than one variable of the same data type can be declared in a single declaration statement. But every variable should be separated by comma.**

There are nine words for data types such as **char** , **int** , **double** , **float**, **void**, **short**, **signed**, **long** and **unsigned** .

**long**, **short**, **signed** and **unsigned** are qualifiers or modifiers that modify a built – in data type with the exception of void.

The internal representation for the integer value 15 is 00000000 00001111 . Integer values are stored in 16 bit format in binary form. Starting from right extreme, 15 bits are used to store data. Maximum value stored in an integer variable is +32767 and the minimum value is –32768, as  $2^{15}$  is **32767 on the positive side and –32768 on the negative side. 16<sup>th</sup> bit, also called as the Most Significant Bit or sign bit. It is used to store sign. The 16<sup>th</sup> bit will have a value 1 if negative value is stored. When the modifier unsigned is used the integer data type will store only positive values, the sign bit is also used to store data. Therefore the range to store data goes upto  $2^{16}$  , hence the maximum value will be 65535.**

✓ **The modifier alters the base data type to yield new data type.**

**The impact of modifiers :**

- **unsigned** modifies the range of the integer values as the sign bit is also used to store data.
- **long** increases the bytes for a particular data type, thus increasing the range of values.

The base data type should be prefixed with the modifiers at the time of declaring a variable. For example :

```
unsigned int registration_number;
long unsigned int index;
short signed char c;
```



✓ **Prefix the data type with modifiers at the time of declaring variables.**

The **const** qualifier specifies that the value of a variable will not change during the run time of a program. Any attempt to alter the value of a variable defined with this qualifier will throw an error message by the compiler. The const qualifier is used like any other modifier where the variable is prefixed with the keyword const followed by data type .

For example :

**const float pi = 3.14;**

The Table 11.15 shows the **data types** with altered lengths and range of values when the qualifiers or modifiers are used

Data Types		
Type	Length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	$3.4 * (10^{**-38})$ to $3.4 * (10^{**+38})$
double	64 bits	$1.7 * (10^{**-308})$ to $1.7 * (10^{**+308})$
long double	80 bits	$3.4 * (10^{**-4932})$ to $1.1 * (10^{**+4932})$

**Table 11.15 Data Types with Modifiers**

### Declaring pointer variables

int \* iptr;

Name of the pointer variable

Instructs the compiler that the variable is pointer ( it will hold an address)

Indicates that the pointer will point to an int data type

The declaration statement `int *ptr` may be read as ptr is a pointer variable of the type int. The variable ptr can only store addresses that hold integer values.

Examples of pointer variable declarations:

char * cptr float * fptr	declaring a pointer to character type a pointer to float type
void *v ptr	a pointer that can point to any data type a generic pointer is declared in this way
const int * ptr	ptr is a pointer to a constant integer (cannot modify the value stored at the address pointed by ptr)
char * const cp	cp is a constant pointer. The address stored in cp cannot be modified

**Table 11.16 Examples of Pointer Variables**

#### 11.4.2 Initialization of variables

Variables are initialized to a specific value at the time of declaration. Initialization is done only once. For example :

```
int num = 10;
int fun(5)
```

In statement (1) the variable num is initialized to 10, whereas in the second statement the variable fun is initialized to 5 through a constructor.

**Implicit conversions:** refers to data type changes brought about in expressions by the compiler. For example consider the following snippet:

```
float f = 7.6;
int x = f;
```

The value stored in the variable x is 7, as float is converted to int. The compiler does this conversion automatically.

Rules for implicit conversion :

Consider a term, having a pair of operands and an operator. The conversions takes place as follows :

1. If one operand is of type **long double** , then the other value is also converted to long double.
2. If one operand is of type **double**, then the other value is also converted to double.
3. If one of the operands is a **float**, the other is converted to a float.
4. If one of the operands is an **unsigned long int**, the other is converted to unsigned long int.

5. If one of the operands is a **long int**, then the other is converted to long int.
6. If one of the operands is an **unsigned int**, then the other is converted to an unsigned int.

```
// demonstrating implicit type conversions
// Program – 11.2
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>

void main()
{
clrscr();

int i; float f;
double d;
long double ld;
unsigned int ui;
unsigned long int uli;
i = -5;
f = 2;
d = 3;
ld = 3;
ui = 6;
uli = 4;
cout << "\nSizeof long double.." << sizeof(ld*d) << '\t' << ld*d;
cout << "\nSizeof double..." << sizeof(d*f) << '\t' << d*f;
cout << "\nSizeof float..." << sizeof(f * i) << '\t' << f*i;
cout << "\nSizeof unsigned long int ..." << sizeof(uli* f) << '\t' << uli * f;
cout << "\nSizeof unsigned int..." << sizeof(ui * i) << '\t' << ui * i;
getch();
}
```

Note : **sizeof** is an operator . It returns the size (memory requirement) in terms of bytes, of the given expression or data type.

**Output displayed by the above program:**

<b>Sizeof long double</b>	<b>...10</b>	<b>9</b>
<b>Sizeof double</b>	<b>...8</b>	<b>6</b>
<b>Sizeof float</b>	<b>...4</b>	<b>-10</b>
<b>Sizeof unsigned long int</b>	<b>...4</b>	<b>8</b>
<b>Sizeof unsigned int</b>	<b>...2</b>	<b>65506</b>

Complete the following Table 11.17 based on the sample program 11.2 , write answers as shown in the reason column for the first value.

Sno.	Size of the result - in terms of bytes	Expression	Reason
1.	10	ld*d	The value generated is of long double type as the variable ld is long double. As long double data type requires 10 bytes to store a value, 10 is displayed.
2.	8	d*f	The value generated is of double type. ....
3.	4	f*l	
4.	4	uli * f	
5.	2	ui * i	

**Table 11.17 Exercise based on Program 11.2**

### Initialization of pointer variables

Pointer variables can store the address of other variables. But the addresses stored in pointer variables are not of the same data type as this pointer variable is pointing to. For example :

```
int *iptr, num1; num1 = 10;
iptr = &num1; // initializing a pointer variable
```

The following initialization is invalid.

```
int *iptr;
float num1 = 10.5;
iptr = &num1 // initializing pointer variable pointing to integer
data type with the address of float variable would throw an error.
```

**Pointer variables are sensitive to the data type they point to.**

**Type cast :** Type cast refers to the process of changing the data type of the value stored in a variable . The statement **(float) 7** , converts the numeric constant 7 to float type. Type cast is achieved by prefixing the variable or value with the required data type. The syntax is : (data type) <variable/value> or data type (variable/constant). Type cast is restricted only to fundamental or standard data types. The statement **x = 8 % 7.7** will throw an error on compilation as, modulus operator % operates on integer data type only. This erroneous statement can be corrected as **x = 8 % (int) 7.7** - the float constant 7.7 is converted to integer constant by type casting it.

Complete the following Table 11.18

int x; x = 7 / 3;	What is the value stored in X?
float x; x = 7 / 3;	What is the value stored in X?
float x; x = 7.0 / 3.0;	What is the value stored in X?
float x; x = (float) 7 / 3;	What is the value stored in X?
float x; int a = 7 , b = 3; x = a/b;	What is the value stored in X?
float x; int a = 7 , b = 3; x = a/ (float) b;	What is the value stored in X?

**Table 11.18 Find the value of X**

## EXERCISES

1. Determine the order of evaluation of the following expressions :

- i.  $a + \text{pow}(b, c) * 2$
- ii.  $a || b \&\& c$
- iii.  $a < b \&\& c || d > a$
- iv.  $(c \geq 50) || (!\text{flag}) \&\& (b + 5 == 70)$
- v.  $(a + b) / (a - b)$
- vi.  $(b * b) - 4 * a * c$

2. Identify errors in the following programs ..

a.

```
# include <iostream.h>
void main()
{
    float f = 10.0;
    x = 50;
    cout << x << f;
}
```

b.

```
# include <iostream.h>
void main()
{
    float f = 10.0;
    x = 50;
    cout << x << f;
}
```

c.

```
# include <iostream.h>
void main()
{
    int x,y,k,l;
    x = y + k---l;
    cout << x;
}
```

### 3. Predict the output

```
a.
# include <iostream.h>
# include <conio.h>

void main()
{
    int i=20;
    cout << i << i++ << ++i;
    getch();
}
```

```
b.
# include <iostream.h>
# include <conio.h>
void main()
{
    clrscr();
    int i = 1, a = 3;
    i = a++;
    cout << i;
    getch();
}
```

```
c.
# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    int i = 3, x;
    x = i ? i++ : ++i;
    cout << x;
    getch();
}
```

```
d.
# include <iostream.h>
# include <conio.h>
void main()
{
    int z, x = 3, y = 2;
    z = --x + y++;
    cout << z;
    getch();
}
```

```
e.
# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    char ch = 'a';
    ch = (ch == 'b') ? ch : 'b';
    cout << ch;
    getch();
}
```

### 4. Evaluate the following C++ expressions

Assume a = 5, b = 3, d = 1.5, c is integer and f is float.

- a.  $f = a + b / a$
- b.  $c = d * a + b$
- c.  $x = a++ * d + a;$
- d.  $y = a - b++ * --b;$
- e.  $(x \geq y) || (!(z == y) \ \&\& \ (z < x))$  where
  - ✓  $x = 10, y = 5, z = 11$  (all are integers)
  - ✓  $x = 10, y = 10, z = 10$
  - ✓  $x = 9, y = 10, z = 2$

### 5. Write the C++ equivalent expressions using the conditional operator. Where

- ✓  $f = 0.5$  if  $x = 30$ , otherwise  $f = 5$
- ✓  $f_n = 0.9$  if  $x \geq 60$ , otherwise  $.7$

6. What are pointer variables ?
7. Write a declarative statement to declare 'name' as a pointer variable that stores the address pointing to character data type.



## CHAPTER 12

# BASIC STATEMENTS

Basic Statements in C++ are constructed using tokens. The different statements are

- Input/output
- Declaration
- Assignment
- Control structures
- Function call
- Object message
- Return

### 12.1 Input/output statements

Input /Output statements such as reading data, processing data and displaying information are the essential functions of any computer program. There are two methods for assigning data to the variables. One method is by assignment statement which we have already seen in the earlier section, and the other method is to read data during the runtime of a program. Data is read from the keyboard during runtime by using the object **cin** (pronounced as C in). **cin** is a predefined object that corresponds to a standard input stream. Input stream represents the flow of data from the standard input device – the keyboard. cin can read data from other sources also which will be dealt later. The declarations for the object cin are available in a **header file** called as **istream.h**. The basic input/output operations are managed by a set of declarations available in the istream.h and ostream.h header files. Iostream.h file comprises the combined properties of istream and ostream.

- A header file comprises of all standard declarations and definitions for predefined functions.
- One can include the header file in the program by using a preprocessor directive
- A preprocessor directive starts with # , which instructs the compiler to do the required job.
- # include <iostream.h> is a typical preprocessor directive, that instructs the compiler to include the header file iostream.h In order to use cin / cout objects one has to include iostream.h in the program.
- The other header files are iomanip.h, stdio.h, ctype.h, math.h, fstream.h etc.

The >> is the extraction or get from operator. It takes the value from the stream object to its left and places it in the variable to its right. For example consider the following snippet :

```
float temperature;  
  
cin >> temperature;
```

The extraction operator (>>) extracts data from the input stream object (cin) and places the value in the variable(temperature) to its right. Multiple values can be read from the input stream and placed in the corresponding variables, by cascading the extraction operator. For example, to read the values for temperature and humidity one can perform it as follows :

```
cin >> temperature >> humidity;
```

**cout** pronounced as (C out) is a predefined object of standard output stream. The standard output stream normally flows to the screen display – although it can be redirected to several other output devices. The operator << is called the insertion operator or put to operator. It directs the contents of the variable to its right to the object to its left. For example consider the following statements;

```
int marks = 85;  
  
cout << marks;  
  
cout << "\n Marks obtained is : " << marks;
```

The value stored in marks is directed to the object cout, thus displaying the marks on the screen.

The second statement **cout << "\n Marks obtained is : " << marks;** directs both the message and the value stored in the variable to the screen. Cascading of insertion operator facilitates sending of multiple output via a single statement.

Examples :

```
cout << "\n The sum of the variables a,b .." << a+b;  
  
cout << a+b << '\t' << a-b << '\t' << a/b;  
  
cout << "\n The difference of numbers ...." << a-b << "\n  
The sum of two numbers .... " << a+b;
```

## 12.2 My first C++ program - Structure of a C++ Program

```
// My first program – Program 12.1
# include <iostream.h>//preprocessor directive
# include <conio.h>
float fact = 1; // declaration of variables
int term;
int main()    // function header
{
clrscr(); // predefined function
cout << "\n This program computes factorial of a number";
cout << '\n' << "Enter a number ..."; cin >> term;
for(int x = 2; x <= term;fact *= x,x++);// looping statement
cout << "\nThe factorail of the given number .."
<< term << " is .." << fact; return 0;
}
```

A C++ program has primarily three sections viz.,

- Include files
- Declaration of variables , data type , user defined functions.
- main ( ) function

On successful compilation, when the program is executed the main() function will be automatically executed. It is from this block, that one needs to give call statements to the various modules that needs to be executed and the other executable statements.

## 12.3 Declaration Statements

Variables used in the declaration statements need to be declared and defined before they are used in a program.

```
int *iptr; // declares a pointer variable to int
iptr = new int;//fetches memory to store data – hence pointer variable gets defined
*iptr = 5; // stores data 5 only after fetching memory
```

Declaration of a variable introduces a variable's name and its associated data type. For example consider the declaration `int *iptr`; This statement may be read as `iptr` is a pointer variable to integer. All pointer variables are defined only when memory is fetched to store data.

Declaration statements are used to declare user defined data type identifiers, function headers, pointer variables and the like. Recall the declaration of user defined data types dealt in 12.3

However, if a declaration also sets aside memory for the variable it is called as definition. For example consider the declaration statement - **int num;** This statement is called as definition statement because memory is set aside to store data. Now consider the following snippet :

`int num; // declares and defines an integer variable num = 5; // The data 5 is stored .` Have you noticed , there is no explicit request for memory. That is because memory is set aside at the time of declaring the variable.

- ✓ **Declaration statement introduces a variable name and associates it with a specific data type**
- ✓ **A variable gets defined when memory is set aside .**
- ✓ **Some variables also get defined when they are declared**
- ✓ **Pointer variables get defined only when memory is fetched. For example by using new memory operator**

## 12.4 Assignment Statements

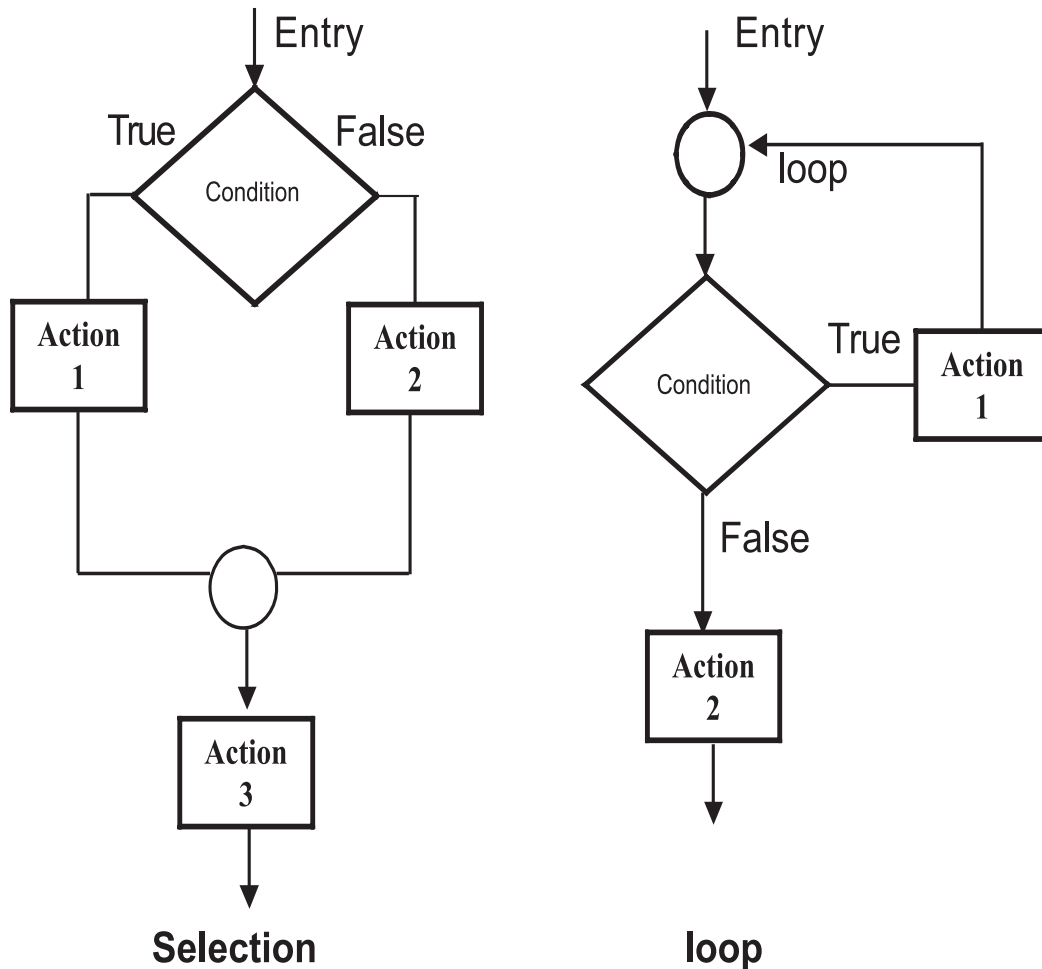
An assignment statement , assigns value on the right hand side of an expression to the variable on the left hand side of the assignment operator. '=' is the assignment operator . For example the different style of assigning values to the variables are as follow :

```
num = 5;
total = english+maths;
sum += class_marks;
```

During assignment operation , C++ compiler converts the data type on the right hand side of the expression to the data type of the variable on the left hand side of the expression. Refer to implicit conversions and Type cast of 11.4.2.

## 12.5 Control Structures

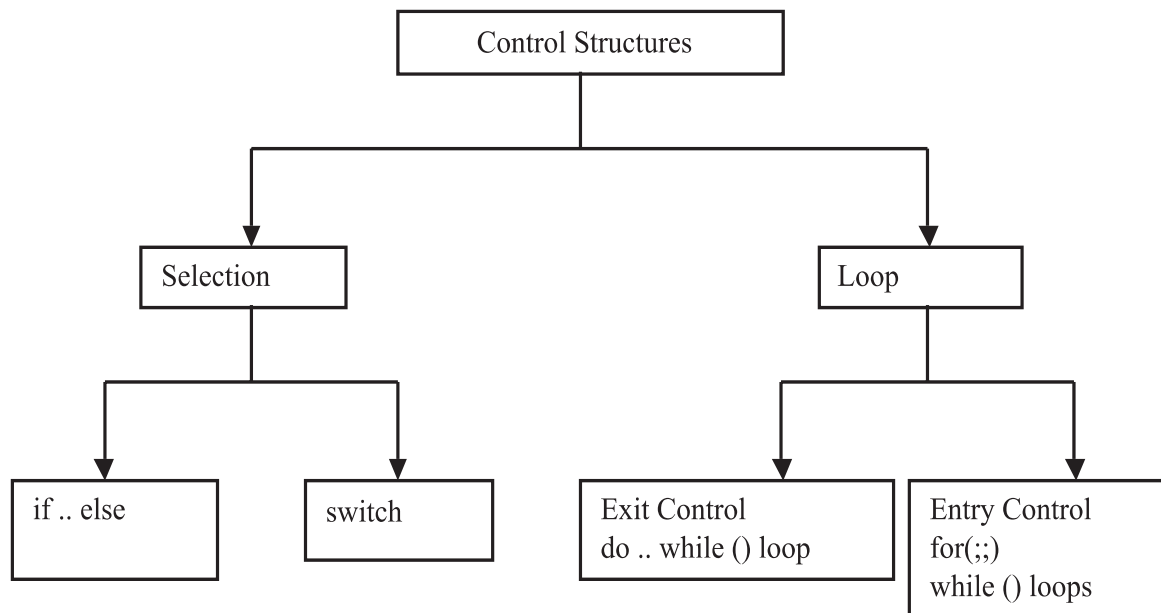
Statements in a program need not necessarily be executed in a sequential order. Some segments in a program are executed based on a condition. In such situations the flow of control jumps from one part of the program to another. Program statements that cause such jumps are called as control statements or control structures. Now look at the following flow charts (Flow chart I & II).



1. Trace out the steps to accept an integer and if it is odd add 1 to it. If it is even do nothing. Print the integer as depicted in Flow Chart 1. The steps are executed in a sequential manner.
2. Trace out the steps to accept a integer, and print the message "EVEN" /"ODD" based on the divisibility of 2. Here the control branches to statement " M = ODD" if there is remainder other wise branches to the statement "M = EVEN". This is depicted in Flow Chart 2.

✓ **Program statements that cause a jump of control from one part of a program to another are called Control Structures**

The two major categories of control structures are Decision making statements and Looping statements. The control structures are implemented in C++ using control statements as indicated in the following figure Fig. 12.1



**Fig. 12.1 Control Structures in C++**

### 12.5.1 Selection Statements

In a program a decision causes a one time jump to a different part of a program. Decisions in C++ are made in several ways, most importantly with **if .. else ...** statement which chooses between two alternatives. Another decision statement , **switch** creates branches for multiple alternatives sections of code, depending on the value of a single variable.

**if statement :** is the simplest of all the decision statements. It is implemented in two forms

- Simple if statement
- if .. else statement

```

if ( condition / expression)
{
    action block
}
  
```

```

If (condition / expression )
{
    action block 1
}
else
{
    action block 2
}
  
```

The following Program - 12.2 demonstrates **if statement** :

```
// Program - 12.2
# include <iostream.h>
# include <conio.h>
// Demonstrates the use and syntax of if statement
void main ()
{
int a;
clrscr();
cout << "\nEnter a number ";
cin >> a;
if ( a%2 == 0)
cout << "\nThe given number " << a << "is even";
getch();
}
```

In the above program the message "The given..." gets printed if the condition is evaluated to true, otherwise the control jumps to getch(); statement directly by passing the statement cout << "\nThe given ....

The following Program -12.3 demonstrates **if .. else ..** statement :

```
// Program – 12.3
// Demonstrates the use and syntax of if else statement
# include <iostream.h>
# include <conio.h>
void main()
{
int a;
clrscr();
cout << "\nEnter a number ";
cin >> a;
if ( a%2 == 0)
cout << "\nThe given number " << a << "is even";
else
cout << "\nThe given number " << a << "is odd";
getch();
}
```



In the above program “**The given number 10 is even**” is printed if the expression is evaluated to true, otherwise statement following else option will be executed.

Examples of if constructs where conditions/expressions are given in different styles :

Condition is expressed using the variable branch

```
int branch = 10 > 20;
if (branch )
{
    action block 1;
}
else
{
    action block 2;
}
```

Condition is expressed as 1, as any positive integer indicates TRUE state

```
if (1)
{
    action block 1;
}
else
{
    action block 2;
}
```

Expression is used for condition. If the value of the expression is evaluated to > 0 then action block 1 is executed other wise action 2 is executed.

```
if (a % 2 )
{
    action block 1;
}
else
{
    action block 2;
}
```

Can you predict as to what will be printed when the following program is executed ?

```
// Program - 12.4
# include <iostream.h>
# include <conio.h>
void main()
{
int count = 1;
if (count > 0)
{
cout << "\nNegating count ....";
count *= -1;
}
else
{
cout << "\nResetting count ...";
count *= 1;
}
getch();
}
```

Output displayed will be **Negating count ...** Why do you think block associated with else option is not executed since count was multiplied by -1 ?

Answer to this is that , once the true block is executed in an if .. else statement, then the else block will not be executed.

Else block is executed only if True block is not executed.

✓ **if .. else ...** statement which chooses between two alternatives , executes the chosen block based on the condition.

The following if constructs are invalid because :

Sno	Invalid construct	Why invalid ?
1.	if a> b cout << "True";	Condition should always be enclosed in a pair of brackets . The correct form is <b>if (a&gt;b)</b> cout << "Condition should True";
2.	if ( a> b) a—; cout<<"\nVariable is decremented"; else a++; cout << "Variable is incremented .."	Error thrown by the compiler is <b>"Misplaced else"</b> . If the action block is compound statements, then it should be enclosed in curly braces .
		The correct form is : if ( a> b) { a--; cout<<"\nVariable is decremented"; }else { a++;    cout << "Variable is incremented .." }
3.	if (a > b); cout << "Greater.. ";else cout << "Lesser ..";	The semicolon placed after condition nullifies the effect of if statement , the compiler throws an error <b>"Misplaced else"</b> .The correct form :if (a > b) cout << "Greater.. ";else cout << "Lesser ..";

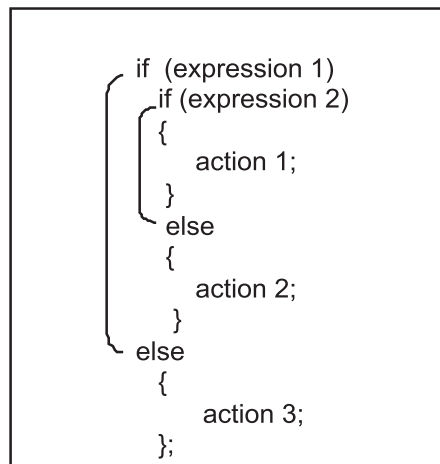
**Table 12.1 if construct**

Write appropriate if constructs for the tasks mentioned in table 12.2

Sno	Task	If construct
1.	Set Grade to 'A' if marks are above 90.	
2.	Set Grade to 'A' if marks are above 90, otherwise set grade to 'B'	
3.	Print the message <ul style="list-style-type: none"> <li>• "Accelerate – traffic to flow " if speed is less than 30 kmph,</li> <li>• "Moderate – accelerate by 10kmph" if speed is between 31– 40 kmph, other wise</li> <li>• "Good – be careful .."</li> </ul>	

**Table 12.2 Using if Constructs**

**Nested if statement :** The statement sequence of if or else may contain another if statement ie., the if .. else statements can be nested within one another as shown below :



In an nested if .. else statement, **“Each else matches with the nearest unmatched preceding if”**

For example

```

if (grade == 'A')
if (basic > 5500)
incentive = basic * 10/100;
else
incentive = basic * 5/100;
else
cout << "Try to attain Grade A";

```

Working of the above example :

- Grade = 'A' and basic == 5501, then incentive gets the value 550.
- Grade = 'A' and basic == 5000, then incentive gets the value 250.
- Grade <> 'A' – the inner if will not be executed , the outer else will be executed and thus prints “Try to attain Grade A”.

Do you think this if construct is equivalent to the above construct ? Write your answers in the Reason it out box.

```

if (grade == 'A' && basic > 5500)
    incentive = basic * 10/100;
else if (grade == 'A' && basic <5501)
    incentive = basic * 5/100;
else
    cout << "Try to attain Grade A";

```

Reason it out .....

**switch Statement :** This is a multiple branching statement where, based on a condition, the control is transferred to one of the many possible points.

This is implemented as follows :

<pre>switch (expression) {     case 1 : action block 1;               break;     case 2 : action block 2;               break;     case 3 : action block 3;               break;     default :               action block 4; }</pre>	<pre>switch (remainder) {     case 1 : cout &lt;&lt; "remanider 1";               break;     case 2 : cout &lt;&lt; "remanider 2";               break;      default :               cout &lt;&lt; "Divisible by 3"; }</pre>
--	--

The following program demonstrates the use of switch statement.

```
// Program - 12.5
// to demonstrate the use of switch statement

# include <iostream.h>
# include <conio.h>

void main()
{
    int a, remainder;
    cout << "\nEnter a number ...";
    cin >> a;
    remainder = a % 3;
    switch (remainder)
    {
        case 1 : cout << "\nRemainder is one"; break;
        case 2 : cout << "\nRemainder is two"; break;
        default: cout << "\nThe given number is divisible by 3";
        break;
    }
    getch();
}
```

The above program displays

- Remainder is two if a = 5 or so
- The given number is divisble by 3, if a = 9 or so

Or in other words the above program checks for divisibility by 3 and prints messages accordingly.

What do you think will be the output of the following program ?

```
// Program - 12.6
// to demonstrate the use of switch statement

# include <iostream.h>
# include <conio.h>

void main()
{
int rank = 1;
char name[ ] = "Shiv";
switch (rank)
{
case 1 : cout << '\n' << name << " secured 1st rank";
case 2 : cout << '\n' << name << " secured 2nd rank";
}
getch();
}
```

Output displayed will be :

Shiv secured 1st rank

Shiv secured 2nd rank

Why do you think both the action blocks of case 1 and case 2 are executed ? Compare the action blocks of Program -12 .5 & Program- 12.6. What do you think is missing in Program-12.6 ? Yes it is the **break;** statement.

What do we infer ?

Every action block should be terminated with a break statement. Otherwise all action blocks are executed sequentially from the point where the control has been transferred based on the condition.

In the above example(Program- 12. 6), control was transferred to case 1, as Rank is 1, hence action blocks of case 1 and case 2 are executed sequentially.

✓ Include **break;** in action block, in order to exit from switch statement.

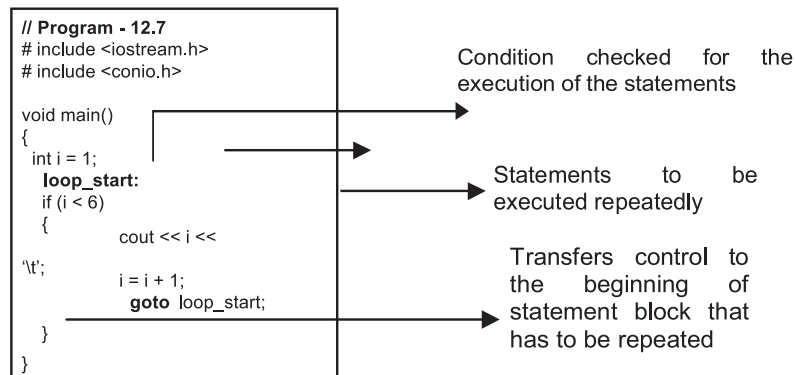
The following switch constructs are invalid because :

1.	<pre>char name[ ] = "Shiv"; switch (name) { case "Shiv" : cout &lt;&lt; '\n' &lt;&lt; name &lt;&lt; " secured 1st rank"; case "Rama" : cout &lt;&lt; '\n' &lt;&lt; name &lt;&lt; " secured 2nd rank"; }</pre>	<p>Compiler throws an error. "Switch selection expression must be of integral type "which means that switch expression should be evaluated to an integer constant only (char, enum,int)</p>
2.	<pre>float value; switch (value) { case 1.5 : cout &lt;&lt; '\n' &lt;&lt; value - 0.5; case 2.9 : cout &lt;&lt; '\n' &lt;&lt; value + 0.1; }</pre>	<p>Value is of float type , hence not a valid switch expression.</p>
3.	<pre>switch (rank) { case 1 to 2 : cout &lt;&lt; '\n' &lt;&lt; "Best rank"; break; case 3 to 4 : cout &lt;&lt; '\n' &lt;&lt; "Good rank"; }</pre>	<p>Case 1 to 2 is an invalid case statement, as case label should have only one integral value. In order to use more than one value for a particular action block one may rewrite the code as :</p> <pre>switch (rank) { case 1 : case 2 : cout &lt;&lt; "Best rank"; break; case 3 : case 4 : cout &lt;&lt; "Good rank"; break; }</pre>



## 12.5.2. Loops

Loops execute a set of instructions repeatedly for a certain number of times. For example consider the following Program – 12.7.



The above program on execution will print numbers between 1 and 5, as the action block of **if statement** is executed 5 times.

The Program - 12. 7 works as follows :

1. Declares and initializes the variable i
2. Checks the relational expression `i<6`
3. If True then executes the action block ( `cout << i; i = i + 1`) and transfers the control back to the `loop_start` (**goto** `loop_start`). This enables the program to execute a set of instructions repeatedly, based on the condition of the relational expression. The variable `i` is referred to as the control variable, as the iterations of the block is totally controlled by this variable.

A looping block therefore consists of two segments viz., the body of the loop and the control statement. The control statement checks the condition, based on which directs the control back to the body of the loop to execute the segment repeatedly. Now look at the following snippets.

<pre>//Program - 12.7 A void main() {     int i = 6;     loop_start:     if (i &lt; 6)     {         cout &lt;&lt; i &lt;&lt; '\t'; i = i + 1;         goto loop_start;     }     cout &lt;&lt; i; }</pre>	<pre>//Program - 12.7 B void main() {     int i = 6;     loop_start:     {         cout &lt;&lt; i &lt;&lt; '\t'; i = i + 1;         if (i &lt; 6)             goto loop_start;     }     cout &lt;&lt; i; }</pre>
--	--

What do you think will be the output generated by the above snippets ?

The Program -12.7 A will display 6, where as Program -12.7 B will display 7. Why do you think the loop is executed in Program-12.7 B? In this program the condition is placed after the statements (cout << i; i = i + 1;),hence these statements are executed once, after which the condition is checked. Since the variable i takes the value as 7, the control is not transferred to loop\_start. So what do we infer ??

- ✓ **Loops are unconditionally executed at least once, if the condition is placed at the end of the body of the loop**
- ✓ **Based on the position of the condition, the loops are classified as Entry-Check loop (as in Program-12.7 A) and Exit Check Loop (as in Program-12.7 B)**

In general, a looping process would work in the following manner :

1. Initializes the condition variable
2. Executes the segment of the body
3. Increments the value of the condition variable as required
4. Tests the condition variable in the form of a relational expression. Based on the value of the relational expression the control is either transferred to the beginning of the block, or it quits the loop.

There are three kinds of loops in C++, the **for** loop, the **while** loop and the **do .. while** loop.

**do .. while Loop** : The construct of a do .. while loop is :

```
do
{
    action block
} while <(condition)>
```

Look at the following program

```
// Program - 12.8
#include <iostream.h>
#include <conio.h>

// to print the square of numbers
// between 2 to 5

void main()
{
    clrscr();
    int num = 2;
    do
    {
        cout << num * num << '\t';
        num += 1;
    }
    while (num < 6);
    getch();
}
```

Answer the following questions based on the Program - 12.8

- A. Identify the
  1. control variable used .
  2. Identify the statements that form the body of the loop
  3. The test expression
- B. How many times will the loop be executed ?
- C. What is the output of the program?
- D. What type of loop is this ?

- A.**
  - 1. The control variable is num**
  - 2. Statements forming the body of the loop are :**  
`cout << num * num << '\t'; num += 1;`
  - 3. num < 6 is the test expression**
- B. 4 times**
- C. 4    9        16        25**
- D. Exit – check loop**

1. Enters the loop
2. Prints the square of num
3. Increments the control variable by 2
4. Evaluates the condition , based on which the control is transferred to step 2
5. End

do ... while <(condition)> is called as exit- check loop, as the condition(test expression) marks the last statement of the body of the loop. The following snippets show the various styles of constructing conditions.

```
Int ctr = 1, sum = 0, check = 1;
do
{
    cout<<ctr;
    sum = sum + ctr;
    ctr = ctr + 2;
    check = (ctr<11);
} While(check);
```

```
Int ctr = 5, sum = 0;
do
{
    cout<<ctr;
    sum = sum + ctr;
    ctr = ctr - 2;
}While(ctr);
```

```
int ctr = 5,sum = 0,c=1;
do
{
    cout << ctr;
    sum = sum + ctr;
    ctr = ctr - 2;
}while(ctr >= 1);
```

What is the output displayed by the following snippets A and B ?

```
// snippet A
# include <iostream.h>
# include <conio.h>

void main()
{
    int i = 10;
    do
    {
        cout << i;
        i--;
    } while (i <= 10);
    getch();
}
```

```
//snippet B
# include <iostream.h>
# include <conio.h>

void main()
{
    int i = 10; choice = 1;
    do
    {
        cout << i;
        i++;
    }while (choice);
    getch();
}
```

Snippet A – the loop will be executed till the variable i gets the value as –32768, and the snippet B will result in infinite loop, as the value stored in the variable choice is 1 thus rendering the test expression to be TRUE all the time in both the snippets . It is very important to construct appropriate conditions that would evaluate to false at some point of time, and also incrementing/updating the control variable that is linked to the test expression in the while loop.

**while <(condition)>{ ... } loop** : is called as the **entry-check** loop. The basic syntax is :

```
while <(condition)>
{
    action block;
}
```

The body of the while loop will be executed only if the test expression results true placed in the while statement. The control exits the loop once the test expression is evaluated to **false**. Let us rewrite all the programs that were discussed under do.. while loop (Program - 12.9)

```
// Program - 12.9
# include <iostream.h>
# include <conio.h>

// to print the square of numbers
// between 2 to 5

void main()
{
    clrscr();
    int num = 2;
    while (num < 6)
    {
        cout << num * num << '\t';
        num += 1;
    }
    getch();
}
```

Condition (test expression)  
is placed at the entry  
of the body of the loop

The working of the above loop as follows :

1. Initialises the control variable num to 2
2. The test expression num < 2 is evaluated, control is transferred to step 3, only if the test expression is **TRUE**
3. Prints the square of the value stored in num
4. Increments num by 1
5. Control is transferred to step 2
6. End

**Answer the following questions based on the Program - 12.10**

```
//Program-12.10  
# include <iostream.h>  
# include <conio.h>  
void main()  
{  
int x = 3, y = 4, ctr = 2,res = x;  
while(ctr <= y)  
{  
res *= x;  
ctr += 1;  
}  
cout << "x to the power of y is : "  
<< res;  
getch();  
}
```

**Answer the following questions based on the Program - 12.10**

**A. Identify the**

- 1. Control variable used .**
- 2. Statements that form the body of the loop**
- 3. The test expression**

**B. How many times will the loop be executed ?**

**C. What is the output of the program?**

**D. What type of loop is this ?**

**Answers :**

- 1. Control variable used is ctr**
- 2. res \*= x; ctr += 1;**
- 3. ctr <= y**
- B. 3 times**
- C. 81**
- D. Entry- check or entry – controlled loop**

What will be the output of the following Program - 12.11 if the values read for choice is y,y,y,y,n?

```
// Program - 12.11
# include <iostream.h>
# include <conio.h>
void main()
{
clrscr();
int counter = 0;
char choice = 'y';
while (choice == 'y')
{
cout << "Continue <y/n> ...";
cin >> choice;
counter = counter + 1;
}
cout << "\The loop is executed .."
<< counter << " times";
getch();
}
```

**The following snippets are invalid. Why are they invalid ? Correct the code for proper execution.**

```
//Program - 12 A
# include <iostream.h>
# include <conio.h>
//to print numbers between
5&10
void main()
{
int start = 5,end = 10;
while (start >= end)
cout << start++;
getch();
}
```

```
//Program - 12 B
# include <iostream.h>
# include <conio.h>
//to print numbers between 5&10
void main()
{
int start = 5,end = 10;
while (start <= end)
cout << start++;
getch();
}
```



```
//Program – 13 A
# include <iostream.h>
# include <conio.h>
// to print numbers between
10&5
void main()
{
    clrscr();
    int start = 5,end = 10;
    while (start <= end)
        cout << start--;
    getch();
}
```

```
//Program – 13 B
# include <iostream.h>
# include <conio.h>
// to print numbers between 10&5
void main()
{
    clrscr();
    int start = 5,end = 10;
    while (start <= end)
        cout << end--;
    getch();
}
```

```
//Program – 14 A
# include <iostream.h>
# include <conio.h>
// to print numbers
        between 1 & 5
void main()
{
    clrscr();
    int start = 1;
    while (Start <=5)
        cout << start++;
    getch();
}
```

```
//Program – 14 B
# include <iostream.h>
# include <conio.h>
// to print numbers
        between 1 & 5
void main()
{
    clrscr();
    int start = 1;
    while (1)
        cout << start++;
    getch();
}
```

**for ( ; ; ) .. loop** : is an entry controlled loop and is used when an action is to be repeated for a predetermined number of times. The syntax is

**for(initial value ; test-condition ; increment)**

```
{
action block;
}
```

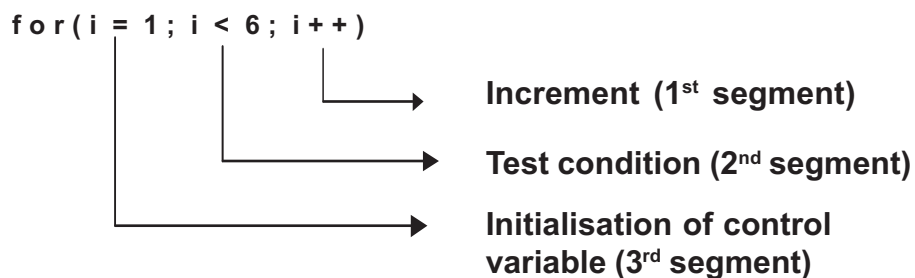
The general working of for(;;)loop is :

1. The control variable is initialized the first time when the control enters the loop for the first time
2. Test condition is evaluated. The body of the loop is executed only if the condition is TRUE. Hence for(;;) loop is called as entry controlled loop.
3. On repetition of the loop, the control variable is incremented and the test condition will be evaluated before the body of the loop is executed.
4. The loop is terminated when the test condition evaluates to false.

The following program illustrates for(;;) loop :

```
// Program - 12.15
# include <iostream.h>
# include <conio.h>

void main()
{
int i,fact = 1;
for(i = 1; i < 6; i++)
fact *= i;
cout << "\nThe factorial of the number is .." << fact;
}
```



- ✓ **Initialisation is executed only once, ie., when the loop is executed for the first time**
- ✓ **Test condition is evaluated before the commencement of every iteration**
- ✓ **Increment segment is executed before the commencement of new iteration.**

Now look at the following programs and write out as to what will be displayed?

```
// Program – 12.16
# include <iostream.h>
# include <conio.h>

void main()
{
int ctr = 10;
for(; ctr >= 6; ctr—)
cout << ctr << '\n';
}
```

**Output**

10  
9  
8  
7  
6

```
//Program – 12.17
#include <iostream.h>
#include <conio.h>

void main()
{
    clrscr();
    for(int i=2,fact =1;i<6;fact*=i,i++)
        cout << "\nThe factorial .." << fact;

    getch();
}
```

Output displayed..

The factorial.. 120

Have you noticed the for statement, comprising of more than one statement in segments incrementation and initialisation ? Syntatically and logically the above statement is valid. Each segment in the for loop can comprise a set of instructions, each instruction should be separated by a comma operator. Can you analyse as to what will be the output of the following segment ?

```
void main()
{
    for (int i = 1, j = 0 ; i < 8,j<3;i++,j++)
        cout << i << '\t';
    for (int i = 1,,j = 0 ; j < 3,i < 8;i++,j++)
        cout << i << '\t';
}
```

Output produced will be :  
 1 2 3 // loop is executed till j < 3  
 1 2 3 4 5 6 7 // loop is executed  
 till i < 8 Recall the working of comma  
 operator.

Now look at the following for..loop constructs.

```
// Program – 12.18
#include <iostream.h>
#include <conio.h>

void main()
{
    clrscr();
    int sum =0, ctr = 1;
    for(;ctr <= 5;)
    {
        sum += ctr;
        ctr = ctr + 1;
    }
    cout << "\nSum :." << sum;
    getch();
}
```

Output displayed will be  
 Sum : 15

Have you noticed,  
 initialization and  
 incrementation segments  
 are not included in the  
 for(..) construct.

```
// Program – 12.19
# include <iostream.h>
# include <conio.h>
void main()
{
    clrscr();
    int sum =0, ctr = 1;
    char ch ='y';
    for(;ch == 'y';)
    {
        sum += ctr;
        ctr++;
        cout << "\nContinue <y/n>
? ..";
        cin >> ch;
    }
    cout << "\nSum : " << sum;
    cout << "\nChoice : " << ch;
    getch();
}
```

```
Continue <y/n> ? ..y
Continue <y/n> ? ..y
Continue <y/n> ? ..y
Continue <y/n> ? ..n
```

sum:10

Choice : n

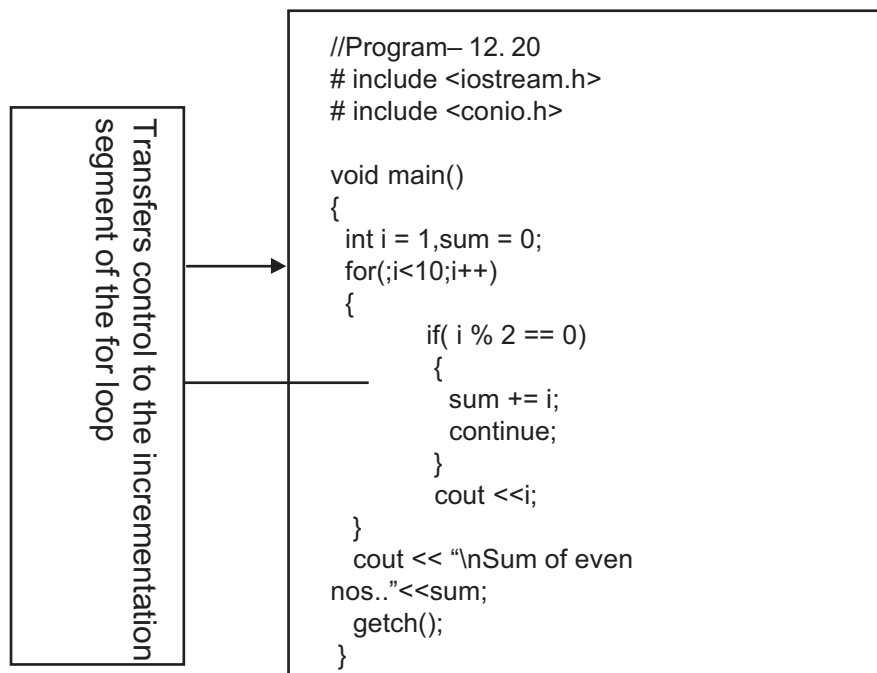
Have you noticed that a for loop is used like a dynamic loop, where the iterations are determined during run time.

What is wrong with the following snippets ?

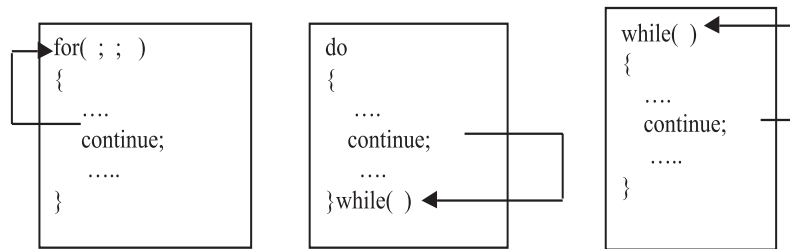
What is the impact of the following statements ?  
int sum = 0;  
for(ctr = 1; ctr < 5; ctr++); sum += ctr;  
cout << sum;  
The output will be 5. Can you reason it out ?  
The reason is a semicolon placed after for loop, hence the statement sum+=ctr is not treated as part of the for loop body.

### 12.5.3 continue

The continue statement forces the next iteration of the loop to take place, skipping any code following the continue statement in the loop body.



Working of continue statement in various loops is as follows:



What will be the output of the following segments ?

```
int ctr = 1;
for( ; ctr < 10; ctr++ )
{
    cout << ctr;
    ctr = 1;
}
```

Since ctr is initialised in the body of the for loop, the loop will result in an infinite loop, and the output displayed will be 1

```
int ctr = 1;
for( ctr = 1; ; ctr++ )
    cout << ctr;
```

Since test expression is missing, the loop will be executed until ctr reaches 32767, the integer data maximum value.

#### 12.5.4 break

A loop's execution is terminated when the test condition evaluates to false. Under certain situations one desires to terminate the loop, irrespective of the test expression. For example consider the Program - 12. 21

```
//Program - 12.21
# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    int a[] = {1,2,3,4,5,6,7,8,9};
    int search_item = 7;
    for(int x=0; x<9;x++)
    {
        if (a[x] == search_item)
        {
            cout << "\nItem found at position .." << x; break;
        }
    }
    cout << '\n' << "value of index position is .." << x;
    getch();
}
```

Output displayed will be :  
Item found at position .. 6  
value of index position is .. 6

- ✓ The control is transferred to cout statement written outside the loop, because of break statement. The loop is terminated when x takes the value as 6, because of break statement.

- ✓ **break statement would exit the current loop only.**
- ✓ **break statement accomplishes jump from the current loop**

**Nested loops :** It is possible to nest loop construct inside the body of another. Look at the following Program - 12.22

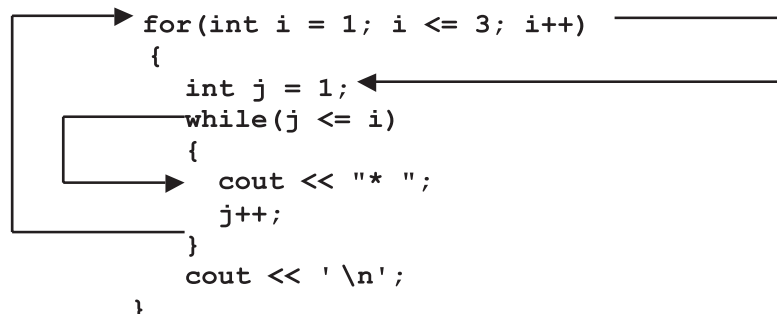
```
// nesting of loops – Program- 12.22
#include <iostream.h>
#include <conio.h>
```

```
void main()
{
    clrscr();
    for(int i = 1; i <= 3; i++)
    {
        int j = 1;
        while(j <= i)
        {
            cout << "*" << " ";
            j++;
        }
        cout << '\n';
    }
    getch();
}
```

**Output displayed :**

```
*
* *
* * *
```

Working of the loops is as follows :



The iterations of the nested loops are as follows :

for ..loop	while loop
i = 1	is executed once (j<= 1)
i = 2	Is executed twice (j = 1 .. 2)
i = 3	Is executed thrice (j = 1.. 3)

**Table 12.4 Nested Loops Example**

Can you write out as to what will be the output of the following program ?

```
# include <iostream.h>
# include <conio.h>

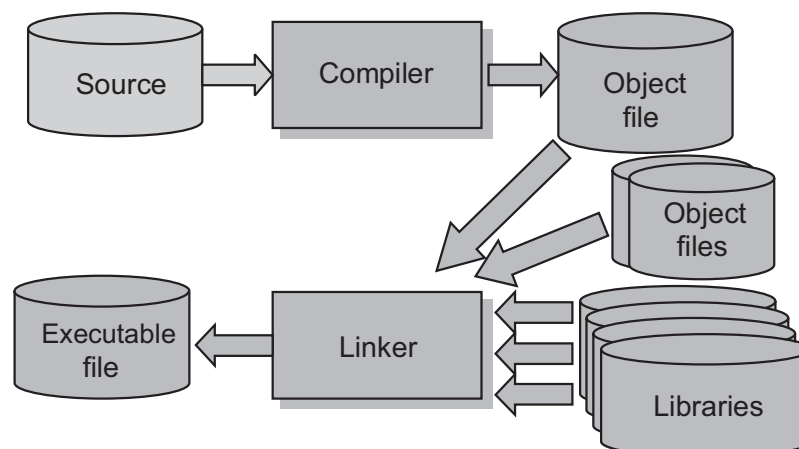
void main()
{
    clrscr();
    int i = 1, j = 1;
    while(i <= 3)
    {
        cout << '\n';
        for(i=1;i<=j;i++)
            cout << '*';
        i++;
        j++;
    }
    getch();
}
```

Output ??

The rules for the formation of nested loops are :

1. An outer loop and inner loop cannot have the same control variable, as it will lead to logical errors
2. The inner loop must be completely nested inside the body of the outer loop.

## 12.6 Program Development



**Fig. 12.1 Program Execution**

Programs are written in high level language using the grammar of a computer language. A Program written in high level language is called as the Source Code. The source code has to be converted to machine-readable form. The machine-readable form of a program is called as Object file. Compilers create object files from source code. Compilers are translator programs that create a machine-readable program from the source code. Compiler checks for the grammar of language (syntax). An object file is created from an error free source code. The object file is linked with the essential libraries to generate an executable file. This sequence of actions is shown in Fig. 12.1.

## EXERCISES

1. Categorise the following declarations as valid/invalid. If invalid, specify the reasons.

Declarations	Valid/Invalid	Reason
int A;a;		
char name(10);		
float f,int;		
double d, float f;		
int 1choice, _2choice		

2. Debug the following program. Rewrite the corrected program.

```
include <iostream.h>
include <conio.h>

void main()
{
    int N1,n1;
    cin << '\nEnter two number s ..';
    result := N1 * n1;
    cout << '\n' << Result;
}
```

3. Write appropriate declaration statements for the following :
  - a. To store the result of the expression  $8/3$  .
  - b. To initialise Emp\_Name with the value "Kalam"
  - c. To accept choice from user indicating Y=yes and N – no
4. Point out errors in the following snippets :
  - a. `int a = 10, b = 5;`  
`if a > b`



- ```
cout << a;
```
- b. if (a<b) && (a<0)
- ```
cout << "a is negative and ..."
```
- c. char option = 'Y';
- ```
do while option == 'y'
{
cout << '*';
.....
}
```
- d. for(int l = 1; l < 10; l++)
- ```
cout << l * 2;
```
- e. do
- ```
{
cout << '*';
} while( cout << "\nContinue <y/n>..."; cin>>ch;ch == 'y');
```
5. What will be the output of the following snippets / programs?

**// 5 a.**

```
#include <iostream.h>
#include <conio.h>
```

```
const int inch_conversion = 12;
```

```
cout << "\nEnter feet ...";
```

```
cout << "\nConverted to inches ..."
<< feet * inch_conversion;
```

**// 5 b.**

```
#include <iostream.h>
#include <conio.h>
```

```
void main()
```

```
{
int l = 1, sum = 0;
clrscr();
while(l++ <= 5)
```

```
{
cout << '\n' << l;
s += l;
```

```
}
```

```
cout << "\nValue of the variable l after
executing the while loop .." << l << "\nSum
:..." << s;
```

```
// 5 c
#include <iostream.h>
#include <conio.h>

void main()
{
    int i = 1, sum = 0;
    clrscr();
    while(++i <= 5)
    {
        cout << '\n' << i;
        sum += i;
    }
    cout << '\n' << i << '\t' << sum;
    getch();
}
```

```
// 5 d
#include <iostream.h>
#include <conio.h>
void main()
{
    int i = 1, sum = 0;
    clrscr();
    for(i = 1; i <= 5; i++)
    {
        cout << '\n' << i;
        sum += i;
    }
    cout << '\n' << i << '\t' << sum;
    for(i = 1; i <= 5; ++i)
    {
        cout << '\n' << i;
        sum += i;
    }
    cout << '\n' << i << '\t' << sum;
}
```

```
//5e
#include <iostream.h>
#include <conio.h>

void main()
{
    int i = 1, j = 1;
    clrscr();
    do
    {
        while (j<=i)
        {
            cout << '#';
            j++;
        }
        cout << '\n';
        i++;
        j = 1;
    } while(i<= 5);
    getch();
}
```

```
// 5 f
#include <iostream.h>
#include <conio.h>

void main()
{
    int num = 1784, s= 0, d = 0, x;
    x = num;
    clrscr();
    for(;num > 0;)
    {
        d = num % 10;
        s += d;
        num = num / 10;
    }
    cout << "\nThe sum of digits of "
        << x << "is : "
        << s;
    getch();
}
```

```
//5 g
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    for(int i = 1,s = 0; ; i++)
    {
        if (i%2 == 0)
            continue;
        s += i;
        if ( i > 9)
            break;
    }
    cout << "\nThe sum is ...." <<
    s;
    getch();
}
```

```
// 5 h
#include <iostream.h>
#include <conio.h>

void main()
{
    clrscr();
    for(int i = 1,x = 0;i <= 5; i++)
        x = x + i%2==0 ? i*1 : i * -1;
    cout << x;
    getch();
}
```

```
//5 j
# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    do
    {
        cout << "\ndo loop ...";
    } while (0);
    getch();
}
```

```
//5 k
# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    int i = 0;
    for(i = -5; i >= 5; i—)
        cout << "Bjarne Stroustrup";
    cout << "\nReturning to Edit Window..";
    getch();
}
```

```
//5 l
# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    int month = 5;
    if (month++ == 6)
        cout << "\nMay ...";
    else if (month == 6)
        cout << "\nJune ...";
    else if (--month == 5)
        cout << "\nMay again ..";
}
```

```
// 5 m
# include <iostream.h>
# include <conio.h>
void main()
{ int day = 3;
  switch (day)
  {
    case 0 : cout << "\nSunday ..";
    case 1 : cout << "\nMonday ..";
    case 2 : cout << "\nTuesday ..";
    case 3 : cout << "\nWednesday ..";
    case 4 : cout << "\nThursday ..";
    case 5 : cout << "\nFriday ..";break;
    case 6 : cout << "\nSaturday ..";
  }
}
```

```
// 5 n
# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    int bool = 2, b = 4;
    while(bool)
    {
        cout << bool << '\t' << ++b << '\n';
        bool—;
        b—;
    }
    getch();
}
```

## 6. Program Writing

- Write a program to compute  $a^b$  where  $a$  and  $b$  are of real and integer types (use while .. loop)
- Write a program to compute the factorial of a given number. (use for( ) loop)
- Write a program to generate fibonacci series upto  $n^{\text{th}}$  term.

Fibonacci series is :      0,1,1,2,3,5,8,12,20,32 ...

- Write a program to print the following patterns :

|   |   |   |   |   |  |   |   |   |   |  |  |
|---|---|---|---|---|--|---|---|---|---|--|--|
| 1 |   |   |   |   |  | 4 |   |   |   |  |  |
| 1 | 2 |   |   |   |  | 3 | 4 |   |   |  |  |
| 1 | 2 | 3 |   |   |  | 2 | 3 | 4 |   |  |  |
| 1 | 2 | 3 | 4 |   |  | 1 | 2 | 3 | 4 |  |  |
| 1 | 2 | 3 | 4 | 5 |  |   |   |   |   |  |  |

|   |   |   |   |  |  |
|---|---|---|---|--|--|
| A |   |   |   |  |  |
| B | C |   |   |  |  |
| D | E | F |   |  |  |
| G | H | I | J |  |  |

Using a switch, write a program to accept the day in a month, and print the messages as :

If day is 1, message is 1<sup>st</sup> day in the month

If day is 2,22 , message is 2<sup>nd</sup> / 22<sup>nd</sup> day in the month

If day is 3,23, message is 3<sup>rd</sup> / 23<sup>rd</sup> day in the month

If day is 4,14,15,16... message is 4<sup>th</sup> /14<sup>th</sup> .. day in the month

## CHAPTER 13

# FUNCTIONS

### 13.1 Introduction

Functions are the building blocks of C++ programs. Functions are also the executable segments in a program. The starting point for the execution of a program is `main ( )`. Functions are advantageous as they

- ✓ reduce the size of the program
- ✓ induce reusability of code

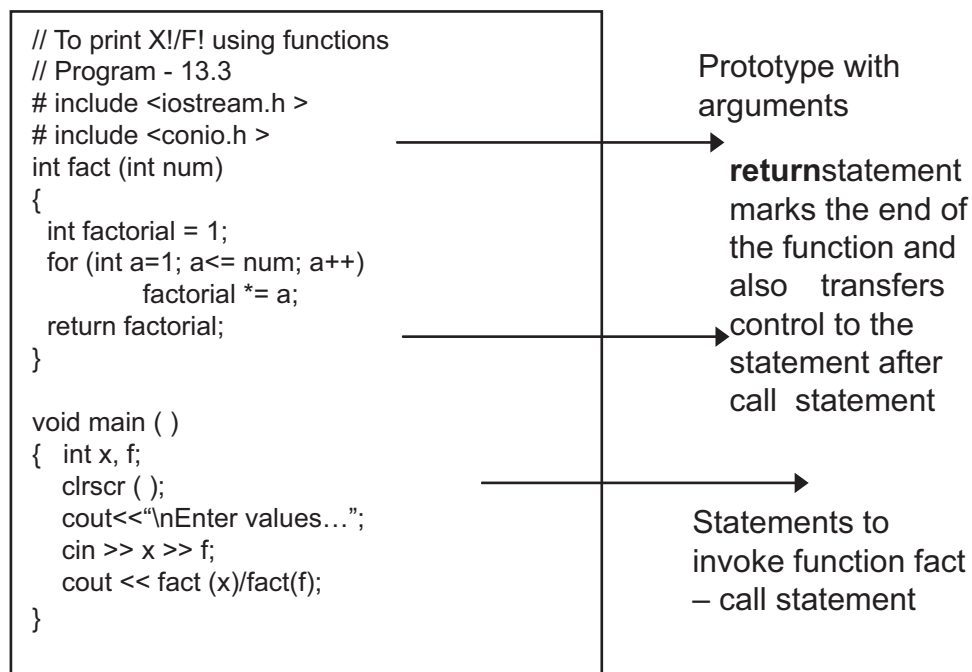
```
// To print X!/F! – Program - 13.1
# include <iostream.h>
# include <conio.h>
void main ()
{ int x, f,xfact=1, ffact = 1;
  clrscr()
  cout << "\nEnter values ...";
  cin >> x >> f;  for (int a=1; a <= x;
  a++)
  xfact * = a;
  for (a = 1; a<= f; a++)
  ffact * = a;
  cout << xfact/ffact;
  getch();
}
```

```
// To print X!/F! using functions
// Program - 13.2
# include <iostream.h>
# include <conio.h>
int fact (int num)
{
  int factorial = 1;
  for (int a=1; a<= num; a++)
  factorial *= a;
  return factorial;
}
void main ( )
{ int x, f;
  clrscr ( );
  cout<<"\nEnter values...";
  cin >> x >> f;
  cout << fact (x)/fact(f);
}
```

In Program –13.1, the code for Factorial evaluation is repeated twice. In the Program –13. 2, the function **fact (int num)** is invoked whenever required. Functions thus encourage :

- ✓ Reusability of code (function fact is executed more than once)
- ✓ A function can be shared by other programs by compiling it separately and loading them together.

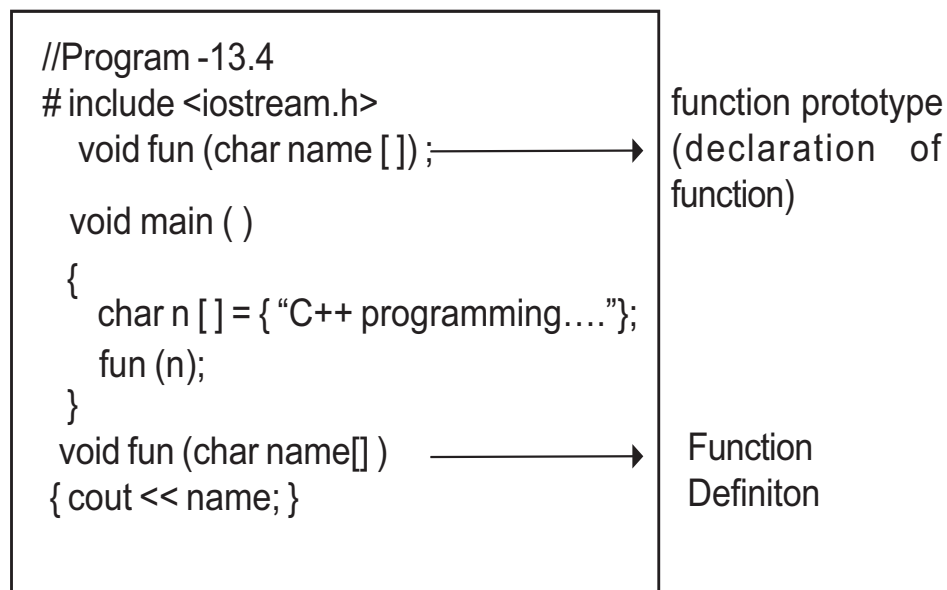
The general syntax showing the various blocks of a function :



## 13.2 Function Prototyping

Functions should be declared before they are used in a program. Declaration of a function is made through a function prototype.

For example look at the Program –13. 4.



The prototype provides the following information to the compiler

1. Number and type of arguments -(char name [ ] - is an argument)
2. The type of return values (in the above example fun does not have any return value, as the data type of the function is void. Recall Program –2 , in which the return type is int for the function Fact ( ) )

The general syntax of a function prototype

<type > <function identifier > <arguments>;

For example :

void fun (char);

int max (int, int);

int max (int a, int b);

The main purpose of function prototype is to help the compiler to check the data requirement of the function. With function prototyping, a template is always used when declaring and defining a function. When a function is called, the compiler uses the template to ensure that proper arguments are passed, and the return value is treated correctly. Any violation in matching of the arguments or the return types will be treated as errors by compiler, and flagged at the time of compilation.

### **Why do you think the prototype int max (int, int) is valid??**

In a function declaration, the names of the arguments are dummy variables and therefore they are optional. The variables in the prototype act as place holders.

The arguments' names are required in function definition, as the arguments are referenced inside the function.

```
void fun (char name [ ] )
```

```
{ cout << name; }
```

```
int fact (int num)
```

```
{    int f = 1; |—————> The argument num is referenced inside function.
```

```
    for (int a = 1;, a <= num; a ++)
```

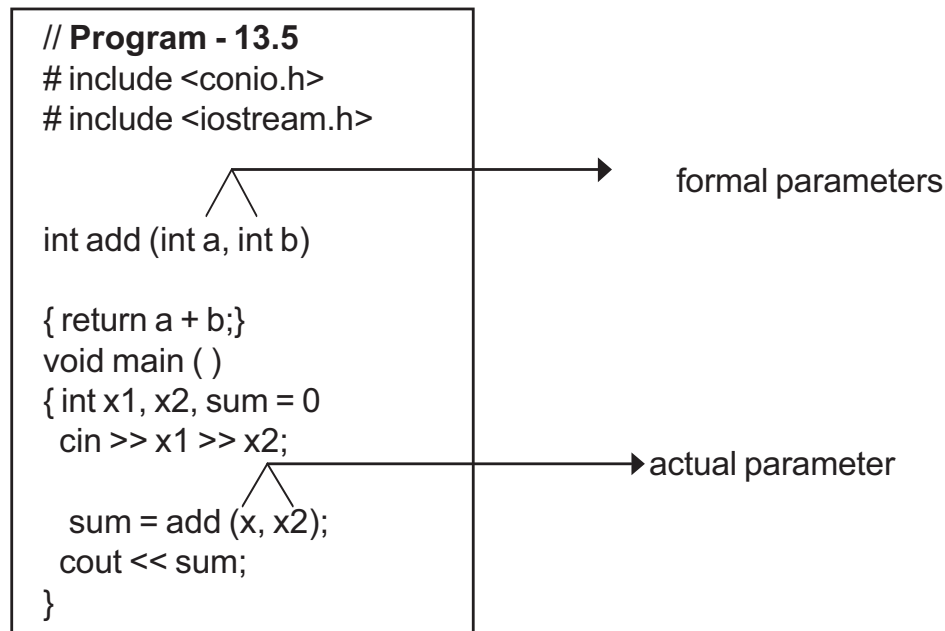
```
    fx = a;
```

```
    return f;
```

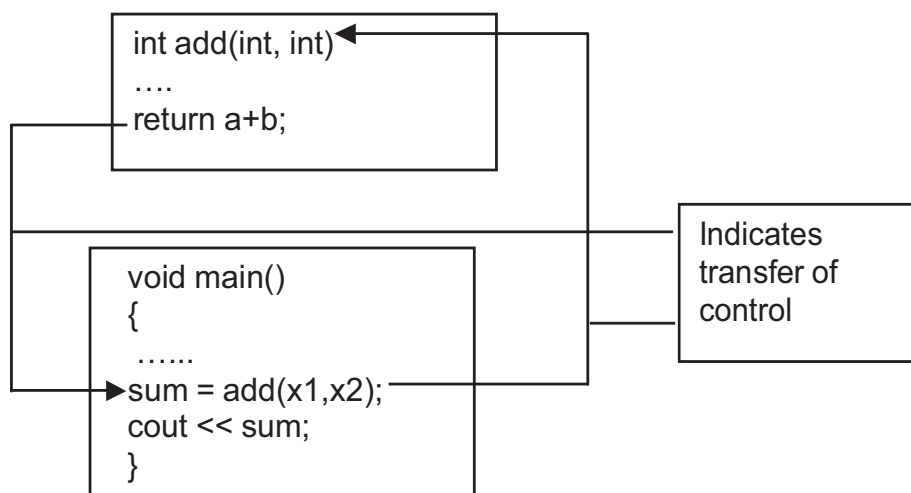
```
}
```

### 13.3 Calling a Function

A function can be called or invoked from another function by using its name. The function name may include a set of actual parameters, enclosed in parentheses separated by commas. For example,



#### Working of a function :



### 13.4 Parameter Passing in Functions

The call statement communicates with the function through arguments or parameters.

Parameters are the channels through which data flows from the call statement to the function and vice versa



In C++, functions that have arguments can be invoked by

- ✓ Call by value
- ✓ Call by reference

### 13.4.1 Call by Value

In this method, the called function creates new variables to store the value of the arguments passed to it. This method copies the values of actual parameters (parameters associated with call statement) into the formal parameters (the parameters associated with function header), thus the function creates its own copy of arguments and then uses them. Recall the example Program - 13.5

| <pre>// Program - 13.5 #include &lt;iostream.h&gt; #include &lt;conio.h&gt; int add (int a, int b) { return a+b;} void main () { int x1, x2, sum; cin &gt;&gt; x1 &gt;&gt; x2; sum = add (x, x2); cout &lt;&lt; sum; } Assume x1 = 5, x2 = 7</pre> | <table><tr><th>Main()</th><th>add()</th></tr><tr><td>x1 = 5</td><td>a = 5</td></tr><tr><td>x2 = 7</td><td>b = 7</td></tr><tr><td>sum =</td><td></td></tr><tr><td colspan="2">Assume address of the variables :</td></tr><tr><td>x1 = 0xf1</td><td>address data</td></tr><tr><td>x2 = 0xf3</td><td>0xf1 5</td></tr><tr><td>a = 0xf7</td><td>0xf2</td></tr><tr><td>b = 0xf9</td><td>0xf3 7</td></tr><tr><td>sum = 0xf6</td><td>0xf4</td></tr><tr><td></td><td>0xf5</td></tr><tr><td></td><td>0xf6 12</td></tr><tr><td></td><td>0xf7 5</td></tr><tr><td></td><td>0xf8</td></tr><tr><td></td><td>0xf9 7</td></tr></table> | Main() | add() | x1 = 5 | a = 5 | x2 = 7 | b = 7 | sum = |  | Assume address of the variables : |  | x1 = 0xf1 | address data | x2 = 0xf3 | 0xf1 5 | a = 0xf7 | 0xf2 | b = 0xf9 | 0xf3 7 | sum = 0xf6 | 0xf4 |  | 0xf5 |  | 0xf6 12 |  | 0xf7 5 |  | 0xf8 |  | 0xf9 7 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|-------|--------|-------|--------|-------|-------|--|-----------------------------------|--|-----------|--------------|-----------|--------|----------|------|----------|--------|------------|------|--|------|--|---------|--|--------|--|------|--|--------|
| Main()                                                                                                                                                                                                                                             | add()                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| x1 = 5                                                                                                                                                                                                                                             | a = 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| x2 = 7                                                                                                                                                                                                                                             | b = 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| sum =                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| Assume address of the variables :                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| x1 = 0xf1                                                                                                                                                                                                                                          | address data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| x2 = 0xf3                                                                                                                                                                                                                                          | 0xf1 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| a = 0xf7                                                                                                                                                                                                                                           | 0xf2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| b = 0xf9                                                                                                                                                                                                                                           | 0xf3 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
| sum = 0xf6                                                                                                                                                                                                                                         | 0xf4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
|                                                                                                                                                                                                                                                    | 0xf5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
|                                                                                                                                                                                                                                                    | 0xf6 12                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
|                                                                                                                                                                                                                                                    | 0xf7 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
|                                                                                                                                                                                                                                                    | 0xf8                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |
|                                                                                                                                                                                                                                                    | 0xf9 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |        |       |        |       |        |       |       |  |                                   |  |           |              |           |        |          |      |          |        |            |      |  |      |  |         |  |        |  |      |  |        |

Have you noticed that the actual parameters x1 and x2 and the formal parameters a&b have been allocated different memory locations? Hence, in call by value method, the flow of data is always from the call statement to the function definition.

```
// Program - 13.6
// To exchange values
#include <iostream.h>
#include <conio.h>
# include <iomanip.h>
void swap (int n1, int n2)
{    int temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
    cout << '\n'<<n1<<'\t'<<n2<<'\n';
}
```

```

void main ( )
{
    int m1 = 10, m2 = 20;
    clrscr ( );
    cout << "\n Values before invoking swap" << m1 << '\t' << m2;
    cout << "\n Calling swap..";
    swap (m1, m2);
    cout << "\n Back to main.. Values are" << m1 << '\t' << m2;
    getch ( );
}

```

Output

Values before invoking swap      10      20

Calling swap .....

20      10

Back to main..... Values are      10      20

Why do you think the exchange of values of the variables m1 and m2 are not reflected in the main program??

When arguments are passed by value, the called function creates new variables of the same data type as the arguments passed to it. The values of these arguments are copied into the newly created variables. Hence, changes or modifications that are made to formal parameters are not reflected in the actual parameters.

In call by value method, any change in the formal parameter is not reflected back to the actual parameter.

### 13.4.2 Call by reference

In this method, the called function arguments - formal parameters become alias to the actual parameters in the calling function. This means that when the function is working with its own arguments, it is actually working on the original data. Recall the example Program 13.6. Let us now rewrite the function using reference parameters.

```

//Program - 13.7
// To exchange values

# include <iostream.h>
#include <conio.h>
void swap (int &n1, int &n2)
{
    int temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
    cout<<'\n'<< n1 <<'\t'<<n2<<'\n';
}

```

```

void main ( )
{
    int m1 = 10, m2 = 20;
    clrscr();
    cout<<"\nValues before swap call" << '\t' << m1 << '\t' << m2;
    swap(m1,m2);
    cout<<"\n Calling swap..";
    cout<<"\n Back to main.Values are"
    << '\t' << m1 << '\t'<< m2;
    getch ( );
}

```

```

Output:
Values before invoking swap 10  20
Calling  swap..
20      10
Back to main. Values are      20  10

```

The modifications made to formal parameters are reflected in actual parameters, because formal and actual parameters in reference type point to the same storage area.

Look at the following depiction:

| <b>Main</b> | <b>Swap</b> |
|-------------|-------------|
| m1 = 10     | n1 = 10     |
| m2 = 20     | n2 = 20     |
|             | temp        |

Assume storage area of m1 is Oxf1, and m2 is Oxf4.

m1 = Oxf1 = 10

m2 = Oxf4 = 20

Reference to formal parameters may be read as

n1 = 10; n1 is a reference to m1, which may be depicted as:

int &n1 = m1

This means that n1 is an alias to m1, hence m1 and n1 refer to same storage area, hence the statements may be rewritten as :

n1 = m1 = 0xf1 = 10  
n2 = m2 = 0xf4 = 20

| Address       | Before Exchange | After exchange |
|---------------|-----------------|----------------|
| 0xf1 (n1, m1) | 10              | 20             |
| 0xf4 (n2, m2) | 20              | 10             |

Hence, changes made to formal parameters are reflected in actual parameters.

In call by reference method, any change made in the formal parameter is reflected back in the actual parameter.

Try out

```
// Reference variables
// Program -13.8
# include <iostream.h>
# include <conio.h>
void main ( )
{
    int num1 = 10, & num2 = num1;
    num2 ++;
    cout << num1;
}
```

Output displayed will be 11  
By virtue of reference num1 and num2 point to the same storage location.  
Hence, change of value in num1 is reflected in num2.

### Rules for actual parameters:

1. The actual parameters can be passed in the form of constants or variables or expressions to the formal parameters which are of value type.

For example,

For a function prototype :     int add (int n1, int n2); - the call statements may be as follows :

x = add (5, 10);

x = add (a1, a2); where a1 and a2 are variables

2. The actual parameters can be passed only as variables to formal parameters of reference type.

For example,

```
int add (int & n1, int & n2);  
x = add (a1, b1) ;   where a1 and b1 are variables
```

The following call statements are invalid:

```
x = add ((a1 + b1), a1);  
x = add (5,10);
```

```
// Program - 13.9  
// To print 5 stars per row and 5 such rows  
# include <iostream.h>  
# include <conio.h>  
  
void fun_starts (int &i)  
{  
    int j = 5;  
    for (i= 1; i <= j; i++)  
        cout << ' ' << '*';  
}  
  
void main ( )  
{  
    int mi = 1;  
    clrscr( );  
    for (; mi<= 5; mi++)  
    {  
        cout << '\n';  
        fun_starts (mi);  
    }  
    getch ( );  
}
```

Why does the above program does not behave the way to produce the result as mentioned in comment line?

The output produced is :

\* \* \* \* \*

Reason – the variable i is a reference to the variable mi. Since the variable i gets a value 6 in the function, mi is also automatically updated to 6, hence, the 'for' loop in main ( ) is executed only once.

### 13.3.4 Default arguments

In C++, one can assign default values to the formal parameters of a function prototype.

For example :

```
// Program - 13.10
// formal parameters with default values
# include <iostream.h>
# include <conio.h>
float power (float n, int p = 1)
{
    float prd = 1;
    for (int i = 1; i <= p; i++)
        prd *= n;
    return prd;
}

void main ( )
{
    clrscr ( );
    int x = 4, b = 2;
    cout << "\n Call statement is power(b, x)..." << power (b, x);
    cout << "\n Call statement is power(b).. " << power (b);
    getch ( );
}
```

Output:

|                                |    |    |
|--------------------------------|----|----|
| Call statement is power (b, x) | .. | 16 |
| Call statement is power (b)    | .. | 2  |

In the call statement power (b,x), initialization is

n= b, p = x

In the second form power (b), the variable n is initialized, whereas p takes the value 1 (default argument), as no actual parameters is passed.

**NOTE:**

- ✓ The default value is given in the form of variable initialization.
- ✓ The default arguments facilitate the function call statement with partial or no arguments.
- ✓ The default values can be included in the function prototype form right to left, i.e., we cannot have a default value for an argument in between the argument list.

Try out the following Program.

```
//Program - 13.11
#include <iostream h>
#include <conio.h>
int area (int side1 = 10, int side2=20
{ return (side1 * side 2); }

void main ( )
{   int s1 = 4, s2 = 6;
    clrscr ( ) ;
    cout << area (s1, s2) << '\n';
    cout << area (s1) << '\n';
    cout << area (s2) << '\n';
    getch ( ) ;
}
```

Output:

24  
80  
120

Variable initialization  
I form - side1 = s1,  
side2 = s2  
II form - side1 = s1  
III form - side1 = s2

What will be the output of the following program?

```
// Program - 13.12
// arguments with default values

#include <iostream.h>
#include <conio.h>

void print (int times, char ch = ' * ')
{
    cout << '\n';
    for (int i = 1; i <= times; i++)
        cout << ch;
}

void main ( )
{
    clrscr ( );
    print (50);
    print ('A', 97);
    print ( );
}
```

### Solution:

print (50)      -      50 is assigned to the argument times. Hence, the output ' \* ' will be printed 50 times.

print ('A', 97) -      'A' is assigned to argument times (implicit conversion of character to integer takes place). Hence, times gets the value as 65.

The constant 97 is assigned to ch, hence ch gets the value as 'a'.

The actual parameters are matched with formal parameters on the basis of one- to -one correspondence.

Hence, 65 times, 'a' will be printed.

print ( )      -      In the absence of actual arguments, the formal parameters takes the default arguments. Hence, the output will be displayed as ' \* ' - 50 times.

### 13.5 Returning Values

The functions that return no value is declared as void. The data type of a function is treated as int, if no data type is explicitly mentioned.

For example,

```
int add (int, int);
```

```
add (int, int);
```

In both prototypes, the return value is int, because by default the return value of a function in C++ is of type int.

Look at the following examples:

| Sl.No. | Function Prototype       | Return type          |
|--------|--------------------------|----------------------|
| 1      | float power (float, int) | float                |
| 2      | char choice ( )          | char                 |
| 3      | char * success ( )       | pointer to character |
| 4      | double fact (int)        | double               |



### 13.5.1 Returning by reference

Reference or alias variables:

```
// Program - 13.13
# include <iostream.h>
# include <conio.h>
void main ( )
{
    int i = 5;
    int &count = i ;
    cout << "\nCount: " << count;
    count ++;
    cout << "\ni: " << i;
    getch ( );
}
```

Output:

Count: 5

i: 6

Why do you think count gets the value as 5?

Why is the variable i updated to 6, when count was incremented???

The reason being that the variables count and i refer to the same data in the memory. Reference variables also behave the same way.

Using this principle, try and find out as to what will be output of the following program:

```
// Program - 13.14
# include <iostream h>
# include <conio.h>
int &maxref (int &a, int &b)
{
    if (a>b)
        return a;
    else
        return b;
}
void main ( )
{
    int x = 20, y = 30, max = 0;
    max = maxref (x,y);
    cout << "\n Maximum is: " << max;
}
```

## Output

Maximum is : 30

In the above program, the function maxref returns a reference to int type of variable. The function call maxref (x,y) will return a reference to either a or b depending upon which one is bigger of the two. Hence, the variable max gets the value of the variable y.

What will be the output of the following program?

```
// Program 13.15
# include <iostream h>
# include <conio.h>
int & maxref (int & a, int & b)
{ if (a > b),
  return a;
    else
      return b;
}
void main ( )
{ int x = 20, y = 30, max = 0;
  maxref (x,y) = -1;
  cout << "\n Value of x is : " << x;
  cout << "\n Value of y is: " <<y;
  getch ( );
}
```

## Output

Value of x is : 20

Value of y is : -1

## NOTE:

1. A function returning a reference can appear on the left-hand side of an assignment.
2. In the above example, the variable y gets the value -1, since the function maxref. establishes reference with the formal parameter b, whose corresponding variable in main block is 'y'.
3. The formal parameters for a reference function should always be of reference parameter type in the sense.

```
int & maxref (int a, int b);
```

will yield compilation error, as the scope of the variables a&b are within the function block maxref.

### 13.6 Inline Functions

We have listed out the advantages of functions as

- ✓ Reusability of code leading to saving of memory space and reduction in code size.

While this is true, we also know that call statement to a function makes a compiler to jump to the functions and also to jump back to the instruction following the call statement. This forces the compiler to maintain overheads like STACKS that would save certain special instructions pertaining to function call, return and its arguments. This reduces the speed of program execution. Hence under certain situations specially, when the functions are small (fewer number of instructions), the compiler replaces the function call statement by its definition i.e., its code during program execution. This feature is called as inlining of a function technically called as **inline** function.

An **inline** looks like a normal function in the source file but inserts the function's code directly into the calling program.

Inline functions execute faster but require more memory space.

Now look at the following example.

```
// Program - 13.16
// inline functions

# include <iostream.h>
# include <conio.h>

inline float convert_feet(int x)
{
    return x * 12;
}

void main()
{
    clrscr();
    int inches = 45;
    cout << convert_feet(inches);
    getch();
}
```

```
// working of Program - 13.16
// inline functions

# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    int inches = 45;
    cout << inches * 12 ;
    getch();
}
```

As shown in the above example, the call statement to the function (convert\_feet(inches)) will be replaced by the expression in the return statement (inches \* 12).

To make a function inline, one has to insert the keyword **inline** in the function header as shown in Program 13.16.

**Note :** inline keyword is just a request to the compiler . Sometimes the compiler will ignore the request and treat it as a normal function and vice versa.

### 13.7 Scope Rules of Variables

Scope refers to the accessibility of a variable. There are four types of scopes in C++. They are:

- |                |                   |
|----------------|-------------------|
| 1. Local scope | 2. Function scope |
| 3. File scope  | 4. Class scope    |

#### 13.7.1 Local scope

```
// Program - 13.17
// to demonstrate local variable
# include <iostream.h>
# include <conio.h>
void main ( )
{
    int a, b ;
    a = 10;
    b = 20;
    if (a > b)
    { int temp; // local to this if block
      temp = a;
      a = b;
      b = temp;
    }
    cout << '\n Descending order...';
    cout << '\n' <<a << '\n' <<b;
    getch ( );
}
```

- A local variable is defined within a block.
- The scope of a local variable is the block in which it is defined.
- A local variable cannot be accessed from outside the block of its declaration.

Program-13.18 demonstrates the scope of a local variable.

```
//Program - 13.18
#include <iostream.h>
#include <conio.h>
void main ( )
{   int a, b;
    a = 10
    b = 20;
    if (a > b)
    {   int temp;
        temp = a;
        a= b;
        b = temp;
    }
    cout << a << b << temp;
    getch ( );
}
```

On compilation, the compiler prompts an error message:  
Error in line no.13  
The variable temp is not accessible.  
The life time of a local variable is the life time of a block in its state of execution.  
Local variables die when its block execution is completed.

- Local variables are not known outside their own code block. A block of code begins and ends with curly braces { }.
- Local variables exist only while the block of code in which they are declared is executing.

A local variable is created upon entry into its block and destroyed upon exit.

Identify local variables, in the Program-13.19 and also mention their scope.

### 13.7.2 Function scope

The scope of variables declared within a function is extended to the function block, and all sub-blocks therein.

|                                                                                                                                                                                                             |                                                                  |                                                                                                                                                          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>// Program - 13.19 #include &lt;iostream.h&gt; void main ( ) { int flag = 1; a = 100;   while (flag)   {     int x = 200;     if (a &gt; x)     { int j;       -     }   } else { int h;   - } }</pre> | <p><u>Local variable</u></p> <p>1. x</p> <p>2. j</p> <p>3. k</p> | <p><u>Scope</u></p> <p>Accessible in while block, and if blocks</p> <p>Accessible only in if (a&gt;x) { } block</p> <p>Accessible only in else block</p> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|

The variable `flag` of Program – 13.19 is accessible in the function `main ( )` only. It is accessible in all the sub-blocks therein - viz, `while` block & `if` block.

The life time of a function scope variable, is the life time of the function block. The scope of formal parameters is function scope.

### 13.7.3 File scope

A variable declared above all blocks and functions (precisely above `main ( )`) has the scope of a file. The scope of a file scope variable is the entire program. The life time of a file scope variable is the life time of a program.

```
// Program - 13.20
// To demonstrate the scope of a variable
// declared at file level
# include <iostream.h>
# include <conio.h>
int i = 10;
void fun ( )
{ cout << i; } void main ( )
{
    cout << i;
    while (i)
    {
        -
        -
        -
    }
}
```

### 13.7.4 Scope Operator

The scope operator reveals the hidden scope of a variable. Now look at the following program.

```
// Program - 13.21
# include <iostream.h>
# include <conio.h>

int num = 15;

void main()
{
    clrscr();
    int num = 5;
    num = num + ::num;
    cout << num << '\t' <<
    ++::num;
    getch();
}
```

Have you noticed the variable **num** is declared both at file scope level and function `main()` level? Have you noticed the reference `:: num ?` is called as scope resolution operator. It is used to refer variables declared at file level. This is helpful only under situations where the local and file scope variables have the same name.

### 13.7.5 Class scope

This will be discussed in Chapter 15.

## EXERCISES

1. Construct function prototypes for descriptions given below:

a) procedural-function ( )

- is a function that takes no arguments and has no return value.

Solution -

void procedural - function (void);

OR void procedural - function ( ) ;

b) manipulative - function ( ) takes one argument of double type and returns int type.

Solution -

i. int manipulative - function (double); OR

ii. int manipulative - function (double d); OR

iii. manipulative - function (double)

c) fun-default ( ) takes two arguments, once with a default integer value, and the other float, has no return type

Solution -

void fun-default (float, int num = 10);

d) return - reference - fun ( ) takes two int arguments and return reference to int type

Solution -

int & return - reference - fun (int &, int &);

e) multi-arguments ( ) that takes two arguments of float, where the 1<sup>st</sup> argument is P1 should not be modified, and the 2<sup>nd</sup> argument is of reference type. The function has no return type.

Solution -

void multi - arguments (float const pi, int & a);

2. Identify errors in the following function prototypes:

- a) float average (a, b);
- b) float prd (int a,b);
- c) int default-arg (int a = 2, int b);
- d) int fun (int, int, double = 3.14);
- e) void strings (char [ ]);

3. Given the function

```
void line (int times, char ch)
{
    cout << '\n';
    for (int i = 1; i <= times; i++)
        cout << ch;
    cout << '\n';
}
```

Write a main ( ) function that includes everything necessary to call this function.

4. Write the scope of all the variables mentioned in this program.

```
# include <iostream.h>
```

```
float a, b ; void f1 (char);
```

```
int main ( )
```

```
{ char ch;
```

```
-
```

```
-
```

```
{ int i = 0;
```

```
-
```

```
-
```

```
-
```

Solution:

a,b - file scope

ch -function scope

- main ( )

i - scope within its

block

x,y,g -function scope - f1 function



```

    }
}

void f1 (char g)

{ short x, y ;

} =

```

4. Identify errors in the following programs

a) # include <iostream.h>

Solution:

```

xyz (int m, int n)
{ int m = 10;
  n = m * n;
  return n;
}

void main( )
{ cout << xyz (9,27) ;}

```

The variable 'm' is declared with function block, which is not permitted.

b) # include <iostream.h>

Solution:

```

void xyz ( );
void main ( )

{ int x = xyz ( ) ; }
void xyz ( )
{ return '10' ; }

```

Function declared as void type, cannot have a return statement, hence the function call cannot be part of an expression

c) # include <iostream.h>

Solution:

```

void counter (int & a)
{ ++ a;}

void main ( )
{counter (50); }

```

The actual parameter cannot be passed in the form of a value, as the formal parameter is of reference type

## 5. What will be the output of the following programs?

a) 

```
# include <iostream.h>
int val = 10;
divide (int) ;
void main ( )
{int val = 5;
val = divide (::val/val);
cout << :: val<<val;
}
divide (int v)
{ return v/2;}
```

Solution: 101

b) 

```
# include <iostream.h>
divide (int v)
{ return v / 10;}
void main ( )
{int val = -1;
val = divide (400) == 40;
cout << "\n Val." << val;
}
```

Solution: 1 - Working

i) divide (400) yields a value 40  
ii) divide (400) == 40 is interpreted as  
40 ==40 since the condition is true  
val gets 1

c) 

```
# include <iostream.h>
int incre (int a)
{ return a++; }
void main ( )
{int x = 10; x = incre (x);
cout << x;}
```

Solution: 10

d) 

```
# include <iostream.h>
# include <iostream.h>
void line( )
{static int v = 5;
int x = v - - ;
while (x)
{cout << ' * ' ; x — ;
}
cout << '\n';
}
void main ( )
{ clrscr ( );
for (int i = 1; i <= 5; i ++
line ( ) ;
getch ( ) ;
}
```

Solution:

```
* * * * *
* * * *
* * *
* *
```

```
e)  # include <iostream.h>
    first (int i)
    { return i++; }
    second (int x)
    { return x —; }
    void main ( )
    { int val = 50;
      val = val * val/val
      val = second (val);
      val = first (val);
      cout << “\n Val: “ << val;
    }
```

Solution:

Val : 50

## 6. Program writing....

a) Write a program in C++ to define a function called float cube (int, int, int). Write main ( ) function, to test the working of cube ( ).

b) Define a function unsigned long it factorial (int);

The factorial of a number is calculated as follows: For example Factorial of 5 is calculated as 1 x 2 x 3 x 4 x 5

Write a main ( ) function to calculate the factorial (n).

c) Define a function called as char odd - even - check (int);

The function should return 'E' if the given number is even, otherwise 'O'. Write a main ( ) to test and execute the function odd-even-check (int) and also print relevant message.

d) Define a function int prime (int);

The function should return a value 1, if the given number is prime, otherwise -1. Write a main ( ) to test and execute the function and also print relevant message.

## CHAPTER 14

### STRUCTURED DATA TYPE - ARRAYS

#### 14.1 Introduction

An array in C++ is a derived data type that can hold several values of the same type.

Processing a collection of data values by reading the data items individually and then processing each item may be very cumbersome and awkward if data is large. For example, consider the following situations:

1. To determine the largest number in the given set of numbers:
  - a) if the set comprises of two numbers then the comparisons would be :  
if (a > b)  
    max = a;  
else  
    max = b;
  - b) if the set comprises of three numbers then the comparisons would be :  
if (a > b) && (a > c)  
    max = a;  
else if (b > c)  
    max = b;  
else  
    max = c;
  - c) if the given set comprises of 4 numbers then the comparisons would be :  
if (a > b && a > c && a > d)  
    max = a;  
else if (b > c && b > d)  
    max = b;  
else if (c > d)  
    max = c;  
else  
    max = d;

Have you noticed the increase in comparisons, as the numbers increase??

In fact, handling large data becomes unwieldy, if one has to adopt the above methods for processing data.

Now look at this:

```
int a [4] = { 10, 40, 30, 20}; max = 0 ; i = 0;
    for (; i < 4; i ++)  
        if a [i] > max  
            max = a [i] ;  
    cout << "\n The largest number is" << max;
```

The above program code determines the largest value in the given list of numbers, i.e., 10, 40, 30, 20, thus storing 40 in max. Have you noticed the construct of if statement? In order to handle large data with ease, elements belonging to the same data type are decaled as ARRAYS.

An array is a collection of variables of the same type that are referenced by a common name.

Arrays are of two types:

One dimensional:                      comprising of finite homogenous elements

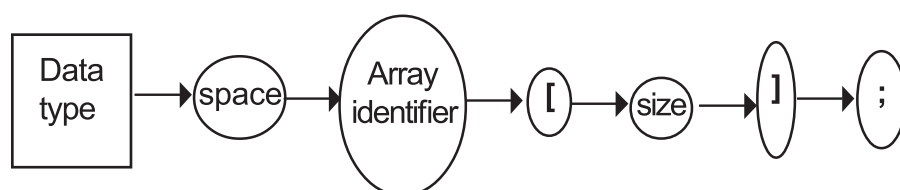
Multi dimensional:                    comprising of elements, each of which is itself a one-dimensional array

## 14.2 Single Dimension Array

These are suitable ways for processing of lists of items for identical types. An array is declared as follows:

```
int num_array [5];
```

Syntax:



**Fig. 14.1 Single Dimension Array**

The size of the array should always be positive. The declaration `int num_array [5];` is interpreted as **num\_array is a one-dimensional array, that stores 5 integer values**. Each element of the array is accessed by the array name and the position of the element in the array. For example, `num_array [3] = 99`, stores the value 99 as the 4<sup>th</sup> element in the `num_array`.

**num\_array** is the array identifier and **[3]** is subscript/position of the element

Memory allotted for num\_array is 10 bytes, as it stores 5 integer element (memory required for one integer is 2 bytes hence  $5 \times 2 = 10$  bytes).

Memory allocation is as follows:

num\_array - identifier

|   |   |   |    |  |                                    |
|---|---|---|----|--|------------------------------------|
| 0 | 1 | 2 | 3  |  | → <b>subscripts</b>                |
|   |   |   | 99 |  | → <b>Value stored as element 4</b> |

The array subscripts always commences from zero, hence the subscripts for the variable num-array is between 0-4. The statement num-array [5] is invalid, because the valid subscripts are only between 0-4. The other valid examples of array declaration are:

- i. `int array [100];`
- ii. `float exponents [10];`
- iii. `char name [30];`
- iv. `const i = 10; double val [i];`
- v. `int days [ ] = {1,2,3,4,5,6,7};`

In example [iv], the size of the array val is indicated through a constant variable i. In example [v], the size of the array days [ ] is indirectly indicated as 7. Can you guess how? Yes, the size is determined through the initialization statement `{1,2,3,4,5,6,7}`. 7 elements determine the size of the array. Now look at the Program – 14.1 that demonstrates basic operations on arrays.

Examples of array processing:

- |                                               |   |                                                                                                 |
|-----------------------------------------------|---|-------------------------------------------------------------------------------------------------|
| <code>cin &gt;&gt; number [4]</code>          | - | Reads fifth element cout                                                                        |
| <code>cout &lt;&lt; number [subscript]</code> | - | displays the element as indicated by subscript                                                  |
| <code>number [3] = number [2]</code>          | - | assigns the contents of the 3 <sup>rd</sup> element of the array to its 4 <sup>th</sup> element |
| <code>number [3] ++</code>                    | - | increments the value stored as 4 <sup>th</sup> element by 1                                     |
| <code>number [++ a] = 10</code>               | - | assigns the value 10 to the element as indicated by ++a                                         |

```
// Program - 14.1
// arrays and basic manipulations on
// arrays
# include <iostream.h>
# include <conio.h> int a[5],ctr = 0, sum;
void main()
{
    for(;ctr<5;ctr++)
    {
        cout << "\nEnter value ..";
        cin >> a[ctr];
    }
    // generating sum of all the elements
    // stored in array
    for(ctr = 0, sum = 0; ctr<5;ctr++)
        sum+= a[ctr];
    clrscr();
    // display values stored in array
    for (ctr = 0; ctr < 5; ctr++)
        cout <<'\t' << a[ctr];
    cout << "\nSum of the elements ..."
    << sum;
    getch();
}
```

```
// try out
// Program – 14.2
# include <iostream.h>
# include <conio.h>
char ch [ ] = {'a', 'b', 'c', 'd', 'e', 'f'};
void main ( )
{
    for (int i = 0; i < 6; i++)
        cout << ch[ i];
    for (j=5; j>=0; j—)
        cout << ch [j];
    getch();
}
```

Output displayed :  
abcdeffedcba

```

// try out
// Program - 14.3
#include <iostream.h>
#include <conio.h>

void main ( )
{
    int even [3] = {0, 2, 4}; int reverse [3];
    for (int i=0, int j = 2; i<3; i++, j —)
        reverse [j] = even [i];
    clrscr ( );
    for (i=0; i<3, i++)
        cout<< '\t' << even [i] << '\t' << reverse [i] << '\n';
    getch ( );
}

```

Output of Program - 14.3

|   |   |
|---|---|
| 0 | 4 |
| 2 | 2 |
| 4 | 0 |

```

// try out
//Program -14.4
#include <iostream.h>
#include <conio.h>
void main ( )
{
    int x[5] = {1,2,3,4,5}, y [5] = {5,4,3,2,1},
    result [5] = { 0,0,0,0,0 };
    int i= 0;
    while (i++ < 5)
        result [i] = x [i] - y [i];
    clrscr ( );
    cout << "\n The contents of the array are: \n";
    i= 0 ;
    do
    {
        cout << '\t' << x [i]
            << '\t' << y [i]
            << '\t' << result [i]<<'\n';
        i++;
    } while (i<5);
    getch ( );
}

```



Output of Program -14. 4

The contents of the array are:

|   |    |    |
|---|----|----|
| 1 | -1 | 0  |
| 2 | 4  | -2 |
| 3 | 3  | 0  |
| 4 | 2  | 2  |
| 5 | 1  | 4  |

```
//Try out
//Program - 14. 5
# include <iostream.h>
# include <conio.h>
void main ( )
{
    int vector [ ] = {2, 4, 8, 10, 0};
    for(int i=4; i>2; i--)
        vector [i]= vector [i-1];
    clrscr( );
    cout << "\n Elements of array before insertion \n";
    for (i= 0; i<5; i++)
        cout << vector[i];
    vector [2] = 6;
    cout << "\n Elements after insertion \n";
    for (i= 0; i<5; i++)
        cout << vector[i];
    getch ( );
}
```

Output of Program - 14.5

Elements of array before insertion

248810

Elements after insertion

246810

One can rearrange the data in a given array either in ascending or descending order. This process is called SORTING.

### 14.3 Strings

Strings are otherwise called as literals, which are treated as single dimensional array of characters. The declaration of strings is same as numeric array. For example,

- i. char name [10];
- ii. char vowels [ ] = {'a', 'e', 'i', 'o', 'u'};
- iii. char rainbow [ ] = VIBGYOR;

A character array (used as string) should be terminated with a '\0' (NULL) character. These arrays can be initialized as in the above examples, viz., (ii) and (iii).

```
// Program - 14.6
// Reading values into an array of characters

# include <iostream.h>
# include <stdio.h>
# include <conio.h>
# include <string.h>
void main()
{
    clrscr();
    char name [30], address [30], pincode[7];
    cout << "\n Enter name ...";
    cin >> name;
    cout << "\n Enter address...";
    gets (address);
    cout << "\n Enter pincode ...";
    cin.getline (pincode, 7,'#');
    clrscr ( );
    cout << "\n Hello " << name;
    cout << "\n Your address is ..." << address;
    cout << "\n Pincode    ....";
    cout.write (pincode, sizeof(pincode));
    getch ( );
}
```

In the above example Program - 14.6, the values for the variables name, address and pincode are read using

cin, gets ( ) and getline.

The instance cin, treats white space or carriage return (enter key) as terminator for string. For example,

cin >> name;

- a) if the value for name is given as K V N Pradyot , then the value stored in name is only K, as white space is treated as string separator or terminator.
- b) if the value for name is given as K.V.N.Pradyot , then the value stored in name is K.V.N.Pradyot.

To treat spaces as part of string literal, then one has to use `gets ( )` defined in `stdio.h` or `getline ( )` - a member function of standard input stream.

Syntax for `gets ( )` is

`gets (char array identifier) or gets (char *)`

Syntax for `getline` is

`cin.getline (char*, no.of characters, delimiter);`

There are two methods to display the contents of string.

1. `cout << name -` this is similar to any other variable.
2. `cout.write (pincode, 7);` or `cout.write (pincode, size of (pincode));`

`write ( )` is a member function of standard output stream, i.e., `ostream`. All member functions of a class, should be accessed through an object /instance of class. The two parameters required for `write ( )` function are identifier string characters, and no. of characters to be displayed.

For example,

```
//Program - 14.7  
# include <iostream.h>  
# include <conio.h>  
void main()  
{  
    clrscr ( );  
    char name[ ] = "Tendulkar";  
    int i=1;  
    while (i<10)  
    {  
        cout.write (name, i);  
        cout << '\n';  
        i++;  
    }  
    getch ( );  
}
```

Output

T  
Te  
Ten  
Tend  
Tendu  
Tendul  
Tendulk  
Tendulka  
Tendulkar

String manipulators defined in string.h are described in Table 14.1.

| Sl. No. | Function   | Syntax                 | Purpose & value returned                                                                                                                                                                                                                                                                      |
|---------|------------|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | strlen ( ) | strlen (char *)        | Returns the number of characters stored in the array. For example, char name =                                                                                                                                                                                                                |
| 2       | strcpy ( ) | strcpy (char *,char *) | Copies source string to target string. For example, strcpy (name, petname)                                                                                                                                                                                                                    |
| 3       | strcmp ( ) | strcmp ( char*, char*) | Compares the two given strings, and returns 0. if strings are equal, value >0, if string 1 is greater than string 2. Otherwise value less than 0. For example, strcmp ("Abc", "Abc") returns 0. strcmp ("Abc", "abc") returns a value less than 0<br>strcmp ("abc", "ABC") returns a value 1. |

**Table 14.1 String Functions**

Strings can be manipulated element by element like a char array. For example,

```
// Program - 14.8
#include <iostream.h>
#include <conio.h>
void main ( )
{
    clrscr();
    char name[ ] = "Pascal", reverse[7];
    int i= 0;
    while (name[i] != '\0')
        i++;
    reverse[i] = '\0';
    —i;
    int j = 0;
    while (i>= 0)
        reverse [i—] = name [j++];
    cout << "\n The contents of the string are: " << name;
    cout << "\n The contents of the reversed string ..." << reverse;
    getch ( );
}
```

What will be the output of the following program?

```
//Program - 14.9
# include <iostream.h>
# include <conio.h>
# include <string.h>
main()
{
    char word [ ] = "test";
    int i=0, k = 4, j = 0;
    clrscr( );
    while (i < 4)
    {
        j = 0;
        while (j<=i)
            cout << word [j++];
        cout << '\n';
        i++;
    }
    return 0;
}
```

#### 14.4 Two-Dimensional Array

A two-dimensional array is an array in which each element is itself an array. For instance, an array marks [3] [4] is a table with 3 rows, and 4 columns.

|     |     |     |     |
|-----|-----|-----|-----|
| 0,0 | 0,1 | 0,2 | 0,3 |
| 1,0 | 1,1 | 1,2 | 1,3 |
| 2,0 | 2,1 | 2,2 | 2,3 |

int marks [3] [4] = {90, 70, 80, 0, 75, 95, 65, 0, 80, 90, 90, 0}; will create a table as;

|   |    |    |    |   |
|---|----|----|----|---|
|   | 0  | 1  | 2  | 3 |
| 0 | 90 | 70 | 80 | 0 |
| 1 | 75 | 95 | 65 | 0 |
| 2 | 80 | 90 | 90 | 0 |

#### Note:

- ✓ The number of elements in a 2-dimensional array is determined by multiplying the number of rows with number of columns. In this example - The array marks has 12 elements.

- ✓ The subscripts always commence from zero. The subscript for rows is from 0 to 2, and for columns - 0 to 3.
- ✓ An element in a 2-D array is referred as Marks [Row] [Column]. For example, marks [0] [3] = marks [0] [0] + marks [0] [1] + marks [0] [2] will sum up the marks of the 1st row, viz., 90, 70, 80.

A 2-D array is declared as:

Type array-id [Rows] [Columns];

Example:

1. int a[3] [2]- declares 3 rows and 2 columns for the array a
2. const i=5; float num [i] [3] - declares a 2-D table num with 5 rows and 3 columns
3. short fine ['A'] ['E'] - declares a 2-D table of 65 rows and 69 columns

#### **Note:**

The dimensions (rows/columns) of an array can be indicated

1. using integer constants
2. using const identifier of integer or ordinal
3. using char constants
4. using enum identifiers

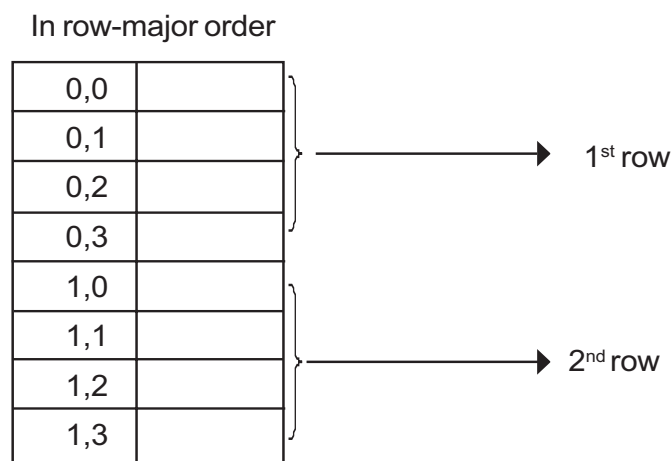
#### **14.4.1 Memory representation of 2-D arrays**

A 2-D array is stored in sequential memory blocks. The elements are stored either

1. row-wise manner (this method is called as row-major order)
2. column-wise manner (this method is called as column-major order)

For example:

int sales [2] [4]; will be stored as follows:



### Column-major order

|     |  |   |                          |
|-----|--|---|--------------------------|
| 0,0 |  | } | → 1 <sup>st</sup> column |
| 1,0 |  |   |                          |
| 0,1 |  | } | → 2 <sup>nd</sup> column |
| 1,1 |  |   |                          |
| 0,2 |  | } | → 3 <sup>rd</sup> column |
| 1,2 |  |   |                          |
| 0,3 |  | } | → 4 <sup>th</sup> column |
| 1,3 |  |   |                          |

Have you noticed the position of sales [1] [0] in row-major order and column-major order?

The size of a 2-D array is calculated as follows:

Number of elements \* memory req. for one element

For example - int sales [2] [4] the size will be calculated as follows:

Number of elements = Rows x columns = 2 x 4 = 8

∴ 8 x 2 (2 bytes is required for integer)

∴ size = 16 bytes

What will be the size of the array - float num [4] [6];

Solution: 4 x 6 x 4 = 96 bytes

Consider the following array:

int num [4] [3] = {8, 7, 6, 4, 5, 8, 9, 7, 6, 1, 2, 3};

|     |   |   |   |   |
|-----|---|---|---|---|
| num |   | 0 | 1 | 2 |
| 0   | 8 | 7 | 6 |   |
| 1   | 4 | 5 | 8 |   |
| 2   | 9 | 7 | 6 |   |
| 3   | 1 | 2 | 3 |   |

Write the appropriate reference for the highlighted elements of the table num.

Solution:

num [0] [1] - (7)

num [1] [1] - (5)

num [2] [0] - (9)

num [3] [1] - (2)

Determine the number of elements in the following declaration:

Solution:

a) int array [10] [12];

a) 120 elements

b) int x [ ] [2] = {0,1,1,2,2,3}

b) 6 elements (rows = 3 columns = 2)

The size of first dimension (first index) is optional in array initialization.

```
#include <iostream.h>
# include <conio.h>
#include <iomanip.h>
void accept (int s [3] [4], int total)
{ int r = 0; c = 0;
  for (; r < 3; r++)
    {cout << "\n Month: " << r+1;
      for (; c < 4; c++)
        {cout << '\n' << c+1 <<
"Quarter..";
```

Arrays can be passed on as arguments to functions. The actual parameter is passed only by the identifier, ignoring dimensions.

Array parameters by default behave like a reference parameter, as the array identifier unlike other identifiers, represents the base address of the array. Hence, it results in sending an address to the formal parameter (like reference parameters).



```

// Program - 14.10
#include <iostream.h>
# include <conio.h>
void accept (int s[3][4], int &total)
{
    int r = 0, c = 0;
    for (; r < 3; r++)
    {
        cout << "\n Month: " << r+1;
        for (c = 0; c < 4; c++)
        {
            cout << '\n' << c+1 << " Quarter..";
            cin >> s[r][c];
            total += s[r][c];
        }
    }
}

void display (int d[3][4], int total)
{
    int r, c;
    clrscr ( );
    cout << "\nSales figures for 3 months & their respective quarters..";
    for (r = 0; r < 3; r++)
    {
        cout << "\n Month ..." << r+1;
        for (c = 0; c < 4; c++)
            cout << '\t' << d[r][c];
    }
    cout << "\n Total sales .." << total;
}

void main ( )
{
    clrscr();
    int sales[3][4], t = 0;
    accept(sales,t);
    display(sales,t);
    getch();
}

```

Now look at the following program.

```
// Program - 14.11
# include <iostream.h>
# include <conio.h>
void accept (int a)
{
    cout << "\n Enter a number ..";
    cin >> a;
}

void display (int a)
{
    cout << '\n' << a;
}

void main ( )
{
    int num [2][2] ={{0,0},{0,0}}, r = 0, c = 0;
    clrscr ( );
    for (; r < 2; r++)
        for (; c < 2; c++)
            accept (num[r][c]);
    clrscr();
    for (r = 0; r < 2; r++ )
        for (c = 0; c < 2; c++)
            display (num[r][c]);
    getch();
}
```

Output: Assume data entered in accept () function is 1,2,3,4

0  
0  
0  
0

Why do you think the array num is not updated with the values 1,2,3,4?

In this example, the parameter passed to void accept ( ) is element by element. Hence, it is treated as value parameter and not reference parameter.

Note: Only the array identifier represents the base address of an array.

Now, rewrite the above program with the change - void accept (int a). On execution, if the same test data 1,2,3,4 is given, then the output displayed will be

1  
2  
3  
4

#### **14.4.2 Matrix**

A matrix is a set of mn numbers arranged in the form of a rectangular array of m rows and n columns. Matrices can be represented through 2-D arrays.

Program 14.12 demonstrates to read values for 2 matrices and check their equality.

#### **14.5 Array of Strings**

An array of strings is a two-dimensional character array. The size of first index (rows) determines the number of strings and the size of second index (column) determines maximum length of each string. For example,

```
char day-names [7] [10] = {"Sunday",  
                           "Monday",  
                           "Tuesday",  
                           "Wednesday",  
                           "Thursday",  
                           "Friday",  
                           "Saturday"};
```

will appear in the memory as shown in Table 14.1

```

//Program - 14.12
# include <iostream.h>
# include <conio.h>
void accept (int mat[3][3])
{
    clrscr();
    int r = 0, c = 0;
    for (; r < 3; r++)
    {
        cout << "\n Enter elements for row.." << r;
        for (c=0; c < 3; c++)
            cin >> mat[r][c];
    }
}
void main ( )
{
    int m1[3][3], m2[3][3];
    accept (m1);
    accept (m2);
    int i=0, j = 0, flag = 1;
    for (; i < 3; i++)
    {
        for (; j < 3; j++)
            if (m1[i][j] != m2[i][j])
            {
                flag = 0;
                break;
            }
        if (flag == 0)
            break;
    }
    if (flag)
        cout << "\n The matrices are equal ...";
    else
        cout << "\n The matrices are not equal..";
    getch ( );
}

```

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  |               |
|---|---|---|---|---|---|---|----|----|----|----|---------------|
| 0 | S | u | n | d | a | y | \0 |    |    |    | day-names [0] |
| 1 | M | o | n | d | a | y | \0 |    |    |    | day-names [1] |
| 2 | T | u | e | s | d | a | y  | \0 |    |    | day-names [2] |
| 3 | W | e | d | n | e | s | d  | a  | y  | \0 | day-names [3] |
| 4 | T | h | u | r | s | d | a  | y  | \0 |    | day-names [4] |
| 5 | F | r | i | d | a | y | \0 |    |    |    | day-names [5] |
| 6 | S | a | t | u | r | d | a  | y  | \0 |    | day-names [6] |

**Table 14.1 Array Elements in Memory**

An individual string is accessed as

day-names [0], i.e., by specifying the 1st index only. A specific character or an element is accessed as day-names [0] [5], i.e., by specifying both 1<sup>st</sup> and 2<sup>nd</sup> indices.

Attaching the null character (\0) to each string literal is optional. Even if we omit it, the C++ compiler will automatically attach it.

## EXERCISES

1. Why do the following snippets show errors?

a) `int a [5.5]`

Dimension of an array should be only an integer

b) `float f [3] = {1.0, 2.0};`

This will not show any error. But the number of elements is one less than the size of the array.

c) `float num [A];`

Dimension of an array should be explicitly mentioned. Here, the identifier A does not have a value. The statement may be rewritten as

`float num ['A']`

Or

`const A = 10; float num [A];`

d) `char a [3] [ ] = {"one", "two", "three"};`

The option for omitting the size of an array is given only for 1<sup>st</sup> index and not the second index. The statement may be rewritten as

`char a [ ] [6] = {"one", "two", "three"};`

e) `char ch [1] = 's';`

Character Array should be initialized using double quotes. The correct statement is

`char ch [1] = "s"`

Or

`char ch [1] = {"s"}`

f) `char test [4]; test = {"abc"};`

An array cannot be assigned in this manner. The correct statements are:

`char test [4] = "abc"` - initializing at the time of declaration

Or

`char test [4];`

`strcpy (test, "abc");`

g) `int num [ ] = {1,2,3}, num2 [3]; num2 = num;`

Group assignment of array is not allowed. One can assign only component by component.

h) `int num [3]; cout << num; cin >> num;`

Such I/O operations are not allowed on arrays. Manipulation of arrays is possible only by specific direction to its elements or components, i.e.

`cout << num [1] / cin >> num [1]`

i) `int roster = {1,2,3,4};`

The variable roster cannot take more than one value. Hence, the statements should be as:

`int roster = 10; or int roster [ ] = {1,2,3,4};`

2. What would be the contents of the array after initialization?

a) `int rate [ ] = {30,40,50};`

b) `char ch [6] = {" bbbb\0 " }`

`ch [0] = 'C';`

ch [4] = 'T';

ch [3] = 'A';

Note b indicates white/blank space.

c) char product-list [ ] [5] = {"nuts", "Bolts", "Screw"};

Solution

a - rate [0] = 30, rate [1] = 40. rate [2] = 50

b - ch [0] = 'C', ch [1] = ' '; ch [2] = ' ', ch [3] = 'A', ch [4] = 'T', ch [5] = '\0',

c - product-list [0] = "Nuts \0", product-list [1] = "Bolts \0",

product-list [2] = "Screw\0"

3. What would be the output of the following programs?

a) # include <iostream.h>

Solution: END

void main ( )

{char ch [ ] = {"END \0"S};

cout << ch;

}

b) # include <iostream.h>

void main ( )

{int a [ ] = {1,2,3,4,5};

for (int i = 0, i < 4, i++)

a[i+1] = a[i];

for (i= 0; i<5; i++)

cout << '\n' << a[i];

Solution:

|   |   |   |   |   |   |
|---|---|---|---|---|---|
|   | 0 | 1 | 2 | 3 | 4 |
| a | 1 | 2 | 3 | 4 | 5 |

i = 0 - a [1] = a [0]

will be as follows:

|   |   |   |   |   |   |  |
|---|---|---|---|---|---|--|
|   | 0 | 1 | 2 | 3 | 4 |  |
| a | 1 | 2 | 3 | 4 | 5 |  |

i = 1 - a [2] = a [1]

i = 2 - a [3] = a [2]

i = 3 - a [4] = a [3]

Hence, the array contents.

The output will be displayed as:

1  
1  
1  
1  
1

c) # include <iostream.h>

# include <conio.h>

void main ( )

{ char name [ ] = {"Jerry \0"}; int k = 5;

for (int i = 0 ; i < 3; i ++, k —)

name [k] = name [i];

cout << name;

getch ( );

}

Solution:

When i = 0 k = 5 name [5] = name [0]

i = 1 k = 4 name [4] = name [1]

i = 2 k = 3 name [3] = name [2]

∴ the output will be displayed as JerreJ

Program Writing

1. Write a program to declare and initialize an array called as int- array, that stores number 10,20,30,40 and 50. Display the sum of all the elements of int-array.
2. Write a program to declare an array of integers that can hold 10 values. Read the elements of the array from the user, and also display the contents in the reverse order.
3. Write a program to read a sentence into an identifier called as word from the user. Using while loop and switch statements, display the count of vowels present in the given sentence. For example:



word [ ] = "The vowel count AEIOU aeiou";  
Vowel count is 14

4. Write a program to create a MATRIX [3] [3]. Display the diagonal elements along with the sum of diagonal elements.
5. Write a program to read values for two matrices, viz., matrix A [4] [4], matrix B [4] [4]. Write program code to create sum-matrix [4] [4] that stores the sum of elements of matrix A and matrix B.

### Example

| matrix A |   |   |   | matrix B |   |   |   | sum-matrix |    |    |    |
|----------|---|---|---|----------|---|---|---|------------|----|----|----|
| 1        | 2 | 3 | 4 | 9        | 8 | 7 | 6 | 10         | 10 | 10 | 10 |
| 5        | 6 | 7 | 8 | 5        | 4 | 3 | 2 | 10         | 10 | -  | -  |
| 9        | 1 | 2 | 3 | 1        | 9 | 8 | 7 | -          | -  | -  | -  |
| 4        | 5 | 6 | 7 | 6        | 5 | 4 | 3 | -          | -  | -  | -  |

## CHAPTER 15

# CLASSES AND OBJECTS

### 15.1 Introduction to Classes

The most important feature of C++ is the “Class”. Its significance is highlighted by the fact that Bjarne Stroustrup initially gave the name ‘C with Classes’. A class is a new way of creating and implementing a user defined data type. Classes provide a method for packing together data of different types. For Example:

```
class student
{
    char name[30];
    int rollno, marks1, marks2, total_marks;
};
```

The data variables, rollno, marks1, marks2, total\_marks define the properties or features of a student, thus packing together data of different types. The class data type can be further extended by defining its associated functions. These functions are also called as methods, as they define the various operations (in terms of accepting and manipulating data) that can be performed on the data.

#### In other words

✓ A class is a way to bind the data and its associated functions together

### 15.2 Specifying a class :

A class specification has two parts :

- 1) Class declaration
- 2) Class Function Definitions

```
// Program - 15.1
# include <iostream.h>
# include <conio.h>
class student
{
    private :
        char name[30];
        int rollno, marks1, marks2 ,total_marks;
    protected:
        void accept()
        {
            cout<<“\n Enter data name, roll no, marks 1 and marks 2.. “;
            cin>>name>>rollno>>marks1>>marks2;
        }
}
```

```

void compute()
{
    total_marks = marks1+ marks2;
}
void display()
{
    cout<<"\n Name "<<name;
    cout<<"\n Roll no "<<rollno;
    cout<<"\n Marks 1.. "<<marks1;
    cout<<"\n Marks 2.. "<<marks2;
    cout<<"\n Total Marks.. "<< total_marks;
}
public:
student()
{
    name[0]='\0';
    rollno=marks1=marks2=total_marks= 0;
    cout<<"\n Constructor executed ... ";
}
void execute()
{
    accept();
    compute();
    display();
}
};
void main()
{
    clrscr();
    student stud;
    stud.execute();
}

```

The form of class declaration is

General Form

```
class class-name
{
    private:
        variable declaration
        function declaration

    protected:
        variable decl.
        function decl.

    public:
        variable decl.
        function decl.
};
```

With respect to the above example

```
class student
{ private:
    char name [10];
    int roll no, mark1, mark2, total marks;

    protected:
        void accept( );
        void compute( );
        void display( );

    public:
        student( );
        void execute( );
};
```

- ✓ The keyword class specifies user defined data type class name
- ✓ The body of a class is enclosed within braces and is terminated by a semicolon
- ✓ The class body contains the declaration of variables and functions
- ✓ The class body has three access specifiers ( visibility labels) viz., private , public and protected
- ✓ Specifying private visibility label is optional. By default the members will be treated as private if a visibility label is not mentioned
- ✓ The members that have been declared as private, can be accessed only from within the class
- ✓ The members that have been declared as protected can be accessed from within the class, and the members of the inherited classes.
- ✓ The members that have been declared as public can be accessed from outside the class also

### 15.3 Data Abstraction

The binding of data and functions together into a single entity is referred to as **encapsulation**.

The members and functions declared under private are not accessible by members outside the class, this is referred to as **data hiding**. Instruments allowing only selected access of components to objects and to members of other classes is called as **Data Abstraction**. Or rather Data abstraction is achieved through data hiding.

Data hiding is the key feature of object oriented programming ( OOPS)

|           |                                                                                                    |
|-----------|----------------------------------------------------------------------------------------------------|
| private   | Accessible by only its own members and certain special functions called as <b>friend functions</b> |
| protected | Accessible by members of inherited classes                                                         |
| public    | Access allowed by other members in addition to class member and objects                            |

#### 15.4 Data Members and Member Functions

Class comprises of members. Members are further classified as Data Members and Member functions. Data members are the data variables that represent the features or properties of a class. Member functions are the functions that perform specific tasks in a class. Member functions are called as methods, and data members are also called as attributes. Now look at the Table 15.1 where information is provided based on Program -15. 1 **class student**. **Classes** include special member functions called as constructors and destructors. These will be dealt in Chapter – 17 Constructors and Destructors.

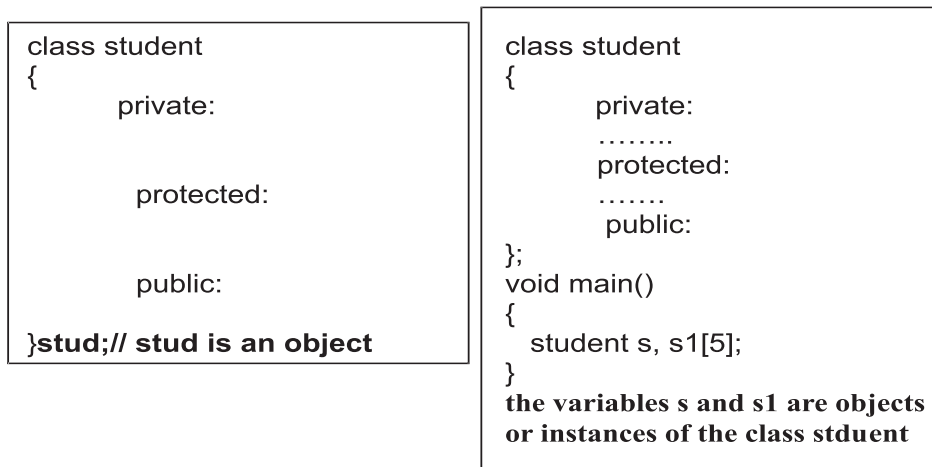
|                                                                               |                                                              |
|-------------------------------------------------------------------------------|--------------------------------------------------------------|
| student                                                                       | The data type identifier student is also called as class tag |
| name, rollno, marks1, marks2, total                                           | data members                                                 |
| public accept ( )<br>compute ( )<br>display ( )<br>execute ( )<br>student ( ) | member functions or methods                                  |
| stud                                                                          | instance/object/variable of class student                    |

**Table 15.1 Class Student of Program - 15.1**

#### 15.5 Creating Objects

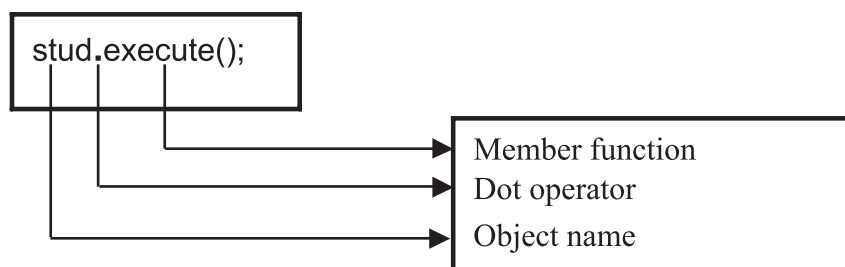
Look at the following declaration statement **student stud**; This statement may be read as **stud is an instance or object of the class student**.

Once a class has been declared, variables of that type can be declared. 'stud' is a variable of type student ,student is a data type of class . In C++ the class variables are known as objects. The declaration of an object is similar to that of a variable of any basic type. Objects can also be created by placing their names immediately after the closing brace of the class declaration.



## 15.6 Accessing Class Members

The members of a class are accessed using the dot operator. For example, the call statement to the function `execute()` of the class `student` may be given as:



The private data of a class can be accessed only through the member functions of its own class and certain special functions called as friend functions.

In the example class `student`, the data members `name`, `marks1`, `marks2`, `rollno`, `total_marks` are accessed only by the member functions `accept()`, `display()`, `compute()`. The objects declared outside the class cannot access members or functions defined under `private` or `protected`.

The member functions declared under `public` can be accessed by the objects of that class. The call statement `stud.execute();` is a valid statement, as `execute()` is a member function defined under `public` visibility mode and hence can be accessed through the object `stud` defined outside the class. Whereas the statements: `stud.accept()`, `stud.compute()`, `stud.display()`, `stud.marks1` etc would force the compiler to throw error messages “not accessible”. Program –15.2 is another example that demonstrates the operation – addition of two numbers. This class wraps three integer variables, and its related member function to accept data, and perform addition. Since the variable `sum` is defined under `public` visibility mode, the object is accessing it.

```

//Program - 15.2
# include <iostream.h>
# include <conio.h>

class add
{
    private:
        int a,b;
    public:
        int sum;

        void getdata()
        {
            a=5;
            b=10;
            sum = a+b;
        }
};

void main()
{
    add s;
    s.getdata();
    cout<<s.sum;
}

```

## 15.7 Defining methods of a class

```

class add
{
    int a,b;
    public:
        add()
        {
            a='\0';
            b='\0';
        }
        void display();
};

void add::display()
{
    int sum;
    sum = a+b;
    cout<<sum;
}

```

Method 1

Method 2

In Method 1, the member function `add()` is declared and defined within class `add`.

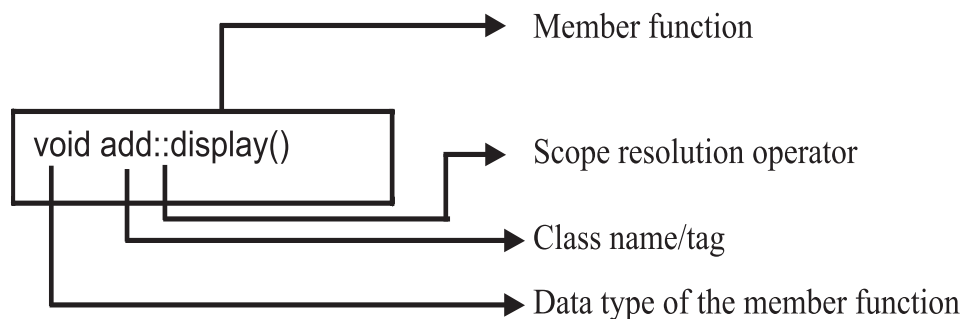
In Method 2, the member function `display()` is declared within the class, and defined outside the class.

Methods of a class can be defined in both ways. The members defined within the class behave like inline functions.

Member functions defined outside the class has the prototype as

```
type class_name :: function name();
```

For example:



The membership label `class_name::` ( `add::` ) tells the compiler that the function `function_name` belongs to the class `class_name`. That is the scope of the function is restricted to the class specified in the function header.

The member function have some special characteristics that are often used in the program development .

- ✓ Several different classes can use the same function name. The 'membership' label will resolve their scope
- ✓ Member functions can access the private data of a class. A non- member function cannot do so.
- ✓ A member function can call another member function directly, without using the dot operator. ( This is called as nesting of member functions )
- ✓ The member functions can receive arguments of a valid C++ data type. Objects can also be passed as arguments
- ✓ The return type of a member function can be of object data type
- ✓ Member functions can be of static type



## 15.8 Memory allocation of objects

The member functions are created and placed in the memory space only when they are defined as a part of the class specification. Since all the objects belonging to that class use the same member function, no separate space is allocated for member functions when the objects are created. Memory space required for the member variables are only allocated separately for each object. Separate memory allocations for the objects are essential because the member variables will hold different data values for different objects

Look at the following class declaration:

```
class product
{
    int code, quantity;
    float price;
    public:
        void assign_data();
        void display();
};
void main()
{
    product p1, p2;
}
```

Member functions `assign_data()` and `display()` belong to the common pool in the sense both the objects `p1` and `p2` will have access to the code area of the common pool.

Memory for Objects for `p1` and `p2` is illustrated:

| objects | Data members             | Memory allotted |
|---------|--------------------------|-----------------|
| p1      | Code, quantity and price | 8 bytes         |
| p2      | Code,quantity and price  | 8 bytes         |

**Table 15.2 Memory Allocation for Objects**

Member functions of a class can handle arguments like any other non member functions as illustrated in Program - 15.3.

## 15.9 Static Data Members

A data member of a class can be qualified as static

The static member variable

- ✓ Is initialized to zero, only when the first object of its class is created . No other initialization is permitted
- ✓ Only one copy of the member variable is created (as part of the common pool ) and is shared by all the other objects of its class type
- ✓ Its scope or visibility is within the class but its lifetime is the lifetime of the program.

```
// Program - 15.3
#include<iostream.h>
#include<conio.h>

class product
{
    int code, quantity;
    float price;
public:
    void assign_data( int c, int q, float p)
    {
        code = c;
        quantity = q;
        price = p;
    }
    void display()
    {
        cout<<"\n Code : "<<code;
        cout<<"\n Quantity : "<<quantity;
        cout<<"\n Price : "<< price;
    }
};

void main()
{
    product p;
    p.assign_data( 101, 200, 12.5);
    p.display();
}
```

```

// Program - 15.4
// To demonstrate the use of static member variables

#include<iostream.h>
#include<conio.h>
class simple_static
{
    int a,b,sum;
    static int count;
public:
    void accept()
    {
        cout<<"\n Enter values.. ";
        cin>>a>>b;
        sum = a+b;
        count++;
    }
    void display()
    {
        cout<<"\n The sum of two numbers ... "<<sum;
        cout<<"\n This is addition... "<<count;
    }
};
int static simple _ count=0;
void main()
{
    simple_static p1,p2,p3;
    p1.accept();
    p1.display();
    p2.accept();
    p2.display();
    p3.accept();
    p3.display();
}

```

OUTPUT :

```

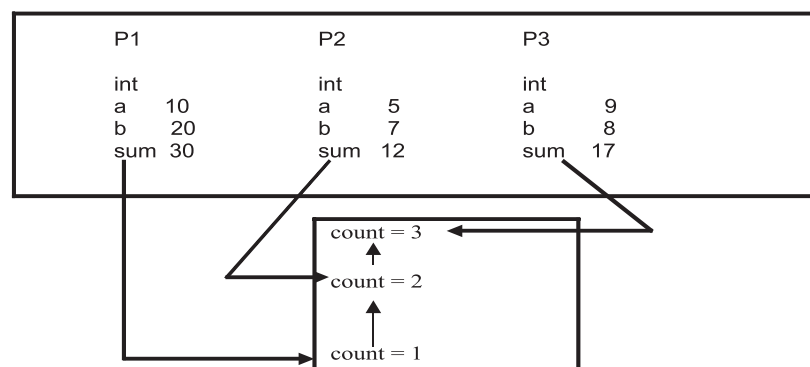
Enter values ..... 10 20
The sum of two numbers ..... 30
This is addition 1

```

Enter values..... 5 7  
 The sum of two numbers.....12  
 This is addition 2

Enter values..... 9 8  
 The sum of two numbers .....17  
 This is addition 3

The static variable count is initialized to zero only once. The count is incremented whenever the sum of the two numbers was calculated. Since the function accept() was invoked three times, count was incremented thrice and hence the value is 3. As only one copy of count is shared by all the three objects, the value of count is set to 3. This is shown in Fig. 15.2.

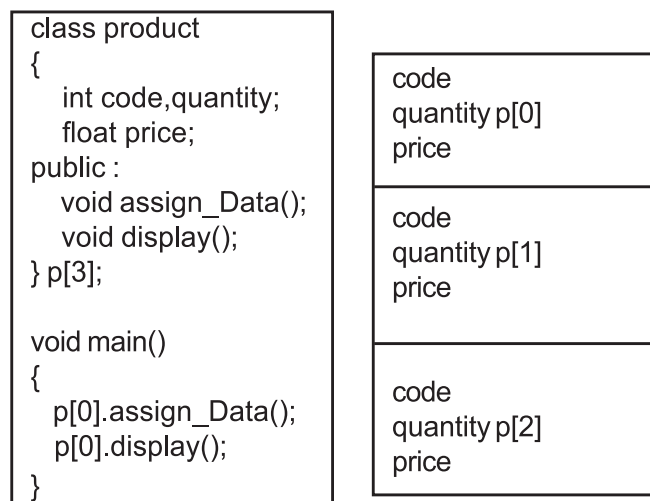


**Fig. 15.2 Static Member Variable - Count**

The initial value to a static member variable is done outside the class.

### 15.10 Arrays of objects

Consider the following class definition and its corresponding memory allocation:



## EXERCISES

### I. Identify and correct the errors in the following

```
class x
{
    public:
        int a,b;
        void init()
        {
            a =b = 0;
        }
        int sum();
        int square();
};
int sum()
{
    return a+b;
}
int square()
{
    return sum() * sum()
}
```

Solution :

int x::sum() and int x::square()

### II

```
#include<iostream.h>
class simple
{
    int num1, num2 , sum = 0;
protected:
    accept()
    {
        cin>>num1>>num2;
    }
public:
    display()
    {
        sum = num1 + num2;
    }
};
```

```

void main()
{
    simple s;
    s.num1=s.num2= 0;
    s.accept();
    display();
}

```

Solution:

- 1) The member sum cannot be initialized at the time of declaration
- 2) The member variable num1 and num2 cannot be accessed from main() as they are private
- 3) s.accept() is invalid. The method accept() is defined under protected
- 4) display() should be invoked through an object

III

```

#include<iostream.h>
#include<conio.h>
class item
{
    private:
        int code,quantity;
        float price;
        void getdata()
        {
            cout<<"\n Enter code, quantity, price ";
            cin>>code>>quantity>>price;
        }
    public:
        float tax='\0';
        void putdata()
        {
            cout<<"\n Code : "<<code;
            cout<<"\n Quantity : "<<quantity;
            cout<<"\n Price : "<<price;
            if( quantity >100 )
                tax = 2500;
            else
                tax =1000;
            cout<<" \n Tax : "<<tax;
        }
};
void main()
{ item i; }

```

Complete the following table based on the above program

|                                  |                      |                     |                                                   |
|----------------------------------|----------------------|---------------------|---------------------------------------------------|
| Memory allocation for instance i | Private data members | Public data members | Methods or data members that can be accessed by i |
|----------------------------------|----------------------|---------------------|---------------------------------------------------|

#### IV

- 1) Define a class employee with the following specification private members of class employee

empno- integer

ename – 20 characters

basic – float

netpay, hra, da, float

calculate () – A function to find the basic+hra+da with float return type public member functions of class employee

havedata() – A function to accept values for empno, ename, basic, hra, da and call calculate() to compute netpay

dispdata() – A function to display all the data members on the screen

- 2) Define a class MATH with the following specifications

private members

num1, num2, result – float

init() function to initialize num1, num2 and result to zero

protected members

add() function to add num1 and num2 and store the sum in result prod() function to multiply num1 and num2 and store the product in the result

public members

getdata() function to accept values for num1 and num2

menu() function to display menu

1. Add...

2. Prod...

invoke add() when choice is 1 and invoke prod when choice is 2 and also display the result.

## CHAPTER 16

# POLYMORPHISM

### 16.1 Introduction

The word polymorphism means many forms (poly – many, morph – shapes). In C++, polymorphism is achieved through function overloading and operator overloading. The term overloading means a name having two or more distinct meanings. Thus an **‘overloaded function’** refers to a function having more than one distinct meaning. Function overloading is one of the facets of C++ that supports object oriented programming.

### 16.2 Function overloading

The ability of the function to process the message or data in more than one form is called as function overloading.

Consider the situation wherein a programmer desires to have the following functions

area\_circle() // to calculate the area of a circle

area\_triangle() // to calculate the area of a triangle

area\_rectangle() // to calculate the area of a rectangle

The above three different prototype to compute area, for different shapes can be rewritten using a single function header in the following manner

float area ( float radius);

float area ( float half, float base, float height );

float area ( float length , float breadth);

Now look at the ease in invoking the function area(...) for any of the three shapes as shown in Program - 16.1

```
// Program - 16.1
// to demonstrate the polymorphism - function overloading
#include<iostream.h>
#include<conio.h>

float area ( float radius )
{
    cout << "\nCircle ...";
    return ( 22/7 * radius * radius );
}
float area (float half, float base, float height)
```



```

{    cout << "\nTriangle ..";
    return (half* base*height);
}
float area ( float length, float breadth )
{    cout << "\nRectangle ...";
    return ( length *breadth ) ;
}
void main()
{
    clrscr();
    float r,b,h;
    int choice = 0 ;
    do
    {
        clrscr();
        cout << "\n Area Menu ";
        cout << "\n 1. Circle ... ";
        cout << "\n 2. Traingle ... ";
        cout << "\n 3. Rectangle ... ";
        cout << "\n 4. Exit ... ";
        cin>> choice;
        switch(choice)
        {
            case 1 :
                cout << "\n Enter radius ... ";
                cin>>r;
                cout<<"\n The area of circle is ... "
                    << area(r);
                getch();
                break;
            case 2:
                cout<< "\n Enter base, height ... ";
                cin>>b>>h;
                cout<<"\n The area of a triangle is .. "
                    << area (0.5, b, h);
                getch();
                break;
            case 3:
                cout<< "\n Enter length, breadth.. ";
                cin>>h>>b;
                cout<<"\n The area of a rectangle is ... "
                    << area(h,b);
                getch();
                break;
        }
    }while (choice <=3);
}

```

Have you noticed the function prototypes for all the 3 functions? The prototypes are:

```
float area ( float radius );
```

```
float area (float half, float base, float height);
```

```
float area ( float length, float breadth );
```

How do you think each function 'area' definition is differing from one and another? Yes, each function prototype differs by their number of arguments. The first prototype had one argument, second one 3 arguments and the third one had 2 arguments. In the example we have dealt , all the three functions has float type arguments. It need not necessarily be this way. Arguments for each prototype can be of different data type. Secondly the number of arguments for each function prototype may also differ. The following prototypes for function overloading is invalid. Can you tell why is it so ?

| Function Prototype                                                                              | Invalid prototype                                                                                                                                      |
|-------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>void fun(int x);<br/>void fun(char ch);<br/>void fun(int y);<br/>void fun(double d);</pre> | <pre>void fun(<b>int x</b>);<br/>void fun(<b>int y</b>);</pre> <p>Both the prototypes have same number and type of arguments. Hence it is invalid.</p> |

### How are functions invoked in function overloading?

The compiler adopts BEST MATCH strategy. As per this strategy, the compiler will

- ✓ Look for the exact match of a function prototype with that of a function call statement
- ✓ In case an exact match is not available, it looks for the next nearest match. That is, the compiler will promote integral data promotions and then match the call statement with function prototype.

For example, in the above example (program –1 ) we have **float area(float radius)** with area(r) where the parameter 'r' should be of float type. In case, the variable 'r' is of integer type, then as per integral promotions integer constant/variable can be mapped to char, or float or double. So, by this strategy the area(r) will be mapped to area(float radius).

Integral promotions are purely compiler oriented. By and large integral promotions are as follows:

- ✓ char data type can be converted to integer/float/double

- ✓ int data type can be converted to char/double/float
- ✓ float data type to integer/double/char
- ✓ double data type to float or integer

Now based on the following call statements to area() of Program -16.1 can you tell as to what will be the output?

| Function call statement | Output displayed |
|-------------------------|------------------|
| area(5.0)               |                  |
| area(0.5,4.0,6.0)       |                  |
| area(3.0,4.5)           |                  |

### Rules for function overloading

- 1) Each overloaded function must differ either by the number of its formal parameters or their data types
- 2) The return type of overloaded functions may or may not be the same data type
- 3) The default arguments of overloaded functions are not considered by the C++ compiler as part of the parameter list
- 4) Do not use the same function name for two unrelated functions

Improper declarations leading to conflict in a function call statement is shown below.

```

void fun ( char a, int times)
{
    for (int i=1; i<=times;i++)
        cout<<a;
}
void fun( char a= '*', int times )
{
    for(int i=1;i<=times;i++)
        cout<<a;
}
void fun( int times)
{
    for(int i=1; i<=times ;i++)
        cout<<'@';
}
void main()
{
    fun ( '+', 60);
    fun(60);
}

```

When the above program is compiled, two errors will be flagged:

- ✓ Conflict between fun(char a, int times) and fun( char a='', int times)
- ✓ Conflict between fun( char a='', int times ) and fun (int times)

The call statement fun( '+', 60) can be matched with fun ( char a, int times ) and fun ( char a='', int times )

The call statement fun(60) can be matched with fun ( char a='', int times ) and fun ( int times )

Overload a function with the help of different function definitions having a unique parameter list. That is, the parameter list differ either by number or types.

### 16.3 Operator Overloading

The term operator overloading, refers to giving additional functionality to the normal C++ operators like +, ++, -, --, +=, -=, \* <, >. The statement sum = num1 + num2 would be interpreted as a statement meant to perform addition of numbers(integer/float/double) and store the result in the variable sum. Now look at the following statement:

```
name = first_name + last_name;
```

where the variables name, first\_name and last\_name are all character arrays. Can one achieve concatenation of character arrays using '+' operator in C++? The compiler would throw an error stating that '+' operator cannot handle concatenation of strings. The user is forced to use **strcat()** function to concatenate strings. Won't it be a lot easier if one is permitted to use '+' operator on strings as used for number data type? The functionality of '+' operator can be extended to strings through **operator overloading**.

Look at the following example:

```
//Program-16.2 – OPERATOR OVERLOADING
#include <iostream.h>
#include <conio.h>
#include <string.h>

class strings
{
    chars[10];
public:
    strings()
    {
        s[0]='\0';
    }
}
```

```
void main()
{
    clrscr();
    strings s1("test"),s2(" run\0");
    char *concatstr ;
    concatstr = s1 + s2;
    cout << "\nConcatenated string ..."
        << concatstr;
    getch();
}
```

```

strings(char*c)
{
    strcpy(s,c);
}

char*operator+(strings x1)
{
    char*temp;
    strcpy(temp,s);
    strcat(temp,x1.s);
    return temp;
}
};

```

The statement **concatstr = s1 + s2** merges two strings, as the operator '+' is given additional function through the member function:

**char \* operator + (strings x1)**

The member function **char \* operator + (strings x1)** takes x1 as the argument. It may be viewed as:

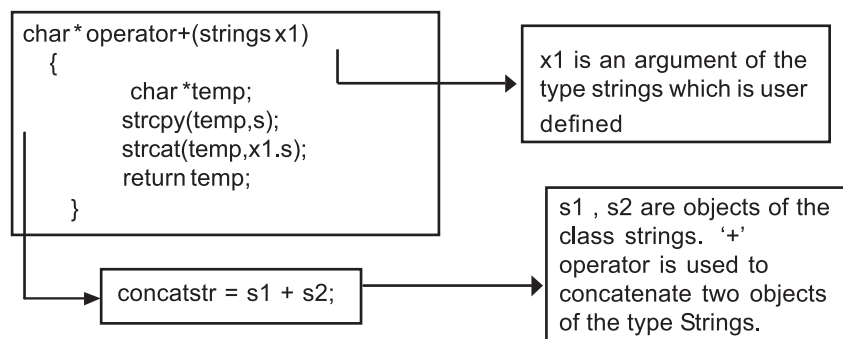
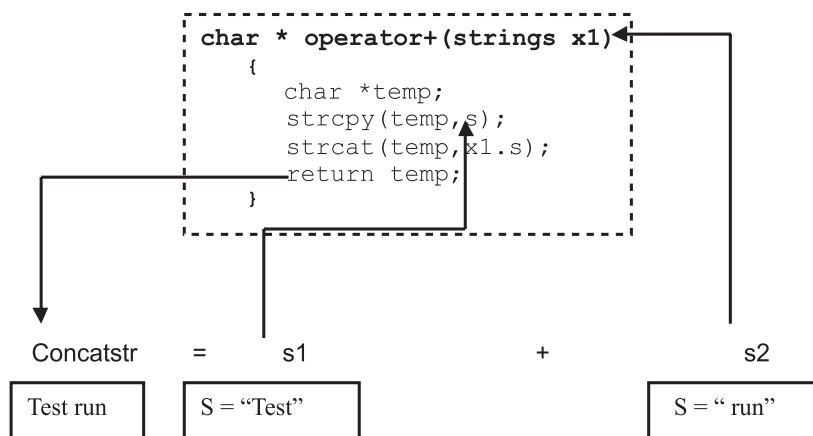


Fig. 16.1 demonstrates the association of variables and their values.



**Fig. 16.1 Association of Variables and Values**

## Operator overloading provides:

- ✓ New function definitions for basic C++ operators like +, \*, -, ++, --, >, <, += and the like. One cannot overload C++ specific operators like membership operator (.), scope resolution operator (::), sizeof operator and conditional operator.
- ✓ The overloaded function definitions are permitted for user defined data type.
- ✓ Operator functions must be either member functions or friend functions. (Friend functions is beyond the scope of this book)
- ✓ The new definition that is provided to an operator does not overrule the original definition of the operator. For example, in the above program – OPERATOR OVERLOADING the '+' operator has been used to merge two strings. In the same program one can also perform addition of numbers in the usual way. The compiler applies user defined definition based on the style of call statement. That is the statements `cout << 5 + 10` will display the result as 15 (original definition of '+' is applied), where as `concatstr=s1 + s2` will invoke the member function **char\*operator+ ( strings s1)** as the operands provided for the '+' operator are s1 and s2 which are the objects of the class strings.

## The process of overloading involves:

- ✓ Create a class that defines the data type that is to be used in the overloading operations
- ✓ Declare the operator function **operator ()** in the public part of the class.
- ✓ Define the operator function to implement the required operations.

The following examples demonstrate the ease of using operators with user defined data types – objects.

Program – 16.3 demonstrates as to how one can negate the data members of a class using the operator – (minus)

|                                                                                                                                                                                                                                                                                                              |                                                                                                                                                      |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>// Program -16.3 #include &lt;iostream.h&gt; #include &lt;conio.h&gt;  class negative {     int i; public :     void accept()     {         cout &lt;&lt; "\nEnter a number ...";         cin &gt;&gt; i;     }     void display()     {         cout &lt;&lt; "\nNumber ..." &lt;&lt; i;     } }</pre> | <pre>void operator-() {     i = -i; }  void main() {     clrscr();     negative n1;     n1.accept();     -n1;     n1.display();     getch(); }</pre> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|

The function void **operator –()** simply negates the data members of the class as one would do with a normal variable as follows: sum = - num1;

Look at the following program and answer the questions:

```
// Program – 16.4
#include <iostream.h>
#include <conio.h>

class distance
{
    int feet,inches;
    public :
void distance_assign(int f, int i)
    {
        feet = f;
        inches = i;
    }

    void display()
    {
        cout << "\nFeet   : " << feet << "\tInches : " << inches;
    }
distance operator+(distance d2)
    {
        distance d3;
        d3.feet = feet + d2.feet;
        d3.inches = (inches + d2.inches) % 12;
        d3.feet += (inches + d2.inches)/12;
return d3;
    }
};
void main()
{
    clrscr();
    distance dist_1,dist_2;
    dist_1.distance_assign(12,11);
    dist_2.distance_assign(24,1);
    distance dist_3 = dist_1 + dist_2;
    dist_1.display();
    dist_2.display();
    dist_3.display();
    getch();
}
```

1. Identify the operator that is overloaded.
2. Write out the prototype of the overloaded member function.
3. What types of operands are used for the overloaded operator?
4. Write out the statement that invokes the overloaded member function.

Program-16.5 demonstrates the overloaded functions of += and -=

```
//Program-16.5
// operator overloading

# include <iostream.h>
# include <conio.h>
# include <string.h>

class library_book
{
    char name[25];
    int code,stock;

    public :

    void book_assign(char n[15],int c,int s)
    {
        strcpy(name,n);
        code = c;
        stock = s;
    }

    void display()
    {
        cout << "\n    Book details ....";
        cout << "\n    1. Name      ...." << name;
        cout << "\n    2. Code      ...." << code;
        cout << "\n    3. Stock     ...." << stock;
    }

    void operator +=(int x)
    {
        stock += x;
    }

    void operator -=(int x)
    {
        stock -= x;
    }
};
```



```

class library_cdrom
{
    char name[25];
    int code,stock;
public :
    void cdrom_assign(char n[15],int c,int s)
    {
        strcpy(name,n);
        code = c;
        stock = s;
    }

    void display()
    {
        cout << "\n CD ROM details ....";
        cout << "\n 1. Name ....." << name;
        cout << "\n 2. Code ....." << code;
        cout << "\n 3. Stock ....." << stock;
    }

    void operator +=(int x)
    {
        stock += x;
    }

    void operator -=(int x)
    {
        stock -= x;
    }
};

void main()
{
    library_book book;
    library_cdrom cdrom;

    book.book_assign("Half Blood Prince",101,55);
    cdrom.cdrom_assign("Know your Basics",201,50);

    char choice,borrow;

```

```

do
{
    cout << "\nBook,cdrom,exit<b/c/e> ...";
    cin >> choice;
    if (choice != 'e')
    {
        cout << "\nBorrow/Return <b/r> ...";
        cin >> borrow;
    }
} while (choice != 'e');
switch (choice)
{
    case 'b' :
        switch (borrow)
        {
            case 'b' : book += 1;break;
            case 'r' : book -= 1;break;
        }
        book.display(); break;
    case 'c' :
        switch (borrow)
        {
            case 'b' : cdrom += 1;break;
            case 'r' : cdrom -= 1;break;
        }
        cdrom.display(); break;

    case 'e' : cout << "\nTerminating ..";
                break;
}
} while (choice != 'e');
getch();
}

```

Have you noticed the ease with which the objects stock is incremented or decremented in a standard style by using the operators += / -=

|       |       |
|-------|-------|
| book  | += 1; |
| book  | -= 1; |
| cdrom | += 1; |
| cdrom | -= 1; |

The mechanism of giving **special meaning to an operator** is called as **operator overloading**.

### Rules for overloading operators:

There are certain restrictions and limitations in overloading operators. They are:

- ✓ Only existing operators can be overloaded. New operators cannot be created.
- ✓ The overloaded operator must have at least one operand of user defined type.
- ✓ The basic definition of an operator cannot be replaced or in other words one cannot redefine the function of an operator. One can give additional functions to an operator
- ✓ Overloaded operators behave in the same way as the basic operators in terms of their operands.
- ✓ When binary operators are overloaded, the left hand object must be an object of the relevant class
- ✓ Binary operators overloaded through a member function take one explicit argument.

## EXERCISES

I. Write a program that uses function overloading to do the following tasks

- a. find the maximum of two numbers ( integers )
- b. find the maximum of three numbers ( integers )

SOLUTION: function prototype – max (int , int) and max( int , int, int)

II. Write function definitions using function overloading to

- a. increment the value of a variable of type float
- b. increment the value of a variable of type char

SOLUTION: function prototype – float incr (float) , char incr (char)

III. Write a program in C++ to do the following tasks using function overloading

a. compute  $x^y$  where x and y are both integers

b. compute  $x^y$  where x and y are both float

SOLUTION: function prototype – `int power(int,int),float power(float,float);`

IV. What is the advantage of operator overloading?

V. List out the steps involved to define an overloaded operator.

VI. List out the operators that cannot be overloaded.

VII. Write a program to add two objects of the class **complex\_numbers**. A complex number has two data members – real part and imaginary part. Complete the following definition and also write a `main()` function to perform addition of the `complex_numbers` objects `c1` and `c2` .

```
class complex_numbers
{
float x;

float y;

public :

void assign_data(float real, float imaginary);

void display_data();

complex_numbers operator +(complex_numbers n1);

}
```

## CHAPTER 17

# CONSTRUCTORS AND DESTRUCTORS

### 17.1 Introduction

When an instance of a class comes into scope, a special function called the constructor gets executed. The constructor function initializes the class object. When a class object goes out of scope, a special function called the destructor gets executed. The constructor function name and the destructor have the same name as the class tag. Both the functions return nothing. They are not associated with any data type.

### 17.2 Constructors

```
// Program - 17.1
// to determine constructors and destructors
#include<iostream.h>
#include<conio.h>
class simple
{
    private:
        int a,b;
    public:
        simple()
        {
            a= 0 ;
            b= 0;
            cout<< "\n Constructor of class-simple ";
        }
        ~simple()
        {
            cout<< "\n Destructor of class – simple .. ";
        }
        void getdata()
        {
            cout<< "\n Enter values for a and b... ";
            cin>>a>>b;
        }
        void putdata()
        {
            cout<< "\n The two integers .. "<<a<<'\t'<< b;
            cout<< "\n The sum of the variables .. "<< a+b;
        }
};
void main()
{
    simple s;
    s.getdata();
    s.putdata();
}
```

When the above program is executed, constructor simple() is automatically executed when the object is created. Destructor ~ simple() is executed, when the scope of the object 's' is lost, i.e., at the time of program termination.

**The output of the program will be as follows:**

Constructor of class - simple ..

Enter values for a & b... 5 6

The two integers..... 5 6

The sum of the variables..... 11

Destructor of class - simple ...

### 17.3 Functions of constructor

- 1) The constructor function initializes the class object
- 2) The memory space is allocated to an object

### 17.4 Constructor overloading

Function overloading can be applied for constructors, as constructors are special functions of classes. Program - 17.2 demonstrates constructor overloading.

The constructor add() is a constructor without parameters(non parameterized). It is called as default constructor. More traditionally default constructors are referred to compiler generated constructors i.e., constructors defined by the computers in the absence of user defined constructor. A non- parameterized constructor is executed when an object without parameters is declared.

```
// Program - 17.2
// To demonstrate constructor overloading
#include<iostream.h>
#include<conio.h>
class add
{
    int num1, num2, sum;
    public:
    add()
    {
        cout<<"\n Constructor without parameters.. ";
        num1= 0;
        num2= 0;
        sum = 0;
    }
}
```

```

add ( int s1, int s2 )
{
    cout<<"\n Parameterized constructor... ";
    num1= s1;
    num2=s2;
    sum=NULL;
}
add (add &a)
{
    cout<<"\n Copy Constructor ... ";
    num1= a.num1;
    num2=a.num2;
    sum = NULL;
}
void getdata()
{
    cout<<"Enter data ... ";
    cin>>num1>>num2;
}
void addition()
{
    sum=num1+num2;
}
void putdata()
{
    cout<<"\n The numbers are..";
    cout<<num1<<'t'<<num2;
    cout<<"\n The sum of the numbers are.. "<< sum;
}
};
void main()
{
    add a, b (10, 20) , c(b);
    a.getdata();
    a.addition();
    b.addition();
    c.addition();
    cout<<"\n Object a : ";
    a.putdata();
    cout<<"\n Object b : ";
    b.putdata();
    cout<<"\n Object c.. ";
    c.putdata();
}

```

## OUTPUT:

Constructor without parameters....

Parameterized Constructor...

Copy Constructors...

**Enter data .. 5      6**

**Object a:**

The numbers are 5 6

The sum of the numbers are ..... 11

**Object b:**

The numbers are 10      20

The sum of the numbers are ... 30

**Object c:**

The numbers are 10 20

The sum of the numbers are ..... 30

The constructor `add ( int s1, int s2)` is called as parameterized constructor .To invoke this constructor , the object should be declared with two integer constants or variables .

**Note:** char, float double parameters can be matched with int data type due to implicit type conversions

**For example:** `add a ( 10, 60 ) / add a ( ivar, ivar )`

The constructor `add (add &a )` is called as copy constructor. A copy constructor is executed:

- 1) When an object is passed as a parameter to any of the member functions.  
Example `void add::putdata( add x);`
- 2) When a member function returns an object For example, `add getdata();`
- 3) When an object is passed by reference to constructor For example, `add a; b(a);`

The following program – 2 demonstrates as to when a copy constructor is executed?



```

// Program – 17.3
// To demonstrate constructor overloading
#include<iostream.h>
#include<conio.h>
class add
{
    int num1, num2, sum;
public:
    add()
    {
        cout<<"\n Constructor without parameters.. ";
        num1= '\0';
        num2= '\0';
        sum = '\0';
    }
    add ( int s1, int s2 )
    {
        cout<<"\n Parameterized constructor... ";
        num1= s1;
        num2=s2;
        sum=NULL;
    }
    add (add &a)
    {
        cout<<"\n Copy Constructor ... ";
        num1= a.num1;
        num2=a.num2;
        sum = NULL;
    }
    void getdata()
    {
        cout<<"Enter data ... ";
        cin>>num1>>num2;
    }
    void addition(add b)
    {
        sum=num1+ num2 +b.num1 + b.num2;
    }
    add addition()
    {
        add a(5,6);
        sum = num1 + num2 +a.num1 +a.num2;
    }
    void putdata()
    {
        cout<<"\n The numbers are..";
        cout<<num1<<"\t"<<num2;
        cout<<"\n The sum of the numbers are.. "<< sum;
    }
};

```

```

void main()
{
    clrscr();
    add a, b (10, 20) , c(b);
    a.getdata();
    a.addition(b);
    b = c.addition();
    c.addition();
    cout<<"\n Object a :  ";
    a.putdata();
    cout<<"\n Object b :  ";
    b.putdata();
    cout<<"\n Object c.. ";
    c.putdata();
}

```

#### Output of the above program

Constructor without parameters..

Parameterized constructor...

Copy Constructor ... Enter data ... 2      3

Copy Constructor ...

Parameterized constructor...

Parameterized constructor...

Object a :

The numbers are..2      3

The sum of the numbers are.. 35

Object b :

The numbers are..0      1494

The sum of the numbers are.. 0

Object c..

The numbers are..10      20

**The sum of the numbers are.. 41**

Have you noticed as to how many times copy constructor is executed?

Copy constructor is for the following statements of Program – 17.3.

add c (b);// object b is passed as argument

a.addition(b);// object b is passed as argument with the member function addition

b=c.addition();// the member function addition is returning an object.

In the above example the function addition is also overloaded. So, primarily functions declared anywhere within the program can be overloaded.

### 17.5 Rules for constructor definition and usage

- 1) The name of the constructor must be same as that of the class
- 2) A constructor can have parameter list
- 3) The constructor function can be overloaded
- 4) The compiler generates a constructor, in the absence of a user defined constructor
- 5) The constructor is executed automatically

### 17.6 Destructors

A destructor is a function that removes the memory of an object which was allocated by the constructor at the time of creating a object. It carries the same name as the class tag, but with a tilde ( ~ ) as prefix.

**Example :**

```
class simple
{
    _____
    _____
    public :
    ~simple()
    {
        .....
    }
}
```

### 17.7 Rules for destructor definition and usage

- 1) The destructor has the same name as that of the class prefixed by the tilde character '~'.
- 2) The destructor cannot have arguments
- 3) It has no return type
- 4) Destructors cannot be overloaded i.e., there can be only one destructor in a class
- 5) In the absence of user defined destructor, it is generated by the compiler
- 6) The destructor is executed automatically when the control reaches the end of class scope

## EXERCISES

I. Complete the following table

|                                  | Constructor | Destructor |
|----------------------------------|-------------|------------|
| 1. Should be declared under      | - scope     | - scope    |
| 2. overloading is                | -           | -          |
| 3. Is executed when an object is |             |            |
| The function of a                |             |            |

II. Why do the following snippets throw error ?

```
Class simple
{
    private :
    int x;
    simple()
    { x = 5; }
};
```

```
Class simple
{
    private :
    int x;
    public :
    simple(int y)
    { x = y; }
};
void main()
{
    simple s;
}
```

```
Class simple
{
    private :
    int x;
    public :
    simple(int y)
    { x = y; }

    simple(int z = 5)
    {
        x = z;
    }
};
void main()
{
    simple s(6);
}
```

## CHAPTER 18

# INHERITANCE

### 18.1 Introduction

Inheritance is the most powerful feature of an object oriented programming language. It is a process of creating new classes called **derived classes**, from the existing or **base classes**. The derived class inherits all the properties of the base class. It is a power packed class, as it can add additional attributes and methods and thus enhance its functionality. We are familiar with the term inheritance in real life (children acquire the features of their parents in addition to their own unique features). Similarly a class inherits properties from its base (parent ) class.

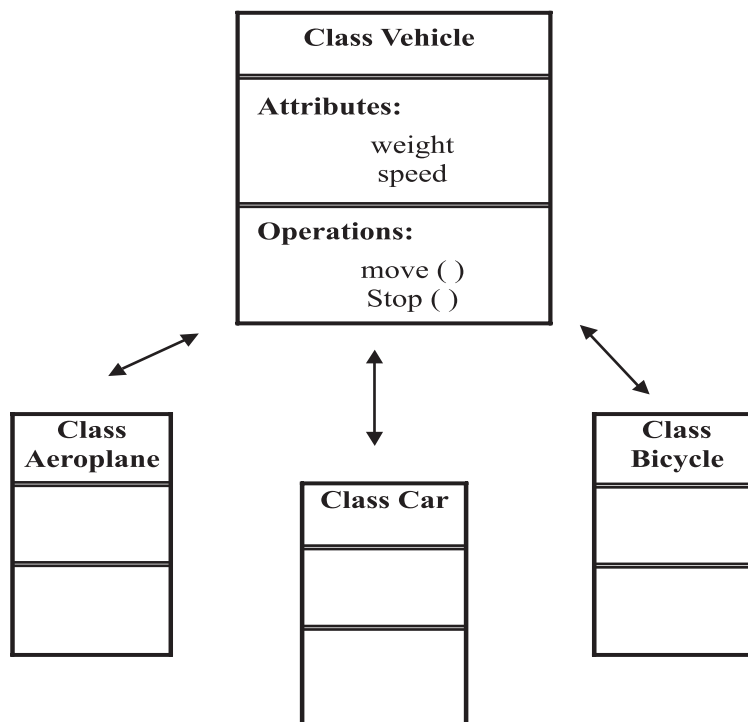


Fig.18.1 Inheritance

### 18.2 Advantages of inheritance

Inheritance has the following basic advantages.

- 1) **Reusability of code** : Many applications are developed in an organization. Code developed for one application can be reused in another application if such functionality is required. This saves a lot of development time.
- 2) **Code sharing** : The methods of the base class can be shared by the derived class.

- 3) **Consistency of interface:** The inherited attributes and methods provide a similar interface to the calling methods. In the Fig. 18.1 the attributes and methods of the class vehicle are common to the three derived classes Aeroplane, Car and Bicycle. These three derived classes are said to be having a consistence interface.

### 18.3 Derived Class and Base class

A base class is a class from which other classes are derived. A derived class can inherit members of a base class.

While defining a derived class, the following points should be observed

- a. The keyword **class** has to be used
- b. The name of the derived class is to be given after the keyword class
- c. A single colon
- d. The type of derivation, namely **private, public or protected**
- e. The name of the base class or parent class
- f. The remainder of the derived class definition

```
// Program - 18.1
#include< iostream.h>
#include<conio.h>

class add
{
    int sum;
    protected:
        int num1, num2;
    public:
        add()
        {
            num1= num2= sum=0;
            cout<<"\n Add constructor .. ";
        }
        accept()
        {
            cout<<"\n Enter two numbers .. ";
            cin>>num1>>num2;
        }
}
```

```

plus()
{
    sum = num1 + num2;
    cout<<"\n The sum of two numbers is .. "<< sum;
}
};
class subtract :public add
{
    int sub;
    public:
    subtract()
    {
        sub = 0;
        cout<<"\n Subtract constructor .. ";
    }
    minus()
    {
        add::accept();
        sub= num1-num2;
        cout<<"\n The difference of two numbers are ... "
            << sub;
    }
};

```

```

void main()
{
    subtract s;
    int choice = 0;
    cout<<"\n Enter your choice ";
    cout<<"\n1. Add..\n2. Subtract ..";
    cin>>choice;
    switch( choice )
    {
        case 1:
            s.accept();
            s.plus();
            break;
        case 2:
            s.minus();
            break;
    }
}

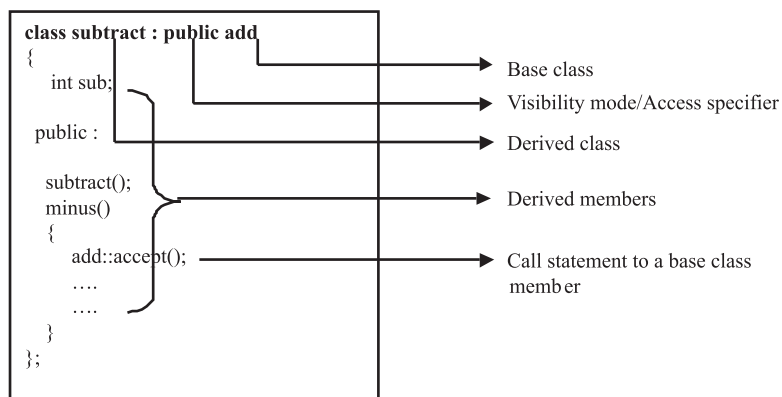
```

In the Program – 18.1 the base class is add and the derived class is subtract. The derived class should be indicated as

## class der\_name : visibility mode base class-id

```
{  
  data members of the derived_class  
  functions members of derived_class  
}
```

In the Program - 18.1 the derived class, subtract is defined as



## 18.4 Visibility Mode/Accessibility specifier

An important feature in Inheritance is to know as to when a member of a base class can be used by the objects or the members of the derived class. This is called as **accessibility**. The three access specifiers are private, protected and public. Access specifier is also referred to as visibility mode. The default visibility mode is private. The following Table 18.1 explains the scope and accessibility of the base members in the derived.

| Base Class members | Derived Class                                   |                                                       |                                                            |
|--------------------|-------------------------------------------------|-------------------------------------------------------|------------------------------------------------------------|
|                    | Private                                         | Protected                                             | Public                                                     |
| Private members    | Are not inherited but they continue to exist    |                                                       |                                                            |
| Protected members  | Inherits protected members as private members   | Inherits protected and public as protected of derived | Protected members are retained as protected of the derived |
| Public members     | Are inherited as private members of the derived | Inherits as protected members of the derived          | Inherits public members as public of derived               |

**Table 18.1 Scope and Access of Base Members in the Derived Class**



When a base class is inherited with private visibility mode the public and protected members of the base class become 'private' members of the derived class

When a base class is inherited with protected visibility mode the protected and public members of the base class become 'protected members' of the derived class

When a base class is inherited with public visibility mode , the protected members of the base class will be inherited as protected members of the derived class and the public members of the base class will be inherited as public members of the derived class.

When classes are inherited publicly, protectedly or privately the private members of the base class are not inherited they are only visible i.e continue to exist in derived classes, and cannot be accessed

The declaration of classes add and subtract of Program-18.1 is as follows

```
class add
{
    private:
        int sum;
    protected :
        int num1, num2;
    public:
        add();
        accept();
        plus();
};
class subtract : private add
{
    int sub;
    public:
        subtract();
        minus();
};
```

The data members and member functions inherited by subtract are:

- int num1 & num2     -     with status as private of subtract
- accept(); plus();     -     with status as private

|                                                                                                                      |                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <p>I</p> <pre>class subtract : private add {     int sub;     public:         subtract( );         minus(); };</pre> | <p>II</p> <pre>class subtract : protected add {     int sub;     public:         subtract();         minus(); };</pre> |
| <p>III</p> <pre>class subtract : public add {     int sub;     public:         subtract();         minus(); };</pre> |                                                                                                                        |

Accessibility of base members is shown in Table 18.2.

The constructors of the base class are not inherited, but are executed first when an instance of the derived class is created.

| class subtract                                                                                                   |                                                                                                                                 |                                                                                                                               |
|------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| I                                                                                                                | II                                                                                                                              | III                                                                                                                           |
| private:                                                                                                         | protected                                                                                                                       | public                                                                                                                        |
| sub<br>num1<br>num2<br>accept()<br>plus()<br>public:<br>subtract()<br>minus();<br>private mode of<br>inheritance | sub<br>protected:<br>num1, num2<br>accept();<br>plus();<br>public:<br>subtract()<br>minus()<br>protected mode of<br>inheritance | sub<br>protected:<br>num1, num2<br>public:<br>accept();<br>plus();<br>subtract()<br>minus();<br>public mode<br>of inheritance |

**Table 18.2 accessibility of Base Members**

Consider the objects declared in the Program – 18.1. Complete the following table based on this program.

Constructors executed are , \_\_\_\_\_,

| The objects of classes | DATA MEMBERS | METHODS/ FUNCTIONS |
|------------------------|--------------|--------------------|
| add                    |              |                    |
| subtract               |              |                    |

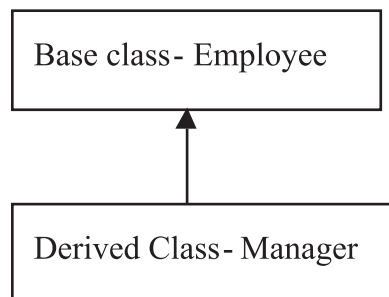
**Table 18.3 Complete this Table**

## 18.5 Types of inheritance

Classes can be derived from classes that are themselves derived. There are different types of inheritance viz., Single Inheritance, Multiple inheritance, Multilevel inheritance, hybrid inheritance and hierarchical inheritance.

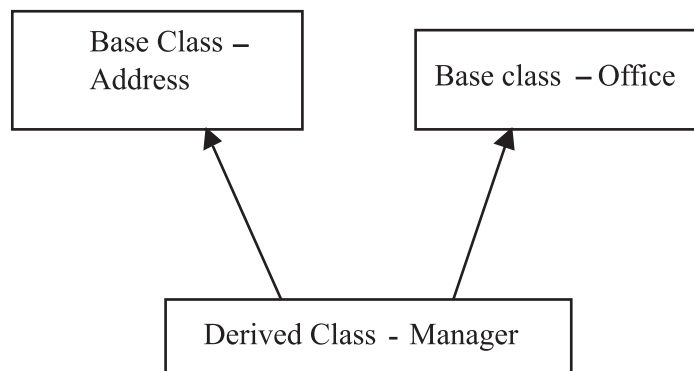
### 1) Single Inheritance

When a derived class inherits only from one base class, it is known as single inheritance



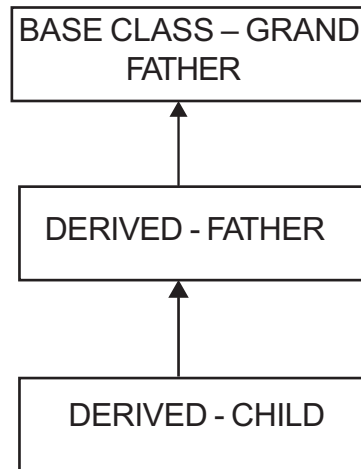
### 2) Multiple Inheritance

When a derived class inherits from multiple base classes it is known as multiple inheritance



### 3) Multilevel Inheritance

The transitive nature of inheritance is reflected by this form of inheritance. When a class is derived from a class which is a derived class itself – then this is referred to as multilevel inheritance.



What will be the output of Program 18.2?

```
// Program - 18.2
#include<iostream.h>
#include<conio.h>
class base
{
    public:
    base()
    {
        cout<<"\nConstructor of base class...";
    }
    ~base()
    {
        cout<<"\nDestructor of base class.... ";
    }
};
class derived:public base
{
    public :
    derived()
    {
        cout << "\nConstructor of derived ...";
    }
    ~derived()
    {
        cout << "\nDestructor of derived ...";
    }
};
```

```

class derived2:public derived
{
    public :
    derived2()
    {
        cout << "\nConstructor of derived2 ...";
    }
    ~derived2()
    {
        cout << "\nDestructor of derived2 ...";
    }
};
void main()
{
    derived2 x;
}

```

## OUTPUT

Constructor of base class...

Constructor of derived ....

Constructor of derived2 ...

Destructor of derived2...

Destructor of derived

Destructor of base class ..

✓ The constructors are executed in the order of inherited class i.e., from base constructor to derived. The destructors are executed in the reverse order

✓ Classes used only for deriving other classes are called as Abstract Classes ie., to say that objects for these classes are not declared.

## EXERCISES

1) Given the following set of definitions

```

class node
{
    int x;
    void init();
    public:

```

```

        void read();
protected:
        void get();
};
class type : public node
{
        int a;
public:
        void func1();
protected:
        int b;
        void getb();
}
class statement :private type
{
        int p;
public:
        void func2();
protected:
        void func3();
};

```

Complete the following table

| Members of the<br>class type       | Accessibility of members /their classes |           |        |
|------------------------------------|-----------------------------------------|-----------|--------|
|                                    | private                                 | protected | public |
| Members inherited<br>by class type |                                         |           |        |
| Defined in class<br>type           |                                         |           |        |

**Table- 4 class node ....**

| Members of the<br>class statement       | Accessibility of members / their classes |           |        |
|-----------------------------------------|------------------------------------------|-----------|--------|
|                                         | private                                  | protected | public |
| Members inherited<br>by class statement |                                          |           |        |
| Defined in class<br>statement           |                                          |           |        |

**Table- 5 class statement**

| Objects         | Can access members |                  |
|-----------------|--------------------|------------------|
|                 | Data members       | Member functions |
| class type      |                    |                  |
| class statement |                    |                  |

**Table- 6 Objects...**

2) Find errors in the following program. State reasons

```
#include<iostream.h>
class A
{
    private :
        int a1;
    public:
        int a2;
    protected:
        int a3;
};
class B : public A
{
    public:
    void func()
    {
        int b1, b2 , b3;
        b1 = a1;
        b2 = a2;
        b3 = a3;
    }
};
void main()
{
    B der;
    der.a3 = 0';
    der.func();
}
```

3) Consider the following declarations and answer the questions given below

```
class vehicle
{
    int wheels;
    public:
        void inputdata( int, int);
        void outputdata();
    protected :
        int passenger;
};
```

```

class heavy_vehicle : protected vehicle
{
    int diesel_petrol;
    protected:
        int load;
    public:
        void readdata( int, int);
        void writedata();
};
class bus: private heavy_vehicle
{
    char marks[20];
    public:
        void fetchdata( char );
        void displaydata();
};

```

- Name the base class and derived class of the class heavy\_vehicle
- Name the data members that can be accessed from the function displaydata()
- Name the data members that can be accessed by an object of bus class
- Is the member function output data accessible to the objects of heavy\_vehicle class

4) What will be the output of the following program

```

#include<iostream.h>
#include<conio.h>
class student
{
    int m1, m2, total;
    public:

        student ( int a, int b)
        {
            m1 = a;
            m2 = b;
            cout<<"\n parameterized constructors..";
        }
};

```



## CHAPTER 19

### IMPACT OF COMPUTERS ON SOCIETY

#### 19.1 Introduction



**“India lives in her seven hundred thousand villages”**

**- Mohandas Karamchand Gandhi**

To reach out the benefits of IT to the common man we need atleast three technical elements :

- Connectivity [Computer networks and Internet facility]
- Affordable computers or other similar devices
- Software

It is interesting to observe that 85% of computer usage is “Word Processing”. Computers can do many more things for the common man than this. Quality IT education will enable the common man to put computers to a better use. This chapter presents the possible ways in which computers can be used to develop the society.

#### 19.2 Computers for Personal Use

Personal computers have totally changed the way we work, live and think. Word Processing, Databases, Spreadsheets and Multimedia presentation packages have increased the efficiency at work. There are many packages that are being used. Desktop Publishing and other impressive packages for graphics are adding value to the work done. Paint, games and a large set of similar packages are providing facilities for people of all age groups to use the computer. Browsing, e-mail and chat have changed our life style.

Today computers come in different sizes and shapes. Some adaptation of the basic computer model is making it more useful in the homes of the user.

### 19.3 Computerized Homes

| Home        | Products and a brief Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Living Room | <ul style="list-style-type: none"> <li>• <b>LCD Screen, Camera and Speakers:</b><br/>Mounted on the Wall to provide better effect and save floor space.</li> <li>• <b>Archive Unit:</b> Enables data storage and management</li> <li>• <b>Emotion Containers:</b> They are small compartments with a screen, speaker and a scent to derive emotional comfort. This can prevent people from acquiring bad habits</li> <li>• <b>Personal Archives:</b> They store personal details like family photographs and personal treasures. In addition it enables connectivity to other people.</li> <li>• <b>Picture Phone and Pad:</b> Picture based personal telephone directory.</li> </ul> |
| Kids Room   | <ul style="list-style-type: none"> <li>• Devices that provide listening access to audio sources in home such as radio, television</li> <li>• Devices with kara-oke features allowing one to sing along with the audio coming from the original source</li> <li>• Robots that function as Electronic Pets</li> <li>• Devices that enable game playing over the net. In addition real world characters are translated into the computer world and a kid can play with them</li> </ul>                                                                                                                                                                                                   |
| Home Office | <ul style="list-style-type: none"> <li>• Packages to make animated stories</li> <li>• <b>Memo Frame:</b> Easy interaction with other people through touch screen, scanner and microphone facilities.</li> <li>• <b>Bookshelf:</b> To manage and study electronic books</li> <li>• <b>Personal Creativity Tool:</b> To draw, capture and work with multimedia elements</li> <li>• Advanced data accessing systems</li> </ul>                                                                                                                                                                                                                                                           |

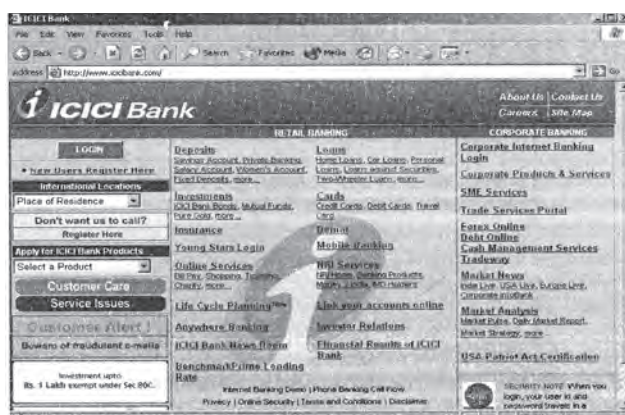
|             |                                                                                                                                                                                                                                                                                                         |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Bed Room    | <ul style="list-style-type: none"> <li>• <b>Touch and Voice Control for various appliances:</b> Display Monitors, Special Headphones and Moving Telephone·</li> <li>• <b>Projection TV:</b> Projects the TV pictures on the ceiling or walls</li> <li>• Alarm Clock</li> </ul>                          |
| Bath Room   | <ul style="list-style-type: none"> <li>• Mirrors, Medical Box and Special Speakers</li> </ul>                                                                                                                                                                                                           |
| Kitchen     | <ul style="list-style-type: none"> <li>• Speakers, Rack Telephone, Intelligent Aprons, Kettle, Toaster, Food Analyzer, Health Monitor, Devices to preserve food</li> </ul>                                                                                                                              |
| Dining Room | <ul style="list-style-type: none"> <li>• Interactive Tablecloth to keep the food sufficiently warm</li> <li>• Ceramic Audio player and speakers·</li> <li>• Communication facilities around the dinner table</li> <li>• Interactive screens to consult with the cook and other kitchen staff</li> </ul> |

**Table 19.1 Computerized Products for Home**

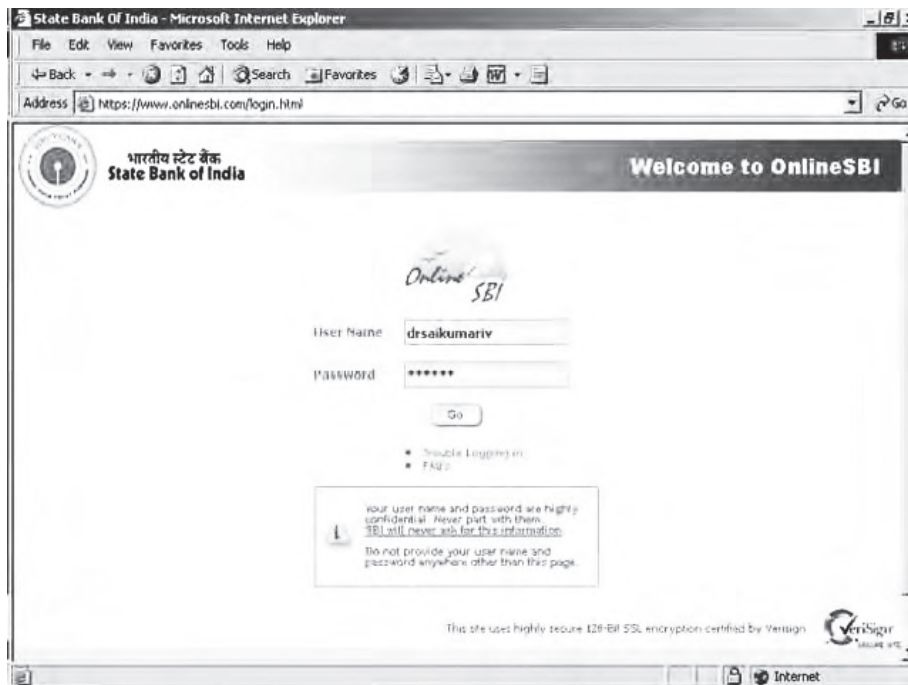
## 19.4 Home Banking and Shopping

Traditional banking needed the user to go to the bank to perform related activities. These activities include depositing or withdrawing money from the account or securing loans. Banks gradually began providing many other services including term deposits, agricultural loans, paying bills related to other services such as telephones, electricity and locker facilities. As the confidence of the common man in banking improved, banks became a key component in the national economy.

This also means long queues at the banks during working hours. Introduction of IT in banks reduced the time required to provide service to a user. Long queues were being handled quickly.



**Fig. 19.1 ICICI Bank**



**Fig. 19.2 State Bank of India**

Advanced machines like ATM enable withdrawal of money from the accounts in a particular bank anytime and anywhere. This helps the user in emergency situations where money is needed during the nights and holidays. However, the user has to go to the nearest ATM facility. It is possible that every branch of any well recognized bank will have a ATM facility soon.

e-Banking permits banking from the comfort of the home by using internet facilities. It has truly improved the reach and services of banks.

Computers are used in many areas even for Shopping. You can purchase any product, any brand, any quantity from anywhere through e-shopping. You need not go to the shop. The pictures and other details of what can be purchased are available on the website of the shop. You have to select and order. Advance payment for these items is assured by various methods. Credit cards and prior registration with the shop are the popular methods. The items purchased will be delivered at your home.

## **19.5 Computers in Education**

Computers are used in many areas of Education including:

- Buying and browsing the latest edition of books by both local & foreign authors
- Educational CDROMs

- Computer Based Tutorials (CBT).
- Spreading information about schools, colleges, universities and courses offered, admission procedures, accommodation facilities, scholarships, educational loans, career guidance.
- e-Learning that enables online educational programs leading to degrees and certifications.

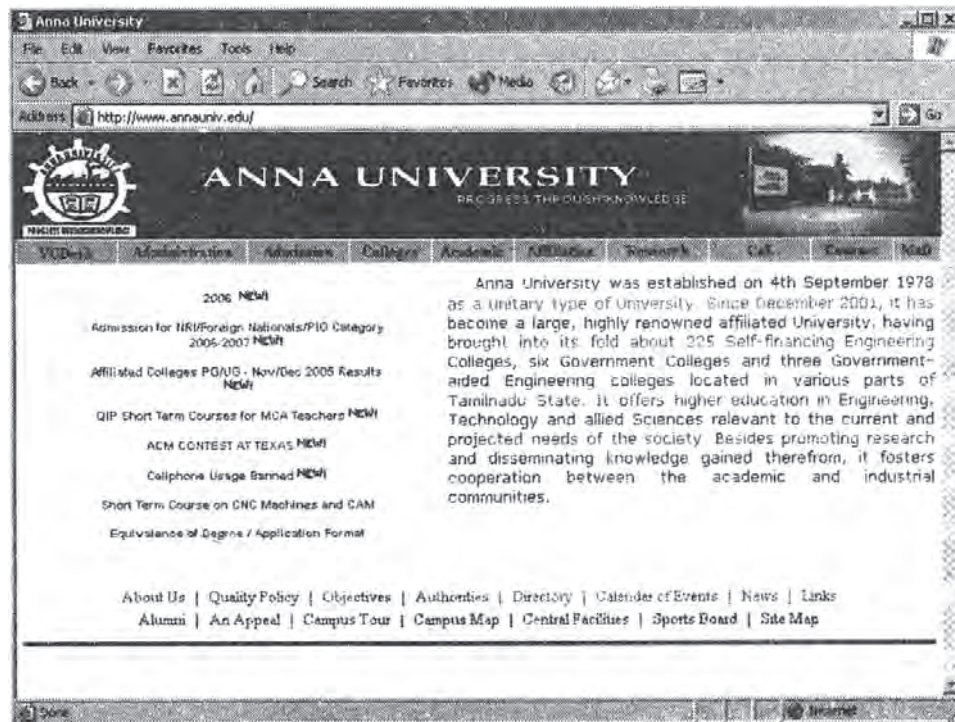


Fig. 19.3 Anna University

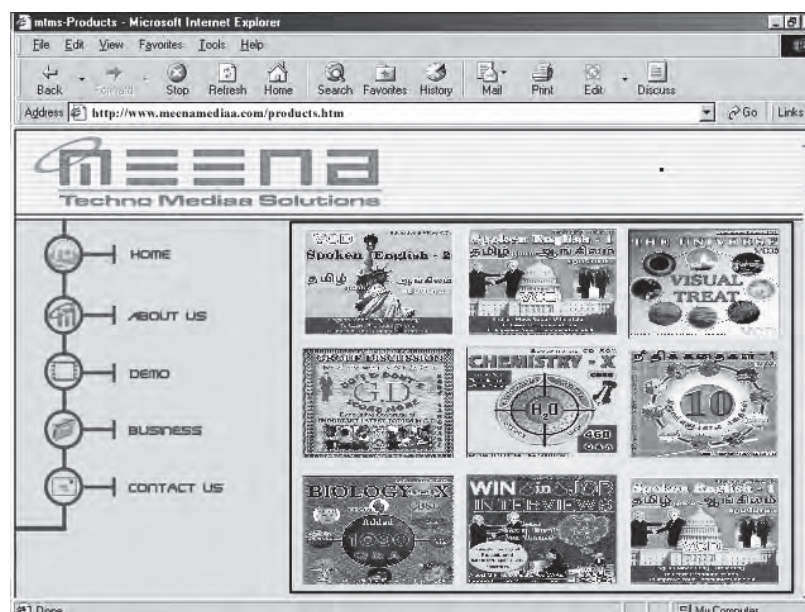


Fig. 19.4 Computer Based Tutorials





**Fig. 19.5 Information on Education around the Globe**

## 19.6 Computers in Entertainment

Computers contribute to entertainment also. You can update your knowledge in fine - arts like painting, music, dance, yoga, games, science, nature, latest news and events. You can know more about various places of worship and places of interest for tourists.



**Fig. 19.6 Computers in Entertainment**

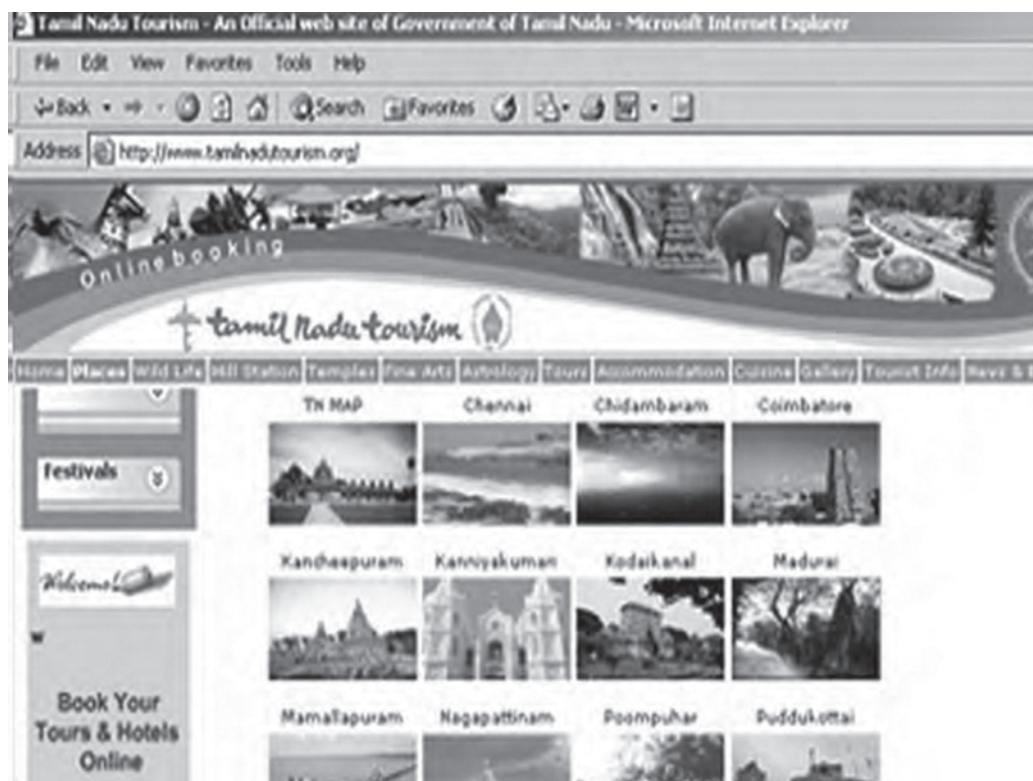


Fig. 19.7 Computers in Tourism (Tamil Nadu)

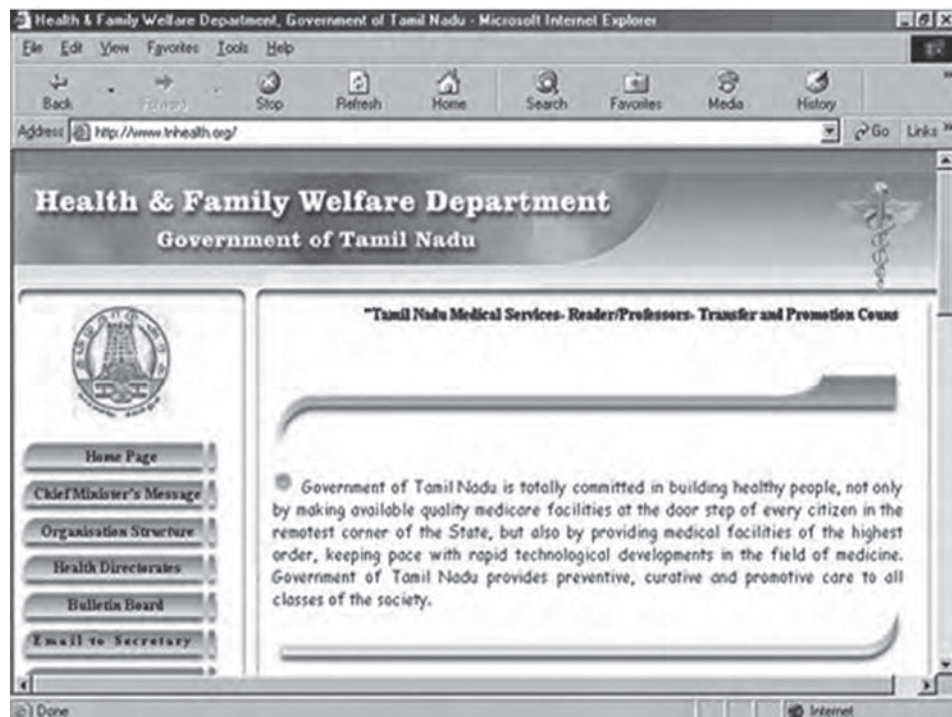


Fig. 19.8 Computers in Tourism (India)

## 19.7 Computers in Healthcare

Healthcare is dominated by large amounts of data and limited financial and human resources and need for accountability of those resources

Healthcare has improved significantly ever since computers found their way into the hands of doctors and health administrators.



**Fig. 19.9 Computers in Health and Family Welfare**

Computers are used in many areas of healthcare including

- Hospital Management System
- Patient Tracking System
- Exchange of diagnostic records between healthcare units
- Tracking and Monitoring Communicable Diseases
- Decision support systems with highly advanced computing techniques

Today many doctors are innovating to suit their needs. It is indeed a good sign for the patients. Tele-medicine is built largely on the foundational systems mentioned above. Internet facilitates remote diagnostics. This ensures expert advice at places where it is not there.



## 19.8 Computers in Agriculture

Farming and agriculture might seem like low technology enterprises, but these industries have benefited from computerization more than the casual observer might think. Farmers, both professional and hobbyists benefit from online resources such as seed estimators and pest information sites. Professional farmers can use revenue estimators to help them plan which crops will produce the highest profits based on weather patterns, soil types, and current market values.



**Fig. 19.10 Computers in Agriculture**

**Some of the areas where software has been developed are:**

- Agricultural Finances and Accounting
- Alternative farming techniques
- Animal Husbandry
- Buildings and Irrigation
- Chat with other agriculturists and scientists
- Farmland Assessment
- Fertilizer Analysis
- Finding links to farm resources, chat boards, classified advertisements, and other farm-related information
- Gardening

- Improving Cow Herds and Increasing revenues
- Land Management
- Livestock
- Milk production
- Use of satellite imagery to decide on the crops

### 19.9 Internet in real time Applications

All applications mentioned above happen in real time and over the net. You can reserve or book air and train tickets from your own place and at your own pace through computers.



**Fig.19.11 Computers in Realtime Applications**

### EXERCISES

This Chapter has the support of multimedia content to understand more about the applications presented. You must see this content and where possible visit the websites indicated.

This multimedia content is provided to your school on a separate CD. Please contact your teacher to get this CD.

## **CHAPTER 20**

### **IT ENABLED SERVICES**

#### **20.1 Introduction**

Information Technology that helps in improving the quality of service to the users is called IT Enabled Services [ITES]. IT Enabled Services are human intensive services that are delivered over telecommunication networks or the Internet to a range of business segments. ITES greatly increases the employment opportunities.

Is typing a letter using the computer an ITES? The answer is No. However, a facility that allows the user to speak into a special device called 'Dictaphone' and then convert the speech into a letter is an ITES.

Word processors, Spreadsheets and Databases have ensured that many traditional services are IT Enabled. However, the user is expected to learn several aspects of these IT tools before gaining from their use. ITES adds value to these services by reducing the learning that needs to be done by the users. ITES thus has the potential to take the power of IT to users who do not know IT.

ITES can improve the quality of the service either directly or indirectly. Improved customer satisfaction, better look and feel and an improved database are some direct benefits. Indirect benefits are seen after sometime. Data collected for one purpose may be useful for some other purpose also after some time.

Some of the IT enabled services presented in this chapter are:

- e-Governance
- Call Centers
- Data Management
- Medical [Telemedicine and Transcription].
- Data Digitization
- Website Services

ITES such as Business Process Outsourcing (BPO), Digital Content Development / Animation, Human Resources Services and Remote Maintenance are other important areas.

ITES requires practical IT skills especially in the area of Databases, Internet and good communication skills in English. A formal training in Soft Skills to understand the basic aspects of Industry Culture, professionalism and etiquette is needed for the effective implementation of ITES.

## 20.2 e-Governance

Computers help you to look at the government websites and the services provided by them. The various websites provided by the government give the details about the departments, specific functions, special schemes, documents, contacts, links, IAS intranet, site map, search, what's new, press releases, feedback. These websites are both in English and Tamil.

தமிழ்நாடு அரசு  
22/4/2002  
பக்கம் 05 : 1 of 1

தலைநகரம் : சென்னை  
பகுதி : 1  
பகுதி : 1

பகுதி : 1  
பகுதி : 1

| பகுதி<br>பெயர் | பகுதி          |                | பகுதி          |                | பகுதி          |                |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
|                | பகுதி<br>பெயர் | பகுதி<br>பெயர் | பகுதி<br>பெயர் | பகுதி<br>பெயர் | பகுதி<br>பெயர் | பகுதி<br>பெயர் |
| 1              | 1              | 1.1            | 1.1            | 1.1            | 1.1            | 1.1            |
| 2              | 2              | 2.1            | 2.1            | 2.1            | 2.1            | 2.1            |
| 3              | 3              | 3.1            | 3.1            | 3.1            | 3.1            | 3.1            |
| 4              | 4              | 4.1            | 4.1            | 4.1            | 4.1            | 4.1            |
| 5              | 5              | 5.1            | 5.1            | 5.1            | 5.1            | 5.1            |
| 6              | 6              | 6.1            | 6.1            | 6.1            | 6.1            | 6.1            |
| 7              | 7              | 7.1            | 7.1            | 7.1            | 7.1            | 7.1            |
| 8              | 8              | 8.1            | 8.1            | 8.1            | 8.1            | 8.1            |
| 9              | 9              | 9.1            | 9.1            | 9.1            | 9.1            | 9.1            |
| 10             | 10             | 10.1           | 10.1           | 10.1           | 10.1           | 10.1           |
| 11             | 11             | 11.1           | 11.1           | 11.1           | 11.1           | 11.1           |
| 12             | 12             | 12.1           | 12.1           | 12.1           | 12.1           | 12.1           |
| 13             | 13             | 13.1           | 13.1           | 13.1           | 13.1           | 13.1           |
| 14             | 14             | 14.1           | 14.1           | 14.1           | 14.1           | 14.1           |
| 15             | 15             | 15.1           | 15.1           | 15.1           | 15.1           | 15.1           |
| 16             | 16             | 16.1           | 16.1           | 16.1           | 16.1           | 16.1           |
| 17             | 17             | 17.1           | 17.1           | 17.1           | 17.1           | 17.1           |
| 18             | 18             | 18.1           | 18.1           | 18.1           | 18.1           | 18.1           |
| 19             | 19             | 19.1           | 19.1           | 19.1           | 19.1           | 19.1           |
| 20             | 20             | 20.1           | 20.1           | 20.1           | 20.1           | 20.1           |
| 21             | 21             | 21.1           | 21.1           | 21.1           | 21.1           | 21.1           |
| 22             | 22             | 22.1           | 22.1           | 22.1           | 22.1           | 22.1           |
| 23             | 23             | 23.1           | 23.1           | 23.1           | 23.1           | 23.1           |
| 24             | 24             | 24.1           | 24.1           | 24.1           | 24.1           | 24.1           |
| 25             | 25             | 25.1           | 25.1           | 25.1           | 25.1           | 25.1           |
| 26             | 26             | 26.1           | 26.1           | 26.1           | 26.1           | 26.1           |
| 27             | 27             | 27.1           | 27.1           | 27.1           | 27.1           | 27.1           |
| 28             | 28             | 28.1           | 28.1           | 28.1           | 28.1           | 28.1           |
| 29             | 29             | 29.1           | 29.1           | 29.1           | 29.1           | 29.1           |
| 30             | 30             | 30.1           | 30.1           | 30.1           | 30.1           | 30.1           |
| 31             | 31             | 31.1           | 31.1           | 31.1           | 31.1           | 31.1           |
| 32             | 32             | 32.1           | 32.1           | 32.1           | 32.1           | 32.1           |
| 33             | 33             | 33.1           | 33.1           | 33.1           | 33.1           | 33.1           |
| 34             | 34             | 34.1           | 34.1           | 34.1           | 34.1           | 34.1           |
| 35             | 35             | 35.1           | 35.1           | 35.1           | 35.1           | 35.1           |
| 36             | 36             | 36.1           | 36.1           | 36.1           | 36.1           | 36.1           |
| 37             | 37             | 37.1           | 37.1           | 37.1           | 37.1           | 37.1           |
| 38             | 38             | 38.1           | 38.1           | 38.1           | 38.1           | 38.1           |
| 39             | 39             | 39.1           | 39.1           | 39.1           | 39.1           | 39.1           |
| 40             | 40             | 40.1           | 40.1           | 40.1           | 40.1           | 40.1           |
| 41             | 41             | 41.1           | 41.1           | 41.1           | 41.1           | 41.1           |
| 42             | 42             | 42.1           | 42.1           | 42.1           | 42.1           | 42.1           |
| 43             | 43             | 43.1           | 43.1           | 43.1           | 43.1           | 43.1           |
| 44             | 44             | 44.1           | 44.1           | 44.1           | 44.1           | 44.1           |
| 45             | 45             | 45.1           | 45.1           | 45.1           | 45.1           | 45.1           |
| 46             | 46             | 46.1           | 46.1           | 46.1           | 46.1           | 46.1           |
| 47             | 47             | 47.1           | 47.1           | 47.1           | 47.1           | 47.1           |
| 48             | 48             | 48.1           | 48.1           | 48.1           | 48.1           | 48.1           |
| 49             | 49             | 49.1           | 49.1           | 49.1           | 49.1           | 49.1           |
| 50             | 50             | 50.1           | 50.1           | 50.1           | 50.1           | 50.1           |
| 51             | 51             | 51.1           | 51.1           | 51.1           | 51.1           | 51.1           |
| 52             | 52             | 52.1           | 52.1           | 52.1           | 52.1           | 52.1           |
| 53             | 53             | 53.1           | 53.1           | 53.1           | 53.1           | 53.1           |
| 54             | 54             | 54.1           | 54.1           | 54.1           | 54.1           | 54.1           |
| 55             | 55             | 55.1           | 55.1           | 55.1           | 55.1           | 55.1           |
| 56             | 56             | 56.1           | 56.1           | 56.1           | 56.1           | 56.1           |
| 57             | 57             | 57.1           | 57.1           | 57.1           | 57.1           | 57.1           |
| 58             | 58             | 58.1           | 58.1           | 58.1           | 58.1           | 58.1           |
| 59             | 59             | 59.1           | 59.1           | 59.1           | 59.1           | 59.1           |
| 60             | 60             | 60.1           | 60.1           | 60.1           | 60.1           | 60.1           |
| 61             | 61             | 61.1           | 61.1           | 61.1           | 61.1           | 61.1           |
| 62             | 62             | 62.1           | 62.1           | 62.1           | 62.1           | 62.1           |
| 63             | 63             | 63.1           | 63.1           | 63.1           | 63.1           | 63.1           |
| 64             | 64             | 64.1           | 64.1           | 64.1           | 64.1           | 64.1           |
| 65             | 65             | 65.1           | 65.1           | 65.1           | 65.1           | 65.1           |
| 66             | 66             | 66.1           | 66.1           | 66.1           | 66.1           | 66.1           |
| 67             | 67             | 67.1           | 67.1           | 67.1           | 67.1           | 67.1           |
| 68             | 68             | 68.1           | 68.1           | 68.1           | 68.1           | 68.1           |
| 69             | 69             | 69.1           | 69.1           | 69.1           | 69.1           | 69.1           |
| 70             | 70             | 70.1           | 70.1           | 70.1           | 70.1           | 70.1           |
| 71             | 71             | 71.1           | 71.1           | 71.1           | 71.1           | 71.1           |
| 72             | 72             | 72.1           | 72.1           | 72.1           | 72.1           | 72.1           |
| 73             | 73             | 73.1           | 73.1           | 73.1           | 73.1           | 73.1           |
| 74             | 74             | 74.1           | 74.1           | 74.1           | 74.1           | 74.1           |
| 75             | 75             | 75.1           | 75.1           | 75.1           | 75.1           | 75.1           |
| 76             | 76             | 76.1           | 76.1           | 76.1           | 76.1           | 76.1           |
| 77             | 77             | 77.1           | 77.1           | 77.1           | 77.1           | 77.1           |
| 78             | 78             | 78.1           | 78.1           | 78.1           | 78.1           | 78.1           |
| 79             | 79             | 79.1           | 79.1           | 79.1           | 79.1           | 79.1           |
| 80             | 80             | 80.1           | 80.1           | 80.1           | 80.1           | 80.1           |
| 81             | 81             | 81.1           | 81.1           | 81.1           | 81.1           | 81.1           |
| 82             | 82             | 82.1           | 82.1           | 82.1           | 82.1           | 82.1           |
| 83             | 83             | 83.1           | 83.1           | 83.1           | 83.1           | 83.1           |
| 84             | 84             | 84.1           | 84.1           | 84.1           | 84.1           | 84.1           |
| 85             | 85             | 85.1           | 85.1           | 85.1           | 85.1           | 85.1           |
| 86             | 86             | 86.1           | 86.1           | 86.1           | 86.1           | 86.1           |
| 87             | 87             | 87.1           | 87.1           | 87.1           | 87.1           | 87.1           |
| 88             | 88             | 88.1           | 88.1           | 88.1           | 88.1           | 88.1           |
| 89             | 89             | 89.1           | 89.1           | 89.1           | 89.1           | 89.1           |
| 90             | 90             | 90.1           | 90.1           | 90.1           | 90.1           | 90.1           |
| 91             | 91             | 91.1           | 91.1           | 91.1           | 91.1           | 91.1           |
| 92             | 92             | 92.1           | 92.1           | 92.1           | 92.1           | 92.1           |
| 93             | 93             | 93.1           | 93.1           | 93.1           | 93.1           | 93.1           |
| 94             | 94             | 94.1           | 94.1           | 94.1           | 94.1           | 94.1           |
| 95             | 95             | 95.1           | 95.1           | 95.1           | 95.1           | 95.1           |
| 96             | 96             | 96.1           | 96.1           | 96.1           | 96.1           | 96.1           |
| 97             | 97             | 97.1           | 97.1           | 97.1           | 97.1           | 97.1           |
| 98             | 98             | 98.1           | 98.1           | 98.1           | 98.1           | 98.1           |
| 99             | 99             | 99.1           | 99.1           | 99.1           | 99.1           | 99.1           |
| 100            | 100            | 100.1          | 100.1          | 100.1          | 100.1          | 100.1          |

Land Ownership Certificate  
(Chitta Extract)

Fig.20.1 Getting Land Certificate using Internet

Accountant General (A&E), Tamil Nadu - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites Media Links Norton AntiVirus

Address <http://www.agae.tn.nic.in/gpf/> Go

Office of the  
Accountant General (A & E), Tamil Nadu  
Chennai

Welcome to Online GPF of Office of the Accountant General (A&E) - Tamil Nadu

Click here to view the details of services offered to the subscribers through this site

**Subscriber Login**

GPF No.  / Suffix

Date of Birth

\* Date of Birth must have 10 Characters in the format dd/mm/yyyy eg. 18/06/1953

Login

Site designed and hosted by  
National Information Centre,  
Chennai

Contents owned & maintained by  
Office of the Accountant General  
(Accounts & Entitlements), Tamil Nadu  
361 Anna Salai  
Teynampet  
Chennai - 600 006

Contacts  
Phone : +91-44-2433 6395  
IVRS : +91-44-2431 4477  
Fax : +91-44-2492 0682  
Gram : ACCOUNTSCENT

Done Internet

Fig.20.2 Office of the Accountant General (A & E), Tamil Nadu

### **20.3 Call Centers**

Information Technology is happening all over the globe. Users or Customers of IT products are all over the world. The customers are in need of a facility that ensures communication services on all days of the year all round the clock – 24 X 365.

A call center is sometimes defined as a telephone based shared service center for specific customer activities and are used for number of customer-related functions like marketing, selling, information transfer, advice, technical support and so on. A call center has adequate telecom facilities, trained consultants, access to wide database, Internet and other on-line information support to provide information and support services to customers. It operates to provide round the clock and year round service i.e. 24 x 365 service.

### **20.4 Data Management**

Data Management is a category of IT Enabled Services pertaining to collection, digitization and processing of data coming from various sources. Traditional data processing services comprise punching data from manually filled forms, images or publications; preparing databases and putting them together. However, with the advent of multimedia and internet, sources have increased to include manually printed documents, images, sounds and video. Equally diverse are the new output media which include databases on servers, hard copy publications, CD-ROM records emanating from internet based queries.

Data management is the key for effective and profitable use of IT in organizations. The range of ITES in this category are:

- ASCII format for upload to your database
- Character Recognition and Processing
- Custom reports
- Data Entry
- Data entry front end edits
- Document Preparation
- Forms are imaged and transferred to CD ROM
- Handwritten, Machine Print, Mark Sense, Bar Coding (Reader Response can be captured and processed from any hard copy or faxed document).
- Image Capturing
- Image Keying
- Image Storage & Retrieval
- Outcome studies
- Statistical analysis

Some of the organizations that can potentially benefit from ITES in this category are:

- Back office Operations such as Accounts, Financial services
- Banking

- Government agencies
- Hospital
- Insurance
- Legal
- Manufacturing
- Municipalities
- Police departments
- Public utilities
- Publishing
- Telecom
- Transportation

Each of the organizations mentioned above presents a huge opportunity in ITES in the critical area of Data Management. Banking, Financial Services and Insurance sectors are popularly termed BFSI. BFSI and Pension Services are high growth areas for ITES.

Data Security and Customer Privacy are two important aspects that must be ensured by the ITES provider in this area. An ITES provider may be serving multiple organizations. The service provider must ensure the privacy aspects of every organization. Computer Ethics is critical for the success of ITES.

## **20.5 Medical Transcription and Tele-Medicine**

Medical Transcription is a permanent, legal document that formally states the result of a medical investigation. It facilitates communication and supports the insurance claims. There are three main steps involved in Medical Transcription. These include:

**Step 1:** Hospitals that want to use this form of ITES sign up with a service provider. Doctors are trained in the process. The doctor dictates into a special device or a free telephone. The sound is then stored on a server at the other end.

**Step 2:** The sound is digitized and sent to the ITES provider. This service provider is usually in a different country. Providing transcription services in countries like USA is becoming very expensive both to the patient and the hospital. So, ITES in this category reduces the cost by having it done in a country where the cost is affordable. The digitized data is converted back to sound. The trained transcriptionists listen to the dictation and transcribe. This is a formal record of the diagnosis made by the doctor.

**Step 3:** The transcribed files are sent out to quality control persons, who listen to the dictation and check the transcription. Corrections are made if required. Then the transcribed reports are transmitted back to hospital as a word document. This is valid for legal purposes and making insurance claims.



## 20.6 Data Digitization

Digitization refers to the conversion of non-digital material to digital form. A wide variety of materials as diverse as maps, manuscripts, moving images and sound may be digitized.

Digitization offers great advantages for access, allowing users to find, retrieve, study and modify the material. However, reliance on digitization as a preservation strategy could place much material at risk. Digital technologies are changing rapidly. Preservation is a long term strategy and many technologies will become obsolete soon. This instability in technology can lead to the loss of the digitized objects. This defeats the purpose of preservation. Some application areas of the digital technology are as follows:

- Annual reports and price list
- Books
- Database archiving
- Electronic Catalogues & Brochures
- Engineering and Design
- Geographical Information System.
- Movies, Sounds and High quality image preservation
- Product/Service Training Manuals
- Research Journals and Conference Papers

### **The steps in data digitization are:**

- Understanding the customer needs
- Customer needs are used as the basis for defining the objectives of digitization
- A pilot application is built
- After the pilot application is approved, the full digitization of data identified by the customer is undertaken.
- Different types of data are digitized using different techniques. Many advance software packages are available to improve the quality of the digitized form of the original document.

- The digitized data is indexed and a table of contents is produced to improve accessibility. Highly advanced and reliable storage facilities are used to stock the digitized data.

**There are many benefits of digitization. Some of the key benefits are:**

- Long term preservation of the documents.
- Storage of important documents at one place.
- Easy to use and access to the information.
- Quick and focused search of relevant information in terms of images and text.
- Easy transfer of information in terms of images and text.
- Easy transfer of information through CD-ROM, internet and other electronic media

## **20.7 Website Services**

Computers also help us in accessing website services such as:

- Agriculture Marketing Network
- Career guidance
- Employment Online
- General Provident Fund
- Results of various Examinations

In the very near future there will be many more ITES that can be utilized even from the remote corners of the world.

## **EXERCISES**

This Chapter has the support of multimedia content to understand more about the applications presented. You must see this content and where possible visit the websites indicated.

This multimedia content is provided to your school on a separate CD. Please contact your teacher to get this CD.



## CHAPTER 21

### COMPUTER ETHICS

Computer ethics has its roots in the work of Norbert Wiener during World War II. Wiener's book included

- (1) An account of the purpose of a human life
- (2) Four principles of justice
- (3) A powerful method for doing applied ethics
- (4) Discussions of the fundamental questions of computer ethics, and
- (5) Examples of key computer ethics topics.

In the mid 1960s, Donn Parker of SRI International in Menlo Park, California began to examine unethical and illegal uses of computers by computer professionals. By the 1980s a number of social and ethical consequences of information technology were becoming public issues in America and Europe: issues like computer-enabled crime, disasters caused by computer failures, invasions of privacy via computer databases, and major law suits regarding software ownership.

During 1990s many universities introduced formal course in computer ethics. Many textbooks and other reading materials were developed. It triggered new research areas and introduction of journals.

Generally speaking, ethics is the set of rules for determining moral standards or what is considered as socially acceptable behaviors. Today, many computer users are raising questions on what is and is not ethical with regard to activities involving information technology. Obviously, some general guidelines on ethics are useful responsibly in their application of information technology.

#### **General guidelines on computer ethics are needed for:**

- Protection of personal data
- Computer Crime
- Cracking

#### **21.1 Data Security**

Personal data is protected by using an appropriate combination of the following methods.

### **21.1.1 Physical Security:**

Physical security refers to the protection of hardware, facilities, magnetic disks, and other items that could be illegally accessed, stolen, damaged or destroyed. This is usually provided by restricting the people who can access the resources.

### **21.1.2 Personal Security:**

Personal security refers to software setups that permit only authorized access to the system. User Ids and passwords are common tools for such purpose. Only those with a need to know have Ids and password for access.

### **21.1.3 Personnel Security:**

Personnel security refers to protecting data and computer system against dishonesty or negligence of employees.

## **21.2 Computer Crime**

A computer crime is any illegal activity using computer software, data or access as the object, subject or instrument of the crime.

### **Common crimes include:**

- Crimes related to money transfer on the internet
- Making long distance calls illegally using computers
- Illegal access to confidential files
- Stealing hardware
- Selling or misusing personal
- Hardware and software piracy
- Virus
- Cracking
- Theft of computer time

It must be observed that 80% of all computer crimes happen from within the company. Over 60% of all crimes go unreported.

Making and using duplicate hardware and software is called piracy. We tend to pirate because:

- We like free things
- Why pay for something when we can get it for free?
- Our thinking and actions are self-serving

- If we have the opportunity to get away with something, benefit financially, and minimal risk is involved, the way in which we've been conditioned by our capitalist society to do it.

## **VIRUS:**

A virus is a self-replicating program that can cause damage to data and files stored on your computer. These are programs written by programmers with great programming skills who are motivated by the need for a challenge or to cause destruction. 57000 known virus programs are in existence. 6 new viruses are found each day.

## **Theft of Computer Time:**

Most of the computers in an organization have lot of free computer time to spare. In other words a lot of computer time is not used. Many solutions for using this spare time are being researched. However, this idle time of computers in an organization is being stolen illegally. Some other software runs on an idle computer without the knowledge of the organization. This is called theft of 'computer time'.

A commonly cited reference is the **Ten Commandments of Computer Ethics** written by the Computer Ethics Institute. This is given below.

- Thou shalt not use a computer to harm other people.
- Thou shalt not interfere with other people's computer work.
- Thou shalt not snoop around in other people's computer files.
- Thou shalt not use a computer to steal.
- Thou shalt not use a computer to bear false witness.
- Thou shalt not copy or use proprietary software for which you have not paid.
- Thou shalt not use other people's computer resources without authorization or proper compensation.
- Thou shalt not appropriate other people's intellectual output.
- Thou shalt think about the social consequences of the program you are writing or the system you are designing.
- Thou shalt always use a computer in ways that insure consideration and respect for your fellow humans.

## **Cyber Laws:**

Computer crimes require special laws to be formed by the government. Different countries have different ways of making the laws and awarding punishment to those who commit the crimes. India has Cyber laws to prevent computer crimes.

### **21.3 Cracking**

Cracking is the illegal access to the network or computer system. Illegal use of special resources in the system is the key reason for cracking. The resources may be hardware, software, files or system information. Revenge, business reasons and thrill are other common reasons for committing this crime.

### **21.4 Work, Family and Leisure**

Portable computers and telecommuting have created the condition where people can take their work anywhere with them and do it any time. As a result, workers find their work is cutting into family time, vacations, leisure, weakening the traditional institutions of family and friends and blurring the line between public and private life. This is becoming an important issue in computer ethics.

## **EXERCISES**

1. What is the need for a password to log into a computer system?
2. How does the Operating System enhance the Security ?