

IT Book

Class XI

INDEX

CHAPTER NO		PAGE NO
CH-1	Overview of fundamental of IT, open office Writer, Impress, Calc	1-12
	Open office writer Open office Impress Open office Calc	
CH-2	Introduction to programming logic and techniques	13-15
	Introduction to Programming Programming Techniques Flow charts and Algorithms	
CH-3	Number system & Boolean Algebra	16-21
	Introduction to binary Octal Decimal & Hexadecimal Complements	
CH-4	Fundamental of C	22-31
	History C Program Decision in C	
CH-5	Networking concept & internet	32-35
	Types of Internet History of Internet Terms used in internet Project work	

Chapter-1

Overview of Fundamental, Open Office Writer, Impress, Calc

Understanding the basic concepts of IT

An Information Technology (IT) concerns the processing, storage and/or transfer of information. Information can take many different forms such as words, numbers, pictures, sounds or video. An IT System can consist of computers, the telecommunications network and other programmable electronic devices. IT is used in business, industry, government, education, health care and in everyday home/social life. IT networks allow us to distribute and share information very quickly (a prime example is the Internet).

A Computer is a programmable machine that execute a prerecorded list of instructions (a program)

Characteristics of Computer

- 1. Speed :** The speed provided by computer is in incredibly faster than what man can possibly record or calculate normally .
- 2. Storage :** In computer the terminology in regard to storages capacity applies to both primary and secondary storages. It is normally measured in terms of Nibble , Byte, Kilobyte[1KB] Mega Byte [MB], Giga Byte[GB] AND Tera Byte [TB]. In computer memory large volumes of data to be maintained more prominently on secondary media Example : Floppy disks, Magnetic disks & Tapes etc.
- 3. Accuracy And Reliability :** The accuracy of operation of computer is always 100%. It is thus reliable .
- 4. Diligence / Endurance:** The computer is capable of operating at exactly the same level of speed and accuracy even if it has to carry out the most voluminous and complex operations for a long period of time.
- 5. Versatility:** The wide use of computer in so many areas such as scientific , commercial applications, Educational industrial areas in day –to –day life there is an ample evidence of its versatility .

Application of Computer

Computers have their application or utility everywhere. We find their applications in almost every sphere of life—particularly.

In Tourism: Hotels use computers to speed up billing and checkout the availability of rooms. So is the case with railways and airline reservations for booking tickets.

In Banks: Banks also have started using computers extensively.

In Industry: Computers are finding their greatest use in factories and industries of all kinds.

In Transportation: Today computers have made it possible for planes to land in foggy and stormy atmosphere also. The aircraft has a variety of sensors, which measure the plane's altitude, position, speed, height and direction. Computer use all this information to keep the plane flying in the right direction.

In Education: Computers have proved to be excellent teachers. They can possess the knowledge given to them by the experts and teach you with all the patience in the world. You may like to repeat a lesson hundred times, go ahead, you may get tired but the computer will keep on teaching you.



In Entertainment: Computers are also great entertainers. Many computer games are available which are like the traditional games like chess, football, cricket, etc.

Hardware and Software

Hardware:

Personal computer hardware are component devices which are typically installed into or peripheral to a computer case to create a personal computer. These are Pieces of equipment that make up a computer system. These are parts/devices connected to the computer..



Software:

Computer software, is a collection of computer programs and related data that provide the instructions for telling a computer what to do and how to do it. These are the instructions that a computer follows. Operating systems, office programs and games are examples of software. There are two types of software:

System Software

System software is to provide the most basic function of computer, and it can be divided into the operating system and support software, of which operating system is the basic operating system software. System software is responsible for managing the computer systems various independent hardware, enabling them to coordinate their work.

Application Software

Application is developed for a certain purpose. It can be a specific program, such as an image browser. It can also be a set of functions that can be closely linked, you can collaborate with each other a collection of programs, such as Microsoft's Office software. It can also be a separate program consisting of many large software systems, such as database management systems.

Computer Memory

Computer Memory is internal storage areas in the computer used to either temporarily or permanently store data or instructions to be processed.

Types of Computer Memory

The vast majority of computer memory can be placed into one of two categories: primary memory and secondary memory.

1. Primary memory, often called main memory, constitutes that device, or group of devices, that holds instructions and data for rapid and direct access by the computer's central processing unit (CPU). Primary memory is synonymous with random-access memory (RAM). As a computer performs its calculations, it is continuously reading and writing (i.e., storing and retrieving) information to and from RAM. Modern RAM is made of semiconductor circuitry, which replaced the magnetic core memory widely used in computers in the 1960s. RAM is a volatile form of information storage, meaning that when electrical power is terminated any data that it contains is lost. There are other semiconductor memory devices accessed by the CPU that are generally considered as being distinct from primary memory. These memory units include cache memory, read-only memory (ROM), and Programmable Read Only Memory (PROM).

ROM (Read Only Memory): This is a special type of memory which contains all the information the computer needs to switch itself on, check that all its systems are working and to tell the PC what things are plugged into it. It cannot be changed or overwritten by you, and stays the same even when the PC is switched off. An example of ROM on a PC is the **BIOS** software (Basic Input Output System) that enables the computer to start up and allows components to communicate with each other.

RAM (Random Access Memory): Random access memory is used in a PC to temporarily store data when you are using applications. RAM is also used to store program instructions and feed information to the CPU to process. RAM is not permanent, when you switch off the PC (or shut down), the contents of RAM are lost or emptied. There are two main uses of RAM in a computer system. These are **main memory** and **cache**.

2. Secondary memory, also called auxiliary memory or mass storage, consists of devices not directly accessible by the CPU. Hard drives, floppy disks, tapes, and optical disks are widely used for secondary storage. The **input and output** of these devices is much slower than for the semiconductor devices that provide the computer's primary memory. Most secondary storage devices are capable of containing much more information than is feasible for primary memory. Secondary memory is non-volatile. This means that data is stored with or without electrical power being supplied to the device, as opposed to RAM, which can retain its data only so long as electrical power is present.

INPUT AND OUTPUT DEVICES

The computer will be of no use unless it is able to communicate with the outside world. Input/ Output devices are required for users to communicate with the computer. In simple terms, input devices bring information INTO the computer and output devices bring information OUT of a computer system. These input/output devices are also known as peripherals

since they surround the CPU and memory of a computer system. Some commonly used Input/ Output devices are listed in table below.

Input Devices	Output Devices
Keyboard	Monitor
Mouse	LCD
Joystick	Printer
Scanner	Plotter
Light Pen	
Touch Screen	

INPUT DEVICES

(a) Keyboard

It is a text base input device that allows the user to input alphabets, numbers and other characters. It consists of a set of keys mounted on a board.



(b) Mouse: The mouse is a small device used to point to a particular place on the screen and select in order to perform one or more actions. It can be used to select menu commands, size windows, start programs etc. The most conventional kind of mouse has two buttons on top: the left one being used most frequently.



(c) Joystick: The joystick is a vertical stick which moves the graphic cursor in a direction the stick is moved. It typically has a button on top that is used to select the option pointed by the cursor. Joystick is used as an input device primarily used with video games, training simulators and controlling robots.



(d) Scanner: Scanner is an input device used for direct data entry from the source document into the computer system. It converts the document image into digital form so that it can be fed into the computer. Capturing information like this reduces the possibility of errors typically experienced during large data entry. Hand-held scanners are commonly seen in big stores to scan codes and price information for each of the items. They are also termed the bar code readers.

(e) Light Pen: It is a pen shaped device used to select objects on a display screen. It is quite like the mouse (in its functionality) but uses a light pen to move the pointer and select any object on the screen by pointing to the object. Users of Computer Aided Design (CAD) applications commonly use the light pens to directly draw on screen.

(f) Digital camera: A digital camera can store many more pictures than an ordinary camera. Pictures taken using a digital camera are stored inside its memory and can be transferred to a computer by connecting the camera to it. A digital camera takes pictures by converting the light passing through the lens at the front into a digital image.



OUTPUT DEVICES

(a) Monitor: Monitor is an output device that resembles the television screen and uses a Cathode Ray Tube (CRT) to display information. The monitor is associated with a keyboard for manual input of characters and displays the information as it is keyed in. It also displays the program or application output. Like the television, monitors are also available in different sizes.

(b) Liquid Crystal Display (LCD): LCD was introduced in the 1970s and is now applied to display terminals also. Its advantages like low energy consumption, smaller and lighter have paved its way for usage in portable computers (laptops).

© **Printer:** Printers are used to produce paper (commonly known as hardcopy) output. Based on the technology used, they can be classified as **Impact or Non-impact printers**.

Impact printers use the typewriting printing mechanism wherein a hammer strikes the paper through a ribbon in order to produce output. Dot-matrix and Character printers fall under this category. Non-impact printers do not touch the paper while printing. They use chemical, heat or electrical signals to etch the symbols on paper. Inkjet, DeskJet, Laser, Thermal printers fall under this category of printers.

Computer Languages

Machine Language: A set of instructions for a specific central processing unit, designed to be usable by a computer without being translated called *machine code*. The set of instructions, encoded as strings of binary bits, interpreted directly by a computer's central processing unit. Each different type of central processing unit has its own machine language. For a given machine language, each unique combination of 1's and 0's in an instruction has a unique interpretation, including such operations as arithmetical operations, incrementing a counter, saving data to memory, testing if data has a certain value, and so on.

Assembly Language: This language uses mnemonic codes 'or symbols in place of 0's AND 1's. Translator programs known as Assemblers were developed to convert the assembly language program into machine language. Assembly language is also machine – dependent and programming in this language is also very time-consuming. Thus, it is also programming as a low

level language.

High Level Language: High level language is quite similar to English language. Basic, C, C++, java etc. are some of the very popular high level languages. High Level languages programs needs to be translated into machines language by using .Translator Programs. There are two types of translator programs.

Open office Writer

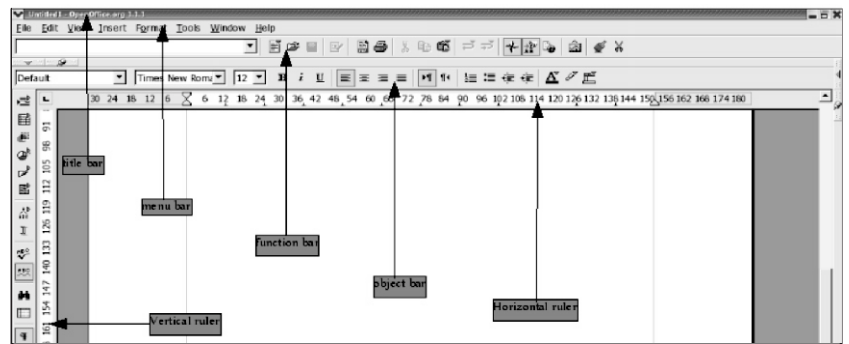
Open Office.org is reasonably intuitive but is sufficiently different to other suites that a period of familiarization required before use is fluid and effective.

Creating & Editing Text

A text document is displayed and edited in the Writer window. Spreadsheet, presentations or drawings are displayed in very similar windows, except the menus and icons change automatically depending on the context.

Entering and Formatting Text

There is no prescribed sequence in which to work. One can enter text while typing. Users can also decide whether or not to split a section of text into two columns immediately or to delay the action. Text never has to be deleted and retyped simply because formatting needs changing.



Basically, to edit text, first select it and then choose the relevant command, such as the one to format the text in italics.

Entering New Text

To enter new text:

1. Open an existing text document or create a new one.
2. Enter text using the keyboard. When special characters, such as the copyright symbol or accented characters that are not available on the keyboard are to be entered, select **Insert > Special Character** and chose what is needed from the table.
3. Press Enter to begin a new paragraph.

Inserting Text

1. Open an existing document.
2. Place the cursor at the point where text is to be added using either the mouse or the arrow Keys, and enter the new text.

Selecting and Deletng Text: Some basic steps to start.

Deleting characters

To delete one character to the left of the cursor, press Backspace (above the Enter key).

To delete one character to the right of the cursor, press the Delete key (may be labeled Del).

Deleting text

1. Left-click to set the cursor on the first character to be deleted or use the arrow keys.
2. Keeping the mouse button depressed, drag the pointer to the last character to be deleted. The characters will be highlighted.
3. Release the mouse button.
4. Press the Delete key to delete the selected text.

Emphasizing Text:

There are many ways of emphasizing text in a special way. Here are some of them:

- Use the icons in the Object bar for regular **Formatting** needs. For example, change the text to bold or to another font style, change the text color and background, or center the text.
- Whole paragraph can be emphasized using borders. Place the cursor in the paragraph that is to be emphasized, right-click to its context menu and select **Paragraph**, then click on, for example, the **Borders** tab. At this point, a border may be selected to frame the paragraph, and also with shadow shading, if desired. If necessary the distance between the border and the paragraph text can be adjusted under **Spacing to contents**.

Using a **Text Frame** provides the following possibilities:

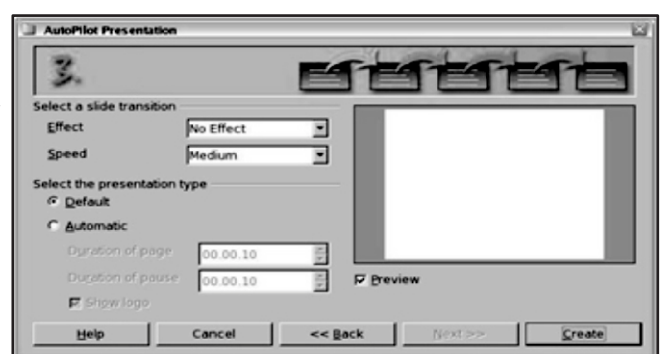
- Text can receive a border
- Text can be placed outside of the text margin on the side of the page.
- Text frames can be linked when text should flow from one frame to another.
- Choose **Format > Paragraph > Background** to apply a background color to the paragraph.
- Use the **Draw Text** "function: Open the **Draw function** toolbar (on the main toolbar), select the **Text** icon, drag open a frame and enter the text. This text can be positioned as desired, which includes rotating at various angles, or curving and slanting the text with the help of **Format > Font Work**.

Open office Impress

Open Office is free office software that is developed and maintained by the organization. OpenOffice.ORG. The Open Office suite of tools-Writer, Calc, Impress, and Draw- offers comparable capabilities to Microsoft Office's suite of tools-Word, Excel, and PowerPoint.

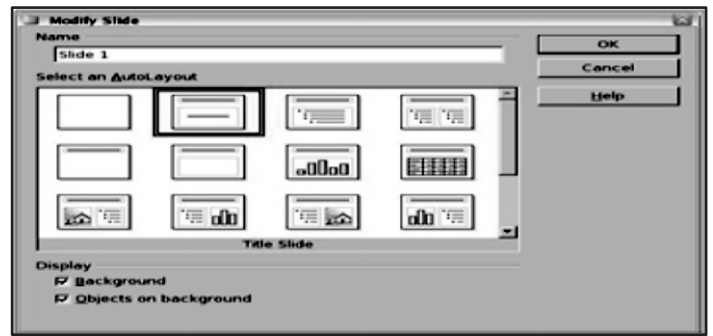
After launching OpenOffice.org an AutoPilot Presentation window appears. **Empty presentation** creates a presentation from scratch. **From template** uses a template design already created as the base of a new presentation. **Open existing presentation** continues work on a previously created presentation.

Creating a new presentation



The Effect option creates transitions between all the slides in the presentation. Select No Effect for no transition effect. Transitions can be added and changed later. Click "Create" to end the AutoPilot.

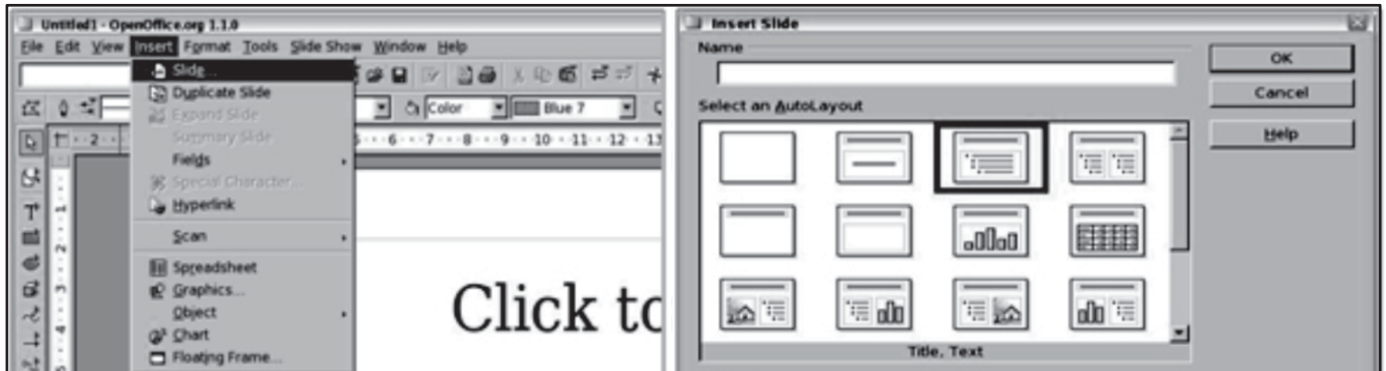
Type in a title for the slide in the area marked Name. Click a thumbnail slide from a "Select an Auto layout" section to select that layout. Click OK



Inserting Slides

To add a slide to the new presentation, go to the Insert menu and select "Slide..."

Insert a title for the slide in the Name field. Choose the slide layout from the "Select an Auto Layout" section. Click OK.



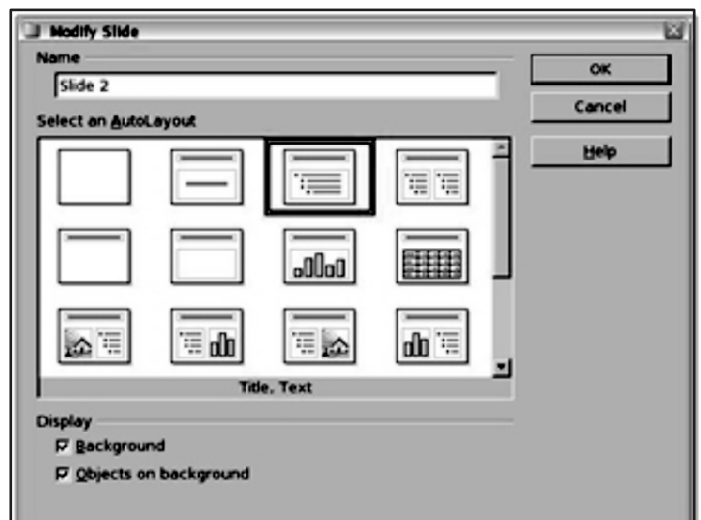
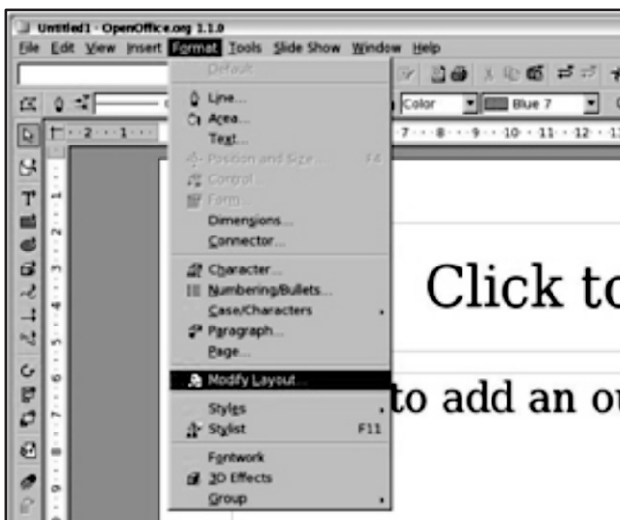
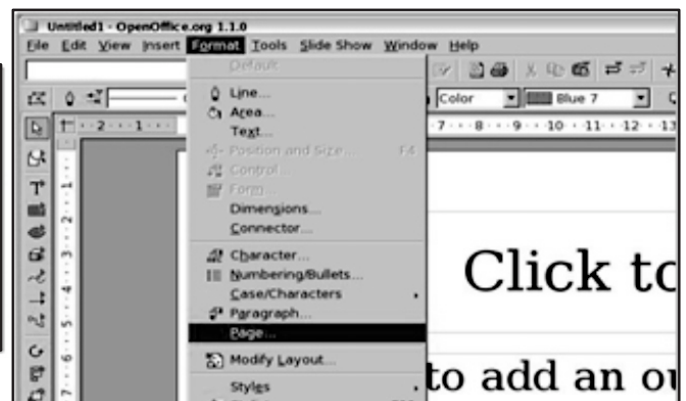
Selecting Slides

A new slide tab appears at the bottom of the workspace for each inserted slide. Click on a slide tab to select and display that tab.

Formatting a Page

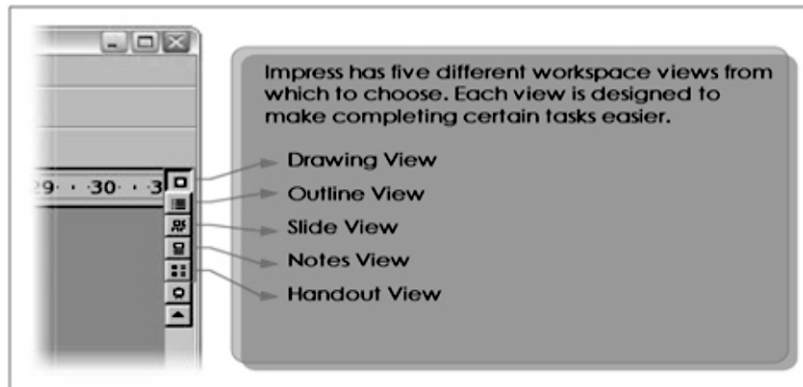
Go to the Format menu and click "Page..." In this window you can change the format, the orientation and the margins of the page.

Select "Modify layout..." from the Format menu. The Modify Slide window appears. Select "Modify layout..." from the Format menu. The Modify Slide window appears.

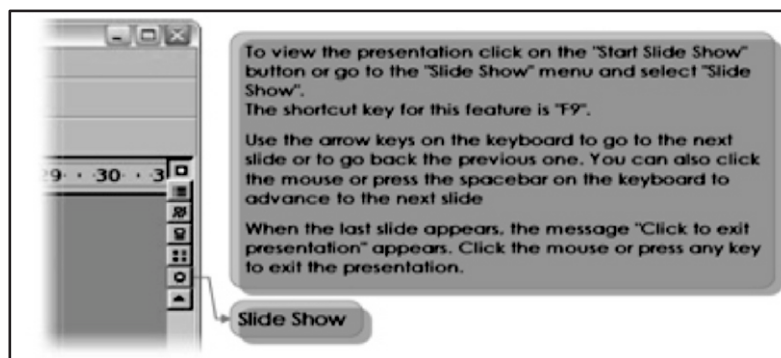


WORKSPACE VIEWS

Modify the layout by choosing a new layout from the "Select an Auto Layout" section.



Running the slide show



OPEN OFFICE CAL-C

Introduction

Open Office is an open source Office Suite package originally designed by Sun Microsystems. Open Office is much like Microsoft Office but free to use. The software can be freely downloaded and used. Open Office Calc is much like using Microsoft Excel.

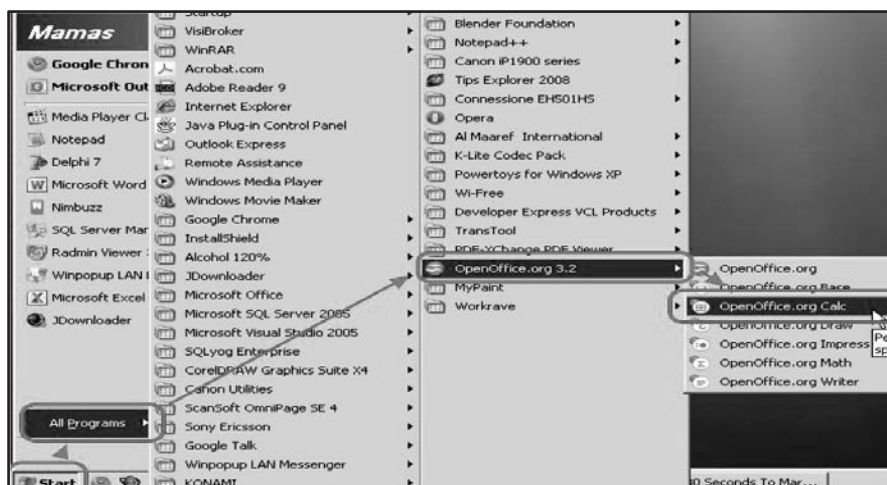
What is a Spreadsheet?

A spreadsheet is the computer equivalent of a paper ledger sheet. It consists of a grid made from columns and rows. Spreadsheets are made up of columns, rows, and cells (intersection of a column and row. A cell can contain data including text (strings or labels), numeric data, and formulas (mathematical equations).

Starting Open Office Calc

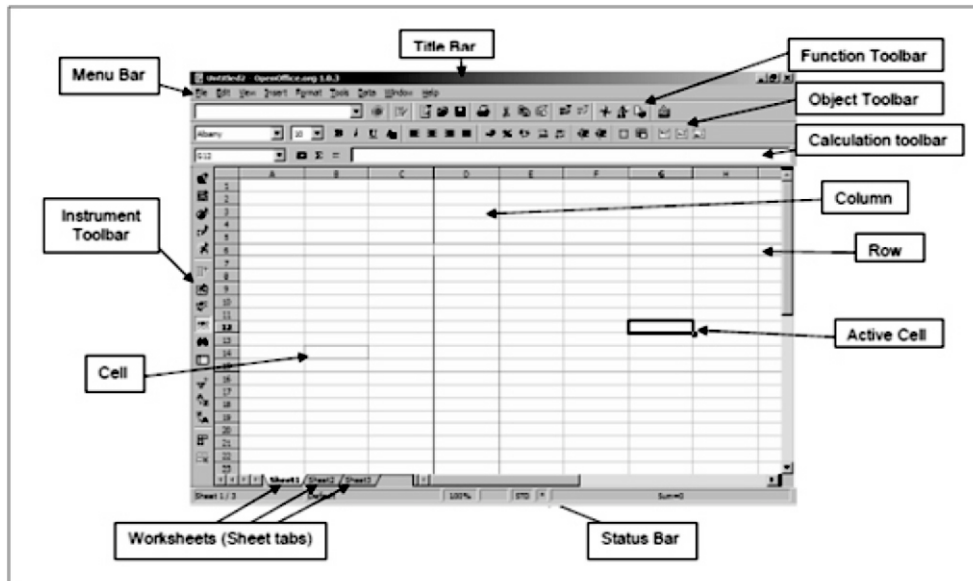
Procedure

1. Click **Start, Programs, Open Office.org 3.2, Open Office Calc.**



Open Office Calc (Spreadsheet) Basics

The picture below shows the Calc screen.



Creating a New Document

1. Type **CTRL+N** on your keyboard (hold down the CTRL key and type N).
2. Select **File, New, Spreadsheet** from the Menu Bar **OR**
3. Click on the **New Document Icon** on the Function Bar and select Spreadsheet.

Entering Data

To enter text, values or a formula:

1. Click on the cell you wish to enter information.
2. Type the information.
3. Press the **ENTER** key on the keyboard or press one of the arrow keys.

Selecting (Highlighting) Cell (s)

To Select One Cell

1. Click in the cell. (The active cell is already selected.)

To Select a Range of Cells (e.g. A1:G5)

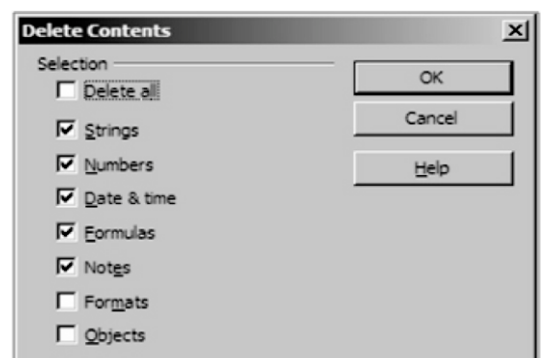
1. On the first cell of the range (e.g. A1), click and hold the left mouse button while dragging the mouse to the last cell of the range (e.g. G5).
2. All cells will turn black except the first cell will remain white.

To Select a Column or Row

1. Click on the column or row heading in gray.

Deleting Cell(s)

1. Select the cell(s) you wish to delete.
2. Press the **DELETE** key on the keyboard. The following window will appear.
3. Check the boxes of what you wish to delete (e.g. checking Formats will delete things like bold, italics, font color, borders).
4. Click the **OK** button.



Opening and Saving a File

The opening, saving, and printing of files are the most common actions in a word processor, and therefore, they have to be very easy and accessible.

Opening a File

1. Select the **File menu, Open** from the Menu Bar, **OR**
2. Click the **Open File icon** on the toolbar.
3. An **Open File Dialog box** will appear as shown below. From the drop down list (shown by the arrow) indicated below, choose the directory that contains the file you wish to open.
4. When a list of files appears, double click on the filename to open the file.



Saving a File

1. Choose **File, Save** from the Menu Bar, **OR**
2. Click the **Save icon** on the Function Bar.
3. If you are saving for the first time, a **Save File Dialog Window** will be displayed.
4. Type a filename and choose a location to save the file.
5. If you wish to save your file as a Microsoft Excel file format, change the Save as type to Microsoft Excel 97/2000/XP.
6. Click the **Save** button.



Creating Formulas/Calculations

You can enter text, numbers or formulae in these cells. Of course, the whole purpose of a spreadsheet application is to be able to carry out calculations within these cells. A calculation can be simply adding two numbers or taking the average of ten numbers.

Formula Basics

Just like in Microsoft Excel, formulas are started with an **EQUAL (=)** sign. (e.g. =b3+b4)

Use cell references (e.g. b3) instead of actual values (e.g. 56) wherever possible.

Use FUNCTIONS (e.g. SUM, AVERAGE) to save time when creating formulas.

e.g. =SUM(A1:A9)

Instead of =A1+A2+A3+A4+A5+A6+A7+A8+A9

By typing the colon [:] between the cell references, you have told the software that you want to add up the values in the range of cells from A1 to A9. The range is indicated on the screen by a red border. By typing "=sum()", you are telling the software the type of mathematical operation that you want to carry out on the referenced cells that are between parentheses.

You can also select the ranges to be added together using the mouse. After having typed "=sum()" into the target cell, click on the first cell and whilst holding the mouse button down, drag the mouse to the last cell of the range, and then let go of the mouse button, and you will see the end of the formula inserted in automatically into the Formula bar.

Cell References

A relative reference is a range whose references are adjusted when the formula is moved: e.g. If you copy the formula "=sum (A1: A9)" to column B, it will become "=sum (B1: B9)".

An absolute reference is used when a calculation has to refer to a precise or absolute cell of the spreadsheet. This is written using dollar signs (\$) around the cell reference. For example, typing \$A\$1 will make column A and row 1 absolute or fixed.

e.g. If you copy the formula “=B\$3 * B8” from cell C8 to cell C9, the formula would read “=B\$3 * B9. Notice the first part of the formula (=B\$3) did not change when copied. A good example of when you might use an absolute cell reference is when you are calculating the tax on an item. The tax (7%) will not change. Each item will be multiplied by the same tax amount.


Formatting Cell(s)

Text Alignment

Use these buttons  found on the toolbar to change the alignment of cell contents.

Merging Cells

Sometimes you might want to center a title between many columns.

1. Select all the cells in which the title is to be centered between.
2. In the menu toolbar, select **Format, Merge Cells, Define**.
3. The cells are now merged into one large cell. Click the **Center** button  on the toolbar to center text within this cell.
4. Click the **OK** button.

Page Settings

Page settings allow you to change settings related to a page as a whole such as changing the page orientation (e.g. portrait - 8.5" x 11" or landscape- 11" x 8.5"), adding headers/footers, and margins.

Changing Page Orientation

1. From the menu toolbar, choose **Format, Page**.
2. Click on the tab **Page**.
3. At the section Orientation, click on the radio button **Landscape**.
4. Confirm with **OK**.

To verify the change in page orientation, click the **File menu, Preview**.

Creating Charts

A chart (e.g. bar chart, pie chart) is a visual representation of data contained in a spreadsheet.

1. Select the cells you wish to include in the chart.
2. Click the **Insert menu, Chart...**
3. **Step 1:** Choose whether you want your chart in the same sheet as your data or a new sheet.
4. Click **Next**.
5. **Step 2:** Choose a chart type from the list.
6. Click **Next**.
7. **Step 3:** Choose a chart sub-type (variant) from the list. Choose data series in rows or columns.
8. Click **Next**.
9. **Step 4:** Add a title to your chart.
10. Click the **Create** button.

Printing a File

It is a good habit to check how the document will look when printed, before printing. It can be done by using the **Page Preview** feature.

1. Select **File, Page Preview** from the menu bar to switch to the **Page Preview Mode**.

2. Click **Page Preview**  button in Page Preview to close Page Preview and return to the main document.

You may want to print part of your spreadsheet, the entire workbook (all worksheets) or even only one sheet.

Chapter-2

Introduction To Programming Logic and Techniques

INTRODUCTION TO PROGRAMMING

If you want to learn any of the programming language you have to adopt the same mythology as you adopt to learn any language. First of all you must have the knowledge of the character used in this language, characters are useful for making words and a group of words make the complete sentence, then you will be able to develop a whole document. Similarly to learn a programming language you must be general term which are used in all programming languages are explained as below: -

1. Variables:-A variable is a name of the memory location which is used to store data. In any programming language which store data, a data store in a variable in infect store in a memory thus many time a variable is also termed as memory variable. When a location is given a name it is called memory variable.

There are certain rules which you should follow to create a memory variable: -

- ✧ A variable no other same should be 1-8 character (alphabets) long. (0-9) and underscore character can also be used in a variable name.
- ✧ Some compilers allow a variable name to be 40 characters long.
- ✧ The first character of a variable must be an alphabet.
- ✧ Within a variable name blank space and commas are not allowed.
- ✧ Accept underscore (_) special characters are allowed.

2. Constant:-Constant are the values which never changes during the execution of program, they remains constants. Example: - 3,100, "India".

3. Keywords:-

Keywords are those words which have special meaning in programming language: they can't be used as memory variable names. Keywords are also called revered words because they are special words for a programming language. In example will see few keyword used in ' C 'Auto, break, case, char, const, continue, default, do etc.

4. Identifier

Identifiers are same as variables; their names are defined by the programmer for different objects. The rules for making an identifier are exactly same as variable. Identifier or variables are named locations in memory that are used to hold a value that may be modified by the program. In C, all variables are to be declared before they can be used. They syntax for declaration is: - <datatype> <variable_list>. Here, type is a valid data type plus any other modifiers, and variable_list may contain one or more identifier names with comma separators.

5. Fundamental Data Type:-

These data type of already mentioned in the compiler of the programming language like char, int, float, double and void.

(b) Derived Data Type:-

This data type is created by the programmers at the time of writing programs for an example in 'C' we have array, function, classes, structure, union, and enumeration.

6. Operators

1. Unary Operator: - Unary operators are those operators which need one operand to perform execution. Some unary operators are: -

-
++
--
+=
-=
*=

2. Binary Operator:- Binary Operator works on two operands, some binary operators are :

✧ Arithmetic Operators :-

There are 5 basic Arithmetic Operators

+ >>>> Addition
- >>>> Subtraction
* >>>> Multiplication
/ >>>> Division
% >>>> Modulus

3. Relation and Logical Operators

Relation Operators are used to test the relationship between two variables, or between a variable and a constant. The test for the larger of two number a and b for example, is made by means of the > sign between the two operands and a and (a>b)

4. Logical Operators

Logical Operators are symbols that are used to combine or negate expressions containing relational operators.

Programming Techniques

Decision Structure Control is a critical factor of programming language, infect you can take the decision with the help of decision control statements which are explained given below: -

if Statement

Like most languages, C uses the keyword if to implement 1 he decision control instruction.

The general form of if statement looks like this: -

if (condition is true)
{ execute statement 1 ; execute statement 2 ; }

Example: if(category == SPORTS)
{ marks = marks + bonus_marks; }

Nested if_else statement

When a series of decisions are involved, we may have to use more than one **if...else** statement in nested form as follows:

if(codition is true)
if(condition2 is true)
{ Statement1; }
else
{ Statement2; }
else
{ Statement3; }

The for Statement

The 'for' loop is an entry-controlled *loop*, that provides a more concise loop control structure. The general form of the for loop is

```
for ( initialization ; test-condition ; increment )  
{ Body of the loop }
```

FLOWCHARTS:

A flow chart is a graphical or symbolic representation of a process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction. Different flow chart symbols have different meanings. The most common flow chart symbols are:

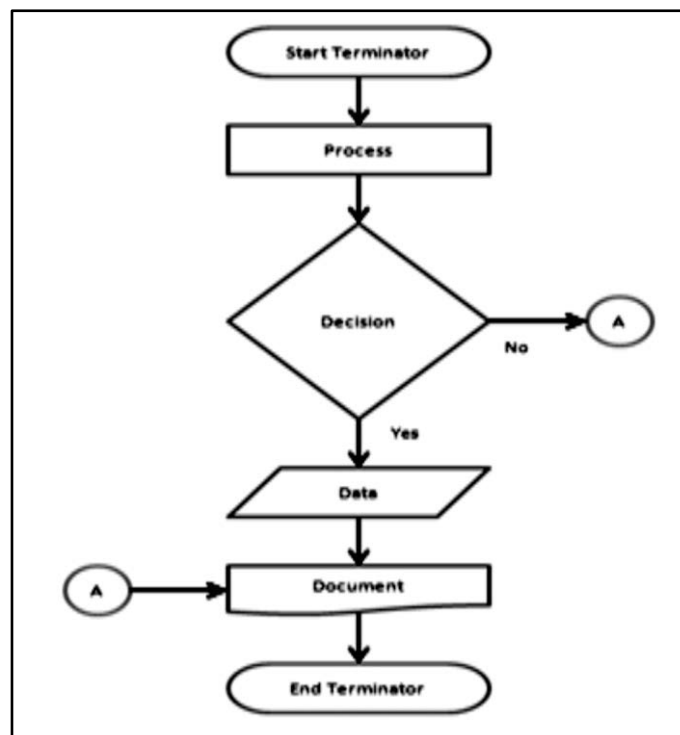
Terminator: An oval flow chart shape indicating the start or end of the process.

Process: A rectangular flow chart shape indicating a normal process flow step.

Decision: A diamond flow chart shape indicating a branch in the process flow.

Connector: A small, labeled, circular flow chart shape used to indicate a jump in the process flow. (Shown as the circle with the letter "A", below.)

- **Data:** A parallelogram that indicates data input or output (I/O) for a process.



ALGORITHM

Step by step representation of a program is called Algorithm. Example: To add two numbers:

Step1 Start

Step2 Input two values a and b.

Step3 Add both values of a and b and store it in c.

Step4 Print c.

PSEUDOCODE

Pseudocode is a detailed yet readable description of what a computer program or algorithm must do, expressed in a formally-styled natural language rather than in a programming language. Pseudocode is sometimes used as a detailed step in the process of developing a program.

Chapter- 3

Number System & Boolean Algebra

Number systems are used to describe the quantity of something or represent certain information. Because of this, I can say that the word

"calculator" contains ten letters. Our number system, the decimal system, uses ten symbols. Therefore, decimal is said to be **Base Ten**.

By describing systems with bases, we can gain an understanding of how that particular system works.

When we count in Base Ten, we count starting with zero and going up to nine in order.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ...

Once we reach the last symbol, we create a new placement in front of the first and count that up.

8, 9, **10**, 11, 12, ... , 19, **20**, ...

This continues when we run out of symbols for that placement. So, after 99, we go to 100. The placement of a symbol indicates how much it is worth. Each additional placement is an additional power of 10. Consider the number of 2853. We know this number is quite large, for example, if it pertains to the number of apples in a basket. That's a lot of apples. How do we know it is large? We look at the number of digits. Each additional placement is an additional power of 10, as stated above. Consider this chart.

10^3	10^2	10^1	10^0
digit	digit	digit	digit
*1000	*100	*10	*1

Each additional digit represents a higher and higher quantity. This is applicable for Base 10 as well as to other bases. Knowing this will help you understand the other bases better.

Binary

Binary is another way of saying Base Two. So, in a binary number system, there are only two symbols used to represent numbers: 0 and 1. When we count up from zero in binary, we run out of symbols much more frequently. 0, 1 ...

From here, there are no more symbols. We do not go to 2 because in binary, a 2 doesn't exist. Instead, we use 10. In a binary system, 10 is equal to 2 in decimal. We can count further.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	1	1	0	1
64+0+0+8+4+0+1						
87						

Binary	0	1	10	11	100	101	110	111	1000	1001	1010
Decimal	0	1	2	3	4	5	6	7	8	9	10

Just like in decimal, we know that the more digits there are, the larger the number. However, in binary, we use powers of two. In the binary number 1001101, we can create a chart to find out what this really means.

Since this is base two, however, the numbers don't get quite as large as it does in decimal. Even still, a binary number with 10 digits would be larger than 1000 in decimal.

Octal

Octal is another number system with less symbols to use than our conventional number system. Octal is fancy for Base Eight meaning eight symbols are used to represent all the quantities. They are 0, 1, 2, 3, 4, 5, 6, and 7. When we count up one from the 7, we need a new placement to represent what we call 8 since an 8 doesn't exist in Octal. So, after 7 is 10.

Octal	0	1	2	3	4	5	6	7	10	11	12...	17	20...	30...	77	100
Decimal	0	1	2	3	4	5	6	7	8	9	10...	15	16...	24...	63	64

Just like how we used powers of ten in decimal and powers of two in binary, to determine the value of a number we will use powers of 8 since this is Base Eight. Consider the number 3623 in base eight.

Each additional placement to the left has more value than it did in binary. The third digit from the right in binary only represented 2^{3-1} , which is 4. In octal, that is 8^{3-1} which is 64.

8^3	8^2	8^1	8^0
3	6	2	3
1536+384+16+3			
1939			

Hexadecimal

The hexadecimal system is Base Sixteen. As its base implies, this number system uses sixteen symbols to represent numbers. Unlike binary and octal, hexadecimal has six additional symbols that it uses beyond the conventional ones found in decimal. But what comes after 9? 10 is not a single digit but two... Fortunately, the convention is that once additional symbols are needed beyond the normal ten, letters are to be used. So, in hexadecimal, the total list of symbols to use is 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. In a digital display, the numbers B and D are lowercase.

When counting in hexadecimal, you count 0, 1, 2, and so on. However, when you reach 9, you go directly to A. Then, you count B, C, D, E, and F. But what is next? We are out of symbols! When we run out of symbols, we create a new digit placement and move on. So after F is 10. You count further until you reach 19. After 19, the next number is 1A. This goes on forever.

Hexadecimal	9	A	B	C	D	E	F	10	11...	19	1A	1B	1C...	9F	A0
Decimal	9	10	11	12	13	14	15	16	17	25	26	27	28	159	160

16^3	16^2	16^1	16^0
2	D	B	7
8192+3328+176+7			
11703			

Digits are explained as powers of 16. Consider the hexadecimal number 2DB7.

As you can see, placements in hexadecimal are worth a whole lot more than in any of the other three number systems.

Conversion

It is important to know that 364 in octal is **not** equal to the normal 364. This is just like how a 10 in binary is certainly not 10 in decimal. 10 in binary (this will be written as 10_2 from now on) is equal to 2. 10_8 is equal to 8. It is important to understand how the number systems work. By using our powers of the base number, it becomes possible to turn any number to decimal and from decimal to any number.

Base to Decimal

we know that 364_8 is not equal to the decimal 364. There is a simple method in converting from any base to the decimal base ten. If you remember how we dissected the numbers above, we used powers, such as 2^4 , and ended up with a number we understand. This is exactly what we do to convert from a base to decimal. We find out the true value of each digit according to their placement and add them together.

In a formula, this algorithm looks like:

$$V_{10} = v_p B^p + v_{p-1} B^{p-1} + \dots + v_1 B + v_0$$

Where V_{10} is the decimal value, v is the digit in a placement, p is the placement from the right of the number assuming the rightmost placement is 0, and B is the starting base. Do not be daunted by the formula! We are going to go through this one step at a time.

Let us use the formula above. Define every variable first. We want to find V_{10} , so that is unknown. The number $2B_{16}$ has two positions since it has two digits. p therefore is one less than that 1, so p is 1. The number is in base 16, so B is 16. Finally, we want to know what v is, but there are multiple v 's. You have v_1 and v_0 . This refers to the value of the digit in the subscripted position. v_1 refers to the digit in position one (the second digit from the right). So, v_1 is 2. v_0 is the first digit which is B. In the case of the conversion, you must convert all the letters to what they are in decimal. B is 11 in decimal, so v_0 is 11.

Now, plug all this into the formula:

$$V_{10} = 2(16^1) + 11(16^0) = 2(16) + 11(1) = 32 + 11 = 43$$

Therefore, $2B_{16}$ is equal to 43.

Now, let me explain how this works. Remember how digit placement affects the actual value? For example, in the decimal number 123, the "1" represents 100 which is 1×10^2 . The "2" is 20, or 2×10^1 . Likewise, in the number $2B_{16}$, the "2" is 2×16^1 , and the B is 11×16^0 .

We can determine the value of numbers in this way. For the number 364_8 , we will make a chart that exposes the decimal value of each individual digit. Then, we can add them up so that we have the whole. The number has three digits, so starting from the right, we have position 0, position 1,

8^2	8^1	8^0
3	6	4

and position 2. Since this is base eight, we will use powers of 8.

Now, 8^2 is 64. 8^1 is 8. 8^0 is 1. Now what?

3×64	6×8	4×1
192	48	4

Remember what we did with the decimal number 123? We took the value of the digit *times* the respective power. So, considering this further...

Now, we add the values together to get 244. Therefore, 364_8 is equal to 244_{10} .

In the same way that for 123, we say there is one group of 100, two groups of 10, and three groups of 1, for octal and the number 364, there are three groups of 64, six groups of 8, and four groups of 1.

In human language: the value of the cipher in the number is equal to the value of the cipher on its own multiplied by the base of the number system to the power of the position of the cipher from left to right in the number, starting at 0. Read that a few times and try to understand it.

Thus, the value of a digit in binary **doubles** every time we move to the left. (see table below)

From this follows that every hexadecimal cipher can be split up into 4 binary digits. In computer language: a nibble. Now take a look at the following table:

Binary Numbers				Hexadecimal Value	Decimal Value
8	4	2	1		
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	A	10
1	0	1	1	B	11
1	1	0	0	C	12
1	1	0	1	D	13
1	1	1	0	E	14
1	1	1	1	F	15

Another interesting point: look at the value in the column top. Then look at the values. You see what I mean? Yeah, you're right! The bits switch on and off following their value. The value of the first digit (starting from the right), goes like this: 0,1,0,1,0,1,0,1,0,1,... Second digit: 0,0,1,1,0,0,1,1,0,0,1,1,0,0... Third digit (value=4): 0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,... And so on...

Now, what about greater numbers? Therefore we'll need an extra digit. (but I think you figured that out by yourself). For the values starting from 16, our table looks like this:

For octal, this is similar, the only difference is that we need only 3 digits to express the values 1- >7. Our table looks like this:

Binary Numbers						
16	8	4	2	1	Hexadecimal Value	Decimal Value
1	0	0	0	0	10	16
1	0	0	0	1	11	17
1	0	0	1	0	12	18
1	0	0	1	1	13	19
1	0	1	0	0	14	20
1	0	1	0	1	15	21
1	0	1	1	0	16	22
1	0	1	1	1	17	23
1	1	0	0	0	18	24
1	1	0	0	1	19	25
1	1	0	1	0	1A	26
1	1	0	1	1	1B	27
1	1	1	0	0	1C	28
1	1	1	0	1	1D	29
1	1	1	1	0	1E	30
1	1	1	1	1	1F	31

Binary Numbers						
16	8	4	2	1	Hexadecimal Value	Decimal Value
1	0	0	0	0	10	16
1	0	0	0	1	11	17
1	0	0	1	0	12	18
1	0	0	1	1	13	19
1	0	1	0	0	14	20
1	0	1	0	1	15	21
1	0	1	1	0	16	22
1	0	1	1	1	17	23
1	1	0	0	0	18	24
1	1	0	0	1	19	25
1	1	0	1	0	1A	26
1	1	0	1	1	1B	27
1	1	1	0	0	1C	28
1	1	1	0	1	1D	29
1	1	1	1	0	1E	30
1	1	1	1	1	1F	31

Binary Numbers					
4	2	1		Octal Value	Decimal Value
0	0	0		0	0
0	0	1		1	1
0	1	0		2	2
0	1	1		3	3
1	0	0		4	4
1	0	1		5	5
1	1	0		6	6
1	1	1		7	7

- Step 1: Check if your number is odd or even.
- Step 2: If it's even, write 0 (proceeding backwards, adding binary digits to the left of the result).
- Step 3: Otherwise, if it's odd, write 1 (in the same way).
- Step 4: Divide your number by 2 (dropping any fraction) and go back to step 1. Repeat until your original number is 0.

An example: Convert 68 to binary:

- 68 is even, so we write 0.
- Dividing 68 by 2, we get 34.
- 34 is also even, so we write 0 (result so far - 00)
- Dividing 34 by 2, we get 17.
- 17 is odd, so we write 1 (result so far - 100 - remember to add it on the left)
- Dividing 17 by 2, we get 8.5, or just 8.
- 8 is even, so we write 0 (result so far - 0100)
- Dividing 8 by 2, we get 4.
- 4 is even, so we write 0 (result so far - 00100)
- Dividing 4 by 2, we get 2.
- 2 is even, so we write 0 (result so far - 000100)
- Dividing 2 by 2, we get 1.
- 1 is odd, so we write 1 (result so far - 1000100)
- Dividing by 2, we get 0.5 or just 0, so we're done.
- Final result: 1000100

From binary to decimal

- Write the values in a table as shown before. (or do so mentally)
- Add the value in the column header to your number, if the digit is turned on (1).
- Skip it if the value in the column header is turned off (0).
- Move on to the next digit until you've done them all.

An example: Convert 101100 to decimal:

- Highest digit value: 32. Current number: 32
- Skip the "16" digit, its value is 0. Current number: 32
- Add 8. Current number: 40
- Add 4. Current number: 44
- Skip the "2" and "1" digits, because their value is 0.
- Final answer: 44

From decimal to hexadecimal.

- Convert your decimal number to binary
- Split up in nibbles of 4, starting at the end
- Look at the first table on this page and write the right number in place of the nibble

(you can add zeroes at the beginning if the number of bits is not divisible by 4, because, just as in decimal, these don't matter). An example: Convert 39 to hexadecimal:

- First, we convert to binary (see above). Result: 100111
- Next, we split it up into nibbles: 0010/0111 (Note: I added two zeroes to clarify the fact that these are nibbles)
- After that, we convert the nibbles separately.
- Final result: 27

From hexadecimal to decimal

*Check the formula in the first paragraph and use it on the ciphers in your hexadecimal number.
(this actually works for any conversion to decimal notation)

An example: Convert 1AB to decimal:

- Value of B = $16^0 \times 11$. This gives 11, obviously
- Value of A = $16^1 \times 10$. This gives 160. Our current result is 171.
- Value of 1 = $16^2 \times 1$. This gives 256.
- Final result: 427

From decimal to octal

- Convert to binary.
- Split up in parts of 3 digits, starting on the right.
- Convert each part to an octal value from 1 to 7

Example: Convert 25 to octal

- First, we convert to binary. Result: 11001
- Next, we split up: 011/001
- Conversion to octal: 31

From octal to decimal

Again, apply the formula from above

Example: convert 42 to decimal

- Value of 2 = $8^0 \times 2 = 2$
- Value of 4 = $8^1 \times 4 = 32$
- Result: 34

One's and two's Complement

To find the 1's complement is to simply
take the bit by bit complement of the binary number.

For example: $N = +6 = 000001102$

$N = -6 = 111110012$

Conversely, given the 1's complement we can find the
magnitude of the number by taking it's 1's complement.

The largest number that can be represented in 8-bit 1's
complement is $011111112 = 127 = \$7F$. The smallest is
 $100000002 = -127$. Note that the values 000000002 and
 111111112 both represent zero.

Calculation of 2's Complement

To calculate the 2's complement of an integer, invert the binary equivalent of the number by
changing all of the ones to zeroes and all of the zeroes to ones (also called 1's complement), and
then add one.

For example,

0001 0001(binary 17) 1110 1111(two's complement -17)

NOT(0001 0001) = 1110 1110 (Invert bits)

1110 1110 + 0000 0001 = 1110 1111 (Add 1)

Chapter-4

Fundamental of C

HISTORY

C is a programming language developed at AT & T's Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL, etc. No one pushed C. It wasn't made the 'official' Bell Labs language. Thus, without any advertisement C's reputation spread and its pool of users grew. Ritchie seems to have been rather surprised that so many programmers preferred C to older languages like FORTRAN or PL/I, or the newer ones like Pascal and APL.

But, that's what happened.

Possibly why C seems so popular is because it is reliable, simple and easy to use. Moreover, in an industry where newer languages, tools and technologies emerge and vanish day in and day out, a language that has survived for more than 3 decades has to be really good.

Structure of C Program

Every C program consists of one or more functions. A function is nothing but a group or sequence of C statements that are executed together. Each C program function performs a specific task. The '**main()**' function is the most important function and **must** be present in every C program. The execution of a C program begins in the **main()** function. The structure of a C program:

Directives

main()

{ Statements; }

Programmers are free to name C program functions (except the main() function). The following is a simple C program that prints a message '**Hello, world**' on the screen:

```
/* First program of C language */
```

```
#include<stdio.h>
```

```
main( )
```

```
{ printf("Hello, world"); }
```

Keywords and Identifiers

Every C word is classified as either a keyword or an identifier. All **keywords** have fixed meanings and these meanings cannot be changed. The lists of all keywords in ANSI C are listed in Table.

Identifiers refer to the names of variables, functions and arrays. These are user defined names consisting of sequence of letters and digits, with a letter as first character.

Constants

Constants in C refer to fixed values that do not change during the execution of a program:

Integer Constants

An integer constant refers to a sequence of digits. There are three types of integers namely decimal, octal and hexadecimal. Decimal integer consists of set of digits, 0 through 9. Valid examples of decimal integer constant are:

123, _321, 0, +78

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

An octal integer constant consists of any combination of digits from the set 0 through 7 with a leading 0. Some examples of octal integers are: 037, 0, 0435, and 0551. A sequence of digits preceded by 0x or 0X is considered as hexa-decimal integer. They may also include alphabets A through F or a through f. Examples of valid hex integers: 0x2, 0x9F, 0xbcd, 0X. The largest integer value that can be stored is machine-dependent. It is 32167 on 16-bit machines and 2,147, 483, 647 on 32-bit machines. It is also possible to store larger integer constants on these machines by appending qualifiers such as U, L and UL to the constants.

Real Constants

Decimal numbers, fractional numbers and exponential numbers are real constants. Examples of real constants are: 0.0083, -0.75, 435.36, +247.0

Character Constants

A single character constant (or simply character constant) contains a single character enclosed within a pair of single quote marks. Examples of character constants are: '5' 'X'. The character constant '5' is not the same as the number 5. The last constant is a blank space. Character constants have integer values known as ASCII values. For example, the statement

```
printf("%d", 'a');
```

would print the number 97, the ASCII value of the letter. Similarly, the statement

```
printf("%c", '97');
```

would output the letter 'a'. Since each character constant represents an integer value it is also possible to perform arithmetic operations on character constants.

String Constants

A string constant is a sequence of characters enclosed in double quotes. The characters may be letters, numbers, special characters and blank space. Examples are:

"Hello!", "1987", "WELL DONE", "?..t", "5+3", "X"

Data Types

C language is rich in its data types. Storage representations and machine instructions to handle constants differ from machine to machine. The variety of data types available, allow the programmer to select the type appropriate to the needs of the application as well as the machine. ANSI C supports four classes of data types:

1. Primary (or fundamental) data types
2. User-defined data types
3. Derived data types
4. Empty data set

All C compilers support four fundamental data types, namely integer (int), character (char), floating point (float), and double precision floating point (double). Many of them also offer extended data types such as long int and long double. The range of the basic four types are given in Table

Data Type	Range of Values
char	-128 to 127
int	-32,768 to 32,767
float	3.4e-38 to 3.4e+38
double	1.7e-308 to 1.7e+308

Variables

A variable is a data name that may be used to store a data value. Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during execution. Rules for naming variables:

1. They must begin with a letter. Some system permits underscores as the first character.

2. ANSI standard recognizes a length of 31 characters, since only the first eight characters are treated as significant by many compilers.

3. White space is not allowed.

Variable Name	Valid?	Remark
First_tag	Valid	char is a keyword
char	Not Valid	Dollar sign is illegal
Priceg	Not Valid	Blank space is not permitted
group one	Not Valid	First eight characters are significant
average_number	Valid	
int_type	Valid	Keyword may be part of name

Some examples of valid variable names are:

Piyush Value T_raise, Delhi x1 ph_value, Mark sum1 distance.

Invalid examples include: t23 (area), % 25th

Operators

C supports a rich set of operators. An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables. C operators can be classified into a number of categories. They include:

1. Arithmetic operators.
2. Relational operators.
3. Logical operators.
4. Assignment operators.
5. Increment and decrement operators.
6. Conditional operators.
7. Bitwise operators.
8. Special operators.

Arithmetic Operators

C provides all the basic arithmetic operators. They are listed in Table. The operators +, -, *, and / all work the same way as they do in other languages. These can operate on any built-in data type allowed in C. Examples of arithmetic operators are:

a - b, a + b, a % b, a/b, a * b

Here a and b are variables and are known as

operands. The module division operator % cannot be used on floating point data.

Relational Operators

We often compare two quantities and depending on their relation take certain decisions. For example, we may compare the age of two persons, or the price of two items, and so on. An expression such as: a < b or 1 < 20 containing a relational operator is termed as a relational expression. The value of a relational expression is either one or zero. It is one if the specified relation is true and zero if the relation is false. For example: 10 < 20 is true. But 20 < 10 is false.

Operator	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
×	Multiplication
/	Division
%	Modulo division

Operator	Meaning
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

Logical Operators

In addition to the relational operators, C has the following three logical operators: `&&` meaning logical **AND**,

`||` meaning logical **OR**,

`!` meaning logical **NOT**

The logical operators `&&` and `||` are used when we want to test more than one condition and make decisions. An example is: `a > b && x == 10`. An expression of this kind which combines two or more relational expressions is termed as a logical expression or a compound relational expression.

Table: Truth Table

<i>op-1</i>	<i>op-2</i>	<i>Value of the Expression</i>	
		<i>op-1 && op-2</i>	<i>op-1 op-2</i>
Non-zero	Non-zero	1	1
Non-zero	0	0	1
0	Non-zero	0	1
0	0	0	0

Some examples of the usage of logical expressions are:

1. **If** (age > 55 && salary < 1000)
2. **If** (number < 0 || number > 100)

Assignment operators

An assignment operator (`=`) is used to assign a constant or a value of one variable to another. e.g.
`a=5;`

Increment and decrement operators

C provides two operators for incrementing and decrementing the value of variables.

- The increment operator `++` add 1 to its operand.
- The decrement operator `--` subtracts 1 from its operand

Conditional operator

A conditional operator checks for an expression, which returns either a true or a false value. If the condition evaluated is true, it returns the value of the true section of the operator, otherwise it returns the value of the false section of the operator. Its general structure is as follows:

`expression1 ? expression 2 (True Section): expression3 (False Section)`

Bitwise operator

C provides six bitwise operators for manipulating bit.

C Bitwise Operators Description

`&` Bitwise AND

| Bitwise inclusive OR
^ Bitwise exclusive OR
<< Bitwise left shift
>> Bitwise right shift

~one's complement The bitwise operators are preferred in some contexts because bitwise operations are faster than (+) and (-) operations and significantly faster than (*) and (/) operations.

DECISION MAKING AND LOOPS

Introduction

We have seen that a C program is a set of statements which are normally executed sequentially in the order in which they appear. This happens when no options or no repetitions of certain calculations are necessary. However, in practice, we have a number of situations where we may have to change the order of execution of statements based on certain conditions, or repeat a group of statements until certain specified conditions are met. This involves a kind of decision-making to see whether a particular condition has occurred or not and then direct the computer to execute certain statements accordingly.

Decision-Making Statement

C language possesses such decision-making capabilities and supports the following statements known as control or decision-making statements.

if Statement

- The **if** statement is a powerful decision-making statement and is used to control the flow of execution of statements. It is basically a two-way decision statement and is used in conjunction with an expression. It allows the computer to evaluate the expression first and then, depending on whether the value of the expression (relation or condition) is 'true' (non-zero) or 'false' (zero), it transfers the control to a particular statement. The **if** statement may be implemented in different forms depending on the complexity of conditions to be tested.
 - Simple **if** statement
 - **if.....else** statement
 - Nested **if....else** statement
 - **else if** ladder.

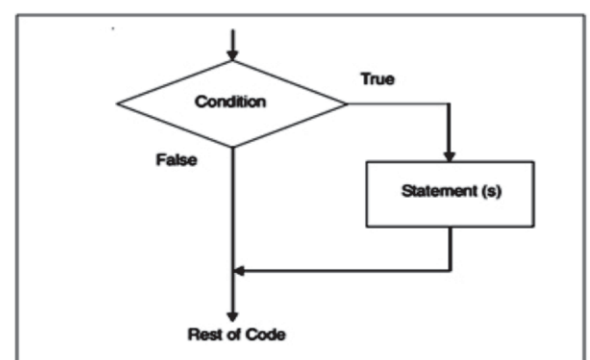
Simple if Statement

The general form of a simple if statement is

```
if(test expression)
{ statement-block; }
statement-x;
```

The 'statement-block' may be a single statement or a group of statements. If the test expression is true, the statement-block will be executed; otherwise the statement-block will be skipped and the execution will jump to the statement-x. Remember, when the condition is true both the statement-block and the statement-x are executed in sequence. This is illustrated in Figure.

Consider the following segment of a program that is written for processing of marks obtained in an entrance examination:



The if...else statement

The **if...else** statement is an extension of the simple if statement.

if (test expression)

{ True-block statement(s) }

Else

{ False-block statement(s) }

statements

If the test expression is true, then the true-block statement (S), immediately following the if statement is executed, otherwise, the false-block (S) are executed. In either case, either true-block or false-block will be executed, not both

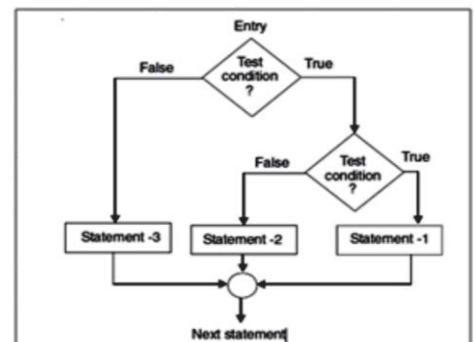
Nesting of if...else Statements

When a series of decisions are involved, we may have to use more than one **if...else** statement in nested form as follows:

The logic of execution is illustrated in Figure.

If the condition-1 is false, the statement-3 will be executed; otherwise it continues to perform the second test.

If the condition-2 is true, the statement-1 will be evaluated; otherwise the statement-2 will be evaluated and then the control is transferred to the statement-x.



The else if Ladder

There is another way of putting ifs together when multipath decisions are involved. A multipath decision is a chain of **ifs** in which the statement associated with each else there is an **if**. It is following general form:

This construct is known as the **else if** ladder. The conditions are evaluated from the top (of the ladder), downwards. As soon as a true condition is found, the statement associated with it is executed and the control is transferred to the statement-x skipping the rest of the ladder.

When all the n conditions become false, then the final else containing the default-statement will be executed. Figure shows the logic of execution of **else if** ladder statements. Let us consider an example of grading the students in an academic institution. The grading is done according to the following rules:

Average marks	Grade
80 to 100	Honors
60 to 79	First Division
50 to 59	Second Division
40 to 49	Third Division
0 to 39	Fail

This grading can be done using the **else if** ladder as follows:

if (marks > 79)

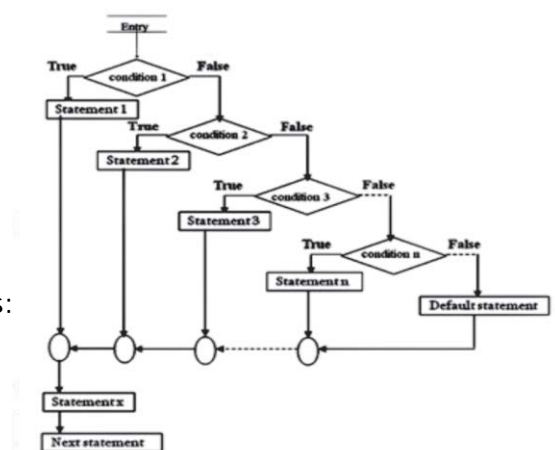
grade = "Honors".

else if (marks > 59)

grade = "First Division";

else if (marks > 49)

grade = "Second Division";




```

else if (marks > 39)
grade = "Third Division";
else
grade = "Fail";
printf ("%s\n", grade);

```

LOOPING CONTROL structure

We have seen to execute a segment of a program repeatedly by introducing a counter and later testing it using the if statement. While this method is quite satisfactory for all practical purposes, we need to initialize and increment a counter and test its value at an appropriate place in the program for the completion of the loop. For example, suppose we want to calculate the sum of squares of all integers between 1 and 10. We can write a program using the if statement as follows:

```

main (){
sum = 0;
n= 1;
loop;
sum= sum+n*n;
if (n == 10)          // n=10,
goto print;           // end of loop
else
{ n = n+1;
goto loop;
}print;}

```

This program does the following things:

1. Initializes the variable n.
2. Computes the square of n and adds it to sum.
3. Tests the value of n to see whether it is equal to 10 or not. If it is equal to 10, then the program prints the results.
4. If n is less than 10, then it is incremented by one and the control goes back to compute the sum again.

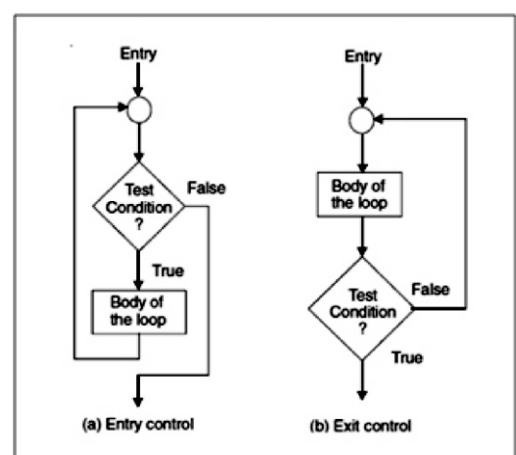
The program evaluates the statement

```
sum = sum + n*n;
```

10 times. That is, the loop is executed 10 times.

In looping, a sequence of statements is executed until some conditions for termination of the loop are satisfied. A program loop therefore consists of two segments, one known as the body of the loop and the other known as the control statement. The control statement tests certain conditions, and then directs the repeated execution of the statements contained in the body of the loop.

Depending on the position of the control statement in the loop, a control structure may be classified either as the entry-controlled loop or as the exit-controlled loop. The



flow charts in Figure illustrate these structures. In the entry-controlled loop, the control conditions are tested before the start of the loop execution. If the conditions are not satisfied, then the body of the loop will not be executed. In the case of an exit-controlled loop, the test is performed at the end of the body of the loop and therefore the body is executed unconditionally for the first time.

A looping process, in general, would include the following four steps:

1. Setting and initialization of a counter.
2. Execution of the statements in the loop.
3. Test for a specified condition for execution of the loop.
4. Incrementing the counter.

The C language provides for three loop constructs for performing loop operations. They are:

1. The **for** statement.
2. The **while** statement.
3. The **do** statement.

The for Statement

The 'for' loop is another entry-controlled *loop*, that provides a more concise loop control structure.

The general form of the for loop is for (initialization ; test-condition ; increment)

{ Body of the loop }

The execution of the 'for' statement is as follows:

1. Initialization of the control variables is done first, using assignment statements such as $i = 1$ and $count = 0$. The variables i and $count$ are known as loop-control variables.
2. As shown in fig., the value of the control variable is tested using the test-condition. The test-condition is a relational expression, such as $i < 10$ that determines when the loop will exit. If the condition is true, the body of the loop is executed; otherwise the loop is terminated and the execution continues with the statement that immediately follows the loop.

When the body of the loop is executed, the control is transferred back to the 'for' statement after evaluating the last statement in the loop. Now, the control variable is incremented using an assignment statement such as $i = i + 1$ and the new value of the control variable is again tested to see whether it satisfies the loop condition. If the condition is satisfied, the body of the loop is again executed. This process continues till the value of the control variable fails to satisfy the test-condition. Consider the following segment of a program:

```
for (x=0; x<=9; x=x+1)
```

```
{ printf ("%d", x); }
```

```
printf("\n");
```

This for loop is executed 10 times and prints the digits 0 to 9 in one line.

Comparison of the Three loops Do while loop

```
n=1;
```

```
do
```

```
{ printf ("%d", n); } while(n<=10);
```

While loop

```
n=1;
```

```
while (n<=10)
```

```
{ printf ("%d", n); }
```

For loop

```
for (n=1; n<=10; ++n)
```

```
{ printf ("%d", n); }
```

This for loop is executed 10 times and prints the digits 0 to 9 in one line.

Comparison of the Three loops

For loop

```
for (n=1; n<=10; ++n)
{
printf("%d", n);
}
```

While loop

```
n=1;
while (n<=10)
{
printf("%d", n);
}
```

Do while loop

```
n=1;
do
{
printf("%d", n);
}while(n<=10);
```

Example : The program in Figure uses a for loop to print the "Powers of 2" table for the power 0 to 20, both positive and negative

Notice that the initialization section has two parts $p = 1$ and $n = 1$ separated by a comma. Like the initialization section, the increment section may also have more than one part. For example, the loop

```
for (n=1, m=50; n<=m; n=n+1, m=m-1)
{ p = m/n;
printf ("%d %d %d\n", n, m, p); }
```

Is perfectly valid. The multiple arguments in the increment section are separated by commas. The third feature is that the test-condition may have any compound relation and the testing need not be limited only to the loop control variable. Consider the example below:

```
sum = 0;
for (i = 1; i < 20 && sum < 100; ++i)
{ sum = sum+i;
printf ("%d %d\n", sum); }
```

The loop uses a compound test condition with the control variable i and external variable sum . The loop is executed as long as both the conditions $i < 20$ and $sum < 100$ are true. The sum is evaluated inside the loop. It is also permissible to use expressions in the assignment statements of initialization and increment sections. For example, a statement of the type

```
for (x = (m+ny2; x > 0; x = x/2) is perfectly valid.
```

65536	16	0.000015258789
131072	17	0.000007629395
262144	18	0.000003814697
524288	19	0.000001907349
1048576	20	0.000000953674

The do...while Statement

In do...while statement, On reaching the do statement, the program proceeds to evaluate the body of the loop first. At the end of the loop, the *test-condition* in the while statement is evaluated. If the condition is true, the program continues to evaluate the body of the *loop* once again. This process continues as long as *the condition* is true. When the condition becomes false, the loop will be terminated and the control goes to the statement that appears immediately after the while statement. Since the test-condition is evaluated at the bottom of the loop, the do. .while construct provides an exit-controlled loop and therefore the body of the loop is always executed at least once. A simple example of a do...while loop is:

```
do{ printf("Input a number\n"); number = getnum(); }
while(number>0);.....
```

This segment of a program reads a number from the keyboard until a zero or a negative. The test conditions may have compound relations as well. For example, the statement **while (number > 0 && number < 100)**; would cause the loop to be executed as long as the number keyed lies between 0 and 100.

Consider another example:

.....

```
l=1
```

```
Sum=0
```

```
Do
```

```
{ Sum=Sum+l;
```

```
L=l+2; }
```

```
while(sum<40 || L<10)
```

```
printf("%d %d\n", l, sum);
```

The loop will be executed as long as one of the two relations is true

```
do {
    body of the loop
}
while (test-condition);
```

```
while (test
condition) {
    body of the
loop
}
```

The while Statement

The simplest of all the looping structures in C is the while statement. We have used **while** in many of our earlier programs. The basic format of the while statement is in the figure.

The while is an *entry-controlled loop* statement. The *test-condition* is evaluated and if the condition is true, then the body of the loop is executed. After execution of the body, the test-condition is once again evaluated and if it is true, the body is executed once again. This process of repeated execution of the body continues until the test-condition finally becomes *false* and the control is transferred out of the loop. On exit, the program continues with the statement

immediately after the body of the loop. The body of the loop may have one or more statements. The body of the loop is executed 10 times for $n = 1, 2, 10$ each time adding the square of the value of n , which is incremented inside the loop. The test condition may also be written as $n < 11$; the result would be the same.

```
sum = 0; n=1;
while(n <= 10) {
    SUM= SUM + n * n;
    n = n+1;
}
printf("sum = %d\n", sum);
```

Chapter - 5

Networking Concept and Internet

Computer Network

A network is any collection of independent computers that communicate with one another over a shared network medium. A computer network is a collection of two or more connected computers. When these computers are joined in a network, people can share files and peripherals such as modems, printers, tape backup drives, or CD-ROM drives. When networks at multiple locations are connected using services available from phone companies, people can send e-mail, share links to the global Internet, or conduct video conferences in real time with other remote users.

Types of Networks:

LANs (Local Area Networks)

A network is any collection of independent computers that communicate with one another over a shared network medium. LANs are networks usually confined to a geographic area, such as a single building or a college campus. LANs can be small, linking as few as three computers, but often link hundreds of computers used by thousands of people.

WANs (Wide Area Networks)

Wide area networking combines multiple LANs that are geographically separate. This is accomplished by connecting the different LANs using services such as dedicated leased phone lines, dial-up phone lines (both synchronous and asynchronous), satellite links, and data packet carrier services. Wide area networking can be as simple as a modem and remote access server for employees to dial into, or it can be as complex as hundreds of branch offices globally linked using special routing protocols and filters to minimize the expense of sending data sent over vast distances.

MAN (Metropolitan Area Network)

A network spanning a physical area larger than a LAN but smaller than a WAN, such as a city. A MAN is typically owned and operated by a single entity such as a government body or large corporation.

Need of Networks

Computer networks help users on the network to share the resources and in communication. Can you imagine a world now without emails, online news papers, blogs, chat and the other services offered by the internet?

The following are the important benefits of a computer network.

File sharing: Networking of computers helps the users to share data files.

Hardware sharing: Users can share devices such as printers, scanners, CD-ROM drives, hard drives etc.

Application sharing: Applications can be shared over the network, and this allows to implement client/server applications

User communication: Networks allow users to communicate using e-mail, newsgroups, and video conferencing etc.

Network gaming: Lot of games are available, which are supports multi-users.

Network Topologies

Network topology

A network topology is the geometric arrangement of nodes and cable links in a LAN. There are three topologies to think about when you get into networks. These are the star, ring, and the bus.

Star: In a star topology each node has a dedicated set of wires connecting it to a central network hub. Since all traffic passes through the hub, the hub becomes a central point for isolating network problems and gathering network statistics.

Ring: A ring topology features a logically closed loop. Data packets travel in a single direction around the ring from one network device to the next. Each network device acts as a repeater, meaning it regenerates the signal.

Bus: The bus topology, each node (computer, server, peripheral etc.) attaches directly to a common cable. This topology most often serves as the backbone for a network. In some instances, such as in classrooms or labs, a bus will connect small workgroups.

Internet

The Internet is a system of linked networks that are worldwide in scope and facilitate data communication services such as remote login, file transfer, electronic mail, the World Wide Web and newsgroups. With the meteoric rise in demand for connectivity, the Internet has become a communications highway for millions of users. The Internet was initially restricted to military and academic institutions, but now it is a full-fledged conduit for any and all forms of information and commerce. Internet websites now provide personal, educational, political and economic resources to every corner of the planet.

History

This marvelous tool has quite a history that holds its roots in the cold war scenario. A need was realized to connect the top universities of the United States so that they can share all the research data without having too much of a time lag. This attempt was a result of Advanced Research Projects Agency (**ARPA**) which was formed at the end of 1950s just after the Russians had climbed the space era with the launch of Sputnik. After the ARPA got success in 1969, it didn't take the experts long to understand that how much potential can this interconnection tool have. In 1971 Ray Tomlinson made a system to send electronic mail.

1973 saw the preparations for the vital **TCP/IP** and Ethernet services. At the end of 1970s, Usenet groups had surfaced up. By the time the 80s had started, IBM came up with its PC based on Intel 8088 processor which was widely used by students and universities for it solved the purpose of easy computing. By 1982, the Defence Agencies made the TCP/IP compulsory and the term "internet" was coined. The domain name services arrived in the year 1984 which is also the time around which various internet based marked their debut. Soon after the world got over with the computer worm, **World Wide Web** came into existence. Discovered by **Tim Berners-Lee**, World Wide Web was seen as a service to connect documents in websites using hyperlinks.

In 1992, internet browser called "Mosaic" came into existence. One of the very popular internet browsers, Netscape Navigator made its debut in 1994 which ultimately went to compete with Microsoft's Internet Explorer.

The advantages/Uses of Internet

Following are the advantages provided by the Internet:

- 1) **Information:** The biggest benefit offered by the Internet is information. It functions as a valuable resource of information. You can find any type of information on any subject with the help of the search engines like Yahoo and Google.
- 2) **Communication:** The primary goal of the Internet is communication. It has done extremely well in this field, however the development process is still going on to make it more dependable and quick. By sending an e-mail, we can contact a person who is physically present thousand miles away within the fraction of a second's time.
- 3) **Entertainment:** Internet functions as a popular medium of entertainment. A wide variety of entertainment including video games, music, movies, chat room, news and others can be accessed through the Internet.
- 4) **E-commerce:** E-commerce is the idea that is implemented for any form of commercial strategy or business transactions that entails transmission of data from one corner of the world to another. E-commerce has become a fantastic option through which you can shop anything.
- 5) **Formation of communities** Internet helps in formation of communities or forums. Here a number of people can participate in different types of debates and discussions express their views and gather valuable knowledge.
- 6) **Services** A variety of services are offered via Internet, for example job searching, online banking, buying movie tickets, hotel reservations and consultation services etc. When you avail these services offline, they become more expensive.

Electronic mail

E-mail is a method of exchanging digital messages from an author to one or more recipients. Modern email operates across the Internet or other computer networks. Some early email systems required that the author and the recipient both be online at the same time, in common with instant messaging. Today's email systems are based on a store-and-forward model. Email servers accept, forward, deliver and store messages.

Web Browsers

A **web browser** is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An *information resource* is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content.[2] Hyperlinks present in resources enable users easily to navigate their browsers to related resources.

Although browsers are primarily intended to access the World Wide Web, they can also be used to access information provided by web servers in private networks or files in file systems. The major web browsers are Firefox, Google Chrome, Internet Explorer, Opera, and Safari.

File Transfer Protocol

File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet. Like the Hypertext Transfer Protocol (HTTP), which transfers displayable Web pages and related files, and the Simple Mail Transfer Protocol (SMTP), which transfers e-mail, FTP is an application protocol that uses the Internet's TCP/IP protocols. FTP is commonly used to transfer Web page files from their creator to the computer that acts as their server for everyone on the Internet. It's also commonly used to download programs and other files to your computer from other servers.

Web Pages:

Web pages are what make up the World Wide Web. These documents are written in HTML (hypertext markup language) and are translated by your Web browser. Web pages can either be static or dynamic. Static pages show the same content each time they are viewed. Dynamic pages have content that can change each time they are accessed. These pages are typically written in scripting languages such as PHP, Perl, ASP, or JSP. The scripts in the pages run functions on the server that return things like the date and time, and database information. All the information is returned as HTML code, so when the page gets to your browser, all the browser has to do is translate the HTML.

Hyper Text : Electronic document text that is connected (hyperlinked) to the other chunks of text to which the reader or user is transferred by a mouse click. It is usually differentiated from the normal text by a different color, by underlining, or by both. The concept of hypertext is based on the idea that words or images in an electronic document do not have to behave linearly like text on paper. Hypertext is non-linear: the reader can jump from anywhere to anywhere while pursuing a chain of thought. The term was coined in 1969 by the US computer visionary Theodore Helm Nelson (1937-).

Hyperlink: A text or a set of texts which will redirect you to another web page/document by clicking on them. These hyperlinks can be generated through the especial kind of computer scripting language HTML.

Search Engines

A program that searches documents for specified keywords and returns a list of the documents where the keywords were found. Although *search engine* is really a general class of programs, the term is often used to specifically describe systems like Google, Alta Vista and Excite that enable users to search for documents on the World Wide Web and USENET newsgroups.

Typically, a search engine works by sending out a spider to fetch as many documents as possible. Another program, called an *indexer*, then reads these documents and creates an index based on the words contained in each document. Each search engine uses a proprietary algorithm to create its indices such that, ideally, only meaningful results are returned for each query.

Bing

Google

Yahoo!

Yebol



Addressing system in a Network Model

Four levels of addresses are used in an internet employing the TCP/IP protocols: physical (link) addresses, logical (IP) addresses, port addresses, and specific addresses

