

# IT Book

Class XII

# INDEX

CHAPTER NO		PAGE NO
CH-1	Overview of fundamental, Open office Writer, Impress ,Calc	1-16
	Open office writer	
	Open office Impress	
	Open office Calc	
CH-2	Overview of networking concept & Internet	17-19
	Introduction to Networking	
	Introduction to Interne	
CH-3	Programming in C	20-36
	History of C language	
	Loops	
	Arrays	
	Introduction to pointer & Functions	
	Structure & unions	
	File management in c	
CH-4	Introduction to software engineering	37-41
	System Development life cycle	
	Phases of system development	
	Initiation, analysis, Development, testing, Maintenance	
	Project work	

# Chapter-1

## Overview of Fundamental, Open Office Writer, Impress, Calc

### Understanding the basic concepts of IT

An Information Technology (IT) system 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.

### What is a Computer ?

A programmable machine. The two principal characteristics of a computer are: It responds to a specific set of instructions in a well-defined manner. It can execute a prerecorded list of instructions (a program)



### Characteristics of Computer

**1. Speed :** The speed however is incredibly faster than what man can possibly record or calculate normally. The speed of computer is usually given in terms of the following time units for the access time.

- Time : Millisecond [ 1ms] - A thousandth of a second or  $10^{-3}$
- Micro second [1us] – A millionth of a second or  $10^{-6}$
- Nano seconds [1ns] – A thousand millionth of a second or  $10^{-9}$
- Pico second [1ps] – A million millionth of a second or  $10^{-12}$

**2. Storage :** One of man's failings is perhaps his inability to remember and store large volumes of information is his brain. In computer the terminology in regard to storage 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]. Example : Floppy disks, Magnetic disks & Tapes etc.

UNITS	MEANING
Nibble	4 bits
Byte	8 bits
1 kilo Bytes [1KB]	$2^{10} = 1024 \text{ bytes}$
1 Mega Byte [1MB]	$2^{20} = 1024 \text{ KB}$
1 Giga Byte [1GB]	$2^{30} = 1024 \text{ MB}$
1 Tera Byte [1TB]	$2^{40} = 1024 \text{ GB}$

**3 Accuracy And Reliability :** Computer's are quite reliable in its calculation. The accuracy of operation of computer is always 100%. Computer is only a machine and does not make errors on its own. It is thus reliable.

**4 Automatic :** The computer is quite capable of functioning automatically, once the process has been initiated. It does not require a prompt from an operator at each stage of the process.

**5 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.

**6. 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 check out 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.

## Concept of 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.

- Pieces of equipment that make up a computer system.
- These are the parts you can touch (although many parts are contained within the computer's case).
- Other parts/devices are connected to the computer (usually by leads/connectors).
- These devices usually allow information (or **data**) to be entered (**input**) and retrieved (**output**).

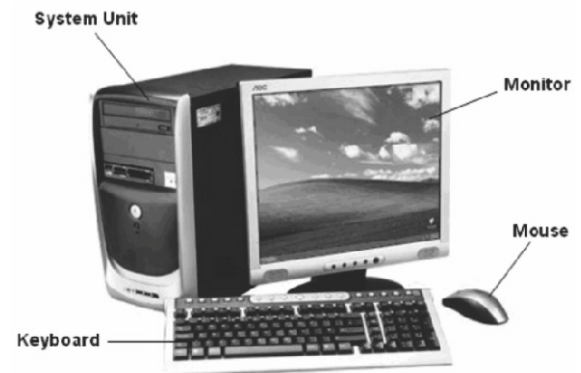
### Software:

**Computer software**, or just **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.

- The instructions that a computer follows (from **computer programs / applications**).
- Operating systems, office programs and games are examples of software.
- Software governs when and how various pieces of hardware can be put to a variety of uses.

### Computer Memory

Computer Memory is internal storage areas in the computer used to either temporarily or permanently store data or instructions to be processed. There are four basic types of computer memory: Cache Memory, RAM, Virtual Memory and Hard Drives. With modern CPU's running at speeds of 1 gigahertz or higher, it is hard for computer memory to keep up with the extreme amount of data being processed.



## Types of Computer Memory

### Primary and Secondary Memory

**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. RAM is not permanent, when you switch off the PC (or shut down), the contents of RAM are lost or emptied. **ROM (Read Only Memory)** 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.

**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.

Like primary memory, many secondary memory devices are capable of storing information, as well as retrieving it. Magnetic technology devices (such as hard drives, floppy disks, and tape) have this read-write capability, as do magneto-optical drives. However, some mass storage devices can only read data, as in the case of **CD-ROM (Compact Disk-Read Only Memory)** drives. CD-ROMs utilize optical technology; however, newer optical technologies, such as CD-RW (compact disk-rewriteable), can both read and write information like magnetic storage devices.

## INPUT AND OUTPUT DEVICES

### 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.



© **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.



**(e) Bar codes:** A bar code is a set of lines of different thicknesses that represent a number. Bar Code Readers are used to input data from bar codes. Most products in shops have bar codes on them. Bar code readers work by shining a beam of light on the lines that make up the bar code and detecting the amount of light that is reflected back.



**(f) Touch Screen:** It allows the user to operate/make selections by simply touching the display screen. Common examples of touch screen include information kiosks, and bank ATMs.

**(g) 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.



## OUTPUT DEVICES

**(a) Monitor:** Monitor is an output device that resembles the television screen and uses a Cathode Ray Tube (CRT) to display information. It also displays the program or application output. Like the television, monitors are also available in different sizes.

**(b) 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 fetch the symbols on paper. Inkjet, DeskJet, Laser, Thermal printers fall under this category of printers.

## Classification of Software

Software is the interface between the user and the hardware. Users mainly interact with the computer through the software. Software is an important basis for computer system design.

**System Software** System software is responsible for managing the computer systems various independent hardware, enabling them to coordinate their work. It enables computer users and other software take the computer as a whole without the need to take into account how each of the underlying hardware works.

**Application Software** Application Software is developed specifically for certain purpose e.g Payroll system for certain organization and Microsoft's Office software. It can also be a separate program consisting of many large software systems, such as database management systems.

## Computer Languages

**Machine Language:** The set of instructions, encoded as strings of binary bits, interpreted directly by a computer's CPU. Each different type of CPU 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.

**Assembly Language:** This language uses 'mnemonic codes' or symbols in place of 0's AND 1's. Instead of remembering the exact memory the exact memory location where data and instruction are stored, symbolic memory addresses are used for data. 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.

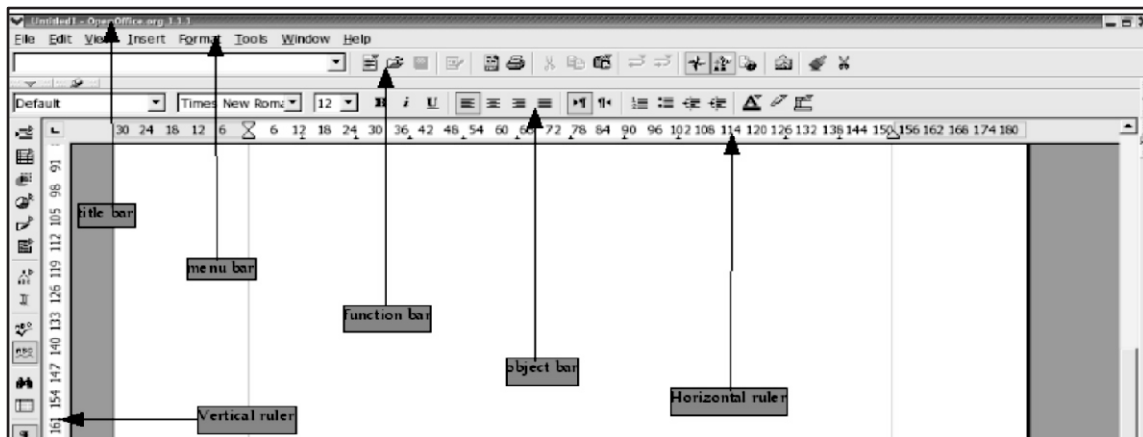
**High Level Language:** High level language is quite similar to English language. Basic, C, C++, java etc. are some of the very popular high level languages. High Level languages programs needs to be translated into machines language by using Translator Programs.

## Open office Writer

This chapter starts with a general explanation of the most common control features in OpenOffice.org, such as toolbars, floating toolbars and dockable windows. This is followed by information on functions that can be used in most of the program modules of OpenOffice.org.

### 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

One can enter text and, let us say, underline it or italicize it while typing or choose to do this later. 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 choose 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.

**Insert mode** is enabled by default, thus any text, following the insertion point, is shifted as new text is entered. To overwrite the existing text at this point, select **Overwrite mode** by clicking the **INSRT** field in the status bar as shown below:



### Switching Between Insert Mode and Overwrite Mode

**Using the keyboard:** If the keyboard has an **Ins** key, press it to toggle between overwrite mode and insert mode. The current mode is displayed on the status bar.



**INSRT:** Insert mode is enabled. The text cursor is a blinking vertical line. Click in the field to enable overwrite mode.

**OVER:** Overwrite mode is enabled. The text cursor is a blinking block. Click in the field to enable insert mode. Default layout of the status bar:



**Entering Text Anywhere on a Page:** Writer enables one to enter text at any position within the type area in the text document. This is the direct cursor function:

Click the **Direct Cursor on/off** icon on the main toolbar. Clicking the icon turns the direct cursor on and off. When the icon appears pressed, the direct cursor is enabled. Click on a free space in the text document. The shape of the mouse pointer shows how the text that is entered will be aligned. Enter the text. Open Office.org automatically inserts the requisite number of blank lines, tabs and spaces.



**Selecting and Deleting 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

#### Selecting text to delete with the mouse

1. Left-click to set the cursor on the first character to be deleted.
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.

#### Selecting text to delete with the keyboard

1. Use the arrow keys to go to the first character to be deleted.
2. Hold down the Shift key.
3. Using the arrow keys, move the cursor to just after the last character to be deleted.
4. Release the Shift key. The text is highlighted.
5. Press the Delete key to delete the selected text..

### Setting the Viewing Zoom Factor



By right-clicking in the area shown above, the user can select the display Zoom factor for the document. The optimal setting is particularly useful when the Stylist or Navigator are docked and opened, off and on, during a session in that the full width page of text is always displayed.



## Inserting Special Characters

This is how to insert special characters (such as check marks, boxes, telephone symbols etc.) in text:

Select Insert > Special Characters. Go view the selection of characters available.

If a special character is required in any text input field (such as in the URL field of the function bar or in the input fields in the Find & Replace dialogue), press Shift+Ctrl+S to pop up the **Special Characters** dialogue.

## 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.

**Changing the color of Text:** Click the **Font color** icon in Writer and other modules, and keep the mouse button pressed to obtain a floating toolbar from which to choose a color from the range of colors, Font color icon.

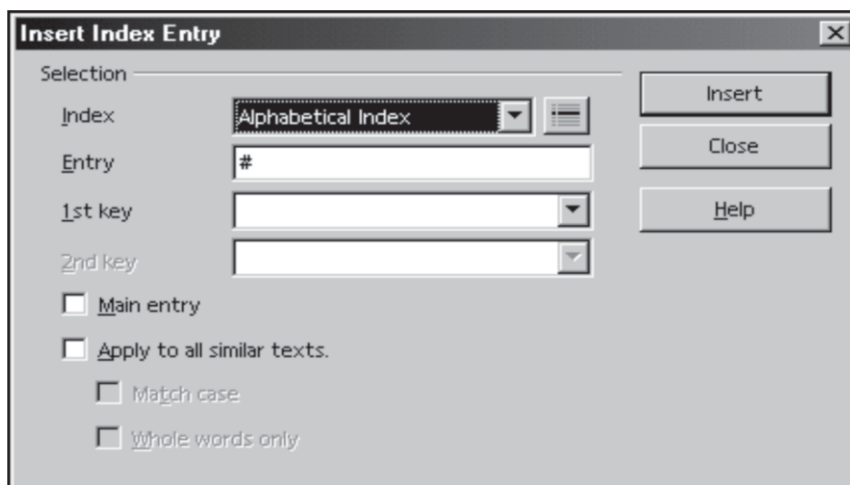
## Working with tables

### Indexes and Tables of Contents in Writer

Writer documents can contain any number of predefined or user-defined indexes. For example, users can have a table of contents, alphabetical index, illustration index and/or bibliography.

### Editing or Deleting Index and Table Entries

All defined index entries are shaded gray in the document for easy recognition but, which, is never be printed. If display of the shaded entries on-screen is unwanted, switch the highlighting on or off using **View > Field Shadings**.



1. To edit an entry, place the cursor immediately before or inside it.
2. Then choose **Edit > Index Entry**.
3. Alter the text in the **Entry** text box as needed.

4. Click on the **Delete** button to delete this entry from the list.

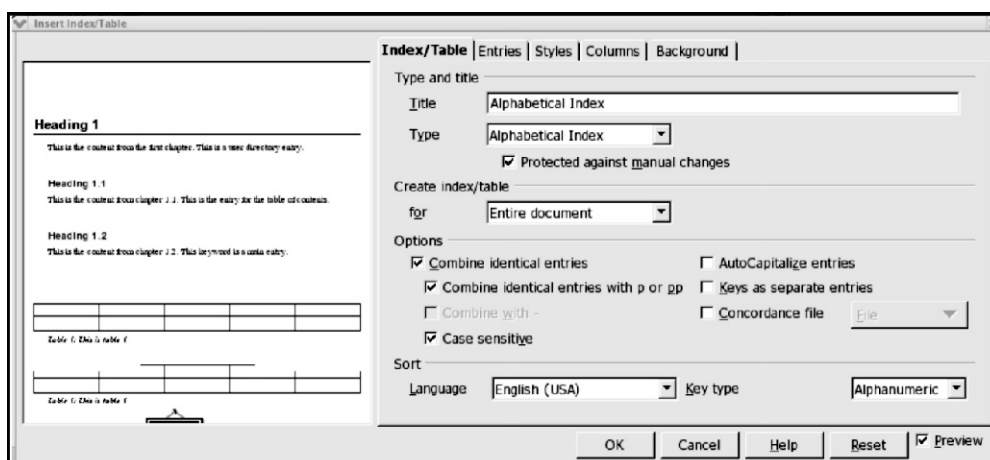
If the displayed text of the entry is “modified”, the altered text is inserted into the generated index. Only a thin gray mark in the document remains where the entry was. To edit an entry such as this, place the cursor directly after the thin mark and choose **Edit > Index Entry**. Use the arrow buttons in the **Edit Index Entry** dialogue to switch to the next or previous entry of the same index type.

### Creating a Table of Contents

1. Place the cursor at the location at which to create the table of contents.
2. Choose **Insert > Indexes and Tables > Indexes and Tables**. The **Insert Index/Table** dialogue appears. Click the **Index/Table** tab and select “Table of Contents” as Click on **OK** to create the table of contents from the headings and entries already defined.
3. To insert additional paragraphs of another Paragraph Style into the table of contents, check **Additional Styles**.
4. Click the icon next to the box to open the dialogue.
5. Define the Paragraph Styles that are also to appear in the table of contents and the level they are to be shown.

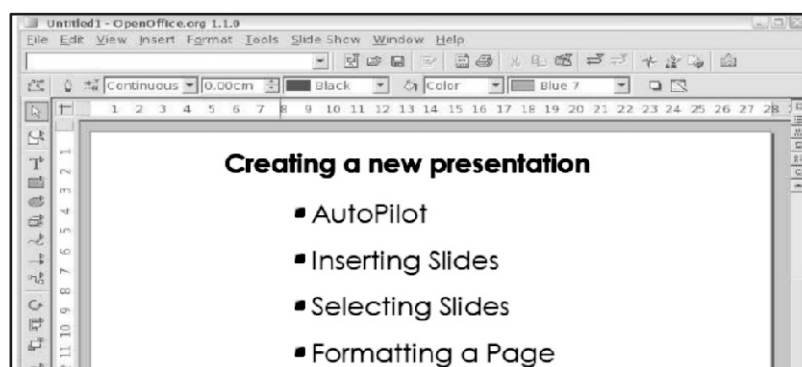
### Creating Alphabetical Indexes

1. Place the cursor in which to create the index.
2. Select **Insert > Indexes and Tables > Indexes and Tables**. The **Insert Index/Table** dialogue appears.
3. On the **Index/Table** tab, select “Alphabetical Index” in **Type**.
4. Click **OK** to generate the alphabetical key word index using the default settings.



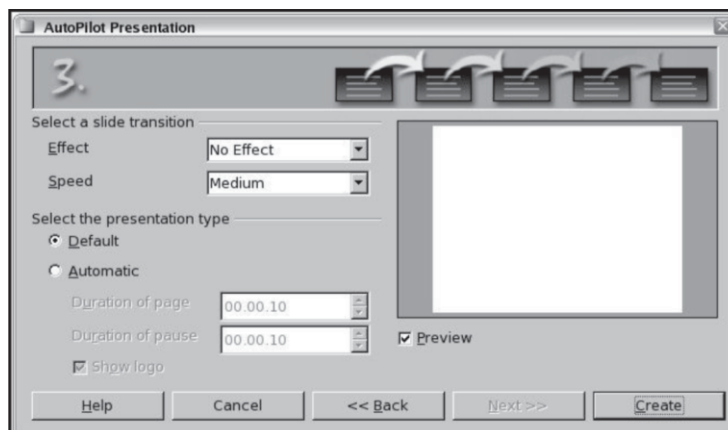
## Open office Impress

Open Office is free office software that is developed and maintained by the organization. Open Office. 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.

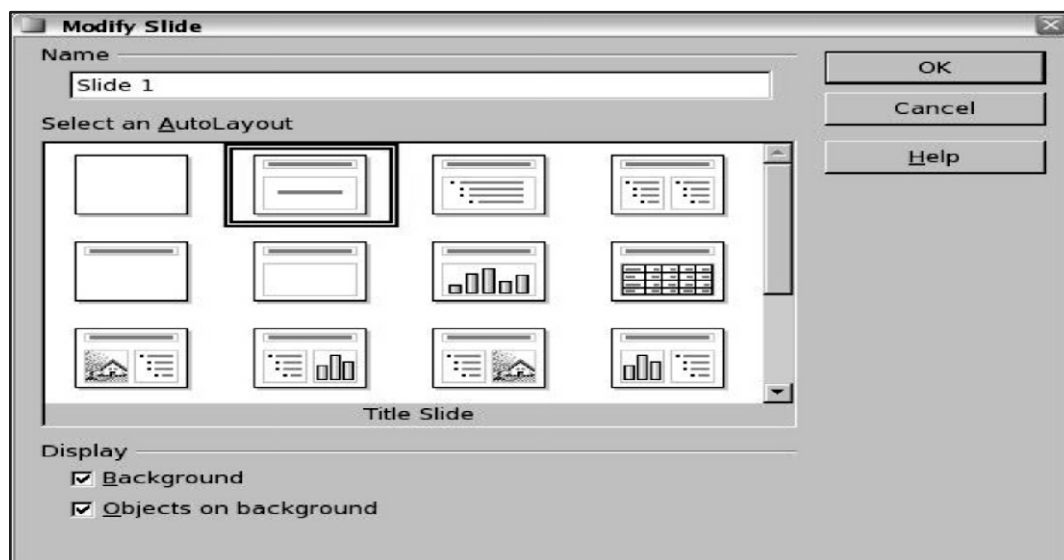


After launching OpenOffice.org an AutoPilot Presentation window appears. **Empty presentation** creates a presentation from scratch. **From template** uses a template design already created as the base of a new presentation. **Open existing presentation** continues work on a previously created presentation.

### 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.

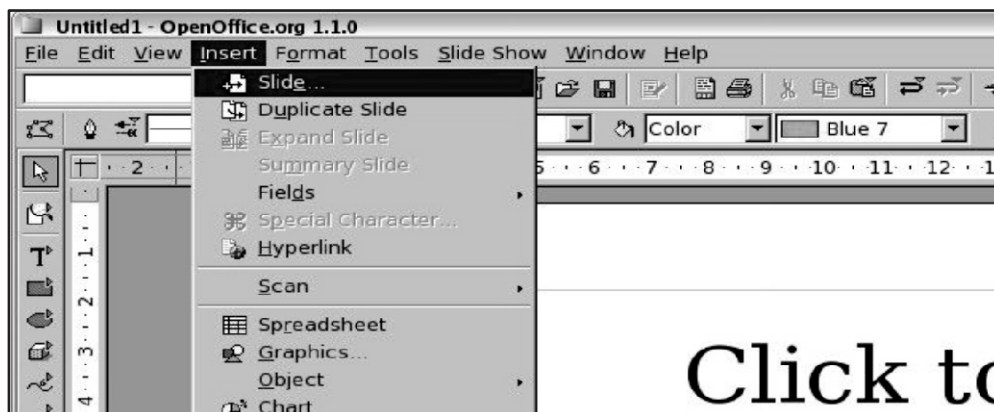


Type in a title for the slide in the area marked Name. Click a thumbnail slide from a "Select an Auto layout" section to select that layout. Click OK

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

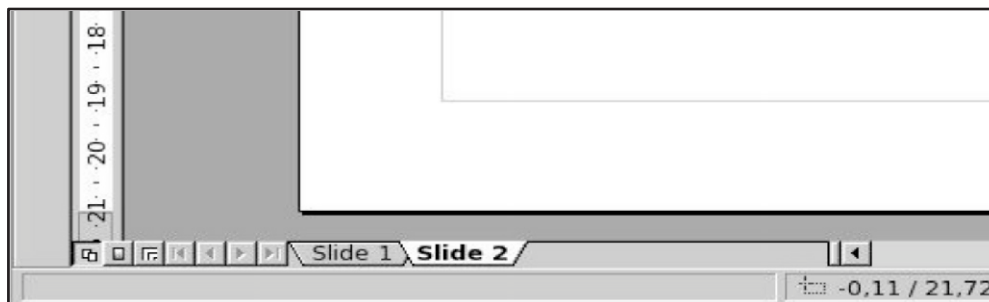
Insert a title for the slide in the Name field. Choose the slide layout from the "Select an Auto Layout" section. Click OK.

### Inserting Slides



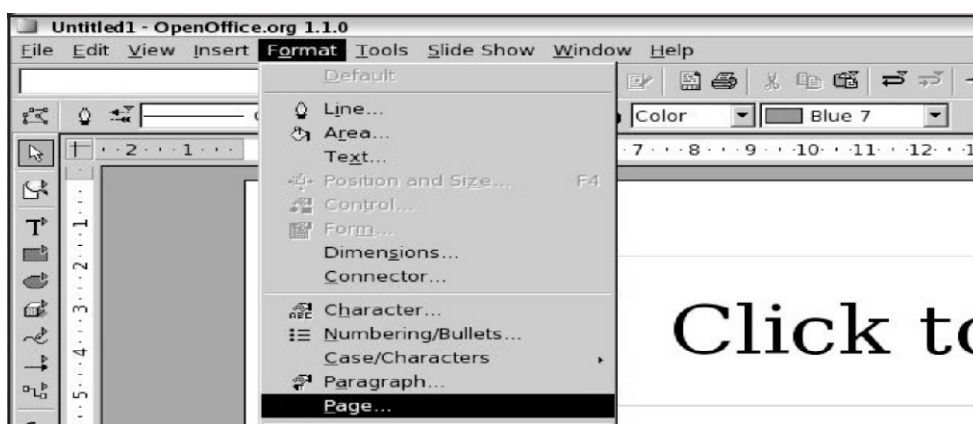
To add a slide to the new presentation, go to the Insert menu and select "Slide..."

## Selecting Slides

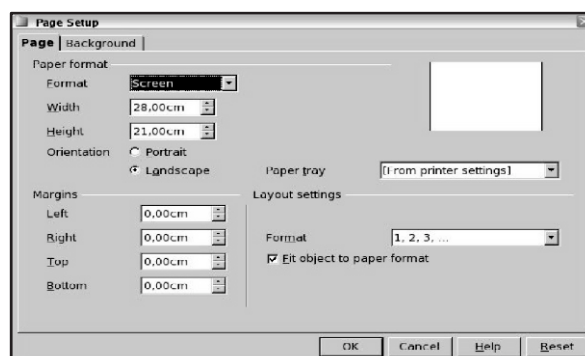


A new slide tab appears at the bottom of the workspace for each inserted slide. Click on a slide tab to select and display that tab.

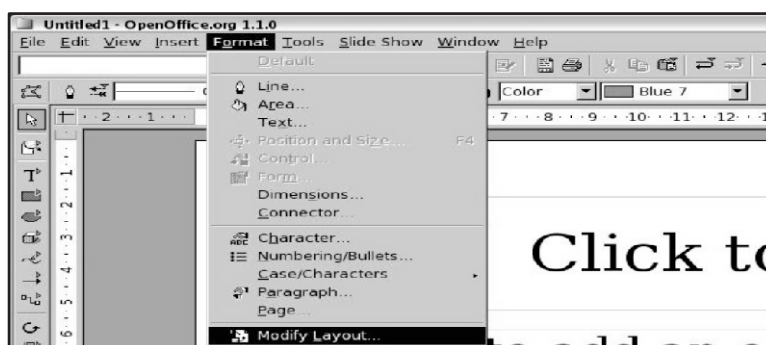
## Formatting a Page



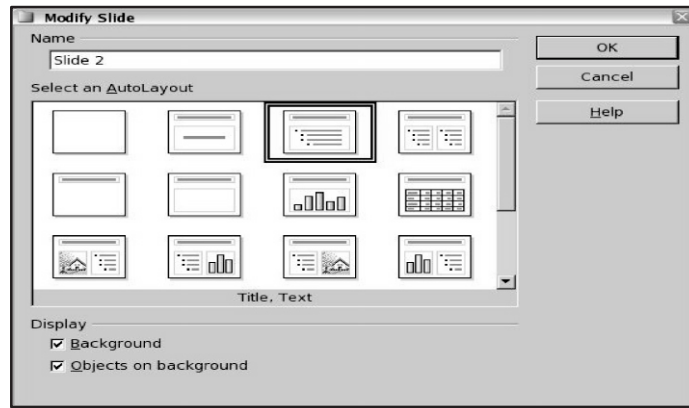
Go to the Format menu and click "Page..."



In this window you can change the format, the orientation and the margins of the page.

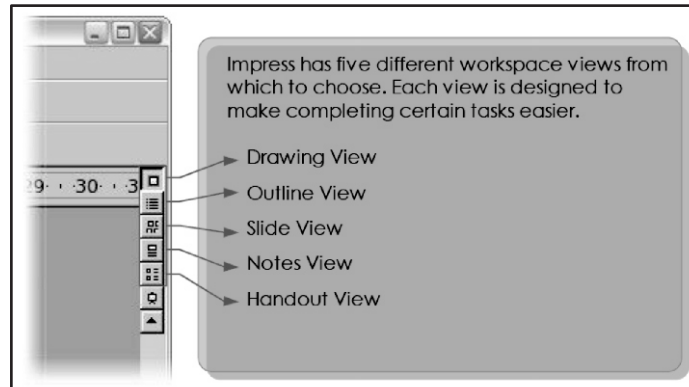


Select "Modify layout..." from the Format menu. The Modify Slide window appears.

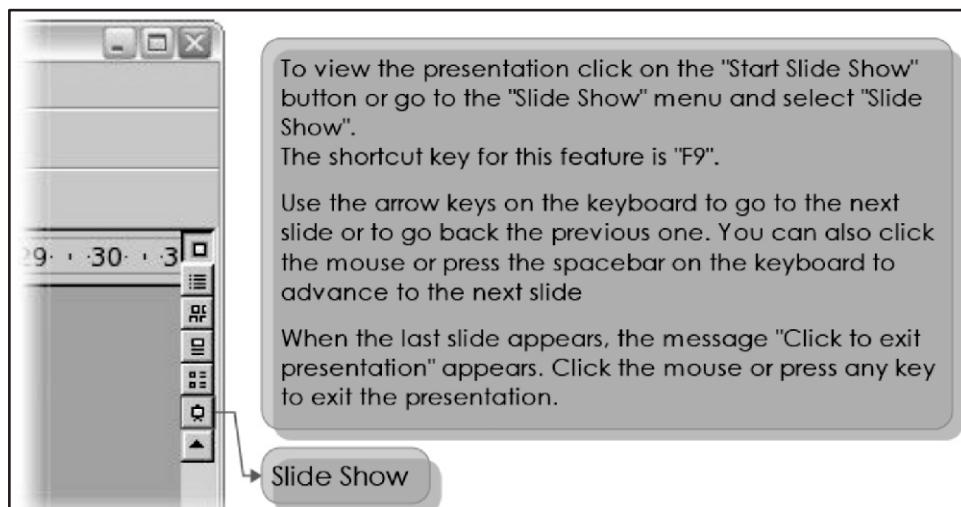


Modify the layout by choosing a new layout from the "Select an Auto Layout" section.

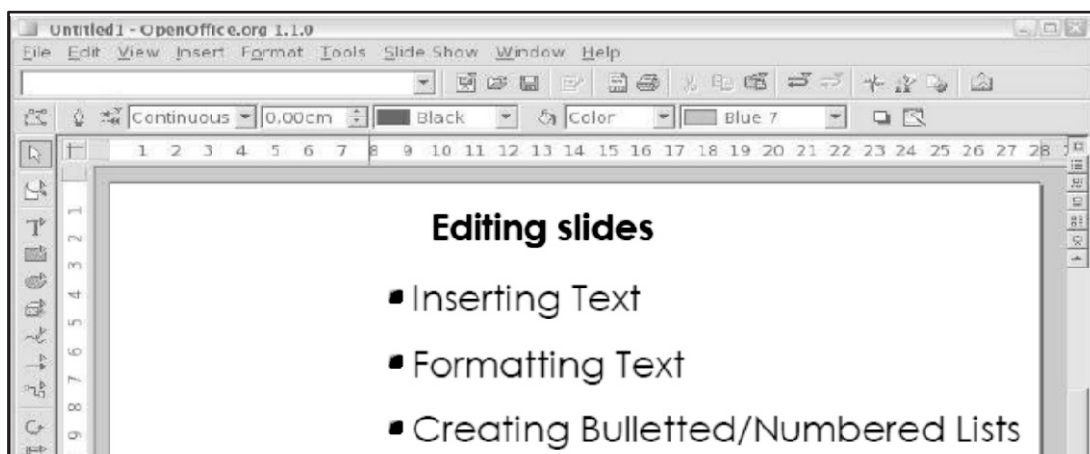
### Workspace views



### Running the slide show



### Editing slides



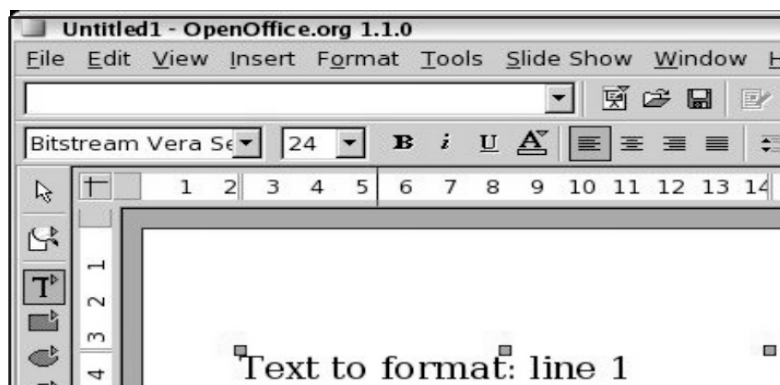
## Inserting Text

- Click on Text icon in the main toolbar.
- Click on the slide and drag to draw a box, release the mouse when finished.
- The cursor appears in the text box which is now in edit mode.
- Type the text in the box.
- Click outside the text box to deselect it.



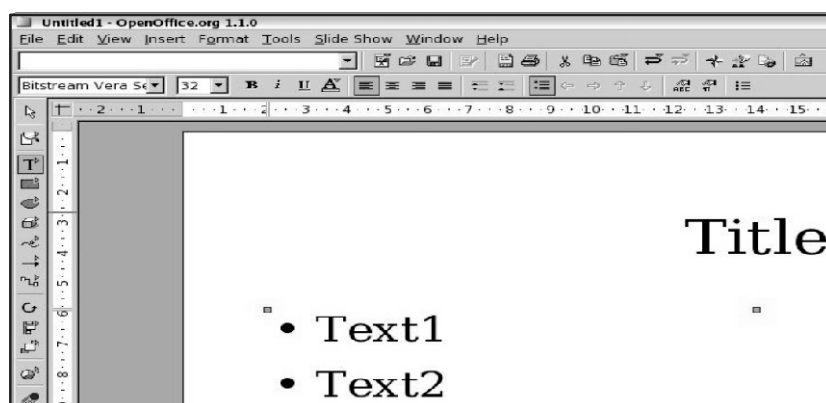
## Formatting Text

- The text must be selected before it can be formatted:
- To format all text in a text box, click on the text, then click once on the border of the text box.
- Now any formatting changing will apply to all text in the box.
- To format only a part of the text, click once on the text, then select the part to be formatted by clicking and dragging (highlighting) over it.
- Formatting changes will apply only to the selected text.



## Creating Bulleted/Numbered Lists

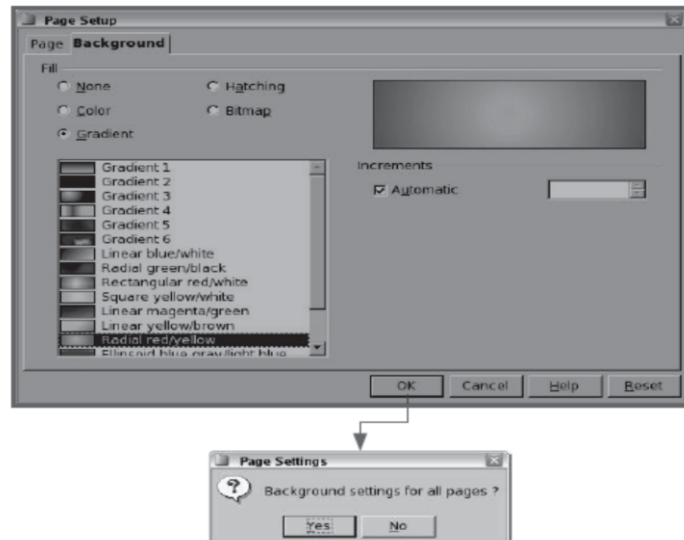
- To create a bulleted or numbered list from Auto Layout text boxes, insert a new slide or modify the current one and then select an Auto Layout that contains a numbered list
- Click in the box that reads "**Click to add an outline**" Type the text, then press *Enter* to start a new bulleted line or the next sequential numbered line.
- Press *Shift + Enter* to start a new line without creating a new bullet or number.





## Changing the Slide Background

To change the slide background select "Page..." from the format menu and then select on the background tab.



## 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?

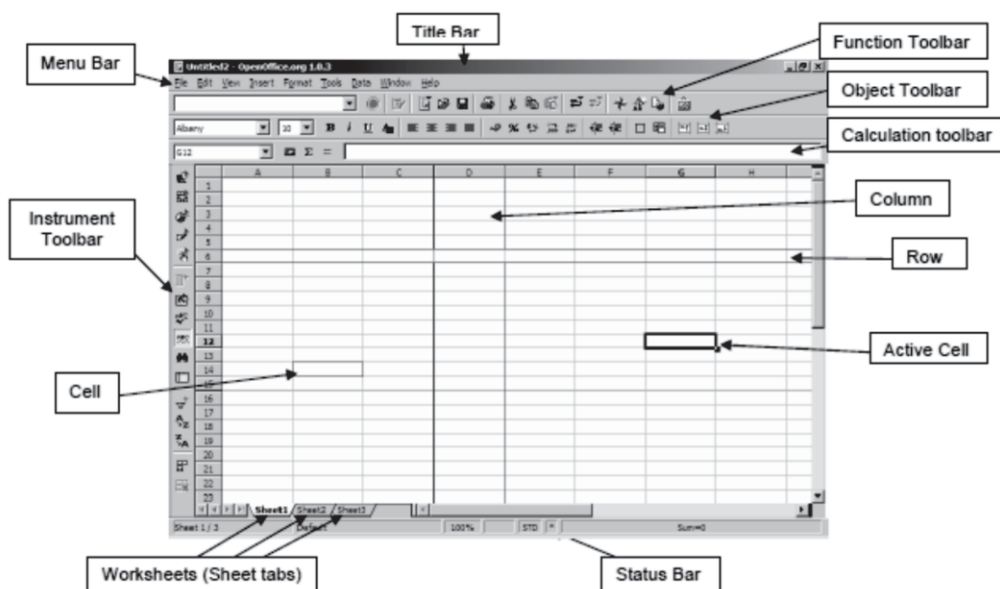
A spreadsheet is the computer equivalent of a paper ledger sheet. It consists of a grid made from columns and rows. Once you have the formulas set up, you can change the variables that are called from the formula and watch the answers change. Spreadsheets are instantly updated if one of the entries is changed. 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, OpenOffice.org 1.0.3, Open Office Calc.**

### Open Office Calc (Spreadsheet) Basics

The picture below shows the Calc screen.





The **Menu toolbar** contains the main menus of the Calc module. The **Function toolbar** provides user access to function icons such as **Open, Save, Copy, Cut, Paste** and other common tasks in Open Office. The **Object toolbar** consists of a set of tools that are specific to calculation and cell formatting (number format, text alignment, borders). Finally, the **Calculation toolbar** is intended for the entry of formulae necessary for your calculations, and also shows you the position of the cursor within the spreadsheet.

The spreadsheet is represented as a grid comprising cells, with each cell bearing a unique reference.

### [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 \(e.g. A1:G5\)](#)

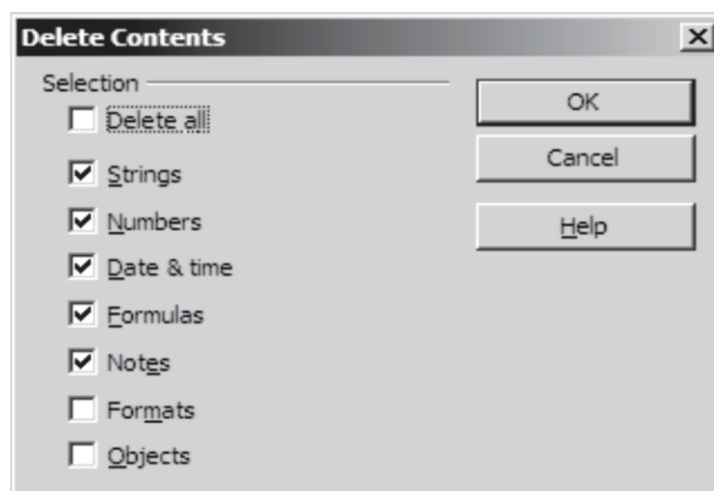
1. On the first cell of the range (e.g. A1), click and hold the left mouse button while dragging the mouse to the last cell of the range (e/g. G5).
2. All cells will turn black except the first cell will remain white.

#### [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.

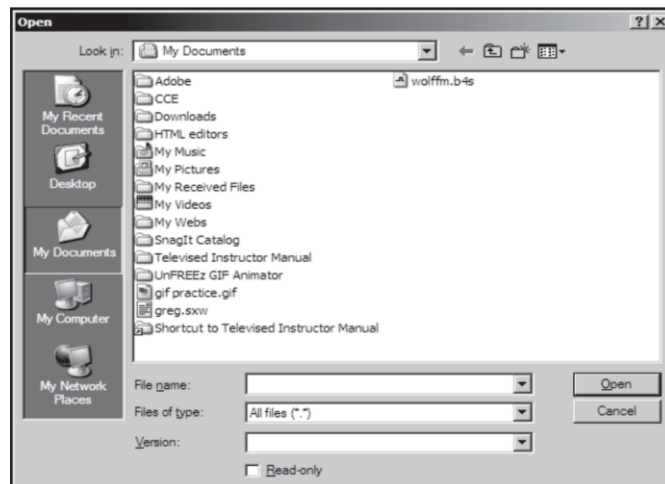


## Opening and Saving a File

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.

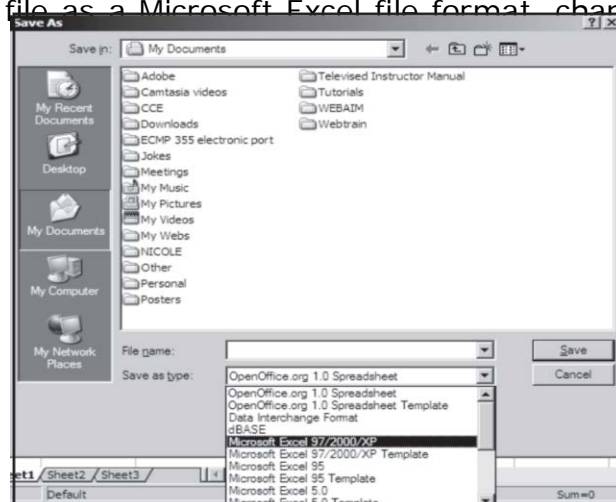
### Opening a File

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.



## Cut, Copy, Paste

You can perform four main actions on a selected text: *copying, cutting, deleting, and formatting*.

### Cut and Paste



This allows the user to move selected text so that it can be placed somewhere else in the document or in another document.

1. Select cell(s) that contain the information you wish to cut (move).

2. Click the **Cut**  button on the toolbar or **CTRL + X** on the keyboard.
3. Click in the cell where you want to paste the information.
4. Click the **Paste**  button on the toolbar or **CTRL+ V** on the keyboard.

### Copy and Paste

This allows the user to create a duplicate (copy) of the selected text to be placed somewhere else in the document or in another document.

1. Select cell(s) you wish to cut (move).
2. Click the **Copy**  button on the toolbar or **CTRL + C** on the keyboard.
3. Place the cursor where to place the cut cell(s).
4. Click the Paste  button on the toolbar or **CTRL+ V** on the keyboard.


### Bold, Italics, Underline

1. Select the cell(s) you wish to format.
2. Click on the attribute icon you wish to apply such as **B** *i* U on the Toolbar. These buttons transform the selected text (in sequence from left to right) into Bold, Italics, and Underlined.
3. With the cell(s) still selected, click the button again to turn the feature off.


### Cell Alignment

1. Select the cell you wish to change.
2. Click on one of the alignment icons  on the function toolbar to modify the text alignment (left, center, right, or justified).



### Font Color

1. Select the cell (s) you wish to change.
2. Click the font color button on  the toolbar and pick the color of your choice.

### Background (highlighting) Color

1. Select the cell (s) you wish to change.
2. Click the background color button  on the toolbar and pick the color of your choice.

### Undo/Redo Buttons

If you perform an action that does not give you the desired result, you can use the Undo button  to reverse the last action. Likewise, the Redo button  can be used to Redo an action that has been undone.


### 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.

### Other Formatting Hints

The 'Cell Attributes' window (click Format, Cells) below includes other tabs for cell formatting (e.g. **Fonts, Font Effects, Alignment, etc**). The function toolbar also contains some of these formatting functions.

# Chapter-2

## 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)

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, satellite links, and data packet carrier services.

#### VPN (Virtual Private Network)

VPN uses a technique known as tunneling to transfer data securely on the Internet to a remote access server on your workplace network. Using a VPN helps you save money by using the public Internet instead of making long-distance phone calls to connect securely with your private network. There are two ways to create a VPN connection, by dialing an Internet service provider (ISP), or connecting directly to Internet.

### 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.

### The advantages of Internet

**1) 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. 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** A wide variety of entertainment including video games, music, movies, chat room, news and others can be accessed through the Internet.

**4) 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.

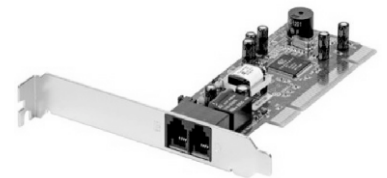
## Web Browsers

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. Hyperlinks present in resources enable users easily to navigate their browsers to related resources. A web browser can also be defined as an application software or program designed to enable users to access, retrieve and view documents and other resources on the Internet. The major web browsers are Firefox, Google Chrome, Internet Explorer, Opera, and Safari.

## Types of Internet Access

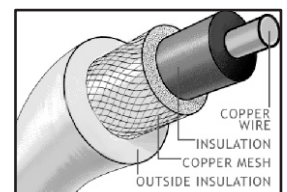
### 1) PCI modem

It is both economical and slow and it is also called dial-up access. If you connect the modem, you get internet but as it uses the analogue telephone line, if you surf on the internet, nobody can call you because the line is busy. Using a modem connected to your PC which is very cheap, users connect to the Internet only if you click on the telephone Access Icon and the computer dials the phone number provided by your ISP ( Internet Service Provider ) and connects to the network.



### 2) DSL

DSL or - an 'always on' connection- uses the existing 2-wire copper telephone line connected to the internet and won't tie up your phone like the old modem does. There is no need to dial-in to your ISP as DSL is always on. DSL is called ADSL ( Short for Asymmetric Digital Subscriber Line) for home subscribers.



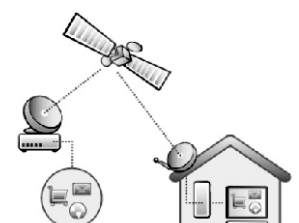
### 3) CABLE

There are two type of cable: Coaxial and optic fibre. The first one is used by cable TV and that is common for data communications . The cross-section of the cable how a single centre solid wire made of copper surrounded by a copper mesh conductor. Fibre-optic cables are strands of a special optical material as thin as a human hair that carry data ( files, videos .. ) over long distances. Now, there is not electrical signal. In Optical fibres data are carried as light signals



### 4) Satellite

IOS short for Internet over Satellite allows a user to access the Internet via a geostationary satellite that orbits the earth. A geostationary satellite is a type of satellite placed at a fixed position above the earth's surface. Because of the large distances between home and satellite, signals must travel from the earth up to the satellite and back again. It causes a slight delay between the request and the answer.



## Services on the web

**A** Web service is a method of communication between two electronic devices over the web. The **W3C** defines a "Web service" as "a software system designed to support **interoperable machine-to-machine** interaction over a **network**".

## Newsgroups

Newsgroups are Internet discussion forums where groups of users with common interests gather to talk about everything from software to comic books to politics. Newsgroups are international in scope, with participants from all corners of the Internet.

## 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.

## 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. E.g

- Bing
- Google

Ÿ

**Yahoo!** Ÿ

**Yebol**

## Chat

Real-time communication between two users via computer. Once a chat has been initiated, either user can enter text by typing on the keyboard and the entered text will appear on the other user's monitor. Most networks and online services offer a chat feature.



## Chapter-3

# Programming in C

### Introduction

#### 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. 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.

C seems so popular is because it is reliable, simple and easy to use.

#### 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 table below shows the structure of a C program:

```
main() function1() function2()
{ { {statement1; statement1; statement1;.....; .....; .....; } } }
```

The following is a simple C program that prints a message '**Hello, world**' on the screen:

```
#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. Keywords serve as basic building blocks for program statements. The list 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. Both upper and lower case letters are permitted, although lower case letters are commonly used.

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

The underscore character is also permitted in identifiers.

#### Constants

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

##### Integer Constants

An "integer constant" is a decimal (base 10), octal (base 8), or hexadecimal (base 16) number that represents an integral value. Use integer constants to represent integer values that cannot be changed.

##### Real Constants

Integer numbers are inadequate to represent quantities that vary continuously, such as distances, heights, temperatures, and so on. These quantities are represented by number containing fractional parts like 17.548.

##### 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 has a concept of 'data types' which are used to define a variable before its use. The definition of a variable will assign storage for the variable and define the type of data that will be held in the location.

- |           |          |
|-----------|----------|
| 1. Int    | 2. Float |
| 3. Double | 4. Char  |

## Variables

A variable is just a named area of storage that can hold a single value (numeric or character). The Programming language C has two main variable types

- Local Variables
- Global Variables

## 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. They usually form a part of the mathematical or logical expressions.

### 1. Arithmetic Operators

C provides all the basic arithmetic operators. 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:  $a - b$ ,  $a + b$ ,  $a/b$ ,  $a * b$

### 2. 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. These comparisons can be done with the help of relational operators. An expression such as:  $a < b$  or  $1 < 20$  containing a relational operator is termed as a relational expression.

### 3. 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.

### 4. Assignment operators

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

### 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

The C increment operator in both prefix or postfix contexts is to add 1 to a variable. But the expression ++variable increments variable before its value is used, whereas variable++ increments variable after its value has been used.

### 6. 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.

### 7. Bitwise operator

C provides six bitwise operator 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

When we want 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 and are known as control or decision-making statements.

### If statement

The **if** statement is used to control the flow of execution of statements. It takes the following form:

**if** (test 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.

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 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) } statement-
s
```

### Example of if-else statement

```
#include <stdio.h>
int main()
{ float temp;
printf("What is the temperature outside?");
scanf("%f",&temp);
if(temp < 65)
{ printf("My but it's a bit chilly out!\n"); }
else if(temp >= 80)
{ printf("My but it's hot out!"); }
else
{ printf("My how pleasant!"); }
return(0); }
```

## 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:

## LOOPING CONTROL STRUCTURE

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 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 general form of the for loop is:

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

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.

### The do...while Statement

The do-while statement lets you repeat a statement or compound statement until a specified expression becomes false.

iteration-statement:

```
do statement while ( expression ) ;
```

The expression in a do-while statement is evaluated after the body of the loop is executed.

Therefore, the body of the loop is always executed at least once.

The expression must have arithmetic or pointer type. Execution proceeds as follows:

1. The statement body is executed.
2. Next, expression is evaluated. If expression is false, the do-while statement terminates and control passes to the next statement in the program. If expression is true (nonzero), the process is repeated, beginning with step 1.

Consider following segment of a program: #include <stdio.h> #include <conio.h>

```
void main()  
{ int x = 5;  
  int i = 0;  
  do{ i++;  
    printf("%d\n", i); } while(i < x); }
```

### The while Statement

The while statement lets you repeat a statement until a specified expression becomes false.

iteration-statement:

while ( expression ) statement

The expression must have arithmetic or pointer type. Execution proceeds as follows:

1. The expression is evaluated.
2. If expression is initially false, the body of the while statement is never executed, and control passes from the while statement to the next statement in the program.

If expression is true (nonzero), the body of the statement is executed and the process is repeated beginning at step 1.

Program to demonstrate while loop:

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a; clrscr(); a=1; while(a<=5)
{ printf("\n Hello");
a+=1 // i.e. a = a + 1 }
getch(); }
```

## ARRAYS

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.

An array is a data structure of multiple elements with the same data type. Array elements are accessed using subscript. The valid range of subscript is 0 to size -1.

### Declaration of Array

```
int arr[10];
```

### Array Initialization

Initializing of array is very simple in c programming. The initializing values are enclosed within the curly braces in the declaration and placed following an equal sign after the array name. Here is an example which declares and initializes an array of five elements of type int. Array can also be initialized after declaration. Look at the following C code which demonstrate the declaration and initialization of an array.

```
int myArray[5] = { 1, 2, 3, 4, 5}; //declare and initialize the array in one statement
int studentAge[4];
studentAge[0]=14;
studentAge[1]=13;
studentAge[2]=15;
studentAge[3]=16;
```

The program is given below:

**Example:** Ten integers of an array are scanned the scan function and

```
printed */#include <stdio.h>
main()
{ int s1[10];
int i;
printf("Enter 10 integers \n");
for(i=0; i<=9; i++)
{ scanf("%d", &s1[i]); }
printf("you have entered: \n");
for(i=0; i<=9; i++)
{ printf("%d\r", s1[i]); }
```

#### Result of the Program

Enter 10 integers

1 2 3 4 5 6 7 8 9 0

you have entered:

1  
2  
3  
4  
5  
6  
7  
8  
9  
0

## Multi-dimensional Arrays

### Multidimensional Arrays

A multi-dimensional array of dimension n (i.e., an n-dimensional array or simply n-D array) is a collection of items which is accessed via n subscript expressions. Multidimensional arrays can be described as "arrays of arrays".

Example:

```
#include <stdio.h>
#include<conio.h>
void main(){
int  a[3][2];
int  i,j;
for(i = 0;i<3;i++){
for(j=0;j<2 ;j++) {
scanf("%d",&a[i][j]); }}
for(i = 0;i<3;i++){
for(j=0;j<2;j++) {
printf("value in array %d\n",a[i][j]); }}
```

## String in C

A string is a collection of characters. Strings are always enclosed in double quotes as "string\_constant". Strings are used in string handling operations such as,

- Counting the length of a string.
- Comparing two strings.
- Copying one string to another.
- Converting lower case string to upper case.
- Converting upper case string to lower case.
- Joining two strings.
- Reversing string.

## Declaration :

The string can be declared as follow :

Syntax:

```
char string_nm[size];
```

Example: char name[50];

## Read Strings :

To read a string, we can use scanf() function with format specifier %s.

```
char name[50];
```

```
scanf("%s",name);
```

The above format allows to accept only string which does not have any blank space, tab, new line, form feed, carriage return.

## Write Strings :

To write a string, we can use printf() function with format specifier %s.

```
char name[50];
```

```
scanf("%s",name);
```

```
printf("%s",name);
```

## String Functions

strcmp - Strcmp compares two strings and returns an integer indicating the difference between the strings. If the strings match, then the number returned is 0. E.g

```
int strcmp(string1, string2);
```

strcat - Strcat combines two strings and returns a pointer to the destination string. In order for this function to work, you must have enough room in the destination for both strings. E.g

```
char *strcat(string1, string2);
```

strcpy - Strcpy copies one string to another. The destination must be large enough to accept the contents of the source string. E.g.

```
strcpy(string1, string2);
```

strlen - Strlen returns the length of a string.

```
char fname[30] = {"Bob"};
```

```
int length = strlen(fname);
```

index - Index returns a pointer to the first instance of a character in a string. If the character cannot be found in the string, a NULL pointer is returned.

## INTRODUCTION TO POINTERS

Pointer is a variable just like other variables of c but only difference is unlike the other variable it stores the memory address of any other variables of c. This variable may be type of int, char, array, structure, function or any other pointers.

In c programming every variable keeps two type of value.

1. Contain of variable or value of variable.
2. Address of variable where it has stored in the memory.

## Pointer Declaration

A pointer variable contains the memory location of another variable. You begin the declaration of a pointer by specifying the type of data stored in the location identified by the pointer. The asterisk tells the compiler that you are creating a pointer variable. Finally you give the name of the pointer variable. The pointer declaration syntax is as shown below:

type \*variable name

e.g: int \*ptr;

float \*string;

## Pointers and Arrays

Array of function means array which content is address of function and pointer to array of function means pointer is pointing to such array.

In other word we can say pointer to array of functions is a pointer which is pointing to an array which contents are pointers to a function.

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

```
int display();
```

```
int(*array[3])();
```

```
int(*(*ptr)[3])();
```

```
int
```

```
main(){array[0]=display;array[1]=getch;ptr=&array;printf("%d",(**ptr)());(*(*ptr+1))();  
return 0;}
```

```
int display(){int x=5;return x++;}
```

Output: 5

## Function

The function is a self contained block of statements which performs a coherent task of a same kind.

C program does not execute the functions directly. It is required to invoke or call that functions. When a function is called in a program then program control goes to the function body. Then, it executes the statements which are involved in a function body. Therefore, it is possible to call function whenever we want to process that functions statements.

Types of functions :

There are 2(two) types of functions as:

### 1. Built in Functions :

These functions are also called as 'library functions'. These functions are provided by system. These functions are stored in library files. e.g.

- scanf()

•

printf() •

strcpy •

strlwr •

strcmp •

strlen

- strcat

### 1. User Defined Functions :

The functions which are created by user for program are known as 'User defined functions'.

Syntax:

```
void main(){// Function prototype
    <return_type><function_name>([<argu_list>]);
    // Function Call
    <function_name>([<arguments>]);}
// Function definition
<return_type><function_name>([<argu_list>]);
{<function_body>;}
```

#### Function Definition

```
return-type function-name(parameters) {
declarations
statements
return value;}
```

- Return-type - type of value returned by function or void if none
- *Function-name*- unique name identifying function
- Parameters - comma-separated list of types and names of parameters
- *value* - value returned upon termination (not needed if return-type void)

The list of parameters is a declaration on the form

type 1 par 1, ..., type n par n

and represents external values needed by the function. The list of parameters can be empty.

All declarations and statements that are valid in the main program can be used in the function definition, where they make up the function body.

### Structure and Unions

**Structure** is user defined data type which is used to store heterogeneous data under unique name. Keyword 'struct' is used to declare structure.

The variables which are declared inside the structure are called as 'members of structure'.

Syntax:

```
struct structure_nm
{<data-type> element 1;
  <data-type> element 2; - - - -
  - - - - -
  - - - - - <data-type>
  element n;}struct_var;
```



## Structure Elements

The structure variable declaration is of no use unless the variables are assigned values. Here each member has to be specifically accessed for each structure variable. For instance to assign the account number for variable a1 we have to specify as follows:

```
a1.number = 0001;
```

There is a dot operator in between the structure variable name and the member name or tag.

Suppose you then want to assign account No. 2 to a2, it can be assigned as follows:

```
a2.number = 2;
```

If you want to know the address where 'a2' number is stored you can use

```
printf("%u",&a2.number)
```

This is similar to other data types. The structure is a complex data type, and have to indicate which structure variable to which the number belongs, as otherwise the number is common to all the structure lru variable such as a1, a2, a3, etc. Therefore, it is necessary to be specific. Assuming that you want to get the value from the keyboard, you can use **scanf** as given below:

```
scanf(" %u ", &a1.number);
```

## Array of Structures

Since structures are data types that are especially useful for creating collection items, why not make a collection of them using an array? Let us now modify our above example object1.c to use an array of structures rather than individual ones.

The program is given below:

```
/*program to demonstrate structures*/
#include<stdio.h>
main()
{ struct account
{ unsigned number;
char name[15];
int balance; }a[5];
int i;
for(i=0; i<: 4; i++)
{ printf("A/c No: =\t Name: :\t Balance: :\n");
scanf(" %u%s%d", &a [1] .number, a [i] .name, &a [i] .balance); }
for(i = 0; i<=4; i++)
{ printf("A/c No: =%u\t Name:= %s\t Balance: = %d\n, a[i] .number, a[i] .name, a
[i].balance); } }
```

## Unions

Unions and Structures are identical in all ways, except for one very important aspect. Only one element in the union may have a value set at any given time. Everything we have shown you for structures will work for unions, except for setting more than one of its members at a time. Unions are mainly used to conserve memory. While each member within a structure is assigned its own unique storage area, the members that compose a union share the common storage area within the memory. Unions are useful for application involving multiple members where values are not assigned to all the members at any one time.

## Difference between Structure and Unions

The difference between structure and union in c are:

1. Union allocates the memory equal to the maximum memory required by the member of the union but structure allocates the memory equal to the total memory required by the members.
2. In union, one block is used by all the members of the union but in case of structure, each member has its own memory space.

## Difference in Their Usage

While structure enables us treat a number of different variables stored at different in memory, a union enables us to treat the same space in memory as a number of different variables. That is a Union offers a way for a section of memory to be treated as a variable of one type on one occasion and as a different variable of a different type on another occasion.

## Difference Between Structure and Union

Structure	Union
i. Access Members	
We can access all the members of structure at anytime.	Only one member of union can be accessed at anytime.
ii. Memory Allocation	
Memory is allocated for all variables.	Allocates memory for variable which variable require more memory.
iii. Initialization	
All members of structure can be initialized.	Only the first member of a union can be initialized.
iv. Keyword	
'struct keyword is used to declare structure.	'Union' keyword is used to declare union.
v. Syntax	
<pre>struct struct_name {     structure element 1;     structure element 2;     -----     -----     structure element n; } struct_var_nm;</pre>	<pre>union union_name {     union element 1;     union element 2;     -----     -----     union element n; } union_var_nm;</pre>
vi. Example	
<pre>struct item_mst {     int rno;     char nm [50]; }</pre>	<pre>union item_mst {     int rno;     char nm [50]; }</pre>
z it;	z it;

## File Management in C

File management in C, File operation functions in C, Defining and opening a file, Closing a file, The getw and putw functions, The printf & scanf functions, Random access to files and seek function. C supports a number of functions that have the ability to perform basic file operations, which include: 1. Naming a file 2. Opening a file 3. Reading from a file 4. Writing data into a file 5. Closing a file

Real life situations involve large volume of data and in such cases, the console oriented I/O operations pose two major problems

It becomes cumbersome and time consuming to handle large volumes of data through terminals. The entire data is lost when either the program is terminated or computer is turned off therefore it is necessary to have more flexible approach where data can be stored on the disks and read whenever necessary, without destroying the data. This method employs the concept of files to store data.

## Opening and Closing Files

Any file has to be opened for any further processing such as reading, writing or appending, i.e. writing at the end of the file. The characters will be written or read one after another from beginning to end, unless otherwise specified. We have to open the file and assign the file pointer to take care of further operations. Hence we can declare:

```
FILE *fp;
```

```
fp = fopen ("filename", "r");
```

The filename is the name of the file which you want to open. You must give the path name correctly so that the file can be opened. "r" indicates that the file has to be opened for reading purposes.

```
fp = fopen ("Ex1.C", "r");
```

 will enable opening file *Ex1 C*.

Therefore, the arguments to **fopen** are the name of the file and mode character. Obviously w is for write, a for append, i.e. adding at the end of the file. If these characters are specified, the operations as indicated can be performed after opening the file. It is therefore essential to indicate the operations to be performed before opening the file. When the file is opened in the w "mode", the data will be written to the file from the beginning. This means that if the named file is already in existence, the previous contents will be lost due to overwriting. If the file does not exist, then a file with the assigned name will be opened. When append mode is specified, the writing will start after the last entry or, in other words, previous contents of the file will be preserved.

*FILE* provides the link between the operating system and the program currently being executed.

*FILE* is a structure containing information about the corresponding files, including information such as:

- the location of the file.

- the location of the buffer.

- the size of the file.

After the command is executed in the read mode, the file will be loaded into the buffer if it is present. If the file is absent, or the file specification is not correct, then the file will not be opened.

If the opening of the file is successful, the pointer will point to the first character in the file, and if not, NULL is returned, meaning that the access is not successful.

The **fopen** function returns a pointer to the starting address of the buffer area associated with the file and assigns it to the pointer, *fp* in this case.

After operations are completed the file has to be closed.

The syntax for closing file is as given below:

`fclose (filepointer)`

**fclose** also flushes or empties the buffer.

The function `fputc` performs putting one character into a file. If for every **fputc**, the computer prints a character to a file then it will get tired.

Therefore it collects all characters to be written on to a file in the buffer. When the buffer is full or when **fclose** is executed the buffer is emptied by writing to the hard disc drive in the assigned file name.

### Formatted I/O with Files

We are familiar with reading and writing. So far we were reading from and writing to standard input/output. Therefore we used functions for formatted I/O with *studio* such as **scanf( )** and **printf( )**. We also used unformatted I/O such as **getch**, **putch** and other statements. When dealing with files, there are similar functions for I/O. The functions **getc()**, **fgetc()**, **fputc()** and **putc()** are unformatted file I/O functions similar to **getch** and **putch**. We will consider formatted file operations in this section. When it pertains to standard input or output we use **scanf( )** and **printf()**. To handle formatted I/O with files we have to use **fscanf()** and **fprintf()**. We can write numbers, characters etc. to the file using **fprintf**. This helps in writing to the file neatly with a proper format. In fact any output can be directed to a file instead of the monitor. However, we have to indicate which file we are writing to, by giving the file pointer. The following syntax has to be followed for **fprintf()**:

**fprintf**(filepointer, "format specifier", variable names);

We are only adding the file pointer as one of the parameters, before the format specifier. This is similar to **sprintf** which helps in writing to a buffer. In case of **sprintf**, buffer was a pointer to a string variable. Here, instead of a pointer to a string variable, a pointer to a file is given in the **fprintf** statement. Like the string pointer in **sprintf**, the file pointer should have been declared in the function and should be pointing to the file.

**Caution** : Before writing to a file, the file must be opened in the write mode.

You can declare the following:

```
FILE *fp;
```

```
fP = fopen("filename", "wb");
```

You have to write `wb` within double quotes for opening a file for writing in binary mode.

Therefore **fopen** searches the named file. If the file is found, it starts writing. Obviously the previous contents will be lost. If a file is not found, a new file will be created. If unable to open a file for writing, `NULL` will be returned.

We can also append data to the file after the existing contents. In this manner we will be able to preserve the contents of a file. However, when you open the file in the append mode, and the file is not present, a new file will be opened. If a file is present then writing is carried out from the current end of the file. After writing is completed either in write mode or append mode, a special character will be automatically included at the end of the text in case of text files. (In case of binary files no special character will be appended). This can be read back as EOF.

Usually it is `-1`, but it is implementation dependent. Hence it is safer to use EOF to check the end of the text files. Let us look at a program to write numbers to a binary file using **fprintf** and then read from the file using **fscanf**. It is given in below example:

table below:

## Writing the file

```
/*digits to a binary file and then reading*/
#include<stdio.h.>
main()
int alpha,i;
FILE *fp;
fp=fopen("ss.doc", "wb");
if(fp==NULL)
printf("could not open file\n")
else
{ for(i=0; i<=99; i++)
fprintf(fp, "%d", i);
fclose(fp);
/*now read the contents*/
fp=fopen("ss.doc", "rb");
for(i=0; i<100; i++)
{ fscanf(f.p, "%d", &alpha);
printf("%d", alpha);}
fclose(fp); }}
```

## Result of program

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
```

## Explanation

(a) The file *ss.doc* is opened in the binary mode for writing. If the opening of the file was not successful, the message will be displayed and program execution will stop. If successful, the program will enter the **else** block. Numbers 0 to 99 are generated one after another and written then and there to the file using the **fprintf** function. There should be space before %d as shown in **fprintf**, otherwise the program may not work.

(b) The file is closed using **fclose**.

(c) Now the same file is opened for reading in binary mode.

(d) Next the text is scanned using **fscanf**, one at a time, and written on the monitor using simple **printf**. The difference between **scanf** and **fscanf** is the specification of the file pointer before the format specifier.

(e) After reading, the file is closed.

The result of the program is read from the file *ss.doc* and printed on the monitor. In all programs involving files, a similar check to see that file opening was successful should be made. For the sake of improved readability, this statement has been skipped in the rest of the programs.

Let us look at one more example of writing, appending and then reading one integer at a time with the help of for loops. Look at the program below:

**Example:** /\* Writing, then appending digits to a file and then reading \*/  
#include<stdio.h>main()

```

{int alpha, i;
FILE *fp;
fp=fopen("ss.doc", "wb");
for(i=0; i<20; i++)
fprintf(fp, "%d", i);
fclose(fp);
fp=fopen("ss.doc", "ab");
for(i=20; i<100; i++)
fprintf(fp, "%d", i);
fclose(fp);
/*now read the contents*/
fp=fopen("ss.doc", "rb");
for(i=0; i<100; i++)
{fscanf(fp, "%d", &alpha);
printf("%d", alpha);}
fclose(fp);}

```

A binary file is opened in the write mode, and digits from 0 to 19 are written on to the file. The file is then closed using **fclose**. The same file is opened in the append mode again, and numbers from 20 to 99 are appended to the file. After the file is closed, the file is opened in the read mode. The contents of the file are read using **fscanf** and written to the monitor. Remember to leave a space before %d in **fprintf**, as otherwise you may have a problem. The file is closed again. We have used the same file pointer, since at any time only one file is in use. If more than one file is to be kept open simultaneously, it may call for multiple pointers.

### Reading Characters from a File

After having worked with formatted I/O, let us now look at unformatted I/O. If you want to read a character from the file, you can use the **getc( )** or **fgetc** functions. If *alpha* is the name of the character variable, you can write.

```
alpha = fgetc(fp);
```

This means the character pointed to by *fp* is read and assigned to *alpha*. You can also go to the help screen of the "C" language system to get more details, as well as search for help on any of the library functions. The help screen gives the syntax of the functions, and also provides examples in which the function or command is used. Even after reading this book or any other book on "C" you will not be able to use all the functions. Hence, the best way is to take the help from help screen whenever other functions are to be used.

Now **fgetc( )** reads the character pointed to by *fp*. It then increments *fp* so that *fp* points to the next character. We can keep on incrementing *fp* till the end of file, i.e. end of data is reached.

When a file is created in the text mode the system inserts a special character at the end of the text. Therefore while reading a file, when the last character has been read and the end of the file is reached, EOF is returned by the file pointer. The following program reads one character at a time till EOF is reached, from an already created text file *ss.doc*. The program is implemented using **do..while** statement.

```

/*Example - reading characters from a file */
#include <stdio.h>

```

```

main()
{ int alpha;
FILE *fp;
fp=fopen("ss.doc", "r");
do
{ alpha=fgetc(fp);
putchar(alpha);
} while(alpha!=EOF);
fclose (fp); }

```

If no change has been made in the file. If we were to read from a binary file EOF may not be recognized. Therefore, a counter can be set up to read a predefined number of characters, as given in the previous examples.

### File Copy

File copy can be achieved by reading one character at a time and writing to another file either in the write mode or append mode. Here it is proposed to read from a file and write to two different files, one in the write mode and another in append mode. This means we have to open 3 files in the following manner:

```
FILE *fr, *fw, *fa;
```

You can assign three file pointers as given above. Three files are then opened. You can use any name for the file pointers and there may be as many file pointers as the number of files to be used.

The program is given below:

```

/*Example - reading from file ss*/
/*writing to file ws and appending to file as, all at a time*/
#include<stdio.h>
main()
{ int alpha;
FILE *fr, *fa;
fr=fopen("ss.doc", "r");
fw=fopen("ws.doc", "w");
fa=fopen("as.doc", "a");
do
{ alpha=fgetc(fr);
fputc(alpha, fw);
fputc(alpha, fa);
putchar(alpha);
} while(alpha!=EOF);
fclose (fr);
fclose(fw);
fclose(fa);

```

After opening the three files, *alpha* gets the character which is written to both the files using **fputc**, and the character is also displayed on the screen. This is continued till EOF is received in *alpha* from ss.doc, the source file. Finally the files are closed. Verify that our program has worked alright. Since we are also writing to the monitor, in addition to writing and appending to files the program output appears as follows:



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32  
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 56 57 58 59 60 61 62  
63 64 65 67 68 69 70 71 72 73 74 75 76 78 79 80 81 82 83 84 85 86 87 89 90 91 92 93 94  
95 96 97 98 99

There are some more mode specifiers with *fopen* like *r+*, *w+* and *a+* which are given in the table below:

Mode Specifier	Purpose
<i>r+</i>	Open already existing file for reading and writing.
<i>w+</i>	Open a new file for writing as well as reading.
<i>a+</i>	Open already e>existing file for appending and reading.

### Line Input/Output

We have discussed writing to and reading from a file, one character at a time, using both unformatted and formatted I/O for the purpose. We can also read one line at a time. This is enabled by the **fgets** function. This is a standard library function with the following syntax:

```
char * fgets(char * buf, int max line, FILE *fp)
```

**fgets** reads the next line from the file pointed to by *fp* into the character array *buf*. The line means characters upto *maxline* - 1, i.e. if *maxline* is 80, each execution of the function will permit reading of upto 79 characters in the next line. Here 79 is the maximum, but you can even read 10 characters at a time, if it is specified.

```
fgets(alpha, 10, fr);
```

Here *alpha* is the name of buffer from where 10 characters are to be read at a time. The file pointer *fr* points to the file from which the line is read, and the line read is terminated with NULL. Therefore it returns a line if available and NULL if the file is empty or an error occurs in opening the file or reading the file.

The complementary function to **fgets** is **fputs**. Obviously **fputs** will write a line of text into the file.

The syntax is as follows:

```
int fouts(char *buf , file *fP)
```

The contents of array *buf* are written onto the file pointed to by *fp*. It returns EOF on error and Zero otherwise. Note that the execution of **fgets** returns a line and **fputs** returns zero after normal operation.

**Did you know:** The functions **gets** and **puts** were used with **stdio** whereas **fgets** and **fputs** operate on files.

We can write a program to transfer 2 lines of text from buffer to a file and then read the contents of the file to monitor. This is shown in example:

```
/* Example: Writing and reading lines on files */
```

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

```
#include<string.h>
```

```
main()
```

```
{ int i;
```

```
char alpha[80];
```

```
FILE *fr, *fw;
```

```

fw=fopen("ws.doc", "wb");
for(i=0; i<2; i++)
{ printf("Enter a line upto 80 characters\n");
gets(alpha);
fputs(alpha, fw); }
fclose(fw);
fr=fopen("ws.doc", "rb");
while
{ fgets(alpha, 20, fr)!=NULL)
puts(alpha);
fclose(fr); } }

```

Note carefully the **fgets** statement. Here alpha is the buffer with a width of 80 characters. Each line can be upto 80 characters, and two lines are entered through alpha to ws.doc. Later on 20 characters are read into alpha at a from same file till NULL is returned. Result of program is given below

```

Enter a line upto 80 characters
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Enter a line upto 80 characters
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
a aa aaa aaaa a aaaaaa aaaaaaaaaa aaaaaaaaaaaaa
aaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

```

More than 20 numbers of a and b were written on to the file. However, since we have specified reading 20 characters at a time, the output appears in 6 lines. Had we specified reading more characters at a time, the number of reads would have reduced. You can try this yourself. Thus you can read and write one line at a time.

### Keywords

**Buffer:** It is also memory which is used to store data temporarily without the knowledge of the user.

**Binary file:** A file in which data stored in Binary form and the storage space will be equal to the number of bytes required for storage of various data types.

**Data file:** A data file is a computer file which stores data for use by a computer application or system.

**File pointer:** It is a pointer to a file, just like other pointers to arrays, structures, etc.

**File copy:** A file copy can be achieved by reading one character at a time and writing to another file either in the write mode or append mode.

**Text files:** A file in data is stored as alpha numeric characters.

# Chapter-4

## Introduction to Software Engineering

### System Development Life Cycle

There are standard phases and processes, however, that all system development projects should follow, regardless of environment and tools. The material in this section is organized according to a generic system development lifecycle. While no two development efforts are exactly alike, all projects should progress through the same six phases:

**1. System Initiation** –This validation effort provides the Project Team with the basis for a detailed schedule defining the steps needed to obtain a thorough understanding of the business requirements and an initial view of staffing needs. In addition, a high level schedule is developed for subsequent system development lifecycle phases.

**2. System Requirements Analysis** – In which the needs of the business are captured in as much detail as possible. The Project Manager leads the Project Team in working with the Customers to define what it is that the new system must do. By obtaining a detailed and comprehensive understanding of the business requirements, the Project Team can develop the Functional Specification that will drive the system design. Dev

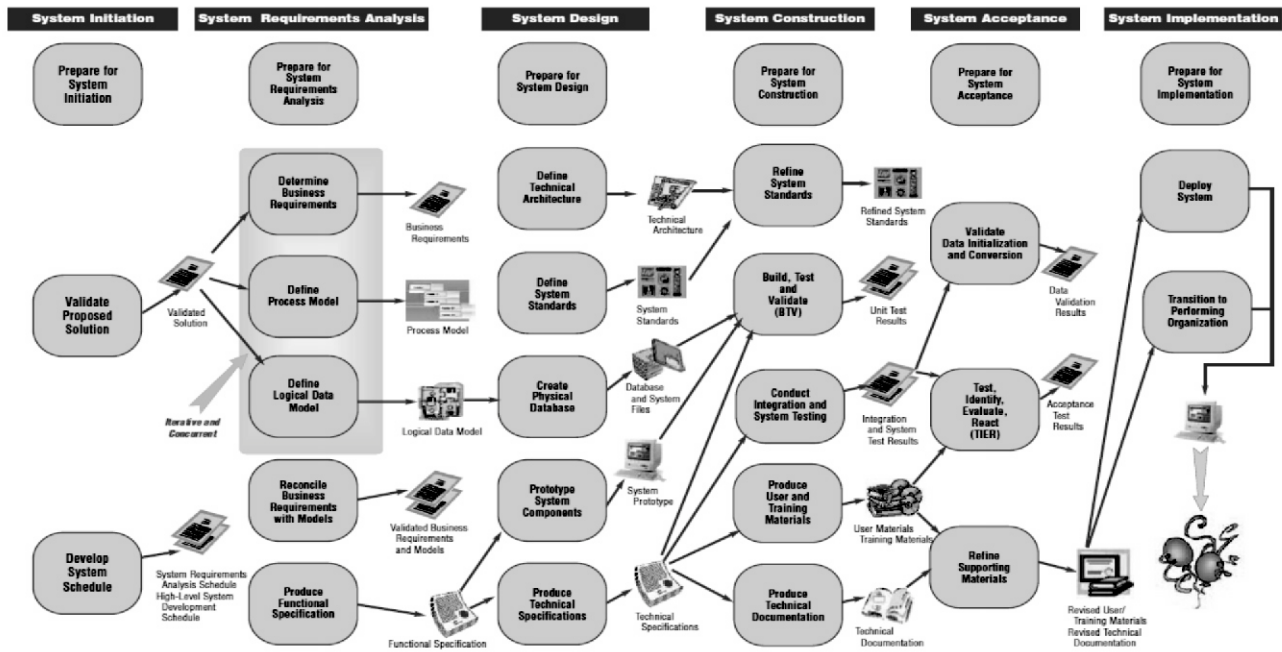
**3. System Design** – which builds upon the work performed during System Requirements Analysis, and results in a translation of the functional requirements into a complete technical solution. This solution dictates the technical architecture, standards, specifications and strategies to be followed throughout the building, testing, and implementation of the system. The completion of System Design also marks the point in the project at which the Project Manager should be able to plan, in detail, all future project phases.

**4. System Construction** – throughout which the Project Team builds and tests the various modules of the application, including any utilities that will be needed during System Acceptance and System Implementation. As system components are built, they will be tested both individually and in logically related and integrated groupings until such time as a full system test has been performed to validate functionality. Documentation and training materials are also developed during this phase.

**5. System Acceptance** – during which the focus of system validation efforts shifts from those team members responsible for developing the application to those who will ultimately use the system in the execution of their daily responsibilities. In addition to confirming that the system meets functional expectations, activities are aimed at validating all aspects of data conversion and system deployment.

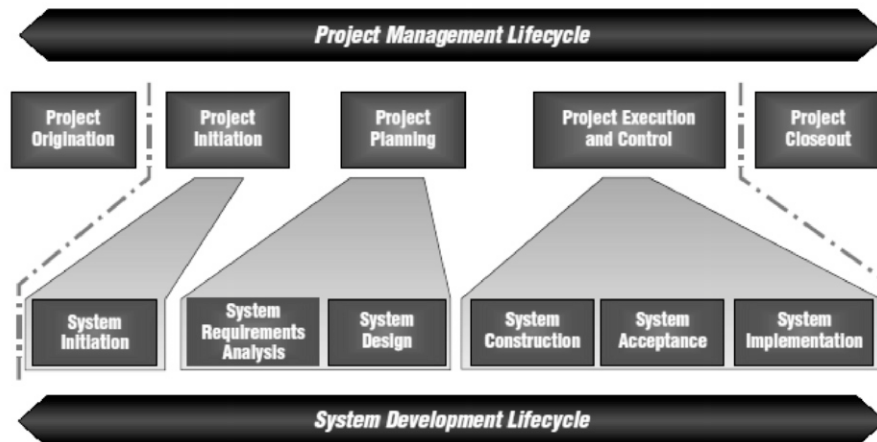
**6. System Implementation** – the final phase of the lifecycle, which comprises all activities associated with the deployment of the application. These efforts include training, installation of the system in a production setting, and transition of ownership of the application from the Project Team to the Performing Organization.

The following diagram illustrates every phase, process and deliverable in the system development lifecycle.



## Mapping the Project Management and System Development Lifecycles

The phases of the system development lifecycle generally align with the phases of the project management lifecycle; however, SDLC phases do not correspond one-to-one with the project management phases. One of the challenges for system development projects is aligning the SDLC with the project management lifecycle. The following diagram demonstrates how the phases of the two lifecycles may be integrated.



The SDLC defined in this section may appear to have characteristics of a classic “waterfall” approach, which assumes that each phase and process is completed and agreed upon before the next phase begins. The reality is, however, that phases generally overlap, with each successive phase introducing changes to the work of the prior phase, resulting in an iterative process.

This SDLC is also consistent with newer techniques for system development, such as Rapid Application Development (**RAD**).

RAD allows users to participate in an iterative design and development process. Conceptually, the project “loops” through the Design, Construction, and Acceptance phases, followed by re-Design, revised Construction, Acceptance, and so on. Project management deliverables such as the Project Scope Statement, Project Schedule, and budget estimates are refined to reflect increasing clarity of scope and requirements with each iteration.

## PHASES OF SYSTEM DEVELOPMENT

### SYSTEM INITIATION

The purpose of **System Initiation** is to validate the Proposed Solution developed during the Project Origination phase of the Project Management Lifecycle, and to estimate the system development effort in greater detail. In this phase, the broad parameters of the new system are defined, and applicable system development activities are identified. Once the overall approach has been confirmed, it is necessary to estimate the effort and resources required for the next phase in elemental detail, and to provide high-level estimates for subsequent phases, to the extent necessary to support the project management lifecycle deliverables and activities of Project Initiation.

#### List of Processes

This phase consists of the following processes:

**Prepare for System Initiation**, where the initial members of the Project Team familiarize themselves with the project's defining documents and plan the activities for the rest of the phase.

**Validate Proposed Solution**, where the original technology direction and system development approach are validated.

**Develop System Schedule**, where a detailed System Requirements Analysis schedule is developed, and a high-level system development schedule is produced.

#### List of Roles

The following roles are involved in carrying out the processes of this phase:

- Project Manager
- Project Sponsor
- Business Analyst
- Technical Lead

### PREPARE FOR SYSTEM INITIATION

#### Purpose

The purpose of **Prepare for System Initiation** is to ensure that the Project Team, and the environment in which it will operate, are ready for successful completion of this phase.

#### Description

In addition to the Project Manager, Business Analyst and Technical Lead roles are required to complete the team for this phase. Environment preparation includes gathering all relevant project and historical documentation, and placing it in the document repository. Any historical data, such as best practices or performance statistics from similar earlier efforts, can also serve to guide the Project Team towards – or away from – certain solutions.

### VALIDATE PROPOSED SOLUTION

#### Purpose

The purpose of **Validate Proposed Solution** is to make sure that the original technology decision and system development approach still represent the optimal solution for the identified business need.

#### Description

Considerable time may have elapsed since the Project Proposal was developed, due to the vagaries of the budget process or to other procedural delays. In the interim, the Performing Organization may have changed its course and the state-of-the-art technology may also have changed significantly.

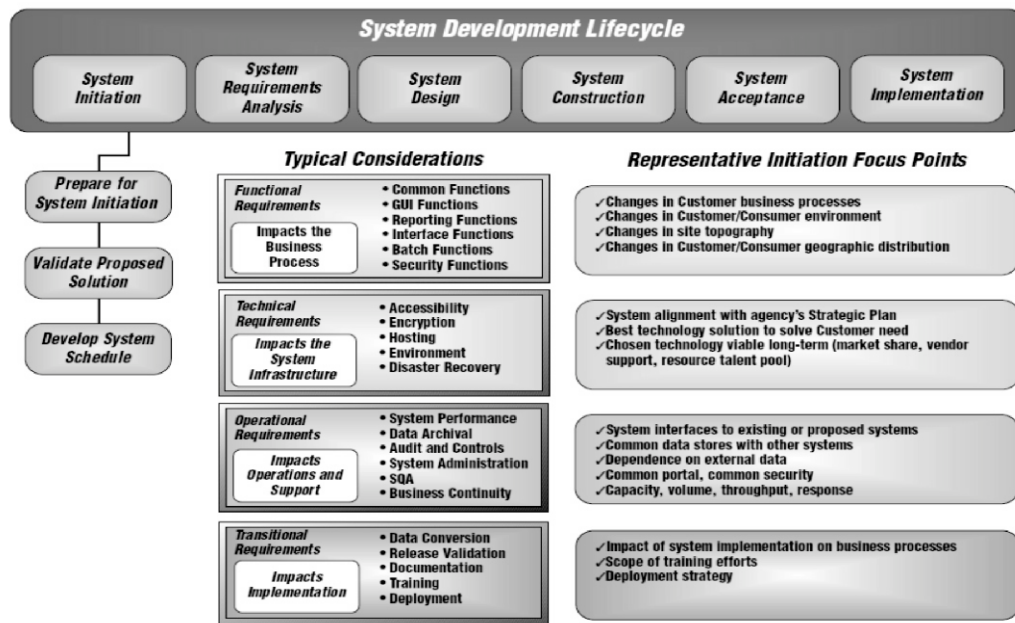
## Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Technical Lead

To validate the Proposed Solution, the Project Team must:

- Understand the agency's current Strategic Plan, and how the new system fits into it;
- Assess the proposed technology solution in view of Customer needs, the Performing Organization's long term technology direction and constraints, and state-of-the-art technology; and
- Confirm feasibility of the proposed system development approach.

## Phases of system development



In deciding whether the proposed technology direction represents an industry trend or a dead end, there are numerous professional journals available by subscription or free on the Web. Once it has been determined that the proposed technical solution fits into the Performing Organization's Strategic Plan, any lingering questions may be resolved through formal reviews, or directed to the Project Sponsor.

Finally, the system development approach needs to be validated against the latest understanding of both the business needs and the technology solution. Certain decisions must be considered even if they cannot yet be made. Among them are:

- Should the system be developed in-house or acquired as a Custom Off-The-Shelf (COTS) solution?
- Are there available resources to develop/customize the system, or is it necessary to contract for additional resources?
- Is the choice of technology platform predicated on the existing environment, or does the system offer an opportunity to upgrade the infrastructure?

## Deliverable

**Validated Solution**– The team should update the original Project Proposal, or recreate the Proposed Solution using the template.

## DEVELOP SYSTEM SCHEDULE

### Purpose

The purpose of **Develop System Schedule** is to create a detailed schedule for System Requirements Analysis and a high-level schedule for the remaining phases.



## Description

After the technical solution has been validated, it is possible to decide how the rest of the System Development Lifecycle will be applied to this particular system development effort. Using a project scheduling tool of choice and referring to the chart of the System Development Lifecycle, the Project Manager should populate the project with the phases and processes and the suggested roles. Then the Project Team, including the Business Analyst and the Technical Lead, should walk through and brainstorm each process, attempting to answer as well as possible the following questions:

1. What are the system development deliverables in this process?
2. What are the tasks necessary to produce this deliverable?
3. Is there a logical way to organize the tasks associated with this process for this system?

## Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Technical Lead

In addition to thinking through the deliverables necessary for implementing the desired functionality, the team should consider the technical, operational and transitional requirements of the system. These additional requirements will influence the definition of necessary tasks and the tasks' order or complexity.

This scheduling process needs to be performed to an elementary level of detail for the next phase, but only at a high level for the subsequent phases.

Deliverables

**System Requirements Analysis Schedule** – Task-level schedule for the System Requirements Analysis phase of the System Development Lifecycle.

**High-Level System Development Schedule** – Process level schedule for the remaining System Development Lifecycle phases.

## Measurements of Success

The success of this phase is measured by how readily the team can perform the next phase. The schedule for System Requirements Analysis should be immediately executable by the team.

The Project Manager can assess how successfully the project is proceeding by utilizing the measurement criteria outlined below. More than one “No” answer indicates a serious risk to the next phase and the eventual success of the system.

Process	Measurements of Success	Yes	No
Prepare for System Initiation	Have you obtained the materials that describe (1) the business needs and benefits, by functional unit; (2) the proposed solution; (3) the reasons that this project and this solution were selected for initiation?		
Validate Proposed Solution	Does the head of the information technology organization (or designate) agree that the system solution fits into the Strategic Plan?		
Develop System Schedule	Have the standard development procedures been customized for the specific system components defined for this system, including functional, technical, operational and transitional requirements?		
	Do you have management commitments for team member availability for the next phase?		
	In the High-Level System Development Schedule, do you know if the effort allocated to system development phases correlate to industry-accepted norms?		