INTERNET AND ITS APPLICATIONS

SYLLABUS

UNIT I

What is Internet - History of Internet - How the Web works? - Web Servers and Clients - Looking at connection as ISP - ISDN - Dial up or Leased Connection - Domain Naming System - Registering our own Domain Name - Intranet - Overview of Web Browsers - hypertext - Hyper Text Markup Language-Basic Components - Formatting the text HTML-URL-Protocol - Server name - Port - Relative URLs Absolute URLs - Linking to other HTML Documents - Linking Inside the same Documents - Linking to other internet servies - File Transfer Protocol (FTP)-Gopher.

UNIT II

List in HTML - Displaing Text in lists - Ordered List - Using Ordered Lists - Using Netscape Extensions - Unordered Lists - Using <UL> Tag - Directory Lists - Defintion Lists - Combining Lists Types - Graphics and Web Pages - Image Format and Drowsers - HTML Tables Aligning Table Elements - Row and Colum Spanning - Netscaps Table - Enhancements Frames in HTML - Frameset Container - HTML Forms - The <Input> Tag - Dynamic Documents - Background Graphhics and colour - Microsoft Internet Extensions - Font Tag Enhancements - Scrolling Marquees.

UNIT III

Data - Steps in Programming Process - Programming Specification - Problem Definition - Requirement Analysis - Desing a Program Model - Determine or correctness of an Algorithm - Code Program - Test and Debug - Debugging - Documentation - Structured Programming Techniques Program Tools - Flow Chart - Why Structured Programs? - Structures - Sequence Program Flow - Decision Structure - Iteration Structure - Tools for Structured Programming - Structure Charts - Pseudo Codes - Strucutred Programming issues - Maintenance - Portability - Readability - Program verification - Modularity - Problem solving - Apporaches - Top - Down Approach - Brute - Force Approach - Testing Methods - Black-Box Approach - Glass Box Approach.

UNIT IV

Client - Side and Server-side Programming Languages n-Declaring variables Commenting - Adding Data and Time Functions to Scripts - Using Mathematical Operators and Functions - Using Conditional Statements.

UNIT V

String Functions - Creating Subroutines - Creating Functions - Using Logical Connectives and Operators - Using Loops to Repeat Code - A Simple Page - VB Script and Forms - Hiding Errors.

UNIT I

LESSON

1

INTERNET

CONTENTS

1.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Understand the concept of internet]
- Discuss history of internet

- Discuss working of web
- Understand web servers and clients
- Discuss ISP, ISDN, etc.
- Understand domain naming system
- Define intranet

1.1 INTRODUCTION

This lesson will provide an introduction to the various terms like web pages, web servers and clients, domain naming system, intranet etc, you might have come across while surfing the Internet. Firstly we will discuss about internet and history of internet.

1.2 INTERNET

"The Internet" is probably one of the most overused technical terms in common man's lingo today. Though everyone seems to have some ideas and opinions about the Internet yet not many can define it precisely. There is no single, generally agreed upon answer to the question – "What is the Internet?" - because the Internet is a different thing to different people. Consider the following few expressions in this context.

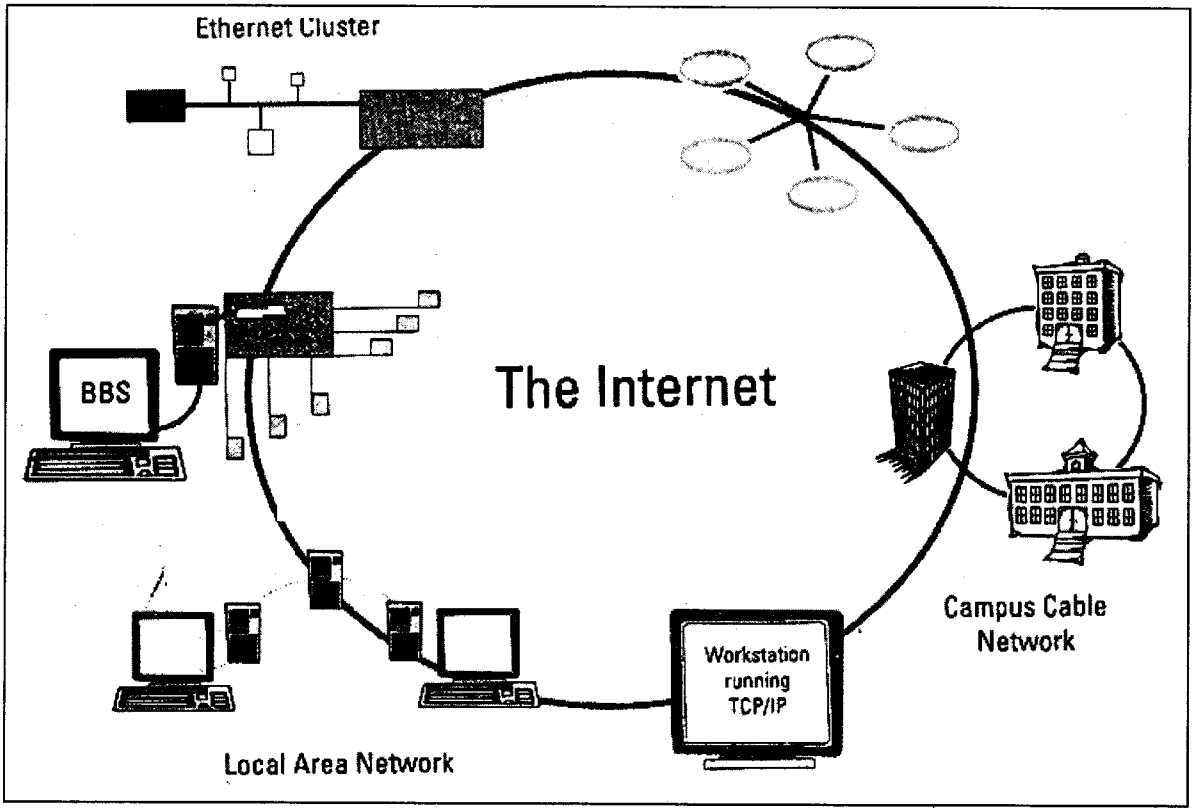
- Internet, the mega network connecting millions of people, is a tremendous phenomenon. The web, an offshoot of this global resource has revolutionized the way information is passed on.
- It is the name given for a vast, worldwide system consisting of people, information, and computers.
- It is a network of networks.
- It is an unlimited commercial opportunity.
- It is a set of computers talking over fibre-optics, phone lines, satellite links, and other media.
- It is cyber space where data surfing can be done.
- It is an ocean of information.
- It is a gold mine of professionals from all fields sharing information about their work.
- It is a world wide interconnected system of thousands of computer networks, each network in turn linking thousands of computers together.

In short, we can say that the Internet is a vast collection of globally available information, which can be accessed electronically – information, which is of practical use for business, research, study, and technical purposes. It is a means for electronic commerce – marketing, buying, services, economic, and financial data research. It is a collection of hundreds of libraries and archives that will open to your fingertips. It is also a vast store of information relating to your hobbies, travel, health, entertainment, games, software, etc.

Today the information can be in the form of Text, Images, Animation, Sound, Video etc., tomorrow it would probably be in the form of smell, touch, taste or some energized form. If information can be

put on computers, it can be digitized and can be made available on the Internet. The only catch is - How fast? Even the future may not be able to tell.

To be technically correct, we can say that the Internet is "an ever growing wide area network of millions of computers and computer networks across the globe, which can exchange information through standard rules (protocols)". The same is illustrated in the following figure.



Each computer participating in the Internet has a unique address. Information is divided into packets, which may travel through different paths to the destination address where it is recombined into its original form.

A few computers can be interconnected to share resources and exchange information using a number of network technologies. Further, such networks can be interconnected to form even larger networks. The Internet is a similar network that connects millions of networks worldwide.

Note that we use "The Internet", with definite article "The" and not just "Internet". This is because though there are many networks existing today the one we call "Internet" is a single network consisting of all the participating networks.

1.2.1 History of Internet

How old is the Internet? Here is a short historical background. The concept, which gave birth to the Internet, as it exists today, started with a project called the ARPANET (around late 1960s) which was sponsored by the United States Department of Defence, Advanced Research Project Agency (ARPA). The Department of Defence (DOD) was interested in building a network that could maintain itself under adverse conditions. The original idea was to build a network capable of carrying military and

government information during a "nuclear event". Thus, the predecessor of Internet, as often called ARPANET, came into being.

Till late 1970s, ARPANET operated to provide connection facility to around a dozen of research sites. Later in 1982, ARPA established the Transmission Control Protocol (TCP) and Internet Protocol (IP), as the protocol suite, commonly known as TCP/IP, which is a connection protocol between sites, which is still in use today and is the primary method of connecting to the Internet.

This technique of information access, which started as a method of sharing files amongst researchers was slowly adopted by a wider clientele and after 1982 the network started expanding faster like the big bang.

The credit remains with the National Science Foundation (NSF), USA, whose network of strategically located supercomputers (NSFnet) allowed people to share information from home or their institutions. The other networks which developed in parallel, like BITNET of IBM, X.25 based in Europe and UUCP based in Bell Labs got slowly absorbed or got connected to Internet, in order to provide a unified information sharing medium. The network although started in US, slowly started reaching other countries by connecting the networks of other countries.

Internet in India started on an auspicious day, 15th of August, 1995. This in a way marks the beginning of free information flow from every nook and corner of the world and thus could well be called an "Independence day" for Information Age in the country. Videsh Sanchar Nigam Limited (VSNL) started a service called GIAS (Gateway Internet Access Service) to allow the Indian users a bite of the Internet.

1.2.2 Internet Usages

The Internet is an important tool for practically everybody. The applications are endless limited only by our imagination. Whatever information is required, it is either already available on the Internet or it is soon going to be available. Here are some interesting application areas:

- Electronic mail, which was until recently considered only an internal mechanism of an enterprise, is quickly becoming the most widely used application on the Internet. The most common of the communication methods used by people on the Internet is the private letter, written by one individual to another (on any subject and in any language), and sent between any two connected Internet sites or through an Internet E-Mail gateway to or from a service which provides an Internet gateway.
- The ability to exchange visual information in readable and reusable formats — such as charts, figures, tables, images, databases, software code — opens up possibilities for collaboration at the global as well as local levels. With the trend specialization, the ability not only to communicate but also to actually work with colleagues in the same field scattered all over the world makes long distance collaboration feasible.
- The resources for on-line research are multiplying at an astounding rate. Searchable databases library holdings, alerting services, pre-prints, and other information systems are all changing the way research is done And it is not only the research community that is responsible for this. Library shelves are overflowing with journals and proceeding, and with acquisitions budgets receiving deep cuts, a likely scenario for the future is one in which libraries archive electronically, share holdings, and become information clearing houses instead of closets.

- Another very important application of Internet is multimedia. Live music concerts, radio broadcasts, live or recorded television shows, interactive audio and webphone, and video conferencing are no more a dream on Internet, even for a desktop PC user.

Internet provides a variety of information to everybody ranging from entertainment to serious business application to areas of daily life such as:

- Magazines and newspapers
- Household shopping items
- Ordering novelties from anywhere in the world
- Radio and TV broadcast schedules and sometimes the broadcast itself
- Tour and travel plan guides and bookings, etc.
- Health consultation
- Tips for doing various things
- Talking to friends and relatives in any part of the globe
- Games of various kinds
- Language interpreter
- On-line education course material, examination conduction, advertising on popular information sites, making payments on the net and getting an item, net banking

1.3 WORKING OF WEB

The World Wide Web or just the web is a collection of millions of files stored on the thousands of computers (Web servers) all over the world. Web is only a few years old, but it is growing at an astounding rate. Its popularity has increased dramatically because it is so easy to use, colorful, and right in content. Basically, the Web is a series of interconnected documents stored on computer sites or Web sites. On a trip through this Web, you can visit colleges, companies, museums, government departments, and other individuals like yourself. As you move through the Web you can read data on almost every imaginable topic. You can visit stores to buy things or transfer movies, pictures, games, and other software to your computer, much of it free. The World Wide Web is in its infancy, but it may soon become a pipeline for telephone, communication, entertainment, and news — challenging existing technologies.

The World Wide Web (also called WWW, or W3, or simply the Web) is a tool that helps you to find and retrieve information, using links to the other WWW pages. Web links are stored within the page itself and when you wish to "jump" to the page that is linked, you select the "hotspot" or "anchor". This technique is some times called hypermedia or hypertext. If you have used a Windows or Macintosh based help system in which you click on an underlined phrase to jump to that topic, you are already familiar with hypermedia. On the Web, a link can point anywhere else on the Internet. Thus, it is possible to follow one link after another, jumping from one computer to another, all around the Net.

The WWW provides a network of interactive documents and the software to access them. To navigate the WWW, you surf from one page to another by pointing and clicking on the hyperlinks in text or graphics. The World Wide Web and HTTP:

- Allow you to create links from one piece of information to another.
- Can incorporate references to text, graphics, sound, and movies (or multimedia).
- Understand other Internet protocols, such as FTP, gopher, and telnet.

The World Wide Web is non-linear with no top, or no bottom. The meaning of non-linear is that you do not have to follow a hierarchical path to information resources. So you can:

- Jump from one link (resource) to another.
- Go directly to a resource if you know the Uniform Resource Locator (URL), i.e., its address.
- Even jump to specific parts of a document.

Since the Web is not hierarchical and can handle graphics, it offers a great deal of flexibility in the way information resources can be organized, presented, and described.

1.4 WEB CLIENT & WEB SERVER

Client-server is a computing technology wherein there are two participating software which may run on the same computer or on different computers connected through a network. One of the software's is called server that while running waits to receive a request from the other software called client. On receipt of the request the server carries out the necessary computations to produce the response which is subsequently sent to the client that originated the request.

Note carefully that web server is actually a software. People often use the term loosely for the machine itself running the web server.

1.4.1 Web Server

Web pages are created using HTML syntax. These pages must be organized and stored at a central computer.

The organization of web pages into directories and files stored on the HDD of a central computer is called 'Web Site' creation.

Computers which stores web pages in the form of directories and files and provide these files to be read, are called 'Servers'. They act like service providers that service the need for information.

The Server Computer runs special software called 'Web Server' software that allows:

- Web Site Management
- Accept a client's request for information
- Respond to client's request by providing the page with the required information.

Some of the most popular software, which Servers run to allow them to respond to client request for information, is Internet Information Server (IIS), Apache Web Server, Netscape Server, and Microsoft Personal Web Server.

We Server Software stores and manages web pages. When required, the Web Server accepts requests for these Web Pages, retrieves these web pages from its HDD, and sends the page back to the client who requested for it.

1.4.2 Web Client/Browser

To access information stored in the form of web pages, users must connect to a Web Server. Once connected, an interface that displays the contents of the web page is required.

Computers that offer the facility to read information stored in web pages are called Web Clients.

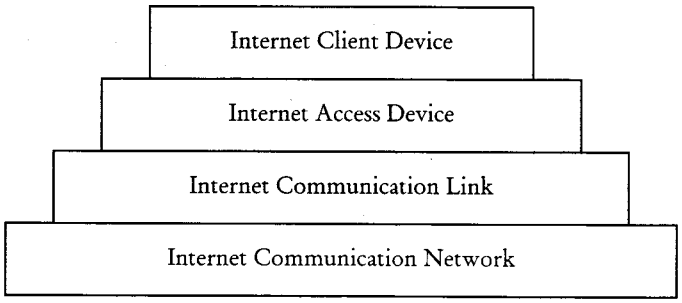
Web Clients run special software called Browser that allows them to:

- Connect to an appropriate Server.
- Query the Server for the information to be read.
- Provides an interface to read the information returned by the Server.

Some of the most popular Browser software that clients run to allow them to query Web Servers for information is Netscape Communicator, Internet Explorer, etc.

1.5 TOWER OF THE INTERNET HARDWARE

The Internet hardware can be presented in a hierarchical manner forming a tower like structure as shown below.

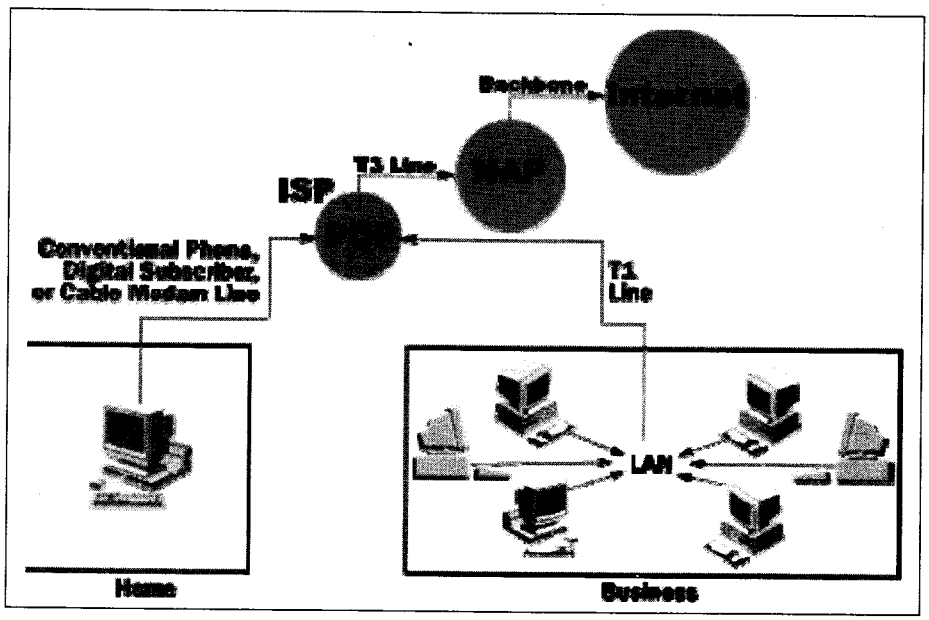


A typical hardware connection for accessing the Internet is shown below. Intercontinental networks include communication backbone. Backbones are typically fiber optic trunk lines. The trunk line has multiple fiber optic cables combined together to increase the capacity.

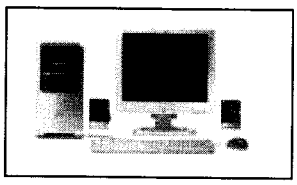
NSFNET was the first high-speed backbone created by National Science Foundation (NSF) in the year 1987. It connected 170 smaller networks together and operated at 1.544 Mbps (million bits per second). IBM, MCI and Merit worked with NSF to create the backbone and developed 45 Mbps backbone the following year. Today there are many companies that operate their own high-capacity backbones. The entire Internet is a gigantic, sprawling agreement between companies to intercommunicate freely.

The Internet interconnects multitude of electronic resources. These resources can be hardware, software information or a combination of them. The end-users of the Internet are largely either an individual like you and me or an organization.

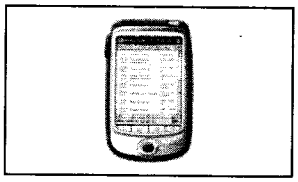
From an end-user point of view, following are the required hardware and software infrastructure to hook onto the Internet.



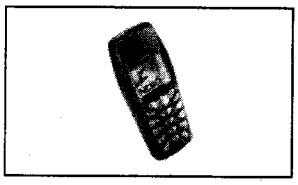
The first requirement is an Internet-access device, which could be a personal computer - desktop, laptop etc., or a mobile device like PDA (Personal Data Assistant).



A Desktop PC



A PDA



A mobile phone

Next a communication line is needed to allow communication between the participating devices. Commonly available communication lines for public use are:

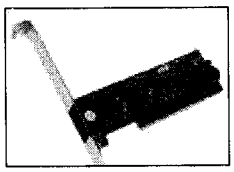
- Telephone lines
- Optical fiber lines
- Microwave links
- Radio links

Then one needs an NIC (Network Interface Card), which connects the above devices to the communication line. The other devices required for Internet access are:

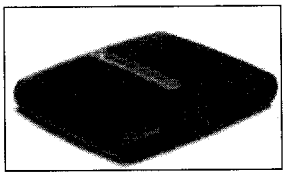
1.5.1 Modem

Modem, an acronym for Modulation - Demodulation, is a device that takes a digital signal from the computer, converts it into an analog signal that the phone line and recognize and transmits the signal through the phone line to another modem that is connected to another computer. The receiving modem then converts the analog signal back to its original digital format then sends it to the computer for processing. Depending on the speed of the modem, the transmission could be fast or slow. Most modems today transmit at speeds of up to 56Kbps.

Modems can be internal to the hardware or external attached as external device to the computer.



Internal Modem



External Modem

1.5.2 ISP

Internet Service Provider is any company which provides Internet facility, either through dial-up or through leased lines. In India, some leading ISPs are Satyam, BSNL, VSNL, Airtel, etc.

1.5.3 ISDN

ISDN stands for Integrated Services Digital Network. There are two versions of ISDN available. The simplest one is Basic Rate Interface (BRI), which takes advantage of the copper twisted pair wiring that is in homes and offices. Instead of a single analog signal, an ISDN line carries three digital channels: two B (Bearer) channels that can carry any kind of data at 64,000 bps, and a D (or Delta) channel, operating at 16,000 bps, that can carry control signals and serve as a third data channel.

1.5.4 Cable Modem

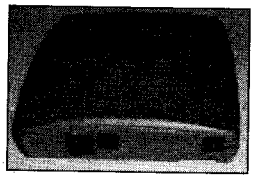
The basic interface between the cable modem and your PC is an Ethernet link. The cable modem plugs into an Ethernet host adapter in your PC. Then you just need to load the drivers to operate the cable modem. One problem with cable modems is with the current wiring of cable systems themselves. They are designed for program distribution, not for two-way communications. The cable itself does not limit the system to distribution, but many cable companies put amplifiers in their systems to allow their signals to reach for miles beyond their offices. All the bandwidth on the coaxial cable running to your home is not yours alone. The same signals travel to you and your neighbors. This creates a problem with bandwidth. Divide the bandwidth by the number of subscribers and you have little bandwidth available to them. To combat this, cable companies slice up the entire system into sub-units.

Cable Modem

1.5.5 DSL

DSL stands for Digital Subscriber Line, is a very high-speed connection that uses the same wires as a regular telephone line. DSL offers a number of advantages over other lines. You don't have to shut down your Internet connection to talk on the phone. The speed taps off at 1.5 Mbps vs. 56 Kbps for a regular modem. DSL doesn't require new wiring it can use the phone line you already have. However,

there are some disadvantages as well. Your connection is better when you are closer to the provider's central office. The connection is faster for receiving data than it is for sending data. You can't get the service everywhere.



ADSL

ADSL is asymmetric DSL. ADSL divides up the available frequencies in a line based on the fact users do more downloading than uploading. The two pieces of equipment includes: for the customer, a DSL transceiver, for the service provider, a DSL Access Multiplexer (DSLAM) to receive customer connections.

Connecting to the Internet is a simple process outlined below.

1. Connect your device to the modem
2. Connect the modem to the communication line
3. Run a modem driver program on your system
4. Subscribe the Internet service from an Internet Service Provider.
5. Run the Internet Access software on your system to connect to the Internet server at the Provider's end.
6. Once connected - after proper authentication - run the Internet applications to use the Internet services.

1.6 DOMAIN NAMING SYSTEM (DNS)

We have two types IP address in the form of decimal numbers and text for the same host. You know that list of all IP addresses are maintained centrally by ICANN in the form of distributed database directory. There are several distributed servers, which maintain this list of IP addresses. The reasons behind the distributed server are very logical and simple. It helps in disaster management and in diverting the load of the traffics in the form of requests from clients to other DNS servers located at different sites. DNS server maintains database in both the form that is textual as well as decimal notations. For example, DNS server maintains the address of google site as www.google.com and 216.23.9.53.99. In this manner, DNS is used to provide host-to-IP address mapping of remote hosts to the local hosts and vice versa. It is now amply clear that the DNS maintains a distributed database to map between hostnames and IP addresses. Whenever a client requests a service from a site, then both the site runs DNS protocol to access the distributed database which is nothing but Domain Name Systems. Therefore, the DNS provides the protocol, which allows clients and servers to communicate with each other. DNS enables a system to use a resolver, which resolves the host name to IP address understandable by server. IP address assigned to a computer on the Internet is a 32-bit number unique to each computer. It is not easy to remember such a number specially when you have to deal with many computers. Domain Name System (DNS) is a mechanism that assigns an easy to remember names to IP addresses.

You may be now thinking of how DNS is able to provide the quick translation of text of the IP addresses within fraction of seconds from a directory of billions of such addresses. This could be made possible by using Domain concepts, which uses hierarchical arrangements of text addresses translation.

You can see from the Figure 1.1 that at the top level is the root server, which has null label. Below this is another level domain or domain as com, edu, int and so on which are grouped together. Below this different sub domains or groups have been created.

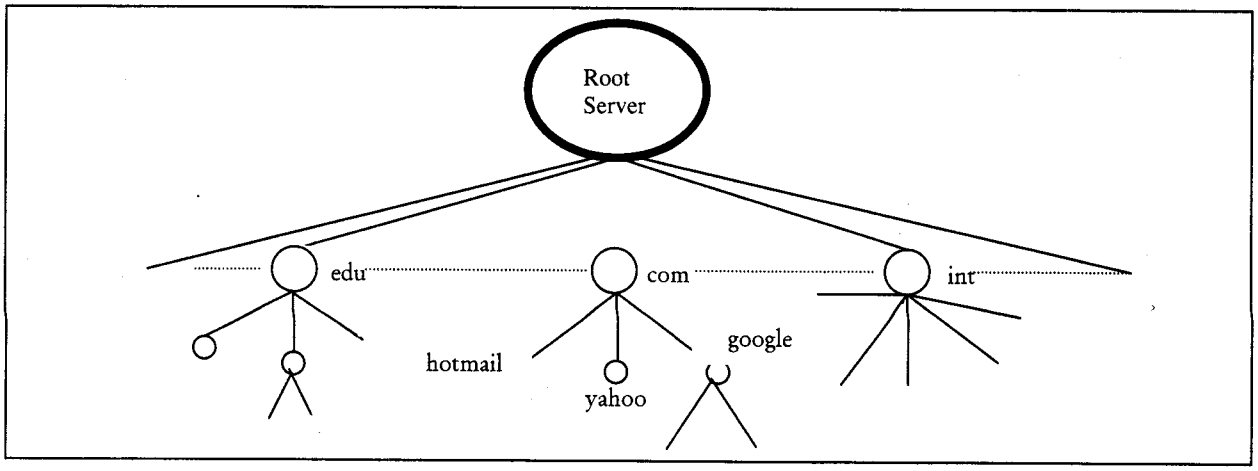


Figure 1.1: DNS Hierarchy

As we know that the servers maintaining addresses are distributed and have locations throughout the world. Then this question arises as to how text addresses are organized in hierarchical arrangement. You may refer Figure 1.1 above. The hierarchy is represented into zones and each zones is a hierarchy of one or more nodes without any overlapping. Each zone is represented by a server and undoubtedly with one backup server. Root server as shown in Figure 1.1 is only one, which is just indicative; there may be several root servers at several locations in the world. Each root is aware of the location of each DNS server of specific domains.

The process is now very simple to understand. When you need to connect with a particular site, you first send your request to your local host. If your local host can provide the translation, your request is completed. If not, your local host then sends your request one level above in the hierarchy. If the server at one level above is able to handle the same, you get your intended website at your desktop through your local server. If not, then the server at one level above from your local server either sends your request again to another server or informs your local server that your request is failed and gives the address of another server to process your request. This process continues till a server is found who knows the address, otherwise, the request is filtered up to the root server. Depending upon the domain address, root server forwards the request to the one of the domain servers represented at the next level of hierarchy. This process continues and the information of text address is returned to Root server and then back to your local server.

Domain is a large group of computers on the Internet. Under this scheme each computer has an IP address and a domain name. Domains have been made on the base of organization type or geographic locations, e.g., the domain name - ibm.com - where .com indicates that ibm is a commercial organization.

In general, while typing Internet addresses it is better to use all lower case letters. There are basically two types of top-level domains:

- Non-geographical domains or those which indicate the type of organization, e.g., in *www.yahoo.com*, the .com indicates that it is a commercial organization.
- Geographical domains indicate the code for individual countries, e.g., www.vsnl.net.in. Here the .in indicates that the network connection is in a country called India.

There are several computers in a network which can be grouped under a domain and can be given friendly names for convenience. Thus if machines/resources are shifted, their names need not change. Domain names are easy to remember.

The latest we have heard is that soon we will have .travel to indicate sites related to travel, .movies to indicate sites related to movies and so on, i.e., extensions will be according to the functions the sites perform.

1.6.1 Non-geographical Domains

Table 1.1 below corresponds to some commonly appearing domain names with their respective sites. The DNS can accommodate almost all kinds of organizations by allowing each group to choose between geographical or organizational naming hierarchies.

The Standard Non-Geographical Domains (Top Domains) are:

Table 1.1: Internet Domains

Domain	Description	Example
com	for commercial organisations	yahoo.com
edu	for higher educational institutions	imt.edu
gov	for Government organisations	whitehouse.gov
mil	for US military organisations	army.mil
net	for network organisations or resources	pacific.net
org	miscellaneous or non-profit organisations which do not form a part of either educational or commercial activities	farnet.org
int	international organisations	tpc.int

1.6.2 Geographical Domains

The geographically based top level domains use two-letter country designations. Examples of Geographic Domains:

Domain	Description
af	Afghanistan
bb	Barbados
by	Belarus
co	Colombia (Republic of)
gu	Guam
kr	Korea (Republic of)
kw	Kuwait (State of)
ng	Nigeria (Federal Republic of)

Contd....

ua	Ukraine
ye	Yemen (Republic of)
zw	Zimbabwe (Republic of)

Each domain corresponds to a unique numeric IP address. The user might be connecting to an address like www.sun.com but as far as the Internet is concerned it is connecting to the web server with the IP address associated with that domain name. The Domain Name System (DNS) implemented as Domain Name Server, automatically converts domain name addresses into IP addresses. DNS is portable. Even if IP addresses change, the domain name may remain the same.

Sub-domains can also be created within domains. They are represented by period(.). The rightmost part is the top level domain which can be a geographic/organisation type domain till you reach the left which could be the name of the individual host computer, e.g., *vsnl.net.in* means in India there is a networking organisation called vsnl.*doc.cis.myuniversity.edu* shows that this host/server is in the documentation department called doc in the (CIS) Computer Information Services division at My University which is an educational organisation.

Registering your own Domain Name(net)

Your Internet service provider may setup your domain name free of charge, except for the domain name registration fees, or for a low fee, so contact them first. If you already have a web site and an Internet service provider to add your domain name to their DNS servers (so that your name can be translated into an IP address), the cheapest and the most straight forward way to get a domain name maybe by contacting the InterNIC yourself.

Although registering your domain name yourself through InterNIC could cut your cost by eliminating the middle man, you can save some time by asking a third party to do the registering for you. This type of service could come in handy, specially if you don't already have a web server running to keep the new domain name active.

1.7 INTRANET

Intranet is a network based on TCP/IP protocols (an internet) belonging to an organization, usually a corporation, accessible only by the organization's members, employees, or others with authorization. An intranet's Web sites look and act just like any other Web sites, but the firewall surrounding an intranet fends off unauthorized access.

Check Your Progress

1. Define web server.
2. What is Intranet?

1.8 LET US SUM UP

Internet, the mega network connecting millions of people, is a tremendous phenomenon. The web, an offshoot of this global resource has revolutionized the way information is passed on. Today the information can be in the form of Text, Images, Animation, Sound, Video etc., tomorrow it would probably be in the form of smell, touch, taste or some energized form. If information can be put on

computers, it can be digitized and can be made available on the Internet. The Internet is an important tool for practically everybody. The applications are endless limited only by our imagination. Whatever information is required, it is either already available on the Internet or it is soon going to be available.

The World Wide Web (also called WWW, or W3, or simply the Web) is a tool that helps you to find and retrieve information, using links to the other WWW pages. Web links are stored within the page itself and when you wish to "jump" to the page that is linked, you select the "hotspot" or "anchor". This technique is some times called hypermedia or hypertext. The WWW provides a network of interactive documents and the software to access them. To navigate the WWW, you surf from one page to another by pointing and clicking on the hyperlinks in text or graphics.

Computers which stores web pages in the form of directories and files and provide these files to be read, are called 'Servers'. They act like service providers that service the need for information. Computers that offer the facility to read information stored in web pages are called Web Clients.

Internet Service Provider is any company which provides Internet facility, either through dial-up or through leased lines. In India, some leading ISPs are Satyam, BSNL, VSNL, Airtel, etc. ISDN stands for Integrated Services Digital Network. There are two versions of ISDN available. The simplest one is Basic Rate Interface (BRI), which takes advantage of the copper twisted pair wiring that is in homes and offices. DSL stands for Digital Subscriber Line, is a very high-speed connection that uses the same wires as a regular telephone line. ADSL is asymmetric DSL. ADSL divides up the available frequencies in a line based on the fact users do more downloading than uploading. Domain is a large group of computers on the Internet. Under this scheme each computer has an IP address and a domain name. Domains have been made on the base of organization type or geographic locations, e.g., the domain name - ibm.com - where .com indicates that ibm is a commercial organization. Non-geographical domains or those which indicate the type of organization, e.g., in www.yahoo.com, the .com indicates that it is a commercial organization. Geographical domains indicate the code for individual countries, e.g., www.vsnl.net.in. Here the .in indicates that the network connection is in a country called India.

1.9 KEYWORDS

Internet: Internet, the mega network connecting millions of people, is a tremendous phenomenon.

Web: The World Wide Web (also called WWW, or W3, or simply the Web) is a tool that helps you to find and retrieve information.

Web Server: Computers which stores web pages in the form of directories and files and provide these files to be read, are called 'Servers'.

Web Client: Computers that offer the facility to read information stored in web pages are called Web Clients.

ISP: Internet Service Provider is any company which provides Internet facility, either through dial-up or through leased lines.

ISDN: ISDN stands for Integrated Services Digital Network.

Domain: Domain is a large group of computers on the Internet.

1.10 QUESTIONS FOR DISCUSSION

1. What is internet? Discuss evolution of internet.
2. Describe the working of web.
3. Differentiate between web servers and web clients.
4. Differentiate between ISP and ISDN.
5. What is domain naming system? Differentiate between geographical and non-geographical domain system.
6. Discuss the process of registering your own domain name.
7. Discuss the process of a typical hardware connection for accessing the Internet.

Check Your Progress: Model Answers

1. Computers which stores web pages in the form of directories and files and provide these files to be read, are called 'Servers'.
2. Intranet is a network based on TCP/IP protocols (an internet) belonging to an organization, usually a corporation, accessible only by the organization's members, employees, or others with authorization.

1.11 SUGGESTED READINGS

Adolfo Rodriguez, John Gatrell, John Karas, and Roland Peschke, *TCP/IP Tutorial and Technical Overview* (7th Edition), Prentice Hall.

Martin W. Murhammer and Eamon Murphy, *TCP/IP Tutorial & Technical Overview*, Prentice Hall.

Erik T Ray, *Learning XML*, Second Edition O'Reilly Media.

Michael Morrison, *HTML and XML for Beginners*, Microsoft Press.

Lisa Lopuck, *Web Design for Dummies*, Bk&CD-Rom edition.

LESSON

2

WEB BROWSERS

CONTENTS

Contd....

2.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss the concept of web browsers
- Discuss hypertext
- Understand hyper text markup language and its components
- Understand URL
- Discuss HTML links

2.1 INTRODUCTION

This lesson will provide an introduction to the various terms like web pages, web server, html, browser etc., you might have come across while surfing the Internet. It will provide an easy but comprehensive explanation of what web pages are and how they are created. It will give you an idea of how communication takes place between web clients and a web server and how web pages are transferred from the web server to the web clients. Further it will familiarize you with hypertext markup language, which is used to create web pages along with various tools, HTML editors and HTML tags. This lesson will familiarize you with the structure of an HTML program which consist of a head and body part, how to put title and footer and text formatting in a web page. For emphasizing material in a web page heading styles and horizontal rules are given which will equip you with the basic skills to write a simply HTML document.

This lesson will familiarize you with how HTML allows linking of one HTML document to other HTML documents. It will provide you with details of how one can link to documents, which are either present in the current working directory, or to a different location in the same document. Images can also act as links and you will be introduced to the concept of images maps, which provides a technique to link multiple documents to the same image.

2.2 WEB BROWSERS

A web browser is the software program used to access the World Wide Web that is the graphical portion of the Internet. The first browser, called NCSA Mosaic, was developed at the National Centre for Supercomputing Applications lab at Illinois in the early 1990s. The easy-to-use point-and-click interface fuelled the growth of the Web. The web browser software program facilitates a user to

display and interact with text, images, videos, music and other information typically located on a web page at a website on the www or a LAN. Text and images on a web page normally designed to provide hyperlinks to other web pages at the same or different website. Thus web browser enables point to point click to reach directly to the targetted web pages.

Some of the popular web browsers are Internet Explorer, Mozilla Firefox, Safari, AOL Explorer, etc. Web browsers belong to HTTP user agent category. Browsers also provide in accessing information provided by web servers in private networks or content in file systems.

Web browsers talk with web servers using HTTP (hypertext transfer protocol) to retrieve web pages. HTTP enables web browsers to provide information to web servers to retrieve web pages from them. Different web pages have a URL address starting with http:// for HTTP access. There may different other URL types and their corresponding protocols and most of the browsers supports them. FTP (file transfer protocol) is one of the examples of such types. Other examples are rtsp: for RTSP (real-time streaming protocol) and https: for HTTPS (an SSL encrypted version of HTTP).

The file format for a Web page is normally HTML (hyper-text markup language) and is identified in the HTTP protocol with a MIME content type. Most of the browsers support different formats such as the JPEG, PNG and GIF image formats including HTML. The combination of HTTP content type and URL protocol specification enables designers to embed images, animations, video, sound and streaming media into a web page or to make them accessible through the web page.

To access information stored in the form of web pages, users must connect to a Web Server. Once connected, an interface that displays the contents of the web page is required.

Computers that offer the facility to read information stored in web pages are called Web Clients.

Web Clients run special software called Browser that allows them to:

- Connect to an appropriate Server.
- Query the Server for the information to be read.
- Provides an interface to read the information returned by the Server.

Some of the most popular Browser software that clients run to allow them to query Web Servers for information is Netscape Communicator, Internet Explorer, etc.

2.2.1 Information Files Creation

If information has to be stored on a central computer, it must be created first. While being created, information can only be stored in the form of files on the computer. These files are created using special software programs or programming environments.

Files that travel across the largest network in the world, the Internet, and carry information from a 'Server' to a 'Client' that requested them are called 'Web Pages'. The individual who develops these web pages is called 'Web Developer'.

2.2.2 Basic Features of Browsers

Before we get involved in all the details, let is discuss some important browser features.

1. The Web browser should be able to look at the Web pages throughout the Internet or to connect to various sites to access information, explore resources, and have fun.

2. The Web browser must enable you to follow the hyperlinks on a Web page and also to type in a URL for it to follow.
3. Another feature of browser is to have a number of other commands readily available through menus, icons, and buttons.
4. Your browser ought to include an easy way to get on-line help as well as built-in links to other resources on the Web that can give you help or answers to your questions.
5. You will definitely want a way to save links to the sites you have visited on the WWW so that you can get back to them during other sessions. Web browsers take care of those in two ways, through a history list, which keeps a record of some of the Web pages you've come across in the current session, and a bookmark list, which you use to keep a list of WWW pages you want to access any time you use your browser. The name of the site and its URL are kept in these lists. The bookmark list is particularly important and the browser will contain tools to manage and arrange it.
6. One of the main features of a browser is to search the information on the current page as well as search the WWW itself.
7. Browsers give you the facility to save a Web page in a file on your computer, print a Web page on your computer, and send the contents of a Web page by e-mail to others on the Internet.
8. Few Web browsers (like Netscape Communicator) are complete Internet package, means they come with components like e-mail client, newsgroup client, an HTML composer, telnet client, ftp client, etc.
9. Web browser should be able to handle text, images of the World Wide Web, as well as the hyperlinks to digital video, or other types of information.
10. To take advantage of some of the most exciting things on the World Wide Web, your browser needs to properly display and handle Web pages that contain animated or interactive items. Netscape Navigator can incorporate these features through its ability to interpret programs written in Java and Java Script.
11. Web browsers interact not just with the Web, but also with your computer's operating system and with other programs, called plug-ins that gives the browser enhanced features.
12. Another important feature to insist on in your browser is caching. A browser that caches keeps copies of the pages you visit so that it does not have to download them again if you want to return to them. Reloading a page from the cache is much quicker than downloading it again from the original source.
13. The most important feature of any browser is ease of use. While all Web browsers are fundamentally simple to use, the one you settle on should be very easy to work with; it should function as a transparent window onto the Web.

14. If you will be browsing the Web from within a secured network, you may have to configure your browser to work through a special computer on your network called a proxy server. Most popular browsers let you configure them to work with a proxy server, but some don't, so find out if you will be working through a proxy before deciding on your browser. If you are, your ISP or system administrator will tell you if you need to do anything special to use your browser.

2.2.3 Understanding how a Browser communicates with a Web Server

Before going into the detailed explanation of how a browser communicates with a Web Server there are some technical terms you should be familiar with. TCP/IP means Transmission Control Protocol/Internet Protocol which is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network either an intranet or an extranet. HTTP means Hyper Text Transfer Protocol, the actual communications protocol that enables Web browsing. FTP refers to File Transfer Protocol which is the language used for file transfer from computer to computer across the WWW. Each machine connected to the Internet has an address known as an Internet Protocol address (IP address). The IP address takes the form of four numbers separated by dots, for example: 123.45.67.890

When a Browser communicates with a Web Server, it results into a four-step HTTP transaction.

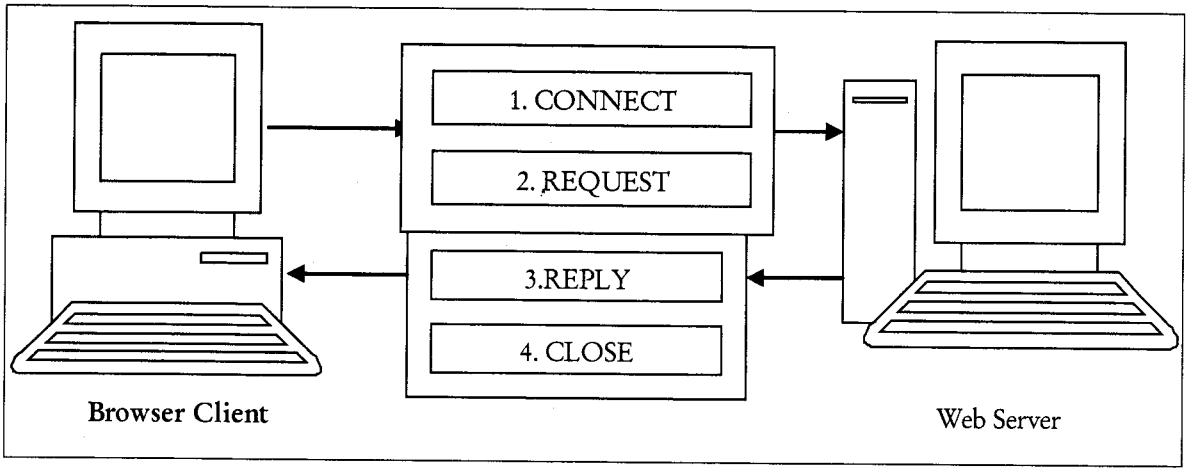


Figure 2.1

As seen in the Figure 2.1, a client's Browser retrieves a web page from the server and displays the web page in the Browser. The communication steps between the client and the server can be summarized as follows:

Establish Connection

Before a client and server can exchange information, they must first establish a connection. TCP/IP is used to let computers establish a link between a Web Server and a Web Browser over the Internet.

To communicate with the Web Server, the client machine must be given the IP address of the server along with the sub protocol that must be used, i.e., HTTP, FTP, etc. The client browser will attempt to locate the server based on the IP address supplied and establish a connection.

A web server supports multiple protocols. For example, a Web Server may support two protocols *viz.* HTTP, FTP. If the protocols are configured on default port numbers, the connection to a Web Server can be established by:

Protocol://IP address

The complete syntax to access and connect to any server would be:

Protocol://servername:port number

Client Issues a Request and Server Sends a Response

When a browser connects to a Web Server using an appropriate protocol name, IP address and port number, and the Web Server treats this connection to be a request for the 'Default Web Document'. The Web Server then dispatches the 'Default Web Document' to the client who connected.

If the client requires viewing any other web page then the client can specify the web page name (if known) along with the connection information. Thus the complete connection and web page information will now be specified as:

Protocol://servername:port number/web page name

A web page, apart from text and HTML tags can also include references to external objects, like GIF's JPEG's, Audio files, Video files, etc.

Thus, the request for a web page can be two-fold:

- The web page itself.
- The request for the objects referenced by the web page, i.e., GIF's JPEG's, Audio Files, Video Files, Executable programs, etc.

Server Terminates the Connection

It is the Server's responsibility to terminate the TCP/IP connection with the Browser after it responds to the Browser's request. However, both the Browser and the Web Server must manage an unexpected closing of a connection as well. In other words, if the user clicks on the Browser's STOP button, the Browser must close the connection.

Also a computer crash by either a Browser or a Web Server must be recognized by the surviving computer, which, in turn, will close the connection.

2.3 HYPERTEXT

Hypertext is most basically a way of constructing documents that reference other documents. Within a hypertext document, a block of text can be tagged as a *hypertext link* pointing to another document. When viewed with a hypertext browser, the link can be activated to view the other document. Of course, if you're reading this document, you're already familiar with the concept.

Hypertext's original idea was to take advantage of electronic data processing to organize large quantities of information that would otherwise overwhelm a reader. Two hundred years ago, the printing press made possible a similar innovation - the encyclopedia. Hypertext's older cousin combined topical articles with an indexing system to afford the researcher one or perhaps two orders of magnitude increase in the volume of accessible information. Early experience with hypertext

suggests that it may ultimately yield an additional order of magnitude increase, by making directly accessible information that would otherwise be relegated to a bibliography. Hypertext's limiting factor appears not to be the physical size of some books, but rather the ability of the reader to navigate increasingly complex search structures. Now, additional increases in human information processing ability seem tied to developing more sophisticated automated search tools, though the present technology presents possibilities that remain far from fully explored.

Supplementing basic hypertext with graphics, more complex user input fields and dynamically generated documents adds considerable power and flexibility to this concept. Hypertext, though still useful for its original goal of organizing large quantities of information, becomes a simple, general purpose user interface that fits neatly into the increasingly popular client-server model. It does not seem difficult to image a day when restaurant orders, for example, will be taken using a hand-held hypertext terminal, relayed directly to the kitchen for preparation, and simultaneously logged to a database for later analysis by management.

Characteristics of Good Hypertext

The flexibility of hypertext gives free range to the author's creativity, but good hypertext appears to have some common characteristics:

- ***Lots of documents:*** Much of the hypertext's power comes from its ability to make large quantities of information accessible. If all the text in your system can be printed on ten pages, it would be just as simple to read through it from beginning to end and forget all this hypertext silliness. On the other hand, if there are ten million pages of text in your system, then someone could follows a link on atomic energy and ultimately hope to find a description of the U-238 decay process.
- ***Lots of links:*** If each document has just one link, then it is little more than normal, sequential text. A hypertext document should present the reader with several links, offering a choice about where to go next. Ideally, a document should present as many relevant links as the reader can easily comprehend and select among.
- ***Range of detail:*** The great advantage of hypertext is that it permits readers to explore to a breadth and depth that is simply not feasible in print. To make this accessible, available hypertext documents should range from the broadest possible overview of a subject, down to its gritty details. Imagine the Encyclopedia Britannica, all thirty-odd volumes of it, searchable online and with each article possessing links to a half dozen reference documents with even more detailed subject coverage. This is the potential of hypertext.
- ***Correct links:*** This may seem trivial, but it's amazing how many Web links point nowhere. In general, be careful linking to any hypertext document not under your direct control. Can you count on it to be there later?

2.4 HYPER TEXT MARKUP LANGUAGE

HTML, or Hyper Text Markup Language is designed to specify the logical organization of a document, with important hypertext extensions. It is *not* designed to be the language of a WYSIWYG word processor such as Word or WordPerfect. This choice was made because the same HTML document may be viewed by many different "browsers", of very different abilities. Thus, for example, HTML allows you to mark selections of text as titles or paragraphs, and then leaves the interpretation

of these marked *elements* up to the browser. For example one browser may indent the beginning of a paragraph, while another may only leave a blank line.

HTML instructions divide the text of a document into blocks called *elements*. These can be divided into two broad categories - those that define how the BODY of the document is to be displayed by the browser, and those that define information `about' the document, such as the title or relationships to other documents. The vocabulary of these elements and a description of the overall design of HTML documents are given in the rest of Section 2. The Last part of the section also describes standard naming schemes for HTML documents and related files.

The detailed rules for HTML (the names of the tags/elements, how they can be used) are defined using another language known as the standard generalized markup language, or SGML. SGML is wickedly difficult, and was designed for massive document collections, such as repair manuals for F-16 fighters, or maintenance plans for nuclear submarines. Fortunately, HTML is much simpler!

However, SGML has useful features that HTML lacks. For this reason, markup language and software experts have developed a new language, called XML (the *eXtensible markup language*) which has most of the most useful features of HTML and SGML.

How do you get a Web page into a Web browser? All you have to do is to enter a HTML code into a file, and then open that file in your Web browser. To create that file, you can use common editors like Windows WordPad or Notepad if you save the page in text format. You can even use a word processor like Microsoft Word, but note that you cannot save normal Web pages in .doc format (which includes many special formatting characters that Web browsers won't understand) - they must be saved in text format from the Save As dialog box.

The name you give the Web page does not matter as long as you give it the extension .html to indicate that it is an HTML document.

In fact, there are many tools called HTML editors to help you with the process - many of which will even write the HTML for you.

2.4.1 HTML Editors

There are a number of HTML editors available for creating Web pages.

- *Adobe PageMill (hhtp://adobe.com)* - An editor that helps automate the whole process of site design and implementation.
- *Allaire's HomeSite (www.allaire.com)* - A simple, but powerful editor.
- *Amaya (www.w3.org/Amaya/)* - A WYSIWYG (what you see is what you get) HTML editor from W3 Corporation.
- *American Cybernetics Multi-Edit (www.amcyber.com)* - An HTML editor that works well for programmers.
- *BBEdit (www.barebones.com)* - An easy-to-use Macintosh HTML editor.
- *Claris Home Page (www.claris.com)* - An editor with mostly WYSIWYG for beginners with some additional tools.
- *ExpertTelligence's Webber (www.webbase.com)* - An editor that includes good support for checking your HTML.

- *Hot Dog Express (www.sausage.com)* - An easy-to-use editor for beginners.
- *Hot Dog Pro (www.sausage.com)* - A more powerful version of the Hot Dog Express editor including many advanced features, such as site management.
- *HoTMetaL Pro (www.softquad.com)* - An extensive, powerful and venerated HTML editor.
- *MicroEdge's Visual SlickEdit (www.slickedit.com)* - An editor that works well for programmers. It lets you work faster than many graphic-based editors.
- *Microsoft FrontPage (www.microsoft.com)* - An editor contained in a powerful package that integrates directly with your Web site.
- *Netscape Composer (www.netscape.com)* - An editor that comes with the Netscape Communicator, more or less WYSIWYG. It is easy to use.

2.4.2 HTML Tags

Tags are instructions that are embedded directly into the text of the document. An HTML tag is a signal to a browser that it should do something other than just throw text up on the screen. By convention all HTML tags begin with an open angle (<) and end with a close angle bracket (>).

HTML tags can be of two types:

Paired Tags

A tag is said to be *paired* tag if it along with a companion tag, flanks the text. For example, the <B> tag is a paired tag with its companion tag </B> causes the text contained between them to be rendered in *bold*. The effect of other paired tag is applied only to the text they contain.

Singular Tags

The second type of tag is the *singular* or *stand-alone* tag. They don't have a companion tag. For example,
 tag will insert a line break. This tag does not require any companion tag.

2.4.3 The Structure of an HTML Program

HTML documents are structured into two parts, the HEAD, and the BODY. Both of these are contained within the HTML element - this element simply denotes this as an HTML document.

The head contains information about the document that is not generally displayed with the document, such as its TITLE. The BODY contains the body of the text, and is where you place the document material to be displayed. Elements allowed inside the HEAD, such as TITLE, are not allowed inside the BODY, and vice versa.

Example of Document Structure

```
<HTML>
    <HEAD>
        <TITLE> Environmental Change Project </TITLE>
    </HEAD>
    <BODY>
        <h1> Environmental Change Project </h1>
```

Welcome to the home page of the Environmental Change Project.

This project is different from other projects with similar names. In our case we actually *wish* to change the climate. For example, we would like hot beaches in Northern Quebec, and deserts near Chicago.

```
<p>So how will we do this. Well we do the following
<ul>
        <li><A HREF="burn.html">Burn</A>more forests.
        <li>Destroy the
            <A HREF="http://who.zoo.do/ozone.html">
            Ozone</A>layer.
        <li>Birth more
            <A HREF="ftp:foo.do.do/cows.gif">cows</a>
</ul>
</BODY>
</HTML>
```

Document Head

Information placed in this section is essential to the inner workings of the document and has nothing to do with the content of the document. With the exception of information contained within the <TITLE> </TITLE> tags, all information placed within the <HEAD> </HEAD> tags is not displayed in the browser. The HTML tags used to indicate the start and end of the head section are:

```
<HEAD>
    <TITLE> </TITLE>
</HEAD>
```

Document Body

The tags used to indicate the start and end of the main body of textual information are:

```
<BODY>
</BODY>
```

Page defaults like background color, text color, font size, font weight and so on can be specified as *attributes* of the <BODY> tag. The attributes that the <BODY> tag takes are:

BGCOLOR	Changes the default background color to whatever color is specified with this tag. The user can specify a color by name or its equivalent hexadecimal number.
BACKGROUND	Specifies the name of the Gif file that will be used as the background of the document. This Gif tiles up across the page to give a background.
TEXT	Changes the body text color from its default value to the color specified with this attribute.

2.4.4 Titles and Footers

Title

A web page could have a title that describes what the page is about without being too wordy. This can be achieved by using the TITLE tag. Text included between the <TITLE> ... </TITLE> tag shows up in the title bar of the browser window.

Footer

Just as a title can be placed in the title bar of the browser window, certain information is commonly placed at the foot of the web page. Copyright information, contact details of the creator of the web page, etc. are the type of information traditionally placed at the foot of the web page. The HTML tags are:

<ADDRESS> ... </ADDRESS>

This tag should ideally be placed immediately after the last line of the textual material of the web page. However, it could also be placed anywhere in the body of the document. The text typed within these tags always appears in italics.

2.5 FORMATTING THE TEXT HTML

2.5.1 Paragraph Breaks

A blank line always separates paragraphs in textual material. The tag that provides this functionality is <P>. On encountering this tag, the browser moves onto a new line, skipping one line between the previous line and the new line.

2.5.2 Line Breaks

When text needs to start from a new line and not continue on the same line (without skipping a blank line), the
 tag should be used. This tag simply jumps to the start of the next line.

2.5.3 Emphasizing Material in a Web Page

Document pages are usually divided into sections and subsections (i.e., pages could have headings and sub headings), which need to be emphasized. HTML provides certain HEADING STYLES and HORIZONTAL RULES, which helps break text into logical sections with visual appeal.

2.5.4 Heading Styles

HTML supports six different levels of headings. The highest-level header for mat is <H1> and the lowest level is <H6>. All the styles appear in BOLDFACE and the size of the heading depends on the level chosen.

Section Headings

HTML allows for six levels of headings, marked by the element names H1, H2... H6. There is no forced hierarchy in these headings, but for consistency you should use the top level (H1) for main headings, and lower levels for progressively less important ones. In general hypertext documents

should be broken up so that each page does not occupy much more than a single screen. In these cases you can use the H1 heading to mark the main document heading, and the others to mark subheadings.

Heading Alignment: The ALIGN Attribute

HTML 3 proposed an ALIGN attribute to the heading element, which allows an author to "hint" at the desired alignment of the heading on the display. The possible values are ALIGN="left" (the default) to left-align the heading, ALIGN="center" to center the heading, and ALIGN="right" to right-align the heading. Several browsers understand left and center alignment, while very few understand right-alignment.

Some examples are shown below.

Examples of Headings

The following examples show the HTML coding for the six heading types, along with the results (note that the results will vary from viewer to viewer)

<H1 align="left"> Heading type H1 </H1>

Heading type H1

<H2 align="center"> Heading type H2 </H2>

Heading type H2

<H3 align="right"> Heading type H3 </H3>

Heading type H3

<H4> Heading type H4 </H4>

Heading type H4

<H5> Heading type H5 </H5>

Heading type H5

<H6> Heading type H6 </H6>

Heading type H6

2.5.5 Drawing Lines

The tag <HR> draws lines and horizontal rules. This tag draws a horizontal line across the whole page, wherever specified. The attributes to the <HR> tag are:

Attributes	Descriptions
ALIGN	Aligns the line on the Browser screen, which is by default, aligned to the center of the screen. ALIGN = LEFT will align the line to the left of the screen ALIGN = RIGHT will align the line to the right of the screen ALIGN = CENTER will align the line to the center of the screen
SIZE	Changes the size of the rule
WIDTH	Sets the width of the rule. It can be set to affixed number of pixels, or to the percentage of the available screen width.

2.5.6 Text Styles

Bold

Displays text in BOLDFACE style. The tags used are <B> ... </B>

Example: <B> Welcome to HTML Programming </B>

Output: Welcome to HTML Programming

Italics

Displays text in ITALICS. The tags used are between <I> ... </I>

Example: <I> Welcome to HTML Programming </I>

Output: Welcome to HTML Programming

Underline

Displays text as UNDERLINED. The tags used are <U> ... </U>

Example: <U> Welcome to HTML Programming </U>

Output: Welcome to HTML Programming

2.5.7 Other Text Effects

Centering (Text, Images, etc.)

<CENTER> ... </CENTER> tags are used to center everything found between them - text, lists, images, rules, tables, or any other page element.

Example: <CENTER> Welcome to HTML Programming </CENTER>

Output: Welcome to HTML Programming

2.5.8 Spacing (Indenting Text)

The tags used for inserting blank spaces in an HTML document is <SPACER>. Its attributes are:

TYPE	To specify whether space has to be left horizontally or vertically. TYPE = "HORIZONTAL" indicates that horizontal space has to be left. TYPE = "VERTICAL" indicates that vertical space has to be left.
SIZE	Indicates the amount of space to be left. Size accepts any integer.

Example:

Welcome to HTML Programming

<SPACER TYPE = "HORIZONTAL" SIZE = 90> Hope you enjoy it

Output:

Welcome to HTML Programming

Hope you enjoy it

2.5.9 Controlling Font Size and Colour

All text specified within the tags <FONT> and </FONT> will appear in the font, size and color as specified as attributes of the tag <FONT>. The attributes are:

FACE	Sets the font to the specified font name
SIZE	Sets the size of the text. SIZE can take values between 1 and 7. The default size used is 3. SIZE can also be set relative to the default size, i.e., SIZE = +x, where x is any integer value and will add up to the default size. For example, SIZE = +3 will display a size of 6.
COLOR	Sets the color of the text. COLOR can be set to an English language color name or to a hexadecimal triplet.

Example:

```
<HTML>
<HEAD>
<TITLE> TRYING AT WRITING HTML </TITLE>
</HEAD>
<BODY BGCOLOR=LIGHTGREEN TEXT=RED>
THIS IS JUST A TRIAL TO CHECK OUT ALL THE HTML TAGS.
YOU CAN CREATE YOUR OWN EXAMPLES TO TRY OUT FOR YOURSELF.<BR>
<P>
THE HEADING SECTION
<H1>FIRST HEADING </H1>
<H2>SECOND HEADING </H2>
<H3>THIRD HEADING </H3>
<H4>FOURTH HEADING </H4>
<H5>FIFTH HEADING </H5>
<H6>SIXTH HEADING </H6>
<P>THE PARAGRAPH TAG: ON ENCOUNTERING THIS TAG THE BROWSER MOVES TO A NEW
LINE  SKIPPING ONE LINE.<BR>
<BR> THE LINE BREAK TAG SIMPLY JUMPS TO THE START OF THE NEXT LINE<BR>
<B> THIS TAG IS USED TO MAKE THE TEXT BOLD </B><BR>
<I> THIS TAG IS USED TO MAKE THE TEXT ITALICS </I><BR>
<U> THIS TAG DISPLAYS TEXT AS UNDERLINED </U><BR>
<CENTER> THIS TAG IS USED TO CENTER TEXT, LISTS, IMAGES, RULES, TABLES AND
ANY OTHER PAGE ELEMENTS
</CENTER>
<P>
TRYING AT INDENTING TEXT <BR>
```

```
<SPACER TYPE="HORIZONTAL" SIZE =90> THIS IS USED TO INSERT BLANK SPACES
 IN AN HTML DOCUMENT
<P>
DRAWING HORIZONTAL LINE
<HR ALIGN=CENTER SIZE=80 WIDTH=10>
</BODY>
<HTML>
```

2.6 URL

Uniform (or Universal) Resource Locator: Though TCP/IP serves its role very well, it won't be easy remembering the TCP/IP address of every website. To simplify this, we have an English equivalent of each web site's address. This English address is also unique and yet, it is easy to remember. This is known as the URL. It includes the protocol, the web server's address and the domain name, such as - http://www.rajbhavanmp.ind.in

2.6.1 Hyper Text Transfer Protocol

In order to access any website, the web browsers are used which are assisted by the URL that uses the http scheme. It is the URL or the port number that assists the browser to link with a Web site. The server indicates a computer connected to the Internet while the port number indicates a type of socket to which the browser plugs in to link with the Web server. The web server not only provides the requisite web pages but also describes a computer program that runs on a computer to provide web pages. When a browser receives an URL will attempt to connect with the server computer having the required web pages by connecting to the specified port number. The URL can be provided to the browser either by typing it at its specified location or by clicking on the link available on some already displayed web page or document.

It is the role of the browser to connect with the server where the requisite requests from client or user is stored or available. When the web server receives the request from browser it replies back to the browser, which is client in this case. The information basically contains the HTTP protocol version, name of the server, the media type of the document and date etc. The media type of the document is quite important information because the browser is required to know what kind of document this is before it can process it. HTML is the most common media type transferred over the Web. Other media types are GIF image and JPEG image. Several times when a response like "HTTP 404 Not Found" is displayed which means that the request document is not available at the link. There are different responses defined in HTTP. Briefly, in order to access a web page, HTTP involves browser that issues a request followed by a few headers. In response, the server replies back with a few headers and a document.

The web server basically maps the URLs to files on its hard disks. The web server interprets the path in any URL to map it with a filename on its hard disk. In order to make it work to map with the requisite file, the web server is configured to contain a "document root" directory relative to which all URLs are resolved as filenames. Let us take an example, suppose the URL is http://myspace. tutorial.in, and the document root is D:\WWWFiles\. When a user types the URL http://myspace. tutorial.in/ lesson1/networking.htm into browser, the browser requests the server for the document

/lesson1/networking.htm. The web server begins searching in the directory D:\WWWFiles\lesson1 for a file called networking.htm. If the requisite file is available it responds with a header followed by the document. If it is not available, it responds a 404 Not Found followed by a helpful error message telling the user to search elsewhere.

2.6.2 Absolute and Relative URLs (net)

An absolute URL contains more information than a relative URL does. Relative URLs are more convenient because they are shorter and often more portable. However, you can use them only to reference links on the same server as the page that contains them.

2.7 HTML LINKS

Links that are the most fundamental part of the World Wide Web provides the facility to navigate from one web page to another web page. Links can be classified in three categories. These are links to anchors on the current page, links to other pages within the current site and links to pages outside the current site. Links could be provided for both texts and images.

HTML allows linking to other HTML documents as well as images. Clicking on a section of text or an image in one web page will open an entire web page or an image. The text or an image that provides such linkages is called Hypertext, *Hyperlink*, or *Hotspot*.

Links available with HTML are hyperlinks that enables user to link to another document on the Web. To facilitate hyperlinks there are the anchor tag <a> and and the Href attribute, which are used by HTML. An anchor can be used to point to any resource on the Web such as an HTML page, an image, a sound file, a movie, etc. The syntax for creating an anchor is: Text to be displayed. For example -

The href attribute is used to address the document to link to and the words between the open and close of the anchor tag will be displayed as a hyperlink. Target attribute is used to define where the linked document will be opened. The name attribute is used to create a named anchor. When using named anchors, links can be created that can jump directly into a specific section on a page, instead of letting the user scroll around to find the needed document. Below is the syntax of a named anchor: Text to be displayed.

Defining colors for the HTML links - With the use of a few settings, colors for all links could be defined on the WebPages. The general color of text links which is blue before the click is given in the <body> tag as: <body link="#C0C0C0" vlink="#800080" alink="#FF0000">. vlink, indicates that the link has been visited by the user and standard color is purple. alink specifies active link which means the color of the link when the mouse is on it. Its standard color is red.

In order to define more links to have different colors than the rest of the page, we can either place the font tags between the <a href> and the tag: here or using a style setting in the <a> tag: here.

Defining link targets – It is evident that a link is usually open in the current window or frame by default. However, in some cases it is required to be opened in another window or frame. It is accomplished by adding a target="" to the <a href>. For example: <a href="http://www.hotmail.com" target="_blank">. The _blank loads the page into a new browser window.

Other targets are _self; _parent; and _top that load the web page into the current window, the frame that is superior to the frame the hyperlink is in and cancels all frames, and loads in full browser window respectively.

Defining link within a page – It is required to create a link pointing to the anchor. As it has already been mentioned that an anchor is created using the <a> tag. For example, if an anchor is created for TOYS in an online shopping mall, this word Toys is simply added where it is anchored. The HTML cod is: and then a link pointing to the anchor using the normal <a href> tag, like this: here.

When it is required to create a link to anchors on external WebPages, the syntax is: blabla

Defining links for a frameset – A link in a frameset may provide link to a web page that is loaded in the other frame window. Take an example website having tutorials in a frameset called Contents where different links such as Lesson 1, Lesson 2, and Lesson 3 etc are created. The HTML code to go at Lesson 3 will be like: Lesson 2 of the tutorial.

Defining image link – A technique called image mapping is used to link one image to several pages by simply specifying which of the areas of the image should link to where. In other words it explains that an user can go to different websites by simply clicking at different portions of an image. For example:

```
<img src="rainbow.gif" usemap = #example border=0>
<map name=example>
<area shape=circle coords=0,0,29,29 Href="http://www.hotmail.com">
<area shape=circle coords=30,30,59,59 Href="http://www.google.co.in">
</map>
```

In the example above, if mouse is clicked at the upper left corner it links to hotmail website and if it is clicked at the lower right corner, it links to Google website.

For other shapes we can use:

```
<area shape=rect coords= x1,y1,x2,y2 Href="http://www.domain.com"> for Rectangles
<area shape=circle coords= x1,y1,x2,y2 Href="http://www.domain.com"> for Circles
<area shape=polygon coords= x1,y1,x2,y2,..,xn,yn Href="http://www.
domain.com"> for Polygons
```

Defining link to the new window – in order to open a page in a new window use the target="_blank" in the <a href> tag. It simply opens a new browser window that will load the linked page. For example linking to the hotmail, the link will be like this: Go to Hotmail.

Defining links to send an email – In order to send email, links are created almost in a similar manner as it is done to link other pages: <a href> tag. The HTML code for the email link is: Email Me.

If a special subject is needed to be added in the email, it can be done using subject= setting: Send Email.

An email link for specific text in the body of the message can be accomplished by simply adding &body=: Send Email

All the above options can be combined in a single email. It will look like: Email Me

The browser distinguished Hyperlinks from normal text. Every Hyperlink,

- Appears blue in color
  - ❖ The default color setting in a browser for Hyperlink text or image
  - ❖ The color can be set dynamically via an HTML program, if required.
- The hyperlink text/image is underlined
- When the mouse cursor is placed over it, the standard arrow shaped mouse cursor changes to the shape of a hand.

The blue color, which appears by default, can be over-ridden. To change these link colors, there are three attributes that can be specified with the <BODY> tag. These are:

LINK	Changes the default color of a hyperlink to whatever color is specified with this tag. The cursor can specify the color name or an equivalent hexadecimal number.
ALINK	Changes the default color of a hyperlink that is activated to whatever color is specified with this tag. The user can specify the color name or an equivalent hexadecimal number.
VLINK	Changes the default color of a hyperlink that is already visited to whatever color is specified with this tag. The user can specify the color name or an equivalent hexadecimal number.

Links are created in a web page by using the <A> </A> tags. Anything written between the <A> </A> tags becomes a hyperlink/hotspot. By clicking on the hyperlink navigation to a different web page or image takes place.

The document to be navigated to needs to be specified. By using the HREH attribute of the <A> tag the next navigable web page or image can be specified.

Syntax

```
<A HREF = "filename.htm">
```

Hyperlinks can be of two types:

- Links to an external document
- Links (jumps) to a specific place within the same document

2.7.1 Linking to other html documents

Example

<A HREF = "details.htm"> Visit my Home Page </A>

Here *Visit my Home Page* becomes a hyperlink, and links to another document, *details.htm*, which is present in the current working directory. If the file is not present in the current directory, a relative or absolute path can be specified.

2.7.2 Linking inside the same documents

Sometimes, a jump is required to a different location in the same document.

Syntax

<A NAME = "location_name")

<A HREF = "# location_name") ..... </A>

Internal links behave identical to external links with one exception - you can use relative URLs for internal links. A relative URL simply drops the common part from the URL and lets the browsers automatically figure out the part that is missing. For example, instead of specifying <A HREF = "http://www.myserver.com/history.htm"> MyPage </A> just specify the part that's different from the current page's URL:

<A HREF = "history.htm"> My Page </A>

Tag: <BASE> ... </BASE>

Tag Name: Relative Addressing base

It occurs within <HEAD> ... </HEAD> and establishes the URL basis for subsequent URL references in <LINK> or anchor statements in the document body.

Attributes:

- HREF = "URL"

  States the absolute or relative URL for the current document.

- TARGET = "window"

There are four pre-defined names that can be targeted by an anchor. They all start with an underscore (_).

<A HREF = "document.htm" Target = "_blank"> My document </A>

Clicking on My document would cause a new browser window to appear, containing the document.htm file.

For example:

<A HREF = "document.htm" Target = "_parent"> My document </A>

Clicking on My document would cause the document.htm file to appear in the parent frameset.

<A HREF = "document.htm" Target = "_top"> My document </A>

Clicking on My document would cancel all of the frames and replace the entire frameset with the document.htm file.

Tag: <Link> ... </Link>

Tag Name: Link

The link element indicates relationships between your documents and other documents or URLS.

Attributes:

- HREF = "URL"

  The address of the current link destination, accessible through normal web linkage mechanisms.

- MEDIA = SCREEN|PRINT|PROJECTION|BRAILLE|SPEECH|ALL

  Identifies the ideal environment for the web page to be conveyed in. The default is ALL.

- REL = "text"

  It indicates a normal relationship to the document specified in the URL.

  e.g.: <LINK REL = "INDEX" HREF = "index.htm">

  This tag would let browsers know where they can find the main index for your Web site.

- REV = "text"

  It indicates a reverse relationship. The referenced document has the indicated relationship to the current document.

  e.g.: <LINK REV = "Index" HREF = "history.htm">

  This tag would let browsers know that there is a two-way relationship between history.htm and index.htm - that index. htm is the index of history.htm, and simultaneously, history.htm is indexed by index.htm.

- Target = "window"

  Specifies loading the link into the targeted window.

- Type = "text"

  Specifies the Internet media type of linked resource. For example, the type for CSSI sheets is "TEXT/CSS".

2.8 LINKING TO OTHER INTERNET SERVICES

2.8.1 File Transfer Protocol (FTP)

The File Transfer protocol is among the oldest protocols still used in the Internet. FTP is widely available on almost all-browsers indicating that all computing platforms, including DOS, OS/2, UNIX, and up to the mainframe level have this service available. You can very well understand from its name that it facilitates the majority of file transfers across the Internet. In other word, FTP is a file server access protocol that enables a user to transfer files between two hosts across the network or Internet using TCP. You may see the versatility of this application layer protocol, it accomplishes its job even intended hosts at separate locations could potentially be running different operating systems, using different file storage systems, and using different character sets. Accessing FTP sites over the Internet requires that the user must have the knowledge of the location and the name of the desired files.

Unlike Telnet, FTP does not require any familiarity with the remote operating system. The user is still required, however, to be familiar with the FTP command set built into the protocol itself so that he or she can productively manage the session.

Modern FTP servers known as ftpd support two different TCP connections, namely control and data connections. First control connection is invoked for the entire duration of transfer of file or FTP session. It facilitates the exchange of commands issued by the client, and replies originating from server. Data connection is established as and when it is required. Its main function is to facilitate transfer of files and directory listings to and from the client at the client's request.

Whenever you wish to do FTP, you need to invoke a few commands. These commands basically are related to transfer a file from remote computer to your computer or from your computer to the remote computer. There are anonymous as well as authorized privileges with regard to transfer of a file from a server. In case of anonymous FTP servers, you can do FTP without authorization. You need to login with a username, which is anonymous, and a password that is your e-mail address. Apart from this, there are authorized servers for which you need to register before you are permitted to do FTP. After registration, you will get a password.

Trivial File Transfer Protocol (TFTP)

TFTP, like FTP, is also an Internet service intended for the transfer of files between from one computer to another over a network. It does not provide password protection or user directory capability. Unlike FTP, however, TFTP does not rely on TCP for transport services. Instead, TFTP uses UDP to shuttle the requested file to the TFTP client. Furthermore, diskless devices that keep software in ROM to use it to boot themselves can use it. It is simpler than the File Transfer Protocol (FTP) but less capable. FTFTP facilitates to quickly send files across the network with fewer security features than FTP.

2.8.2 Gopher Service

Gopher is client/server-oriented software that uses a simple set of rules to hunt for and retrieve files from Gopher servers on the Internet. The Gopher service was developed by the University of Minnesota in 1991 to conquer some limitations of the FTP service. Gopher has an easier-to-use interface and also allows administrators to create links to other computers or services, to gloss files and directories, and to create custom menus.

Gopher is not just an Internet tool. Many organizations use Gopher on their local area network to help people within the organization find the information they need quickly and efficiently.

The user of the Gopher client can download files, switch directories, or link to other Gopher servers by using a sequence of menus. The Gopher server generates menus, links, and annotations by using a series of tag files.

Gopher presents information in a hierarchical structure. Relying on which client software is used and what choices are obtainable on the Gopher server, the user can opt how to vision information—for instance, as a text file, as a Microsoft Word for Windows document, or in a particular language.

A Gopher client presents the individual user with directory lists. If the user chooses a subdirectory from the displayed list, the listing for that subdirectory is displayed. If the user selects a file, it is downloaded. Each directory and file can be on a different Gopher server.

You can also organize a Gopher server to search local Wide Area Information Server (WAIS) databases.

Gopher uses TCP as its transport protocol for all communication and data exchanges between the client and the server. Internet Information Server communicates with Windows Sockets, then Windows Sockets interacts with TCP.

TCP is a connection-oriented protocol (that is, the communications session is established between the client and the server before data is transmitted). However, unlike FTP, Gopher does not maintain the connection between requests; this is also called as a stateless connection.

Gopher Service Configuration

Administrators use Internet Service Manager to configure the following Internet Information Server Gopher Service components:

- Gopher User Access

  Manages user access to Gopher Service
- Gopher IP Access

  Manages computer access to Gopher Service
- Gopher Connection Parameters

  Manages connections to the Gopher Service
- Gopher Resource Location

  Manages location and view of Gopher Service resources
- Gopher Logging

  Manages Gopher Service logs

Gopher Service user access mentions which Windows NT Server logon account the Internet Information Server uses. Most Internet sites use anonymous Gopher logons.

Gopher Service IP access controls which computers can use the Gopher Service. It uses the customer computers IP address to control access.

Gopher Service connection parameters control connection time-out periods, maximum number of Gopher Service connections, and anonymous logons to the Gopher Service.

A Gopher Service URL resource location is an a.k.a. for an absolute path name on a Windows NT Server.

An example of a Gopher URL is

gopher://www.infomax.com/gopher/

Gopher Service information is logged to a database, file, or not at all. A new log file is produced at specified intervals.

The assigned protocol port number for Internet Information Server Gopher service is 70.

Check Your Progress

1. Define URL.
2. What are html tags?

2.9 LET US SUM UP

Files travel across the Internet and carry information from the Server to the Client in the form of web pages. Computers called Servers store web pages in the form of directories and files and provide these files to be read. To access web pages users connect to a Web Server and computers called Web Clients provide the facility to read the information stored in web pages. Web Clients run special software called Browser, which helps connect web clients to appropriate web Server that takes place in four steps namely establishing connection, requesting from the client and reply from the server and finally the last step in which the server terminates connection. Hyper Text Markup Language (HTML) along with standard generalized markup language (SGML) and XML are used to specify the logical organization of a document with important hypertext extensions. HTML editors like Adobe PageMill and Amaya help to write these languages or even write them. HTML tags are instructions embedded directly into the text of the document and are of two types, paired tags and singular tags. HTML documents are structured into head and body parts. Head contains information about the document such as its TITLE and the body contains the body of the text and where the document material is displayed. Web pages could have a title and a footer which is done with the TITLE tag and <ADDRESS>..</ADDRESS> tag. For text formatting paragraph breaks and line breaks are provided with the tags <P> and
 respectively. For emphasizing material in a web page horizontal styles and horizontal rules help break text into logical sections with visual appeal. Various text styles like Bold, Italics and Underline, Centering can be done with tags along with controlling font size and colour.

HTML links allow linking of HTML documents and to images. Hypertext, Hyperlink or Hotspot are the text or image providing the linkages. Hyperlinks are distinguished from normal text by the browser. Hyperlinks can be of two types. Links maybe to an external document or to a specific place within the same document. Images can also act as hyperlinks. Clicking on any part of the image will lead to opening of the document specified in the link attribute. Image maps provide links to multiple documents. Linked regions of an image map are called hot region. A two-step process in which the image is first divided into various areas and secondly the image map is applied to particular images creates image map.

2.10 KEYWORDS

Web Pages: Files, which carry information across the Internet.

HTML: Hyper Text Markup Language, which are designed to specify the logical organization of a document along with important hypertext extensions.

HTML Editors: Tools, which help to write HTML.

HTML Tags: They are instructions embedded directly into the text of a document.

Structure: Parts of the HTML document.

Hyperlinks: Image that provides linkages of one HTML document to another.

Hypertext: Text that provides linkages of one HTML document to another.

HTML Link: HTML links allow linking of HTML documents and to images

2.11 QUESTIONS FOR DISCUSSION

1. Write about the Web browsers.
2. What is Hyper Text Markup language? What are the various HTML Tags?
3. Write a HTML code having some meaningful text, using the italics, center, paragraph, break, and font tags.
4. Explain how to get a web page into a web browser.
5. What is HTTP?
6. Differentiate between paired and singular tags.
7. Create a simple HTML file named Trial.htm with a title 'Welcome to HTML Programming'.
8. Experiment with different attributes of the <BODY> tag such as BGCOLOR by changing the color of the page and the text color and also by using a Gif file as background by using the BACKGROUND attribute.
9. Use the paragraph breaks and line breaks for text formatting and see the difference with and without these formatting.
10. Experiment with drawing lines and aligning them at different positions on the browser with different sizes and widths.
11. Create a HTML code in which hyperlinks are there and experiment with the various attributes for changing the link colors.
12. Create a HTML code, which has images acting as links to another image and so on.
13. Create two files named linkfrom.htm and linkto.htm and link these two files together, by putting the two files in the same current directory.
14. Create a document with two links to an external document. The first link should lead to the beginning of the external document. The second link should lead to a particular section in the external document.
15. Differentiate between HTTP and FTP? Give some advantages of both protocols.

Check Your Progress: Model Answers

1. Though TCP/IP serves its role very well, it won't be easy remembering the TCP/IP address of every website. To simplify this, we have an English equivalent of each web site's address. This English address is also unique and yet, it is easy to remember. This is known as the URL.
2. Tags are instructions that are embedded directly into the text of the document. An HTML tag is a signal to a browser that it should do something other than just throw text up on the screen. By convention all HTML tags begin with an open angle (<) and end with a close angle bracket (>).

2.12 SUGGESTED READINGS

Adolfo Rodriguez, John Gatrell, John Karas, and Roland Peschke, *TCP/IP Tutorial and Technical Overview (7th Edition)*, Prentice Hall.

Martin W. *Murhammer and Eamon Murphy, TCP/IP Tutorial & Technical Overview*, Prentice Hall.

Erik T Ray, *Learning XML*, Second Edition O'Reilly Media.

Michael Morrison, *HTML and XML for Beginners*, Microsoft Press.

Lisa Lopuck, *Web Design for Dummies*, Bk&CD-Rom edition.

Danny Goodman, *Dynamic HTML: The Definitive Reference* (2nd Edition), O'Reilly Media; 2nd edition.

Ed Tittel, Stephen J. James, *HTML for Dummies* (Paperback), For Dummies; 3rd edition.

UNIT II

LESSON

3

HTML LISTS

CONTENTS

3.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Define lists
- Discuss Types of lists such as unordered list, ordered list, etc.

3.1 INTRODUCTION

HTML provided several elements for making lists. This lesson will provide an introduction to the types of lists namely the regular list under which comes the ordered and unordered list and the definition list or the glossary list. In this lesson we will show displaying text in lists through various list types.

3.2 LISTS

HTML supports several elements for making lists. They can be divided into two types: glossary lists, and regular lists. Glossary Lists are denoted by the element <DL>, while regular lists are denoted by the elements <UL>, <OL>, <MENU> and <DIR>. Lists can be nested. Thus you can have a regular list within a regular list, a regular list within a glossary list, and so on.

3.3 REGULAR LISTS

A regular list is a sequence of paragraphs, each of which may be preceded by a special mark, sequence number, or nothing at all. The syntax is:

<UL>

<LI> list element </LI>

<LI> another list element ... </LI>

</UL>

where the opening element defining the list type can be one of

- ***UL:*** A list of multi-line paragraphs, listed separately and usually marked by a bullet or similar symbol (Unordered List)
- ***OL:*** A list of multi-line paragraphs, listed separately and ordered numerically in some way (Ordered List)
- ***MENU:*** An Unordered List of smaller paragraphs, this is similar to UL but is formatted (if possible) in a more compact manner. *Note:* MENU was dropped from the language as of HTML 4.0 - you should use UL instead.
- ***DIR:*** A list of short elements, typically less than 20 characters in length. These may be arranged in columns across the page, as opposed to one above the other. This is browser dependent. ***Note:*** DIR was dropped from the language as of HTML 4.0 - you should use UL instead.

Here are four examples, showing the rendering for the four different types. The text we will format is as follows:

Hi. The following is an example list.

<UL>

<LI> list element. The quick brown fox jumped over the

lazy dog. The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

<LI> another list element ...

</UL>

3.3.1 Netscape Extensions to List Elements

Netscape introduced several "extension" attributes for list elements, all of which were subsequently integrated into standard HTML. Essentially all browsers now support these attributes.

OL Element – START and TYPE Attributes

These attributes control the numbering of ordered lists.

- START -- specifies the starting number for a numbered list. Thus START="4" means that the list will start with the number 4.
- TYPE = "A", "a", "I", "i" or "1" -- specifies *how* the items are numbered. "A" or "a" indicate letter ordering (upper or lower case), while "I" or "i" indicate roman numerals (upper or lower case). "1" is the default, and indicates numerical ordering.

UL Element – TYPE Attribute

The Netscape browser supports a TYPE attribute to the UL element for specifying the *type* of bullet to be used with the list items. Possible values are "disc" (filled disc), "circle" (open circle) and "square" (open square).

LI Element – TYPE and VALUE Attributes

The Netscape browser supports TYPE and VALUE attributes for the LI element. VALUE is valid only inside an OL list, and specifies the *item number* for the given list item. All subsequent items are ordered from this number. The TYPE attribute specifies the type of numbering ("A", etc.) or bulleting ("circle", etc.) depending on whether the LI is inside an OL or UL list.

3.3.2 Unordered List (Bullets)

An unordered list starts with the tag <UL> and ends with </UL>. Each list item starts with the tag <LI>. The attributes that can be specified with <LI> are:

TYPE	Specifies the type of the bullet. TYPE = FILLROUND will give a solid round black bullet TYPE = SQUARE will give a solid square black bullet.

3.3.3 Ordered Lists (Numbering)

An ordered list starts with the tag <OL> and ends with </OL>. Each list items start with the tag <LI>. The attributes that can be specified with <LI> are:

TYPE	Controls the numbering scheme to be used. TYPE = "1" will give counting numbers (1, 2, ...) TYPE = "A" will give Uppercase letters (A, B, C, ...) TYPE = "a" will give Lowercase letters (a, b, c, ...) TYPE = "I" will give Uppercase Roman Numerals (I, II, III, ...) TYPE = "i" will give Lowercase Roman Numerals (i, ii, iii, ...)
START	Alters the numbering sequence. Can be set to any numeric value.
VALUE	Changes the numbering sequence in the middle of an ordered list. It is to be specified with the <LI> tag.

3.3.4 List Examples

Here are some examples.

<ol start=4 type="i">

First item

Second item

<li value=2>Third item; value=2

<li type="A">Second item; type="A"

which is rendered as:

(i) First item

(ii) Second item

(iii) Third item; value=2

(iv) Second item; type="A"

3.4 DEFINITION LISTS

Definition list values appear within tags <DL> and </DL>. Definition lists consists of two parts.

Definition term	Appears after the tag <DT>
Definition description	Appears after the tag <DD>

DL Element: Glossary Lists

This list type, also known as a definition list, is used to present a list of items along with descriptive paragraphs. This can be used for glossaries, but is also useful for presenting a named list of items and their meanings. The items within the list are introduced by the two elements:

<DT> - The `Term' (a single line)

<DD> - The `Definition' (may be multiple lines)

DL can take a single attribute, COMPACT, to signify that the list is small (or large) and should be rendered in a physically compact way. This attribute is ignored by several browsers.

Example of DL Lists

Here is an example from the Paragraphs section of this manual (with a small addition to show a nested regular list:)

```
<dl>
<dt> Things to Avoid: </dt>
    <dd> You should NOT use elements that define paragraph
    formatting within the PRE element. This means you should
    not use <code> &lt;P>, &lt;ADDRESS>, &lt;Hn>
```

```
    </code>and so
    on. You should avoid the use of tab characters - use single
    blank characters to space text apart. </dd>
<dt> Things That are OK: </dt>
    <dd>You <em> can </em> use Anchor.  A typed
    carriage return will cause a new line in the presented text.
    People you should never let format lists include:
    <ul>
    <li> Bozo the Clown </li>
    <li> Uncle Fester </li>
    <li> Knights who say nii </li>
    </ul>
    Which would be downright silly in the first place. </dd>
</dl>
```

Example

```
<DL>
      <DT> Keyboard
      <DD> An input device
      <DT> Printer
      <DD> An output device
</DL>
```

Output

```
Keyboard
      An input device
Printer
      An output device
```

In addition to the above types of lists, HTML also allows two other types of lists that are not very commonly used:

- Directory lists
- Menu lists

3.5 DIRECTORY LISTS

Tag: <DIR> ... </DIR>

Tag Name: Directory List

This block-level element marks unbulleted list of short elements, such as filenames.

```
e.g.: <DIR>
      <LI> Item 1
      <LI> Item 2
      <LI> Item 3
      </DIR>
```

3.6 MENU LIST

Tag: <MENU> ... </MENU>

Tag Name: Menu List

It encloses a menu list in which each element is typically a word or short phrase that fits on a single line.

This list is rendered more compactly than most other list types.

Attributes: COMPACT

u Renders the list as compactly as possible by reducing line leading and spacing.

e.g.: <MENU>
<LI> Sourdough
<LI> Butter milk
<LI> Rolls
</MENU>

3.7 COMBINING LIST TYPES

We can also use a new list within itself or even one type of list inside another type of list. Each time you use a list within another list, you are nesting lists. Perhaps the best example for nested lists is an outline like those formed for a term paper.

I. Introduction

II. Part 1

A. Description

B. Examples

1. Reference One

2. Reference Two

III. Part 2

IV. Summary

Can you visualize what the HTML code would appear like for the preceding outline? The finest solution would be to use a sequence of nested ordered lists as shown in the following illustration and code.

```
<ol type="I">
<li>Introduction</li>
```

```
<li>Part I
<ol type="A">
<li>Description</li>
<li>Examples
<ol type="1">
<li>Reference One</li>
<li>Reference Two</li>
</ol>
</li>
</ol>
</li>
<li>Part 2</li>
<li>Summary</li>
</ol>
```

As we discussed before, you can also shell one type of list within another type. For instance, you could comprise a bulleted list inside a definition list to provide additional clarification to a definition description. Look at the following illustration and code to observe what I mean.

```
<dl>
<dt><b>Morning</b></dt>
<dd>Corinna wakes up around 7:00.</dd>
<dd>Change her diaper and dress her.</dd>
<dd>Feed her breakfast. Some of her favorites are:
<ul>
<li>Waffles</li>
<li>Oatmeal</li>
<li>Cereal</li>
<li>Fruit</li>
</ul>
</dd>
</dl>
```

Check Your Progress

1. Define regular list.
2. What is glossary list?

3.8 LET US SUM UP

The various types of list are unordered list and ordered lists. Unordered list starts with the tag <UL> and ends with </UL> and ordered list starts with the tag <OL> and ends with </OL>. Definition list starts with the tag <DL> and ends with </DL> and consists of two parts, definition term and definition description.

3.9 KEYWORDS

Unordered List: Regular list provided for making list of items.

Ordered List: Regular list provided for making list of 8 items and provides control of the numbering scheme to be used.

Definition List: List used to present a list of items along with descriptive paragraphs.

Directory List: It marks unbulleted list of short elements, such as filenames.

Menu List: It encloses a menu list in which each element is typically a word or short phrase that fits on a single line.

3.10 QUESTIONS FOR DISCUSSION

1. Write about the various types of lists.
2. What is Netscape Extension to List Elements?
3. Create a HMTL code to show an ordered list of 8 items using the <OL></OL> tag and experiment with the various types of numbering schemes that can be used with the attribute TYPE of <LI> tag.
4. Create a HTML code which will show the given below list on the web page:
   (a) Apple
   (b) Mango
   (c) Orange
   (d) Grapes
5. Differentiate between ordered and unordered list.
6. Create a HTML code to show an unordered list of 8 items using the two types of bullet (fillground and square)
7. Change the numbering sequence so that apple starts with C instead of A.
8. What are the parts of the definition list and the tags used for them?
9. What are definition list used for?
10. Write a HTML code which will give a list of 5 terms with their definition description using the <DL></DL> tag.
11. Give an example of combining list types.

Check Your Progress: Model Answers

1. A regular list is a sequence of paragraphs, each of which may be preceded by a special mark, sequence number, or nothing at all.
2. Glossary also known as a definition list, is used to present a list of items along with descriptive paragraphs. This can be used for glossaries, but is also useful for presenting a named list of items and their meanings.

3.11 SUGGESTED READINGS

Steven Holzner, HTML Black Book: *The Programmer's Complete HTML Reference Book*, Paraglyph Press.

Natanya Pitts-Moultis, Dynamic HTML Black Book: *The Web Professional's Guide to Using and Interacting with Dynamic HTML*, Coriolis Group Books; Bk&CD-Rom edition.

Lisa Lopuck, *Web Design for Dummies: For Dummies*; Bk&CD-Rom edition.

LESSON

4

GRAPHICS AND WEB PAGES

CONTENTS

4.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss image format and browsers
- Understand graphics and HTML documents
- Discuss image and hyperlink anchors
- Understand image maps

4.1 INTRODUCTION

This lesson will introduce to how you can add graphics to HTML documents. One can add static as well as animated pictures. You will learn the use of Border attribute, which allows you to specify the sizes of borders that you are placing on the html documents. Along with the border attribute you will learn other attributes like the Width and Height attribute, Align attribute with which you can manipulate your pictures according to the shape you want as well as using the ALT attribute enables you to place some text in case a particular browser is unable to display your image.

4.2 IMAGE FORMAT AND BROWSERS

To display an image on a page, you need to use the SRC attribute. SRC stands for "source". The value of the src attribute is the URL of the image you want to display on your page.

The syntax of defining an image i.e. Here's the format for placing an image:

```
<img src="url">
```

The URL points to the location where the image is stored.

SRC give the URL of the image document -- this attribute must be present. The naming scheme is the same as for hypertext links: thus relative URLs such as SRC="foo.gif" or SRC="../foo.gif" are commonly used. At present you can only inline GIF images and X-bitmaps (standard extensions are .gif, .xbm, .xpm), although many (but not all) browsers also support inline JPEG (standard extensions .jpeg or .jpg) images as well.

ALT = "alternative text"

Some browsers cannot display images, while users connecting via phone lines often turn off image loading. The ALT attribute lets you specify a text alternative to the image, for use in these circumstances. You should *always* include an ALT alternative, particularly if the image is a button linked to some other resource. If the image is unimportant, you can always put ALT="".

ALIGN = "bottom", "middle", "top"

ALIGN tells the browser how to align the image with the neighbouring text. "Bottom" aligns the bottom of the image with the baseline of the text, and is the default. "Middle" aligns the middle of the image with the baseline of text, and "top" aligns the top of the image with the top of the tallest item in the line. This attribute is optional. "left", "right" (HTML 3.2) HTML 3.2 supports left and right-aligned images. In this case, the image "floats" to the left or right margin, with the text following the image element flowing around the image. This attribute value is supported by all current browsers. "absmiddle", "absbottom", "texttop", "baseline" (HTML 3.2) Netscape introduced these extra attributes, which work like "top", "middle" and "bottom" but which give additional vertical control of image placement, when the image flows with the text. These are illustrated in the example document.

```
HEIGHT="n", WIDTH="n" (HTML 3.2)
```

HEIGHT and WIDTH specify the display height and width (in pixels) for the image -- if the picture does not fit, the browser should rescale the image to fit in the specified box. An example is HEIGHT="30" WIDTH="50" This is supported by most current browsers. Some also support percent values (a percentage of page size, or sometimes a percentage of available size inside a table cell) -- but you had better test percentage values, to make sure things work as expected.

4.2.1 Using the Border Attribute

Example 1:

```
<HTML>
    <HEAD>
        <TITLE> Working with Images </TITLE>
```

```
        </HEAD>
        <BODY NACKGROUND = "images/texture1.gif">
                <B> Controlling Image Borders! </B>
                <CENTER>
                        <I> Image without a Border</I> <BR><BR>
                        <IMG SRC = "images/corp.gif"><BR><BR>
                        <I> Image with BORDER = 3</I><BR><BR>
                        <IMG BORDER = 3 SRC = "images/corp.gif"><BR>
                </CENTER>
        </BODY>
</HTML>
```

4.2.2 Using the Width and Height Attribute

Example 2:

```
<HTML>
        <HEAD>
                <TITLE> Working with Images </TITLE>
        </HEAD>
        <BODY BACKGROUND = "images/texture1.gif">
                <B>Controlling Images Sizes </B>
                <CENTER>
                <I> Normal Image Size </I><BR><BR>
                <IMG SRC="images/computer.gif"><BR>
                <I>Image With Size (Height and Width) Set To 200</I>
<BR><BR>
        <IMG WIDTH=200 HEIGHT=200 SRC="images/computer.GIF"><BR>
        </CENTER>
        </BODY>
</HTML>
```

4.2.3 Using the Align Attribute

Example 3:

```
<HTML>
        <HEAD>
                <TITLE> Working with images </TITLE>
        </HEAD>
        <BODY BACKGROUND = "images/texture1.gif">
```

```
            <B><I> Image Aligned Left</I></B>
      <IMG SRC = "images/sctonly2.gif" ALIGN=left><BR><BR>
      <B><I> Image Aligned Right</I><B>
      <IMG SRC="images/scronly2.gif" ALIGN=right><BR><BR>
      </BODY>
</HTML>
```

4.2.4 Using the ALT Attribute

Example 4:

```
<HTML>
      <HEAD>
            <TITLE> Working with Images </TITLE>
      </HEAD>
      <BODY BACKGROUND="images/texture1.gif">
            <B> Use of ALT attribute</B><BR>
      <CENTER><I> Available Image: Javacup.gif</I><BR><BR>
      <IMG SRC="images/javacup.gif"><BR><BR>
      <I>Unavailable Image: Javac.gif - Without the ALT Attribute
</I><BR><BR>
      <IMG SRC="images/javac.gif"><BR><BR>
      <I>Unavailable Image: Javac.gif - With the ALT Attribute set to
"Java"</I><BR><BR>
      <IMG SRC="images/javac.gif" ALT="The Java Cup"><BR>
      </CENTER>
      </BODY>
</HTML>
```

4.3 ADDING GRAPHICS TO THE HTML DOCUMENTS

Other than text, HTML allows placing of static and/or animated images in an HTML page. HTML accepts two picture file formats .gif and .jpg. Using tools such as Gif Constructor or Adobe PhotoShop, images can be created to suit the requirements of a web page and saved in these file formats.

Once an image is ready and stored in the above-mentioned formats, it can be inserted into a web page using the tag <IMG>, which takes the name of the image file (*filename*.gif, *filename*.jpg or *filename*.jpeg) as an attribute. In addition, HTML also allows control of the height, width, border, etc. of every image placed on the web page. The <IMG> tag takes the following attributes.

ALIGN	Controls alignment of the text following the image ALIGN = TOP indicates the text after the image to be written at the top, next to the image. ALIGN = MIDDLE indicates the text after the image to be written at the middle, next to the image ALIGN = BOTTOM indicates the text after the image to be written at the bottom, next to the image CONTROLS ALIGNMENT OF THE IMAGE WITH RESPECT TO THE SCREEN ALIGN = LEFT indicates the image is aligned to the left with respect to the screen ALIGN = CENTER indicates the image is aligned to the center with respect to the screen ALIGN = RIGHT indicates the image is aligned to the right with respect to the screen.
BORDER	Specifies the size of the border to place around the image
WIDTH	Specifies the width of the image in pixels
HEIGHT	Specifies the height of the image in pixels
HSPACE	Indicates the amount of space to the left and right of the image
VSPACE	Indicates the amount of space to the top and bottom of the image
ALT	Indicates the text to be displayed in case the Browser is unable to display the image specified in the SRC attribute

Example

```
<IMG WIDTH = 447 HEIGHT = 57 BORDER = 0 HSPACE = 0 SRC = "IMAGE1.GIF"
ALIGN=CENTER>
```

The attributes taken by the <IMG...> are explained in the following examples:

4.4 IMAGES AS HYPERLINKS

Just as text can act as a hyperlink, so also images can act as hyperlinks.

Example

```
<A HREF = "details.htm"><IMG SRC = "micky.gif"> </A>
```

4.4.1 Image Maps

When a hyperlink is created on an image, clicking on any part of the image will lead to opening of the document specified in the <A HREF ...> tag. If the image is a large image and there is a need to link multiple documents to the same image, there has to be a technique that divides the image into multiple sections and allows linking of each section to a different document.

The technique that is implemented to achieve this is an *Image Map*. Image maps can be created and applied to an image so specific portions of the image can be linked to a different file/image.

Linked regions of an image map are called *hot region* and each hot region is associated with a *filename.htm* document that will be loaded into the browser (navigated to) when the hot region is clicked.

Creating an image map is two-step process:

Step One

Create an image map, i.e., divide the image into various areas. This is done using the <MAP> </MAP> tags. The <MAP> tag takes an attribute, NAME, via which the map can be referenced in an HTML file.

Syntax

<MAP NAME = "map name">

Within the <MAP> </MAP> tags the <AREA> tag is specified. This tag defines the specific region within the image. The <AREA> tag takes certain attributes.

The attributes are:

SHAPE	The shape of a region can be one of the following: Rect, circle, polygon, default
COORDS	Each of the above shapes takes different coordinate parameters. A rectangle will take four coordinates: x1, y1, x2, y2 A circle will take three coordinates: centerx, centery, and radius A polygon will take three or more pairs of coordinates denoting a polygon region. A default shape will not take any parameter and it indicates the portion of the image not specified under any Area tag.
HREF	Takes the name of the .htm file that is linked to the particular area on the image.

Step Two

Deals with applying the image map to a particular image. For this purpose, the <IMG> tag takes an attribute called USEMAP that takes the name of the image map as a value, and applies the map specification to the respective image. The value is always preceded with the # sign.

Syntax

<IMG USEMAP = "*#map_name*">

ISMAP

Most browsers also support the ISMAP attribute. This attribute marks the image as an active *image map*. This allows the user to click the mouse over the image and have different regions of the image cause different actions.

BORDER="n" (HTML 3.2)

If ISMAP is used, then the image is a hypertext link, and will be surrounded by a border to indicate this. BORDER specifies the width of this border. In particular, BORDER=0 specifies no border, which is rather cute.

HSPACE="n", VSPACE="n" (HTML 3.2)

These attributes specify the horizontal and vertical space, in pixels, to leave between the image and the surrounding elements.

LOWSRC="URL" (Netscape only)

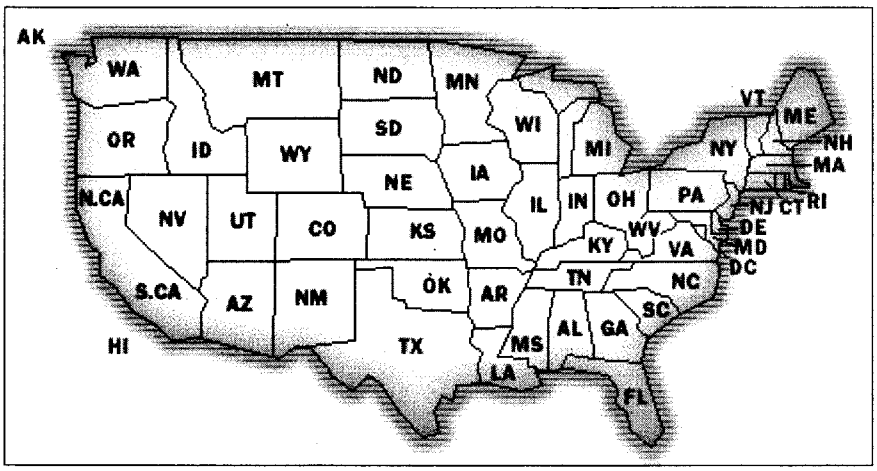
Specifies a low-resolution image file. A netscape browser will first load the smaller, low-resolution file specified by LOWSRC and will then load the larger SRC-specified image file.

USEMAP="url" (HTML 4)

Specifies a URL pointing to a client side imagemap - that is, a MAP element. For example, if the map element starts with <MAP NAME="foo">, then the map would be referenced as <IMG USEMAP="#foo"... >. smaller, low-resolution file specified by LOWSRC and will then load the larger SRC-specified image file.

Example:

As example, the following image is used as a client-side image map *with* alt text on all the areas. As one moves the mouse over areas of the map, the tool top shows the alt text for the area



Check Your Progress

1. Define SRC attribute.
2. Define image maps.

4.5 LET US SUM UP

One can add static as well as animated images along with text in an HTML page. Images in the file format .jpg and .gif are inserted into web pages using the tag <IMG>. For manipulating the layout of the image, the tag <IMG> provides various attributes. The BORDER attribute specifies the size of the border to place around the image. The Width and height attribute specifies the width and height of the image respectively in pixels. The ALT attribute indicates the text to be displayed when the browser used by the users is unable to display the required image.

4.6 KEYWORDS

Graphics: Static or animated images.

Align Attribute: Controls alignment of text following the image.

Border Attribute: Size of the border around the image.

Width: Width of the image in pixels.

Height: Height of the image in pixels.

Alt: Gives alternate text to be displayed in case the image to be displayed is not available to the viewer.

4.7 QUESTIONS FOR DISCUSSION

1. Design a web page using the image files 'House.gif, 'Javacup.gif', and 'Computer.gif' according to the following specifications:

   (a) Use a Border for 'House.gif'

   (b) Resize the Width and Height of 'Javacup.gif' and 'Computer.gif' to 200 pixel each.

   (c) Align the text with respect to the images.

2. What are SRC attributes? Explain with example.
3. What type of picture file formats does HTML accept?
4. Name some tools, other than the one given in the book, with which images can be created for adding to web pages.
5. Discuss the steps for creating image maps.

Check Your Progress: Model Answers

1. To display an image on a page, you need to use the SRC attribute. SRC stands for "source". The value of the src attribute is the URL of the image you want to display on your page.

   The syntax of defining an image i.e. Here's the format for placing an image:

   <img src="url">

2. When a hyperlink is created on an image, clicking on any part of the image will lead to opening of the document specified in the <A HREF ...> tag. If the image is a large image and there is a need to link multiple documents to the same image, there has to be a technique that divides the image into multiple sections and allows linking of each section to a different document. The technique that is implemented to achieve this is an *Image Map*.

4.8 SUGGESTED READINGS

Bud E. Smith, Peter Frazier, and Bud Smith, *Creating Web Graphics for Dummies*, For Dummies; Bk&CD-Rom edition.

Lisa Lopuck, *Web Design for Dummies*, For Dummies; Bk&CD-Rom edition.

LESSON

5

HTML TABLES AND FRAMES

CONTENTS

5.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss HTML tables
- Understand aligning table elements
- Discuss row and column spanning

5.1 INTRODUCTION

A table is a two dimensional matrix, consisting of rows and columns. Tables are intended for displaying data in columns on a web page. All table related tags are included between the <TABLE> </TABLE> tags. Each row of a table is described between the <TR> </TR> tags. Each column of a table is described between the <TD> </TD> tags. HTML tables are discussed in this lesson in detail.

The HTML tags that divide a browser screen into two or more HTML recognizable unique regions is the <FRAMESET> </FRAMESET> tags. Each unique region is called a frame. Each frame can be loaded with a different document and hence, allow multiple HTML documents to be seen concurrently.

The HTML frame is a powerful feature that enables a web page to be broken into different unique sections that, although related, operated independently of each other. We will discuss frames in this lesson in detail.

5.2 TABLES

Table rows can be of two types:

- Header rows *(A row that spans across columns of a table)*

A table header row is defined using <TH> </TH> tags. The content of a table header row is automatically centered and appears in boldface.

- Data rows *(Individual data cells placed in the horizontal plane creates a data row)*

There could be a single centered data cell (i.e., a single column table) or multiple data cells (i.e., a multi column table).

Data cells hold data that must be displayed in the table. A data row is defined using <TR> </TR> tags. Text matter displayed in a data row is left justified by default. Any special formatting like boldface or italics is done by including appropriate formatting tags inside the <TR> </TR> tags. An image can also be displayed in a data cell.

The attributes that can be included in the <TABLE> tag are:

ALIGN	Horizontal alignment is controlled by the ALIGN attribute. It can be set to LEFT, CENTER, or RIGHT.
VALIGN	Controls the vertical alignment of cell contents. It accepts the values TOP, MIDDLE, or BOTTOM.
WIDTH	Sets the WIDTH to a specific number of pixels or to a percentage of the available screen width. If width is not specified, the data cell is adjusted based on the cell data value.
BORDER	Controls the border to be placed around the table. The border thickness is specified in pixels.
CELLPADDING	This attributes controls the distance between the data in a cell and the boundaries of the cell.
CELLSPACING	Controls the spacing between adjacent cells.
COLSPAN	The COLSPAN attribute inside a <TH> or <TD> tag instructs the browser to make the cell defined by the tag to take up more than one column. The COLSPAN attribute can be set equal to the number of columns the cell is to occupy. This attribute is useful when one row of the table needs to be a certain number of columns wide.
ROWSPAN	The ROWSPAN attribute works in the same way as the COLSPAN attribute except that it allows a cell to take up more than one row. The attribute can be set by giving a numeric value. For example, ROWSPAN = 3

5.3 ALIGNING TABLE ELEMENTS

Often tables need to be given a heading, which gives the reader a context for the information in the tables. Table Headings are called Captions. Captions can be provided to a table by using the <CAPTION> </CAPTION> tags. This paired tag appears within the <TABLE> </TABLE> tags. The table caption can be made to appear above or below the table structure with the help of the attribute **ALIGN.**

ALIGN	It controls placing of the caption with respect to the table. ALIGN = BOTTOM will place the caption immediately below the table. ALIGN = TOP will place the caption immediately above the table.

By passing a row's <TR> tag the VALIGN and ALIGN attributes, vertical or the horizontal alignment can be made identical for every cell in a given row.

By passing the <TH> and/or <TD> tags, VALIGN or ALIGN attributes, vertical or horizontal alignments in both header and data cells can be done. Any alignment specified at the cell level overrides any default alignments and any alignments specified in a <TR> tag.

An opening table tag, <TABLE >, may have several attributes in the tag but some attributes are optional or not needed, depending upon the type of table you are making. The value for each attribute inside the tag is enclosed in quotation marks, only around the value and not around the attribute.

```
<TABLE attribute="value">.
```

The ALIGN attribute of the table element defines the horizontal alignment of the table element (in relationship to the next outer "container", usually the body "container", but a table may be made inside another table. Example: ALIGN="center" would center a table inside the next outer container. If the next outer container is the page BODY itself, then the table is centered horizontally within the body of the page. I still use this attribute when editing code using Internet Explorer 7.0, even though this attribute is said to have been deprecated in HTML 4.0. Test your code in more than one browser. If the align attribute is not working the newer fix is to use the <DIV align="center"> and the </DIV> tags before and after your table. I am also currently using the ALIGN attribute inside the <TD ALIGN="right"> to align the cell contents to the right side of the cell. Values for ALIGN may be left, right, or center.

The ALIGN attribute has been used in HTML 3.2 specification and "Transitional HTML" before HTML 4.0, but was officially deprecated in the HTML 4 specification to be supplanted by attributes of style sheets, a planned topic in the DHTML section of this site.

Future browser versions may eventually drop support of the ALIGN attribute and supplant it with attributes of style sheets and positioning with style sheets.

Let me repeat before we go further that several attributes of the <TABLE> element may be used at the same time but many of these attributes are optional. When working with tables and page layout you will want to set values for CELLPADDING and CELLSPACING for each of your tables. Here is an example of several attributes coded for some fictitious table element (not for the table containing the code):

```
<TABLE WIDTH="250" BORDER="2" BORDERCOLOR="blue" CELLPADDING="7"
CELLSPACING="3" VALIGN="top" ALIGN="right">
```

Quotation marks were used for each value. Line breaks when you are writing HTML code in the text editor are acceptable and white spaces between your lines or paragraphs are also acceptable and will not show up in the complete Web page. (White space in the HTML code of the Web document does not get into the Web page.)

ALIGN values of *left* | *center* | *right* have been defined by the HTML 3.2 specification. The default value is "left". If you do not code the align attribute the browser will assume that the table is aligned to the left. If you code <DIV ALIGN="center"> before a table and </DIV> after a table and do not code an align attribute, the table will still be centered.

Note:
The <CENTER> element has been deprecated by HTML 4.0 to be supplanted by coding <DIV ALIGN="center"> and </DIV>.

```
<TABLE ALIGN="center">
```

Text wrapping did not occur. You do not see text on either side of the table aligned in the center. Advanced techniques of style can force the issue but advanced techniques are not covered in this tutorial. The important thing to remember is that when a table is aligned in the center using only HTML coding that the text will not appear on either side of the table.

ALIGN was not mentioned when this table was coded.

Align was not specified for this table. ALIGN="left" is the default value so the browser aligned this table to the far left, even avoiding any margin settings. The text did not wrap to the right of the table because the attribute ALIGN was not coded. Do not assume that if you need a left aligned table and left is the default, then you will not have to code the ALIGN="left" to get text wrap. (See next table).

<TABLE ALIGN="left" STYLE="margin-right:15px;">

When ALIGN="left" is coded text will wrap to the right of the table.

5.4 USING THE WIDTH AND BORDER ATTRIBUTE

Example

```
<HTML>
    <HEAD>
        <TITLE> Table Attributes </TITLE>
    </HEAD>
    <BODY BGCOLOR=LIGHTGREY>
        <B> Specifying the BORDER and WIDTH of the Table</B>
        <BR><BR><BR><BR>
        <CENTER>
            <TABLE BORDER=5 WIDTH=50%>
            <CAPTION ALIGN=bottom>
            <B>Personal Information</B>
            </CAPTION>
```

```
                <TR>
                        <TH>NAME</TH>
                        <TH>AGE</TH>
                </TR>
                <TR ALIGN=CENTER>
                        <TD Tapas</TD>
                        <TD>21</TD>
                </TR>
                <TR ALIGN=CENTER>
                        <TD>Pradeep</TD>
                        <TD>25</TD>
                </TR>
        </TABLE>
        </CENTER>
    </BODY>
</HTML>
```

5.5 USING THE CELLPADDING ATTRIBUTE

Example:

```
<HTML>
    <HEAD>
        <TITLE>Working with Table</TITLE>
    </HEAD>
    <BODY BGCOLOR=LIGHTGREY>
        <B>Specifying CELLPADDING</B><BR>
        <HR>
        <I> Without Cellpadding</I>
            <CENTER>
                <TABLE BORDER=1 WIDTH=25% ALIGN=CENTER>
                <TR>
                    <TH>NAME</TH>
                    <TH>AGE</TH>
                </TR>
                <TR ALIGN=CENTER>
                    <TD>Tapas<</TD>
                    <TD>21</TD>
```

```
                        </TR>
                        <TR ALIGN=CENTER>
                              <TD>Pradeep</TD>
                              <TD>25</TD>
                        </TR>
                  </TABLE>
            </CENTER>
            <HR>
            <I> With Cellpadding of 10</I>
            <CENTER>
            <TABLE BORDER=1 WIDTH=25% CELLPADDING=10 ALIGN=CENTER>
                  <TR>
                        <TH>NAME</TH>
                        <TH>AGE</TH>
                  </TR>
                  <TR ALIGN=CENTER>
                        <TD>Tapas</TD>
                        <TD>21</TD>
                  </TR>
                  <TR ALIGN=CENTER>
                        <TD>Pradeep</TD>
                        <TD>25</TD>
                  </TR>
            </TABLE>
      </CENTER>
      </BODY>
</HTML>
```

5.6 USING THE CELLSPACING ATTRIBUTE

The program can be written in the same manner as above. Instead of CELLPADDING use CELLSPACING.

5.7 USING THE BGCOLOR ATTRIBUTE

The program can be written in the same manner as above. Introduce the following changes in between.

```
.
.
<TABLE BORDER=1 WIDTH=50% ALIGN=CENTER>
      <TR>
            <TH Bgcolor=gray>NAME</TH>
            <TH Bgcolor=gray>AGE</TH>
      </TR>
.
.
.
```

5.8 USING THE COLSPAN AND ROWSPAN ATTRIBUTES

Example:

```
<HTML>
      <HEAD>
            <TITLE>Working with Table </TITLE>
      </HEAD>
      <BODY BGCOLOR=LIGHTGRAY>
            <B>Specifying ROWSPAN and COLSPAN Attributes</B>
            <BR><BR><BR>
            <CENTER>
                  <TABLE BORDER=1 WIDTH=50% ALIGN=CENTER>
                  <TR>
                        <TH ROWSPAN=2>NAME
                        <TH COLSPAN=3>MARKS
                  </TR>
                  <TR>
                        <TH>PowerBuilder
                        <TH>VisualBasic
                        <TH>Oracle
                  </TR>
                  <TR ALIGN=CENTER>
                        <TD>Tapas
                        <TD>25
                        <TD>35
                        <TD>45
```

```
                    </TR>
                    <TR ALIGN=CENTER>
                        <TD>Pradeep
                        <TD>26
                        <TD>36
                        <TD>46
                    </TR>
                    <CAPTION ALIGN=bottom><B><BR> Mark Sheet </B> </CAPTION>
                </TABLE>
                </CENTER>
        </BODY>
</HTML>
```

5.9 NETSCAPE TABLE ENHANCEMENTS

After being contented with standard HTML tagging, you may choose that you are required more advanced techniques to express your information more efficiently and enhance the look of your web site. Netscape offers two useful HTML extensions which can complete both of these goals: tables and frames. Neither tables nor frames have become part of the "official" HTML standard language, but their high utility value and extreme popularity among page designers will likely earn them a standard specification in HTML 3.0 or later.

In the intervening time, use discretion when selecting either extension for your page. Netscape 1.1 or later will support tables, but frames are evident only on Netscape 2.0 or higher. Both are outstanding features for streamlining the organization of your sites, but as with any other HTML enhancement, knowing when and how often to use them is the key to victorious page design.

5.10 THE <FRAME> TAG

Once the browser screen is divided into rows (Horizontal Sections) and columns (Vertical Sections), each unique section defined can be loaded with different HTML documents. This is achieved by using the <FRAME> tag, which takes the following attributes:

SRC="url"	Indicates the URL of the document to be loaded into the frame
MARGINHEIGHT="n"	Specifies the amount of white space to be left at the top and bottom of the frame
MARGINWIDTH="n"	Specifies the amount of white space to be left along the sides of the frame
NAME="name	Gives the frame a unique name so it can be targeted by other documents. The name given must begin with an alphanumeric character
NORESIZE	Disables the frames resizing capability
SCROLLING	Controls the appearance of horizontal and vertical scrollbars in a frame. This takes the values YES / NO /AUTO.

5.11 THE <FRAMESET> TAG

The splitting of a browser screen into frames is accompanied with the <FRAMESET< and </FRAMESET tags embedded into the HTML document. The <FRAMESET> </FRAMESET> tags require one of the following two attributes relying on whether the screen has to be separated into rows or columns.

ROWS	This attribute is used to divide the screen into multiple rows. It can be set equal to a list of values. Depending on the required size of each row, the values can be: A number of pixels Expressed as a percentages of the screen resolution. The symbol *, which indicates the remaining space.
COLS	This attribute is used to divide the screen into multiple columns. It can be set equal to a list of values. Depending on the required size of each column, the values can be: A number of pixels Expressed as a percentage of the screen resolution The symbol *, which indicates the remaining apce.

5.12 TARGETING NAMED FRAMES

Whenever a hyperlink which loads a document in a frame is created, the file referenced in the hyperlink will be opened and will replace the current document that is in the frame.

In a situation where the new document needs to be opened in a different frame while keeping the document from which the new document was navigated open in a different frame, a simple HTML coding technique must be used.

Since the hyperlink must open in HTML file in another frame, the frame in which the HTML file is to be opened needs to be named. This is done by using NAME attribute of the <FRAME> </FRAME> tags. The NAME takes one parameter, which is its frame name.

The hyperlink tag will have to be supplied with the following information:

- The filename.htm file that has to be opened (navigated to).
- The name of the frame where the filename.htm file has to be opened.
- The attribute, via which the frame name is specified is the TARGET attribute, which is a part of the <A> ... </A> tag. This information is given as:

  TARGET = "Framename"

- The attribute, via which the HTML file name is specified is the HREF attribute which is a part of the <A> </A> tag. This information is given as:

  <A HREF = :index.htm" TARGET = "Main"> Visit us </A>

Example:

Frame Identification:

```
<FRAMESET Cols=30%, 70%>
        <FRAME NAME = "Part">
```

```
        <FRAME NAME = "Main">
    </FRAMESET>
```

The above command will divide the browser screen into two vertical frames, the first frame called Part that will occupy 30% of the browser area and the second frame called Main will occupy 70% of the browser area.

Hyperlink Specification:

```
<A HREF = "Index.htm" TARGET = "AMIN"> Visit us </A>
```

Here an HTML file called Index.htm is loaded into the frame named Main when the hyperlink 'Visit us' is clicked.

5.13 USING NESTED FRAMESET

To create both horizontal and vertical frames in a document, you can use nested <FRAMESET> tags. In one tag, you set up the rows, and in the other tag, you set up the columns.

```
<HTML>
    <HEAD>
        <TITLE> Row and Column Frames </TITLE>
    </HEAD>

    <FRAMESET COLS = "50%, 50%">
        <NOFRAMES>
Your browser does not support frames.
        </NOFRAMES>
            <FRAMESET ROWS = "25%, 50%, 25%">
                <FRAME SRC="frame1.htm">
                <FRAME SRC="frame2.htm">
                <FRAME SRC="frame3.htm">
            </FRAMESET>

            <FRAMESET ROWS = "25%, 25%, 50%">
                <FRAME SRC="frame1.htm">
                <FRAME SRC="frame2.htm">
                <FRAME SRC="frame3.htm">
            </FRAMESET>
        </FRAMESET>
</HTML>
```

Check Your Progress

1. Define ALIGN attribute.
2. What is frameset tag?

5.14 LET US SUM UP

Tables in HTML are intended for displaying data in columns on a web page. HTML tables are contained within a TABLE element. The TABLE element denotes the range of the table and uses attribute to define properties of it. Header rows and Data rows are the two types of table rows. The table layout can be manipulated using attributes like Align,Valign,Width, Border cellpadding, cellspacing, colspan and rowspan. Table Headings are called Captions, which are provided to a table by using the CAPTION element. CAPTION provides information for what is given in the tables. Using the Width and BORDER attribute, the width and the border of the table can be manipulated. Using the Cellpadding attribute, the distance between the data in a cell and the boundaries of the cell can be controlled. Using the Cellspacing attribute, the spacing between the adjacent cells can be controlled. The Colspan and Rowspan attributes allow the cell to take up more than one column or row respectively.

HTML tag FRAMESET divides a browser screen into two or more HTML recognizable unique regions called frames. Frames allow multiple documents to be seen concurrently. The Frameset element provides two attributes depending on whether the screen has to be divided into rows or columns. When a new document is to be opened in a different frame while still keeping the document from which the new document was navigated open, the NAME attribute is used to name the frame in which the HTML file is to be opened. The attribute via which the frame name is specified is the TARGET attribute.

5.15 KEYWORDS

Table: A two dimensional matrix, used for displaying date in columns on a web page.

Header Row: Row, which spans across the columns of a table.

Data Rows: Individual data cells placed in a horizontal plane.

Caption Tag: Used for providing table caption, which gives the reader a context for the information in the tables.

Cellpadding Attribute: It controls the distance between the date in a cell and the boundaries of the cell.

Cellspacing Attribute: It controls the spacing between adjacent cells.

Colspan Attribute: It instructs the browser to make the cell to take up more than one column.

Rowspan: It instructs the browser to make the cell to take up more than one row.

Frames: Unique HTML recognizable regions into which the browser screen is divided.

Named Frames: The frame in which a new document is opened while still keeping the one from which the new document was linked.

Target Attribute: Attribute via which the frame name is specified.

5.16 QUESTIONS FOR DISCUSSION

1. Create a web page giving the following plane (flight) details:
   (a) Flight (plane) name
   (b) Starting Place
   (c) Destination
   (d) Arrival and Departure Time
   (e) Fare

   Place a border for the table and use cell padding to present the cell data with clarity. Align the table in the center of the screen. You should enter, at least, details of 5 flights.
2. What are the types of table rows?
3. Create a simple table with 3 columns and 5 rows. Name each column heading as column 1, column 2 and column 3 and enter some random numbers in each cell.
4. Create a simple table with 3 columns and 5 rows. The table should have a border of 6 with cellpadding of 6 and cellspacing of 6 and width of 60%.
5. Create a specimen of a corporate web page. Divide the browser screen into two frames. The frame on the left will be a menu consisting of hyperlinks. Clicking on any one of these links will lead to a new page, which must open in the target frame, which is on the right hand side.
6. What are the advantages and disadvantages of using frames in designing web pages and the care one should take while using frames in web pages?
7. Write the HTML code for creating two vertical frames named frameone and frametwo and in which frameone should occupy 1/3rd of the browser and the rest by frametwo.
8. Write an HTML code for loading a file named load.htm into a frame named framename when you click on a hyperlink named "click here".
9. Create a web page of a travel agency. You can divide the browser window into two frames, in which one frame shows navigation buttons that allow the visitor to the site to move through a set of related pages that are displayed in separate frame.

Check Your Progress: Model Answers

1. The table caption can be made to appear above or below the table structure with the help of the attribute **ALIGN**.

ALIGN	It controls placing of the caption with respect to the table. ALIGN = BOTTOM will place the caption immediately below the table. ALIGN = TOP will place the caption immediately above the table.

2. The splitting of a browser screen into frames is accompanied with the <FRAMESET> and </FRAMESET tags embedded into the HTML document. The <FRAMESET> </FRAMESET> tags require one of the following two attributes depending on whether the screen has to be divided into rows or columns.

5.17 SUGGESTED READINGS

Lisa Lopuck, *Web Design for Dummies* (Paperback), For Dummies; Bk&CD-Rom edition

Danny Goodman, *Dynamic HTML: The Definitive Reference* (2nd Edition), O'Reilly Media.

Ed Tittel, Stephen J. James, *HTML for Dummies (Paperback),* For Dummies; 3rd edition.

LESSON

6

HTML FORMS AND DYNAMIC DOCUMENTS

CONTENTS

6.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss html forms
- Understand <input tag>
- Discuss the concept of dynamic documents such as background graphics and colors, Microsoft internet extensions, etc.

6.1 INTRODUCTION

Forms are the only method of two-way communication between Web pages and Web sites. Getting feedback is where HTML forms come into play. HTML supports a rich variety of input capabilities to let you solicit feedback.

Most new browsers - Netscape Navigator or Communicator 4.0 (and higher) and Microsoft Internet Explorer (Windows 95 version 3.0.3 and higher), plus NCSA Mosaic and its variants - already include HTML 4.0 level forms support, but other browsers do not.

The first step in developing a form is determining which information to include and how to present it, that is, how to break it down into manageable pieces. You then need to ensure that visitors can easily provide the information you want from them, which means that your form needs to be both functional and visually appealing.

Also we will discuss the concept of dynamic documents in this lesson.

6.2 INFORMATION ISSUES

When deciding which information to include and how to break it down, consider your purposes for creating the form. Begin by answering the following questions:

- What information do I want? Customer contact information?
- Why will visitors access the form? To order something on-line? To request information?
- What information can visitors readily provide? Contact information? Previous purchases?
- How much time will visitors be willing to spend filling out the form?

After determining what information you want and what information your visitors can provide, break the information into the smallest chunks possible.

6.3 USABILITY ISSUES

Usability refers to how easily your visitors can answer your questions. Some usability guidelines to consider when you are creating forms are as follows:

Group Similar Categories

When you group similar categories, the form appears less daunting, and visitors are more likely to fill it out and submit it.

Make the Form Easy

- Whenever possible, provide a list from which visitors can choose one or more items. Lists are easy to use, and they result in easy-to-process information.
- Ask visitors to fill in only a small amount of text. This takes minimal time, and it provides you with data that is fairly easy to process.

Many visitors are likely to ignore a request that requires them to enter lots of text.

Provide Incentives

Provide visitors with incentives to fill out the form and submit it. Studies show that a penny or a stamp included in mailed surveys often significantly improves the response rate. Consider offering a chance in a drawing for a free product, an e-mailed list of tips and tricks, or a discount on services.

Design Issues

A well designed form, helps and encourages visitors to give you the information you want. A good form is visually appealing, graphically helpful, and consistent with the remainder of the site. Here are some guidelines:

- Use headings to announce each new group of information. This helps visitor's move easily through the form.
- Be sure to visually separate groups. This makes the forms easier to use because sections become shorter and easier to read through.
- Use text emphases to draw the audience to important information.
- Specify how visitors are to move through the form. Do not make your visitors scroll horizontally to access information. Consider making a narrow, longer form rather than a wider, shorter form to accommodate those who have lower monitor resolution.
- Use arrows to direct visitors through the page. This can help visitors move through the page in a specified order.
- Be sure that it is clear which check boxes and fields go with the associated descriptive information. Use line breaks and spacing to clearly differentiate.
- Specify which fields are optional.
- Use a background image. Be sure, that the image does not outweigh the content and that the text adequately contrasts with the image.
- Make all the text entry fields the same width and put them on the left if you have a vertical column of name and address information; this way all the text will align vertically and look much better.

6.4 CREATING FORMS

Forms have two basic parts:

- The part you can see (that a visitor fills out).
- The part you can not see (that specifies how the server should process the information).

- When adding forms support to a Web page, you must include special tags to solicit input from users. You also include tags to gather input and ship it to your Web server. Here is how this works:
- On a particular Web page, you include tags to set up a form and solicit input from users. This essentially amounts to filling out the form that you supply.
- After users fill out your form, they can then direct their input to the program running on the Web server that delivered the form.
- The information collected from a form can be:
  (i) Written to a file.
  (ii) Submitted to a database, such as Informix or Oracle.
  (iii) E-mailed to someone in particular.

6.4.1 Setting the <Form> Environment

The two key attributes within the <FORM> tag are METHOD and ACTION. Together, these attributes control how your browser sends information to the web server and which input-handling program receives the form's contents.

ACTION = "..." Indicates the program on the HTTP server that will process the output from the form.

METHOD = "..." Tells the browser how to send the data to the server, with either the POST method or the GET method.

e.g.: <FORM METHOD = "POST" ACTION = "http://www.myserver.com/cgi-bin/myperlone.egi"

The line points to the program itself.

6.4.2 Understanding Widgets

- Forms consist of several types of widgets (they are also called controls), which are fields you can use to collect data:
- Submit and Reset buttons send the form information to the server for processing and return the form to its original settings.
- Text fields are areas for brief text input. Use these for several word responses, such as names, search terms, or addresses.
- Select lists are lists from which visitors can choose one or more items. Use them to present a long but finite list of choices.
- Checkboxes allow visitors to select none, one, or several items from a list. Use them to elicit multiple answers.
- Radio buttons give visitors an opportunity to choose only one item.
- Textareas are areas for lengthy text input, as in open-ended comments.

In general, radio buttons, checkboxes, and select lists are all better choices for accepting input than textareas.

6.4.3 Submit and Reset Buttons

The first step in creating a form is to insert the <FORM> tags and add Submit and Reset buttons. Submit and Reset buttons allow visitors to submit information and clear selections.

Basic Form Tags

Tag/Attribute	Use
<FORM>	Marks a form within an HTML document.
<INPUT TYPE = "SUBMIT"	Provides a submit button for a form. The VALUE = "...">
	Value = attribute produces text on the button.
<INPUT TYPE = "IMAGE"	Provides a graphical submit button. The attribute
	indicates the image source file, and the BORDER =
Name = "POINT" SRC = "..."	attribute turns off the image border.
SRC = BORDER = 0>	
<INPUT TYPE = "RESET"	Provides a reset button for a form. The Value =
	attribute VALUE = "..." > produces text on the button.

6.5 THE <INPUT> TAG

Traditionally, the Submit and Reset buttons go at the bottom of the form immediately above the closing </FORM> tag. The following example creates a Submit and Reset button by using the <INPUT> tag, the TYPE = attribute, and the VALUE = attribute.

```
<HR WIDTH = 80% SIZE = 8 NOSHADE>
<FORM>
<INPUT TYPE = "SUBMIT" VALUE = " Submit">
<INPUT TYPE = "RESET" VALUE = "Start Over">
</FORM>
```

If you want to use an image for your submit button, substitute the following code for the submit button.

```
<INPUT TYPE = "IMAGE" NAME = "POINT" SRC = "Submitbutton.gif"
BORDER = 0>
```

The TYPE = "IMAGE" attribute specifies that an image will be used to click on and submit the form. The NAME = "POINT" attribute specifies that the x, y coordinates where the mouse is located will be returned to the server when the image is clicked. Finally, the SRC = and BORDER = attributes work just as they do with regular images - they specify the URL of the image and turn off the border.

6.5.1 Text Fields

A text field is a blank area within a form and is the place for visitor supplied information.

Input Field Tag and Attributes

Tag/Attribute	Use
<INPUT>	Sets an area in a form for visitor input.
TYPE = "..."	Sets the type of input field. Possible values are TEXT, PASSWORD, CHECKBOX, RADIO, FILE, HIDDEN, IMAGE, SUBMIT, AND RESET.
NAME = "..."	Processes form results.
VALUE = "..."	Use this attribute with radio buttons and checkboxes because it does not accept any other input. You can also use this attribute with text fields to provide initial input.
SIZE = "n"	Sets the visible size for a field. Use this attribute with text input fields.
MAXLENGTH = "n"	Sets the longest set of characters that can be submitted.

Traditionally, the Submit and Reset buttons go at the bottom of the form immediately above the closing </FORM> tag. The following example creates a Submit and Reset button by using the <INPUT> tag, the TYPE = attribute, and the VALUE = attribute.

```
<HR WIDTH = 80% SIZE = 8 NOSHADE>
<FORM>
<INPUT TYPE = "SUBMIT" VALUE = " Submit">
<INPUT TYPE = "RESET" VALUE = "Start Over">
</FORM>
```

If you want to use an image for your submit button, substitute the following code for the submit button.

<INPUT TYPE = "IMAGE" NAME = "POINT" SRC = "Submitbutton.gif"

BORDER = 0>

The TYPE = "IMAGE" attribute specifies that an image will be used to click on and submit the form. The NAME = "POINT" attribute specifies that the x, y coordinates where the mouse is located will be returned to the server when the image is clicked. Finally, the SRC = and BORDER = attributes work just as they do with regular images - they specify the URL of the image and turn off the border.

6.5.2 Text Fields

A text field is a blank area within a form and is the place for visitor supplied information.

Input Field Tag and Attributes

Tag/Attribute	Use
<INPUT>	Sets an area in a form for visitor input.
TYPE = "..."	Sets the type of input field. Possible values are TEXT, PASSWORD, CHECKBOX, RADIO, FILE, HIDDEN, IMAGE, SUBMIT, AND RESET.
NAME = "..."	Processes form results.
VALUE = "..."	Use this attribute with radio buttons and checkboxes because it does not accept any other input. You can also use this attribute with text fields to provide initial input.
SIZE = "n"	Sets the visible size for a field. Use this attribute with text input fields.
MAXLENGTH = "n"	Sets the longest set of characters that can be submitted.

Consider the following html script.

```
<FORM>
<INPUT TYPE = "TEXT" NAME = "firstname" SIZE = "30" MAXLENGTH = "30">
</FORM>
```

Guidelines for Including Multiple Text Fields

1. As a rule, forms are much more attractive if the fields are aligned.
2. Here are some guidelines to follow when you include multiple text fields in your form:
   (i) Place the fields at the left margin of your page, followed by the descriptive text.
   (ii) Set the text fields to the same size, when appropriate.
   (iii) As you add descriptive labels, remember to also add line breaks (
 or <P>) in appropriate places.
   (iv) Optionally, add a VALUE = attribute to the text input tag to "seed" the field with a value or to provide an example of the content you want.

6.5.3 Radio Buttons

A radio button is a type of input field that allows visitors to choose one option from a list. Each choice is mutually exclusive - choosing one excludes the remainder. For example:

<P>

Please choose the most appropriate statement.

 <INPUT TYPE = "RADIO" NAME = "one" VALUE = "text book">

I regularly purchase items on-line.

 <INPUT TYPE = "RADIO" NAME = one" VALUE = "novel">

I have on occasion purchased items on-line.

 <INPUT TYPE = "RADIO" NAME = "one" VALUE = "magazine" CHECKED>

I have not purchased anything on-line, but I would consider it.

 <INPUT TYPE = "RADIO" NAME = one" VALUE = "journal">

Use the same NAME = attribute for all radio buttons in a set. Browsers use the name attribute on radio buttons to specify which buttons are related and therefore which ones are set and unset as a group. Add the attribute CHECKED to one of the items to indicate the default selection.

6.5.4 Checkboxes

Each checkbox works independently from others; visitors can select or deselect any combination of checkboxes. Using checkboxes is appropriate for open questions or questions that have more than one " right" answer. In most browsers, checkboxes appear as little squares that contain a checkmark when selected. For example:

<P> I'm interested in (choose all that apply):

 <INPUT TYPE = "CHECKBOX" NAME = "reading"

VALUE = "reading" > Reading

 <INPUT TYPE = "CHECKBOX" NAME = "writing"

VALUE ="writing"> Writing

 <INPUT TYPE = "CHECKBOX" NAME = "editing"

VALUE="editing" > Editing

 <INPUT TYPE = "CHECKBOX" NAME = "collecting"

VALUE = "collecting" > Colleting

6.5.5 Password Fields

Password fields are similar to text fields, except the contents of the field are not visible on the screen. Password fields are appropriate whenever the content of the field might be confidential - as in passwords. Example to establish a password field:

<INPUT TYPE = "password" NAME = "newpass" SIZE = "10" MAXLENGTH = "10">

When viewed in the browser, each typed character appears as an asterisk (*).

6.6 DYNAMIC DOCUMENTS

The standard HTML/XHTML document model is motionless. After displaying on the browser, a document does not modify until the user begin some activity, like choosing a hyperlink with the

mouse. The Netscape developers found that limitation intolerable and built in some special features to their browser that let you change HTML document content vigorously. Actually they provide two different mechanisms for dynamic documents.. Internet Explorer supports some of these mechanisms, which we'll discuss as well.

Certainly you could implant animated GIFs or applets that animatedly update the display, but the primary HTML document itself doesn't change.

We should declare that many of the features of dynamic documents have been showed by plug-in browser accessories and, in specific, applets. However, Netscape and Internet Explorer carry on to support dynamic documents, and we suppose the technology has virtues you should be aware of, if not take advantage of, in your HTML documents.

6.7 BACKGROUND GRAPHICS AND COLOUR

Displaying Wallpaper in the Background

The background of your page can be either an image or a particular colour. To place an image as the background of your page, use the BACKGROUND attribute and position the name of your image file within quotes:

```
<BODY BACKGROUND="image.gif">
```

Like tags, attribute names can emerge in upper or lowercase. For the filename, be sure to use the same capitalization that is used in the actual filename. When a browser displays the page, the image in the file that you mention is cemented to fill the background of the Web page, that is, it is repetitive across and down the page.

Choosing a Background Colour

As an alternative of using wallpaper, you can state a solid background colour for your Web page. Use the BGCOLOR attribute in the <BODY> tag. For the colour, you can enter either a hexadecimal value that represents the colour, or one of 16 standard colour names. The colour names are the following: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. To specify a standard colour, use this tag:

<BODY BGCOLOR="BLUE">

Using a hexadecimal value is safer, if you want your colours to be read by a broad variety of browsers. The hexadecimal value is six characters that signify the amount of red, green, and blue in the colour. The first two characters specify the amount of red, the next two characters the amount of green, and the final two characters the amount of blue. For example, this code specifies light aqua:

<BODY BGCOLOR="#99FFFF">

6.8 MICROSOFT INTERNET EXTENSIONS

Not to be defeated by Netscape, Microsoft has introduced its own browser-specific extensions to HTML. Like Netscape, Microsoft has presented most of these extensions to the World Wide Web Consortium for deliberation in future versions of the HTML standard. Don't be too astonished if you see some of the tags and attributes showing up as standard HTML .

In the intervening time, if you're designing pages for guests that are principally using Internet Explorer 3, you can make use of the HTML extensions mentioned here to enrich your pages and your spectator's experience.

6.8.1 Adding a Background Sound

You can have a background sound play when your Web pages are open by using the <BGSOUND> tag in your document. <BGSOUND> takes the SRC attribute, which is set equal to the URL of a file containing the sound. The file can be in .Wav, .Au, or .Mid (MIDI) format.

<BGSOUND> also takes the LOOP attribute, which lets you mention how many times to play the sound. LOOP can be set to a particular number of times to duplicate the sound or to INFINITE to play the sound as long as the page is open. For instance:

<BGSOUND SRC="greeting.wav" LOOP=3>

prompts Internet Explorer to deliver your greeting three times when the page is opened. The following HTML:

<BGSOUND SRC="greeting.wav" LOOP=INFINITE>

causes Internet Explorer to deliver your greeting as long as the page is open.

6.8.2 Creating a Watermark

You can use an image as the background of your documents by means of the BACKGROUND attribute of the <BODY> tag. BACKGROUND is set identical to the URL of the image to be used. If the image is not big sufficient to fit the entire screen, it will be tiled (both horizontally and vertically) to fill the available space.

If you use the BACKGROUND attribute of the <BODY> tag to tile a graphic as your document background, the background scroll as you scroll through the document. Internet Explorer provides you superior control over scrolling by supporting a BGPROPERTIES attribute. If you set BGPROPERTIES to FIXED, the background image will not scroll as you move through the document, creating a "watermark" effect.

6.8.3 Controlling Your Margins

LEFTMARGIN and TOPMARGIN attributes of the <BODY> tag are supported by internet explorer. You can set either one to the number of pixels of white space you desire Internet Explorer to leave along the left and top edges of the browser window. Adding some space at the margins frequently improves the readability of your documents.

6.8.4 Adding Colour

Internet Explorer 3 supports all of the usual attributes having to do with color (BGCOLOR, LINK, VLINK, and ALINK), plus a few others that are often useful. Specifically, with Internet Explorer 3, you can paint table borders, table cells, and horizontal rule with the color of your choice.

6.8.5 Table Backgrounds and Borders

In the <TABLE> tag, you can mention the BORDERCOLOR attribute to manage which color Internet Explorer uses when depicting table borders. BORDERCOLOR should be set equal to the hexadecimal RGB triplet or an English language color name that explains the desired color.

BORDERCOLOR is functional when depicting a table on a light background. Frequently, there is not a sharp contrast between the background and the table border when the background is white or a light shade of gray. By darkening the border, you produce greater contrast and make the table easier to read.

If you would like to use the shaded rule to achieve a three-dimensional effect, then you can manage the two colors used to generate the shading with BORDERCOLORDARK and BORDERCOLORLIGHT. Each can be set to an RGB hexadecimal triplet or to an English language color name.

You can also use the BGCOLOR attribute in a <TABLE>, <TD>, <TH>, or <TR> tag to modify the background color of a table, table cell, or table row, correspondingly. BGCOLOR is also set identical to a hexadecimal RGB triplet or an English language color name.

Microsoft makes good use of colored table cells on its site. By turning off the borders in the table, the cell looks like a colored, rectangular block floating on the page, and content in the floating block stands out nicely.

6.8.6 Font Tag Enhancements

IE inserts two attributes to Netscape's <FONT> tag: COLOR and FACE. In fact, you may memorize that you were able to alter the on the whole text color in Netscape. In IE, you can alter the color for a single word (or even individual letters, if you've got a lot of time on your hands).

To modify the color of a font in the middle of your document's text, use the <FONT> container with the COLOR attribute, like this:

```
<FONT COLOR="#rrggbb/color name">new color text</FONT>
```

The COLOR attribute can admit either three two-digit hex numbers to explain a color, or a color name itself. For example, both of the following result in red text:

```
<FONT COLOR="#FF0000">This is red text</FONT>
```

<FONT COLOR="Red">This is also red text</FONT>

The FACE attribute can be used to modify the actual typeface used in the IE browser window. Because different systems can offer different fonts, this attribute permits you to propose a list of font names. Each name will be tried in succession until a matching font name is found. The FACE attribute takes the following format:

```
<FONT FACE="name, name2, name3,...">
```

Look at the following example:

```
<FONT FACE="Arial, Helvetica, Times Roman">
```

Your browser will effort to use the font Arial, and then go back to Helvetica and Times Roman until it locate a font match on the user's computer system. If none of the fonts are found, a default font is used.

6.8.7 Scrolling Marquees

The <MARQUEE> and </MARQUEE> tag pair locates a scrolling text marquee on your Web page. The text that scrolls is the text found between the two tags.

The <MARQUEE> tag can take a number of attributes that give you very well control over the look and behavior of the marquee. These attributes are summarized in Table 6.1.

Table 6.1: Attributes of the <MARQUEE> Tag

Attribute	Purpose
BGCOLOR="*RGB triplet*"	Specifies the background color of the marquee window
BEHAVIOR=SCROLL |	
SLIDE | ALTERNATE	Specifies how the text should move in the marquee window
DIRECTION=LEFT | RIGHT	Controls the direction in which the marquee text moves
SCROLLAMOUNT=*n*	Sets the number of pixels of space between successive presentations of marquee text
SCROLLDELAY=*n*	Sets the number of milliseconds to wait before repeating the marquee text
HEIGHT=*pixels* | *percent*	Specifies the height of the marquee window in either pixels or a percentage of the browser window height
WIDTH=*pixels* | *percent*	Specifies the width of the marquee window in either pixels or a percentage of the browser window width
HSPACE=*n*	Specifies how many pixels to make the left and right margins of the marquee window
VSPACE=*n*	Specifies how many pixels to make the top and bottom margins of the marquee window
LOOP=*n* | INFINITE	Controls how many times the marquee text should scroll
ALIGN=TOP | MIDDLE |BOTTOM	Specifies how text outside the marquee window should be aligned with the window

Notice in Table 6.1 that many of the attributes are the same as for the <IMG> tag and that those attributes work in a comparable way. HEIGHT and WIDTH define the dimensions of the marquee, HSPACE and VSPACE administer the space around the marquee, and ALIGN controls where subsequent text appears comparative to the marquee window.

Of the enduring attributes, BEHAVIOR requires some explanation. Setting BEHAVIOR to SCROLL makes the marquee text scroll on, and then off, the marquee window in the direction mentioned by the DIRECTION attribute. If BEHAVIOR is set to SLIDE, the text will slide into the window and stay there. If BEHAVIOR is set to ALTERNATE, the text will rebound back and forth in the window.

Marquee text is a nice Web page effect that you can complete on other browsers only by using something more advanced, like Java or Shockwave. Figure 6.1 shows some marquee text sliding onto a page.

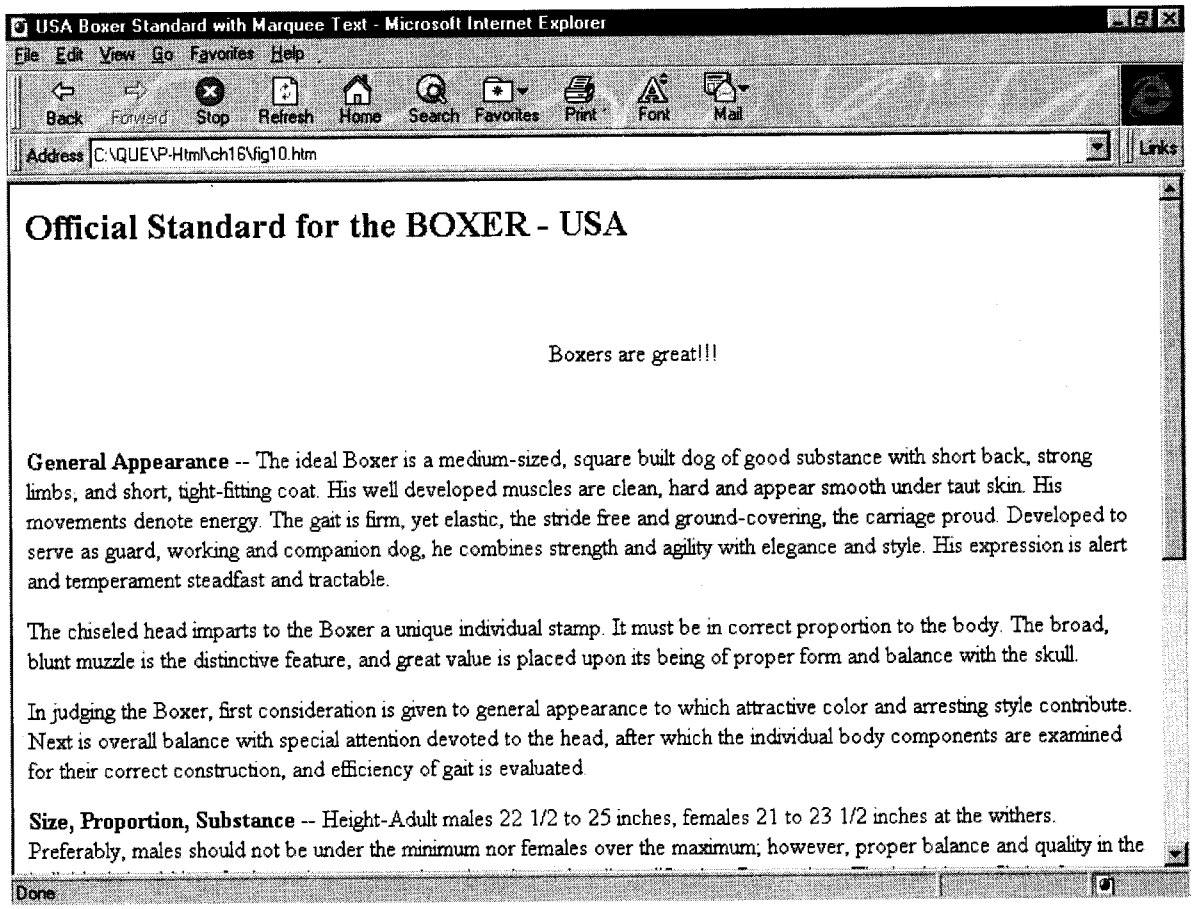


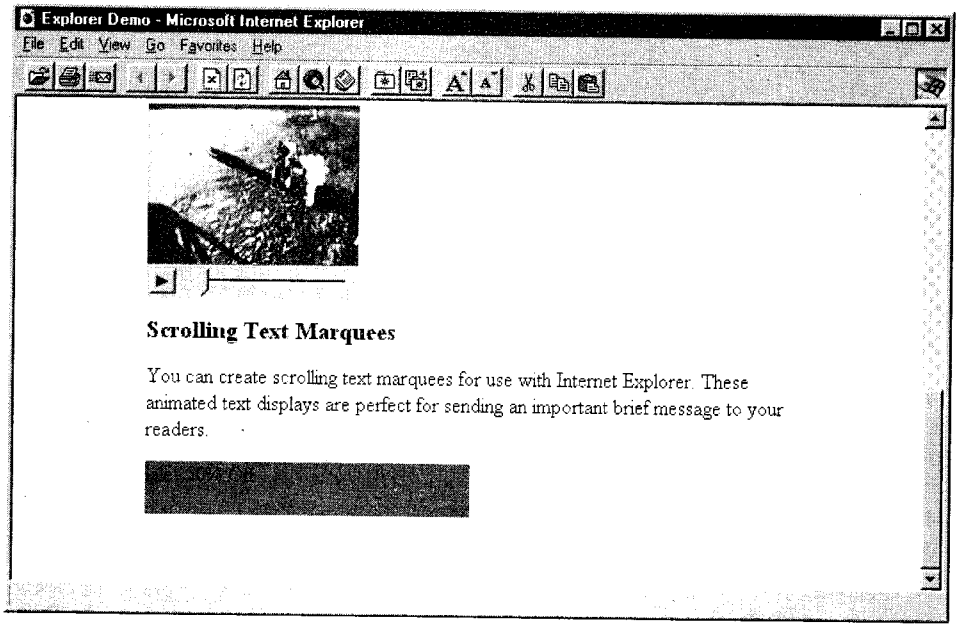
Figure 6.1

Sometimes you desire to get a simple message across to your readers and you want it to stand out from the rest of the document. Internet Explorer lets you generate flashy text messages with scrolling marquees.

Marquees functions as electronic billboards. The text is vigorous and it scrolls across the marquee, where it simply catches the eye. You can manage the speed, style, and direction of the text.

1. Text marquees can be used for visual effect in your HTML document, but the only browser that supports marquees is Internet Explorer. Most other browsers will display the text, but it will not scroll, so be careful with the length of your scrolling image.

   You can surround the <MARQUEE> tag pair with the <CENTER> tag pair to center the marquee on the screen.



2. To place a scrolling text marquee in your HTML document, type <MARQUEE>, followed by the text and a closing </MARQUEE> tag. This will cause the text inside the tag to scroll across the screen. For example, to scroll the words *Sale! 50% Off!*, type <MARQUEE>Sale! 50% Off</MARQUEE>.

```
<MARQUEE>Sale! 50% Off</MARQUEE>
```

3. There are several attributes that you can add to the <MARQUEE> tag that will allow you to control its appearance. First, you can control the style of the marquee using the BEHAVIOR tag. You can make the text scroll, slide, or alternate (bounce back and forth) inside the marquee. The default style is scrolling text. To change this to sliding text, type **BEHAVIOR=SLIDE** inside the <MARQUEE> tag.

```
<MARQUEE BEHAVIOR=SLIDE>
```

4. By default, text in a <MARQUEE> tag moves from the right side of the marquee to the left. You can reverse the direction by typing DIRECTION=RIGHT inside the <MARQUEE> tag.

```
<MARQUEE BEHAVIOR=SLIDE DIRECTION=RIGHT>
```

5. You can specify the height and width of the marquee by placing HEIGHT and WIDTH attributes inside the <MARQUEE> tag. The height and width can be expressed as either specific pixel values or a percentage of the screen. For example, to set the marquee to be exactly 40 pixels high with a width of 50 percent of the visible screen, you would type **HEIGHT=40 WIDTH=50%** inside the <MARQUEE> tag.

```
<MARQUEE BEHAVIOR=SLIDE DIRECTION=RIGHT
HEIGHT=40 WIDTH=50%>
```

6. You can specify how many times the marquee text will loop by using the LOOP attribute inside the <MARQUEE> tag. For example, to set the marquee text to loop five times, type **LOOP=5**. If you do not use the LOOP attribute, the marquee text will cycle infinitely.

```
<MARQUEE BEHAVIOR=SLIDE DIRECTION=RIGHT
HEIGHT=40 WIDTH=50% LOOP=5>
```

7. The SCROLLAMOUNT attribute allows you to specify how many pixels the marquee moves each time it is redrawn. This directly affects the smoothness of the scrolling text as well as the speed with which it moves across the screen. For example, to get the text to scroll slowly and smoothly across the screen one pixel at a time, type SCROLLAMOUNT=1 inside the <MARQUEE> tag.

```
<MARQUEE BEHAVIOR=SLIDE DIRECTION=RIGHT
HEIGHT=40 WIDTH=50% LOOP=5 SCROLLAMOUNT=1>
```

8. You can also control the speed of the marquee text display using the SCROLLDELAY attribute. This attribute specifies the number of milliseconds that will elapse between each redraw of the marquee text. For example, to set the delay to 100 milliseconds, type **SCROLLDELAY=100** inside the <MARQUEE> tag.

```
<MARQUEE BEHAVIOR=SLIDE DIRECTION=RIGHT
HEIGHT=40 WIDTH=50% LOOP=5 SCROLLAMOUNT=1
SCROLLDELAY=100>
```

9. If you want to change the background color of the marquee, use the BGCOLOR attribute inside the <MARQUEE> tag. The color must be specified using an RGB hexadecimal code and preceded with a pound sign. For example, to set the background color of the marquee to light green, type BGCOLOR=#008800.

```
<MARQUEE BEHAVIOR=SLIDE DIRECTION=RIGHT
HEIGHT=40 WIDTH=50% LOOP=5 SCROLLAMOUNT=1
SCROLLDELAY=100 BGCOLOR=#008800>
```

Check Your Progress

1. What are submit and reset buttons?
2. What are dynamic documents?

6.9 LET US SUM UP

Forms are the only method of two-way communication between Web pages and Web sites. Getting feedback is where HTML forms come into play. HTML supports a rich variety of input capabilities to let you solicit feedback. The two key attributes within the <FORM> tag are METHOD and ACTION. Together, these attributes control how your browser sends information to the web server and which input-handling program receives the form's contents.

The standard HTML/XHTML document model is static. Once displayed on the browser, a document does not change until the user initiates some activity, like selecting a hyperlink with the mouse.[1] The Netscape developers found that limitation unacceptable and built in some special features to their browser that let you change HTML document content dynamically. In fact, they provide two different mechanisms for dynamic documents, which we describe in detail in this chapter. Internet Explorer supports some of these mechanisms, which we'll discuss as well. The <MARQUEE> and </MARQUEE> tag pair places a scrolling text marquee on your Web page. The text that scrolls is the text found between the two tags.

6.10 KEYWORDS

Forms: Forms are the only method of two-way communication between Web pages and Web sites.

Scrolling Marquees: The <MARQUEE> and </MARQUEE> tag pair places a scrolling text marquee on your Web page. The text that scrolls is the text found between the two tags.

6.11 QUESTIONS FOR DISCUSSION

1. Discuss the process of creating forms.
2. Discuss how <input tag> is used in the following:
   (i) Text fields
   (ii) Radio buttons
   (iii) Check boxes
   (iv) Password fields
3. What are Microsoft internet extensions?
4. What are scrolling marquees? Discuss various uses of marquee tag.
5. Create a page that plays a background sound three times, while displaying a background image as a watermark.

Check Your Progress: Model Answers

1. The first step in creating a form is to insert the <FORM> tags and add Submit and Reset buttons. Submit and Reset buttons allow visitors to submit information and clear selections.
2. The standard HTML/XHTML document model is static. Once displayed on the browser, a document does not change until the user initiates some activity, like selecting a hyperlink with the mouse. The Netscape developers found that limitation unacceptable and built in some special features to their browser that let you change HTML document content dynamically

6.12 SUGGESTED READINGS

Bud E. Smith, Peter Frazier, and Bud Smith, *Creating Web Graphics for Dummies*, For Dummies; Bk&CD-Rom edition.

Lisa Lopuck, *Web Design for Dummies*, For Dummies; Bk&CD-Rom edition.

UNIT III

LESSON

7

PROGRAM DEVELOPMENT

CONTENTS

7.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss steps in programming
- Discuss design a program model
- Determine correctness of algorithm

- Understand code program
- Discuss test and debug the program
- Discuss documentation

7.1 INTRODUCTION

A program is a set of instructions to be executed by the computer to accomplish a particular task. In this lesson we will discuss Steps in Programming Discuss, Design a Program Model, Determine Correctness of Algorithm, Understand Code Program, Discuss Test and Debug the Program, Discuss Documentation.

7.2 STEPS IN PROGRAMMING

The programming process is a set of activities that are carried out to develop and implement a computer program. Writing a program comprises of the six activities listed below:

1. Understand Program Specification
2. Design a program model
3. Determine correctness of the program
4. Code the Program
5. Test and debug the program
6. Document the program

Now let's discuss these steps in detail:

7.3 UNDERSTANDING PROGRAM SPECIFICATION

Program specification includes two activities:

- Problem-definition
- Requirement Analysis

7.3.1 Problem-definition

The first prerequisite you need to fulfill before designing the program model is a clear statement of the problem that the program is supposed to solve. A problem definition defines what the problem is without any reference to the possible solutions. It's a simple statement, may be one to two pages and it should sound like a problem.

The problem definition comes before the requirement analysis, which is a more detailed analysis of the problem.

The problem definition should be in user language and the problem should be described from the user's point of view. It usually should not be stated in technical computer terms.

7.3.2 Requirement Analysis

Requirements describe in detail, what a program is supposed to do and they are the first step towards a solution. The requirements activity is also known as "functional specification". An explicit set of requirements, is important for several reasons. Explicit requirements help to ensure that the user rather than the programmer drives the program's functionality. If the requirements are explicit, the user can review them and agree to them. Explicit requirements keep you from guessing what the user wants. Specifying requirements adequately is a key to the program's success.

Essentially, we must look for three main components which are:

- What is given as input
- What is expected as output and
- How to arrive at the solution

You are already familiar with the above three items i.e., input (data) process and output (information). Hence, while determining program requirements, we have to discern from the problem statement what exactly constitutes input, what is expected as output and how the processing is to be done. It will not be out of place to mention here, that a given problem or business solution may be solved in a particular way manually and we may or may not choose to adopt the same processing logic while developing a solution to be computerised.

7.4 DESIGN A PROGRAM MODEL

Once the problem is clearly defined, an algorithm, another term for processing logic, can be developed. This is the most creative part of programming. At this stage, the algorithm may be constructed, in broad terms to help visualise possible alternatives.

An algorithm is a formula, a recipe, a step-by-step procedure to be followed, in order to obtain the solution to a problem. To be useful as a basis for writing a program the algorithm must:

- Arrive at a correct solution within a finite time
- Be clear, precise unambiguous and
- Be in a format which lends itself to an elegant implementation in a programming language.

The important tools used in developing a solution and in the preparation of an algorithm are flowcharts and pseudocodes. Among others, flowcharts provide a visual and graphical representation of the solution while pseudocodes mean writing the program logic in a simple English like language. The Logic depicted using these tools can then be written using any programming language. In other words, these are gynoecia tools. We shall look at them in detail through the course of this book.

7.5 DETERMINE CORRECTNESS OF ALGORITHM

One of the most difficult and sometimes most tedious step in the development of an algorithm is ensuring and providing that the algorithm is correct.

Correctness of an algorithm or a program means that it gives the correct output and it does what it is supposed to do.

Correctness of an algorithm or a program can be checked by any one of the following methods

1. Dry Run
2. Independent inspection
3. Structured walk-through

7.6 CODE PROGRAM

This is the process of converting the solution i.e., the algorithm (flowchart/pseudocode) into an actual computer program. Here, we choose a programming language and using the syntax and semantics of that language, convert the algorithm thus far expressed, in the form of a flowchart/pseudocode.

Syntax means the correct way or "grammar" of writing a command or series of commands, including all the proper options and command-line statements.

Semantics means the logical meaning of a statement, separate from the grammatical structure.

7.7 TEST AND DEBUG THE PROGRAM

Once coding is complete, you will enter the program into the computer. The computer will then compile the program into machine code.

Compilation means converting the source code i.e. the user's program which is written using a high level language, into machine code which is machine understandable language.

Compilation results in certain kinds of errors which are indicated by the computer after the process is over. These errors have to be corrected and then the program recompiled. The process continues until the program is error free. Next, we must verify that our program does everything that it is supposed to do. Invariably, computer programs do not work properly the first time. The three types of errors that are normally encountered are:

1. ***Syntax Errors:*** Syntax errors occur when a computer language compiler cannot understand a command entered by the programmer. When such an error is encountered, the computer rejects the program and prints out an error message. These errors are encountered during compilation and are very easy to correct. In many cases, these errors arise due to spelling mistakes, missing commas, etc. and incorrect syntax. For example, in FoxPro if we type APPEND instead of APPEND this causes a syntax error.
2. ***Execution Errors:*** These errors are encountered after error-free compilation, at the time of execution of the program. Execution errors are also called runtime errors. Typical examples are:
   (i) infinite loops
   (ii) correct outputs only for selected data
   (iii) data incorrect or in wrong order
3. ***Logical Errors:*** Logical errors are due to mistakes made by the programmer while coding the program. The computer does not detect logical errors.

For example, for calculating the net salary of an employee the formula is

Net Salary = Basic Salary + Allowances - Deductions

But through oversight, while coding, the formula is written as

Net Salary = Basic Salary -Allowance + Deductions

This will obviously produce a wrong result. Such errors can be detected only by a dry run.

Debugging

Debugging is the process of identifying the root cause of an error and correcting it. It contrasts with testing, which is the process of detecting the error initially.

7.8 DOCUMENTATION

Once the program has been written and debugged, it is ready for use and hence, requires documentation or a written procedure of how to run the program, enter data, what problems to expect, how to handle them, etc. Documentation of a program will consist of flowcharts/ pseudocodes, program listings and detailed written statements of algorithms and procedures involved. Documentation is necessary for program maintenance. Without proper documentation, it will be difficult to change a program at a later date.

It is a mis-held notion that documentation is the last step in the algorithm development stage. In fact, it should be interwoven with the entire phase of the programming process, especially with the design and implementation, because documentation is supposed to enable individuals to understand the logic of programs.

7.9 PROGRAMMING TECHNIQUES

7.9.1 Linear Programming

Linear program is a method for straightforward programming in a sequential manner. This type of programming does not involve any decision making. General model of these linear programs is:

1. Read a data value.
2. Compute an intermediate result.
3. Use the intermediate result to compute the desired answer.
4. Print the answer.
5. Stop.

7.9.2 Structured Programming

One of the most versatile properties of a digital computer is that it can make a decision, thus creating a branching point. There are also times when it becomes necessary for a program to *Look Back* over a set of statements, a number of times. If branching and looping can be used, then much more complex iterative algorithms can be written, which in turn result in more complex programs. There are procedures that can be used for writing these complex programs that make them much less error

prone and much easier to debug. The techniques for writing such programs are referred to as structured programming.

7.9.3 Example of Structured Programming

Structured programming refers to the process in which we break the overall job down into separate piece of modules. Figure 7.1 shows how a salary program is broken down into a number of small modules. These modules, in turn, are broken down into smaller pieces which can also be further subdivided. Modules must be chosen in such a way that we can specify how they are to interact. In effect, there is a contact between each pair of modules. This contact specifies two things:

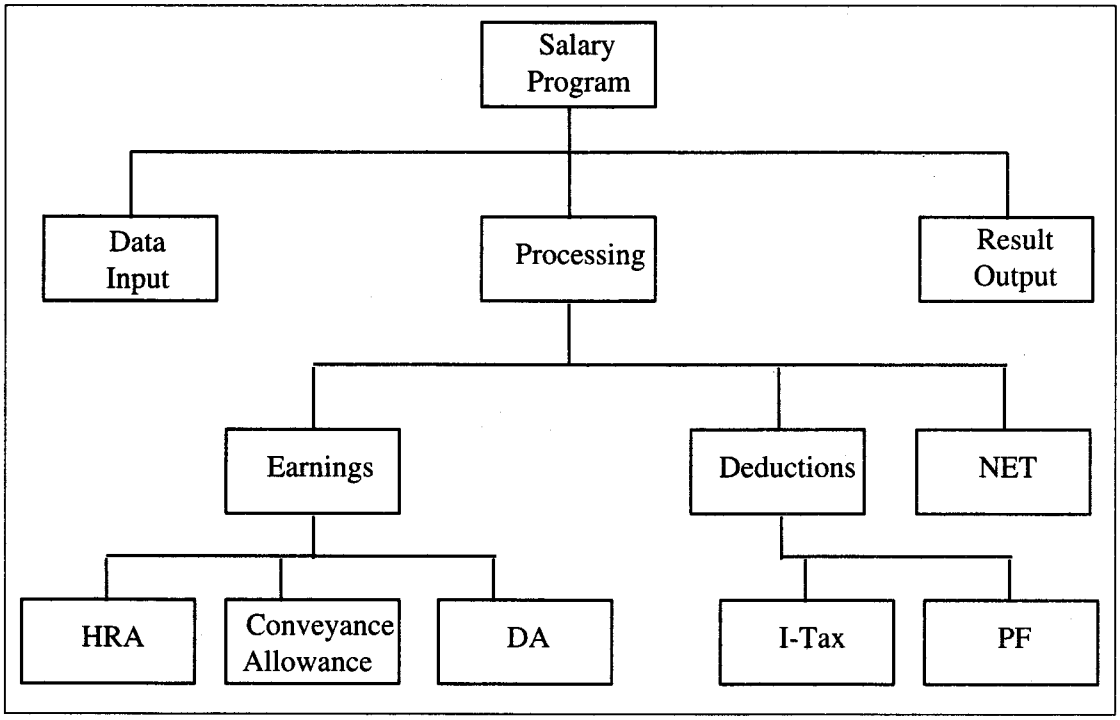


Figure 7.1: Example of Structured Programming

1. What the module will do?
2. What assumptions is it making about the behaviour of the other modules? In particular, we must specify explicitly what inputs a particular module is to receive from the various other modules and what outputs it is required to provide for them.

7.9.4 Advantages of Structured Programming

1. Decreases the complexity of the program by breaking it down into smaller logical units.
2. Allows several programmers to code simultaneously.
3. Allows common functions to be written once and then used in all the programs needing them.
4. Decreases debugging time because modules make it easier to isolate errors.
5. Amendments to single modules do not affect the rest of the program.

6. It saves time to use modular structures rather than using self-made structures. If a job can be done well by using what is already available and known to be well tried and tested then trying out something new for the sake of trying is a waste of effort.
7. Structured programming is a standard method; so, less time is required in writing programs.
8. It is easier to name modules in such a way that they are easy to locate in the documentation, and consistent.

7.9.5 Structured Programming Techniques

There are three widely used programming constructs:

1. Sequence
2. Selection
3. Repetition (iteration)

Now let us discuss each in detail.

Sequence

Sequence structure consists of an action followed by another, and so on, till the desired result is obtained.

```
Statement 1
Statement 2
---------
---------
---------
Statement n
```

Selection

In conditional execution, there is a need to carry out a logical test and then take some particular action which depends upon the result of that test.

The selection structure consists of a text for a condition followed by two alternative paths for the program to follow. The program selects one of the program control paths, depending upon the outcome of the test condition. After performing one of the two paths, the program control returns to a single point. This pattern can be termed as if. . . else because of the logic.

```
If (condition is true) then
    sequence of statements
else
    another sequence of statements
endif
```

Iteration

In most cases, programs require that a group of consecutive instructions be executed repeatedly until some logical condition has been satisfied. Such an iteration is called conditional looping.

Another type of repetition is unconditional looping. In such a looping the instructions are repeated for a specified number of times.

The control structures are easy to use because of the following reasons:

1. They are easy to recognize.
2. They are simple to deal with as they have just one entry and one exit point.
3. They are free of the complications of any particular programming language.

Check Your Progress

1. Define code program.
2. What are logical errors?

7.10 LET US SUM UP

A program is a set of instructions to be executed by the computer to accomplish a particular task. A problem definition defines what the problem is without any reference to the possible solutions. It's a simple statement, may be one to two pages and it should sound like a problem. Requirements describe in detail, what a program is supposed to do and they are the first step towards a solution. The requirements activity is also known as "functional specification". An explicit set of requirements, is important for several reasons. Explicit requirements help to ensure that the user rather than the programmer drives the program's functionality. Once the problem is clearly defined, an algorithm, another term for processing logic, can be developed. This is the most creative part of programming. Correctness of an algorithm or a program means that it gives the correct output and it does what it is supposed to do. Syntax means the correct way or "grammar" of writing a command or series of commands, including all the proper options and command-line statements. Semantics means the logical meaning of a statement, separate from the grammatical structure. Once coding is complete, you will enter the program into the computer. The computer will then compile the program into machine code. Debugging is the process of identifying the root cause of an error and correcting it. Once the program has been written and debugged, it is ready for use and hence, requires documentation or a written procedure of how to run the program, enter data, what problems to expect, how to handle them, etc. Documentation of a program will consist of flowcharts/pseudocodes, program listings and detailed written statements of algorithms and procedures involved.

Structured programming refers to the process in which we break the overall job down into separate piece of modules. Sequence structure consists of an action followed by another, and so on, till the desired result is obtained. The selection structure consists of a text for a condition followed by two alternative paths for the program to follow. In most cases, programs require that a group of consecutive instructions be executed repeatedly until some logical condition has been satisfied. Such an iteration is called conditional looping.

7.11 KEYWORDS

Program: A program is a set of instructions to be executed by the computer to accomplish a particular task.

Requirements Analysis: Requirements describe in detail, what a program is supposed to do and they are the first step towards a solution.

Syntax: It means the correct way or "grammar" of writing a command or series of commands, including all the proper options and command-line statements.

Semantics: It means the logical meaning of a statement, separate from the grammatical structure.

Debugging: It is the process of identifying the root cause of an error and correcting it.

Structured Programming: It refers to the process in which we break the overall job down into separate piece of modules.

7.12 QUESTIONS FOR DISCUSSION

1. Describe the steps in programming.
2. Describe activities included for program specification.
3. How will you determine the correctness of an algorithm?
4. How would you test and debug the program?
5. What is structured programming? Discuss the techniques of structured programming.
6. What is documentation?

Check Your Progress: Model Answers

1. Code program is the process of converting the solution i.e., the algorithm (flowchart/ pseudocode) into an actual computer program. Here, we choose a programming language and using the syntax and semantics of that language, convert the algorithm thus far expressed, in the form of a flowchart/pseudocode.
2. Logical errors are due to mistakes made by the programmer while coding the program. The computer does not detect logical errors.

7.13 SUGGESTED READINGS

Kenneth Leroy Busbee, *Programming Fundamentals: A Modular Structured Approach*, University Press of Florida

Byron S. Gottfried, *Schaum's outline of theory and problems of programming with Basic*, McGraw-Hill

C. Joseph Sass, *BASIC programming and applications*, Allyn and Bacon

LESSON

8

PROGRAM TOOLS

CONTENTS

8.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss program tools
- Discuss flowcharts

- Discuss Pseudocodes
- Understand decision structure
- Discuss structure charts

8.1 INTRODUCTION

As programming is very typical, and mind crushing work, some aids are required to make the task simpler and perform it in an efficient way. Some of the aids, or tools are required to develop a program are:

1. Program Flowcharts
2. Pseudocodes
3. Decision Tables

8.2 PROGRAM FLOW CHARTS

A program flowchart is a pictorial representation of the logic required to accomplish a task. It includes all necessary steps of a program. It is called a flowchart because it charts the flow of a program. It is a symbolic representation of each input, output and processing step. Besides, being a good method of writing down the algorithm, it is also a part of program documentation and helps in understanding, debugging and maintaining programs.

In the flowcharting technique, operations are represented by drawing appropriate symbols for the actions.

These flowchart symbols are connected by arrows to illustrate the sequence of operations.

The basic symbols used in a program flowchart are represented in Figure 8.1.

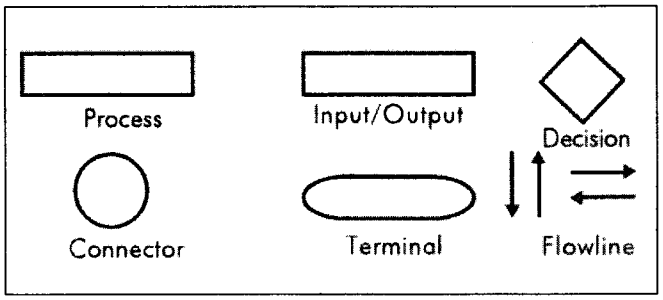


Figure 8.1

8.2.1 The Input/Output Symbol

This symbol is used to represent the Input/Output operations (I/O operations) such as read and write. This is illustrated in Figure 8.2.

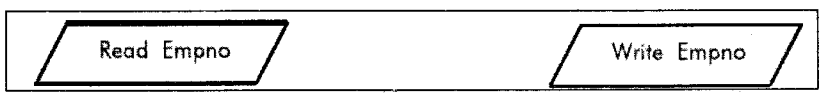


Figure 8.2

8.2.2 The Process Symbol

This symbol is used to represent process like assigning a value to a variable or adding a number. It has one arrow going inside to denote the input data to the process and another arrow leaving it to denote the processed data leaving this process. In other words, it has one entry and one exit. Figure 8.3 illustrate this:

In Figure 8.3, "Sum = N1 + N2" is a process which adds the two numbers and stores the result in Sum. For this process, the input is numbers flowing in (represented by in-going arrow from the process).

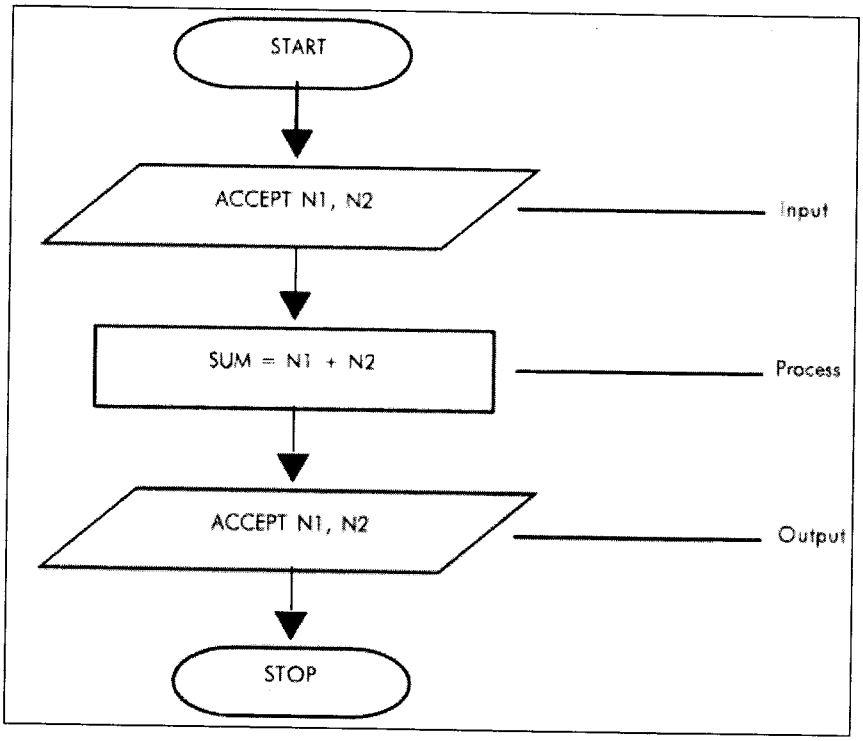


Figure 8.3

8.2.3 The Terminal Symbol

This symbol indicates the beginning or the end of a flowchart as shown in Figure 8.4.

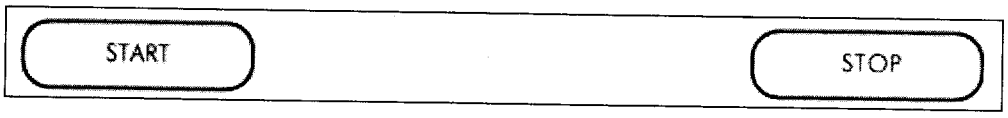


Figure 8.4

8.2.4 The Connector Symbol

The connector symbol, is used to maintain links between two or more flowcharts when the flowchart runs longer than one page or when the same diagram is continued at different locations of the same page. If the same flowchart is continuing on the next page, then at the end of the first page this symbol is used with a marker which can be an alphabet or a number. The same symbol is used, with the same marker as was used in the first page, at the beginning of the second page. Figure 8.5 illustrates this.

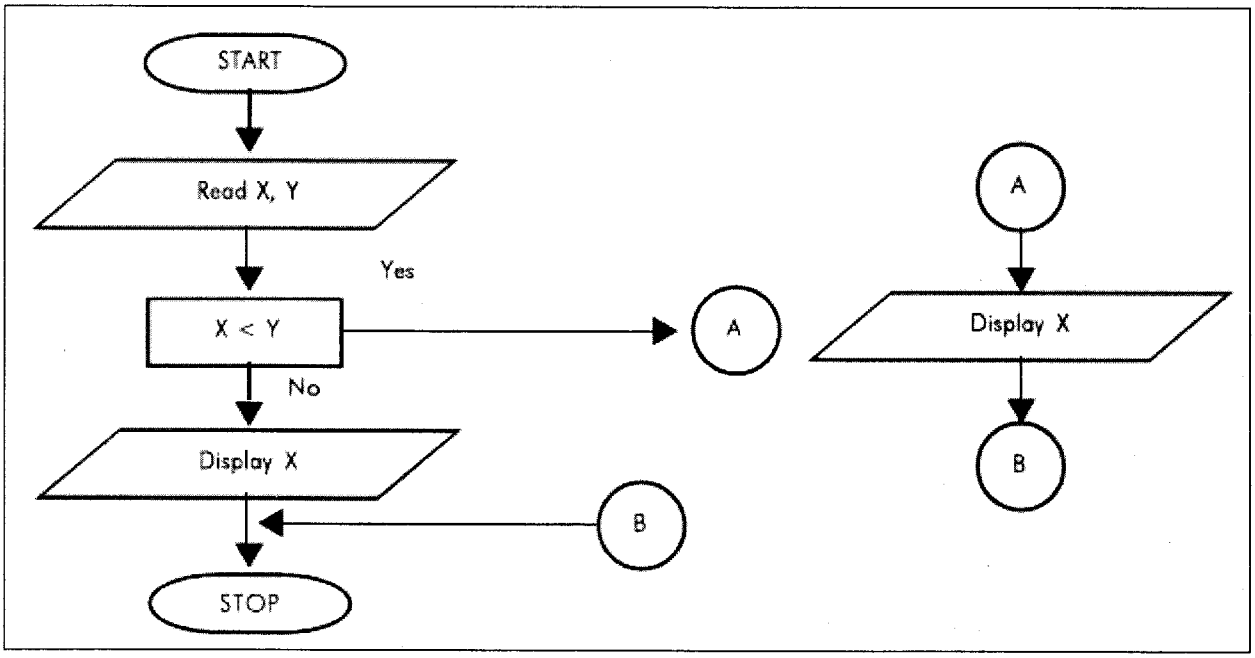


Figure 8.5

8.2.5 The Flowline Symbol

This symbol represents the direction of the flow of data in a flowchart. These are straight lines with arrow heads. They are normally drawn from top to bottom and left to right to left.

Examples of Flowcharts

Example 1:

A student appears for a test in 3 subjects. Each test is out of 100 marks. The percentage of each student has to be calculated and depending on the percentage calculated, grades are given as next page.

Percentage	Grade
= >90	E+
80-90	E
70-79	A+
60-69	A
50-59	B+
<50	FAIL

Flowchart for the same is given in Figure 8.6.

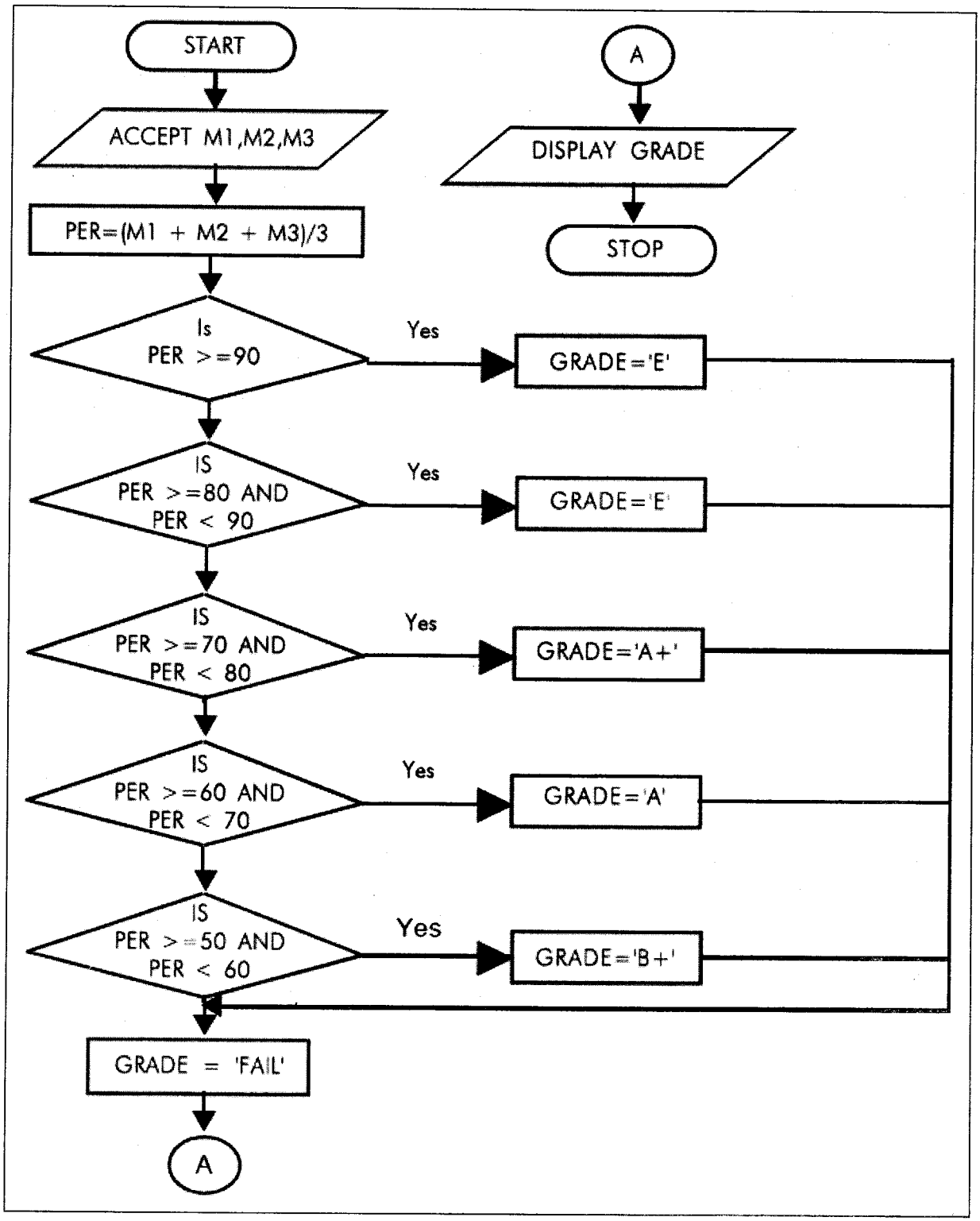

START
ACCEPT M1,M2,M3
PER=(M1 + M2 + M3)/3
Is
PER >=90
Yes
GRADE='E'
IS
PER >=80 AND
PER < 90
Yes
GRADE='E'
IS
PER >=70 AND
PER < 80
Yes
GRADE='A+'
IS
PER >=60 AND
PER < 70
Yes
GRADE='A'
IS
PER >=50 AND
PER < 60
Yes
GRADE='B+'
GRADE = 'FAIL'
A
A
DISPLAY GRADE
STOP

Figure 8.6

Example 2:

The flowchart to find the factorial for any given number is shown in Figure 8.7. Factorial means the product of the positive integers 1, 2, 3 up to and including a given integer. Factorial is derived by applying the following equation:

n! (factorial of n) = n* (n-1)* (n-2)* (n-3).....*3*2*1

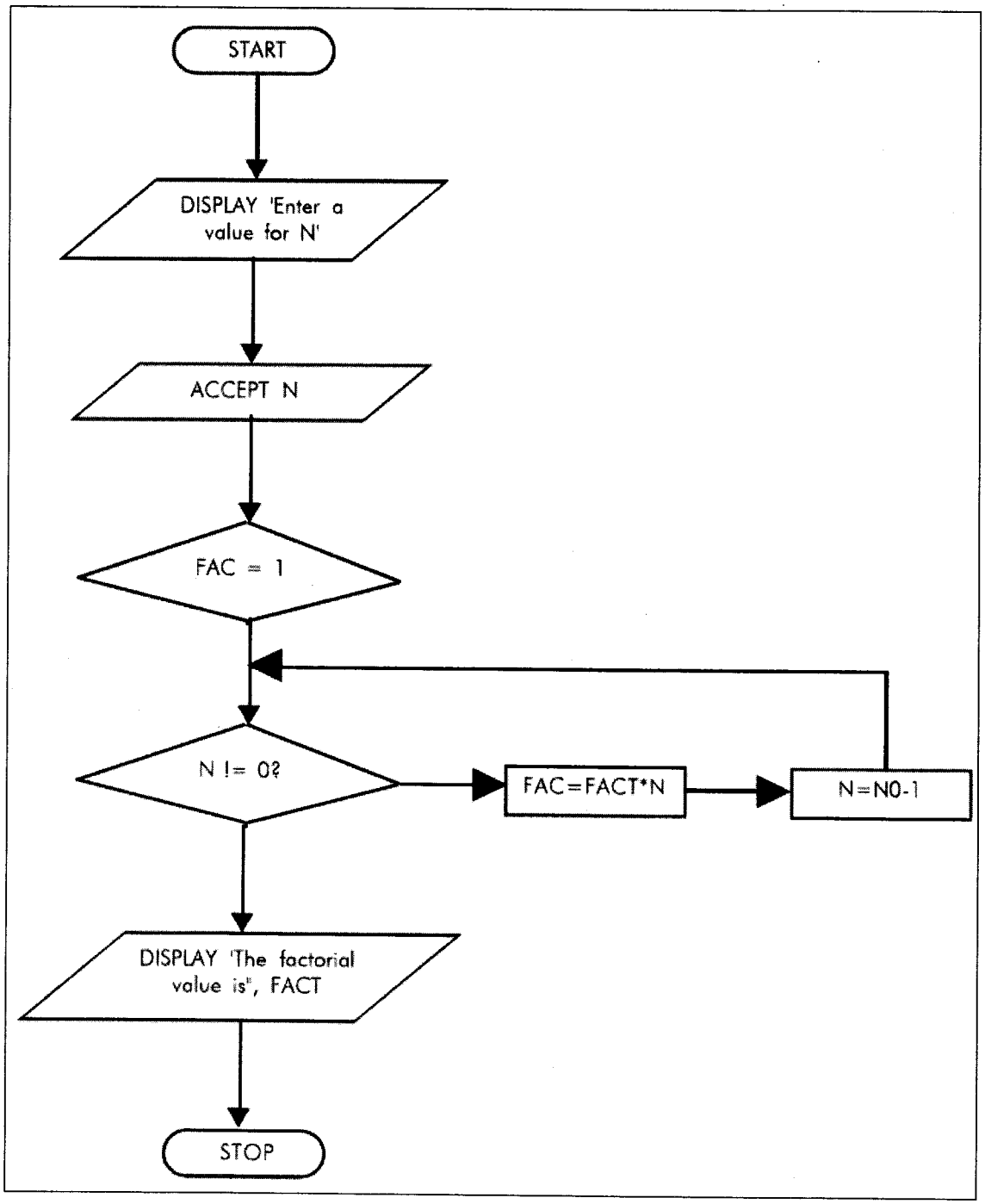


Figure 8.7

Example 3:

Generate the first 20 numbers in the Fibonacci series.

Fibonacci series is a series of integers in which each integer is equal to the sum of the two preceding integers in the series. The series is formulated mathematically by Xi = Xi-1 + Xi-2, where X0 = 0 and X1=1; that is 0,1,2,3,5,8....

The series looks like:

0+1 = 1, 1+1 = 2, 1+2 =3, 2+3 = 5, 3+5 =8, etc.

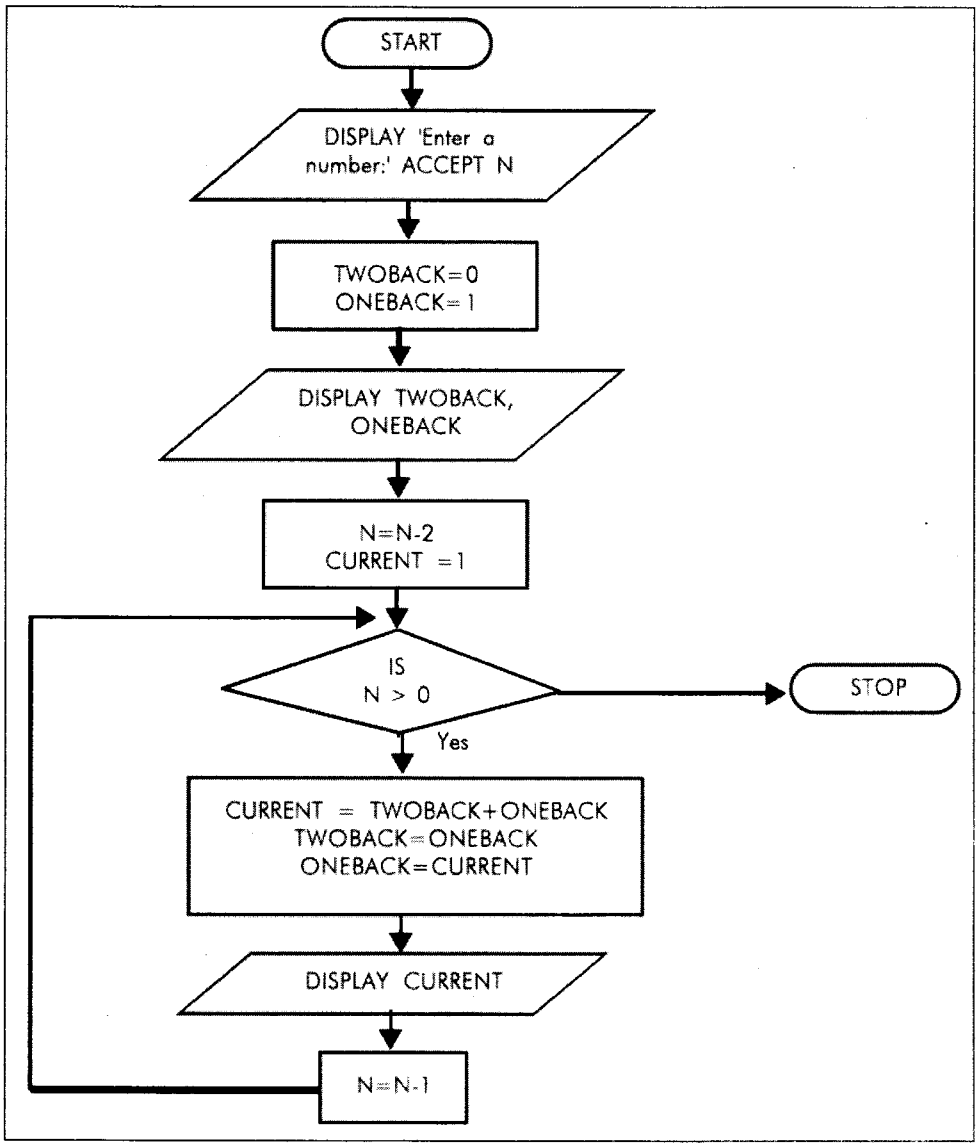


Figure 8.8

Example 4:

The sum of all odd numbers in the range 1 to 10 (i.e. 1+3+5+7+9=25).

The flowchart to solve above problem is shown in Figure 8.9.

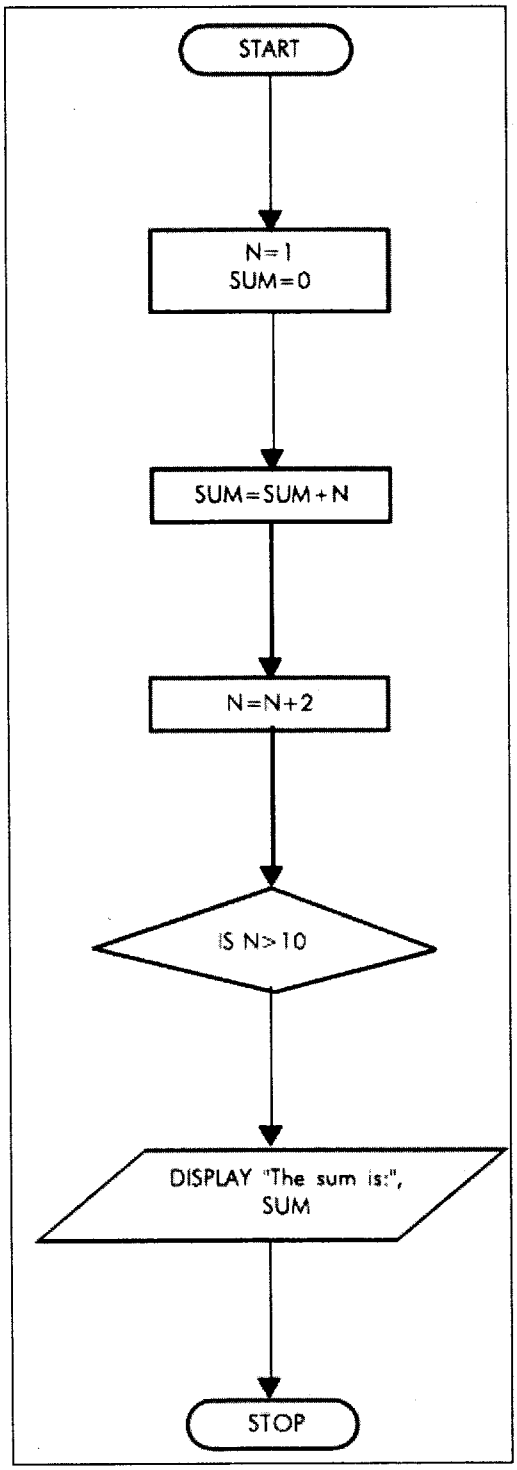


Figure 8.9

8.3 PSEUDOCODES

An alternative method of representing program logic is a pseudocode Instead of using symbols to represent the program logic steps, a pseudocode uses statements which are a bridge between actual programming and ordinary English. In a pseudocode, each step is written using a simple English phrase which is also called a construct.

Pseudocode notation can be used both at the preliminary stage as well as the advanced stage. It can be used at any abstraction level. In a Pseudocode, a designer describes system characteristics using short English language phrases with the help of keywords like while, if..the..else, End, etc. Keywords and indentation describe the flow of control while the English phrases describe the processing action that is being taken. Using the top-down strategy, each English phrase is expanded into a more detailed Pseudocode till the point it reaches the level of implementation.

8.3.1 Functional Procedural Layers

Functions are built in layers. Layers are used to depict additional information.

Level 0:

- Function or procedure name
- Relationship with other systems
- Description of the function purpose
- Author, date

Level 1:

- Functional parameters
- Global variables
- Routines called by functions
- Side effects
- Input/output assertions

Level 2:

- Local data structures, variables, etc
- Timing constraints
- Exception handling
- Any other limitations

Level 3:

- Body including Pseudocode, structure chart, decision tables, etc.

Some of the conventions which are used while writing pseudocodes are as follows.

1. All statements in a loop should be indented.
2. All alphanumeric values should be enclosed in single or double quotes.

3. The beginning and end of a pseudocode is marked with keywords like 'start' and 'end' respectively.
4. All statements must include certain key words which denote an operation.

Some books follow the convention of ending each statement with a semicolon.

8.3.2 The Input Statement

The following verbs can be used to accept or input data from the keyboard or from an existing form like a file.

Accept or Read

For example

Accept Name

Read Name

8.3.3 The Output Statement

The following verbs can be used to output data

Write or Display

For example

Write Grosspay

Display Grosspay

Now let us take few examples and write pseudocodes and then draw flowcharts for the same.

Example 1:

Accept two numbers, add them and display the result.

The steps for the above problem statement are:

START

ACCEPT N1

ACCEPT N2

SUM = N1 + N2

DISPLAY SUM

END

The above is a simple pseudocode for our problem. It may be noted here, that whether we input N1 first or N2 first is immaterial here. However, the statement SUM = N1 + N2 cannot come before the two numbers have been input. What will happen if the DISPLAY SUM statement was put before SUM = N1 + N2?

Let us now draw the flowchart for the same example as shown in Figure 8.10:

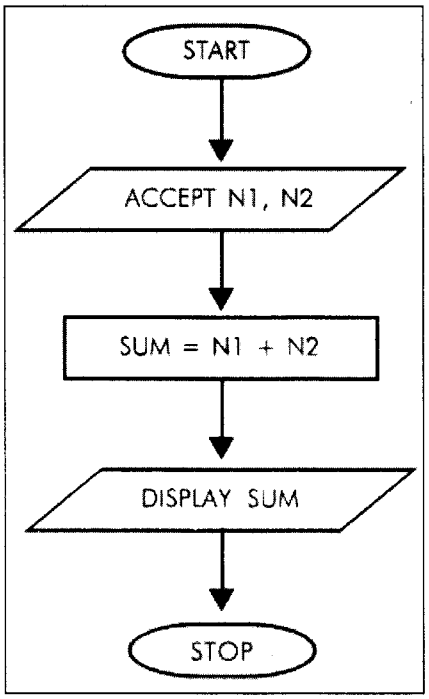


Figure 8.10

In Figure 1 "ACCEPT N1, N2" denotes that two numbers are accepted from the user and stored in variables N1 and N2. "SUM = N1+ N2" denotes that a process is taking place, which is adding the numbers N1 and N2 and the resultant output stored in the variable SUM. "DISPLAY SUM" denotes that the resultant SUM is displayed on the screen.

Example 2:

Mohan's monthly salary consists of Basic salary, travelling allowances and a 15% commission on sales made. At the end of the month, we need to calculate his salary which is done in the following pseudocode.

```
START
ACCEPT BAS_SAL,    TCL_ALL,     SALE_AMT
COMM = SALE_AMT * 0.15
NET_SAL = BAS _ SAL + TVL _ ALL + COMM
DISPLAY NET_SAL
END
```

It may be noted here, that we are accepting 3 variables - BAS_SAL, TVL_ALL, SALE_AMT with one accept statement. This is valid. Alternatively, three ACCEPT statements could have been written - one for each of the three variables, as in Example 1.

Here BAS_SAL, TVL_ALL, SALE_AMT, NET_SAL are variables which hold the value for basic salary, travelling allowance, sales amount and net salary respectively.

The flowchart for this example would be as shown in Figure 8.11.

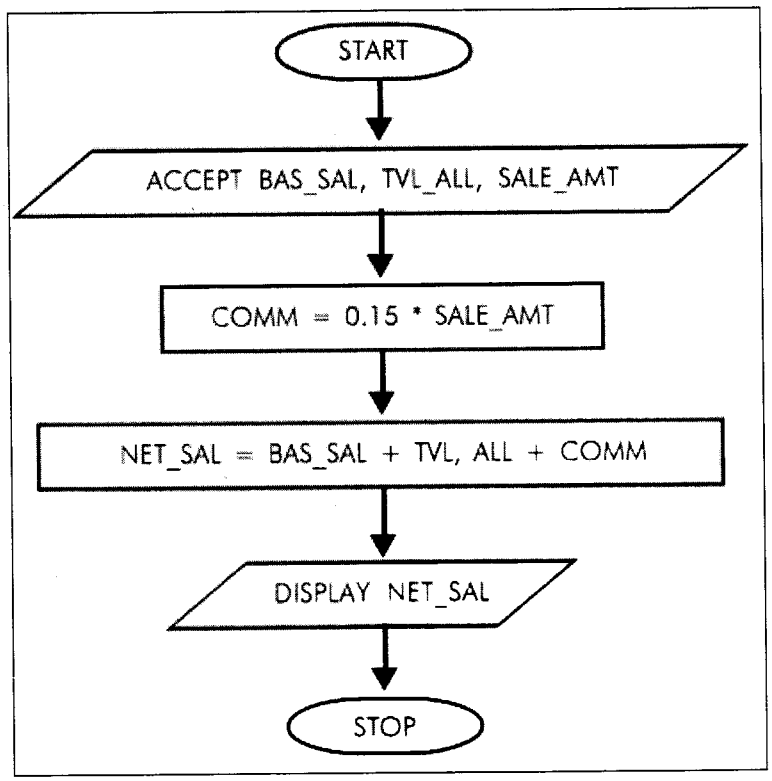


Figure 8.11

Example 3:

SAGA, a grocery store gives discount to its regular customers. The discounts are given irrespective of the amount of purchase. The store gives two types of discounts. A trade discount of 20% on total value of purchase and a cash discount of 5% on the value of the purchase less trade discount.

Let us calculate and display the gross amount due, the net amount due, the value of the quantity discount, the value of the cash discount and the total discount along with the product code.

Pseudocode

```
START
ACCEPT PR _ CODE, PR + RATE, PR _ QTY
GROSS _ AMT = PR _ QTY * PR _ RATE
TRADE _ DIS = 20 / 100 * GROSS _ AMT
CASH _ DIS = 5 / 100 * (GROSS _ AMT - TRADE _ DIS)
NET _ AMT = GROSS _ AMT = (TRADE _ DIS + CASH _ DIS
TOT _ DIS = TRADE _ DIS + CASH DIS
DISPLAY PR_CODE, GROSS_AMT, NET_AMT, TRADE_DIS,
CASH_DIS,   TOT_DIS
END
```

Note the use of parenthesis (brackets). This is to give precedence to the arithmetic operations given therein. Hence, in the statement.

CASH _ DIS = 5/100 * (GROSS_AMT - TRADE_DIS)

TRADE_DIS will be subtracted from GROSS_AMT first then the operations outside the bracket will take place.

```
Here  PR_CODE = Product Code
      PR_RATE = Product rate
      PR_QTY = Product quantity
      GROSS_AMT = Gross amount
      TRADE_DIS = Trade discount
      CASH_DIS = Cash discount
      NET_AMT = Net amount
      TOT_DIS = Total discount
```

The flowchart for the above example in shown Figure 8.12.

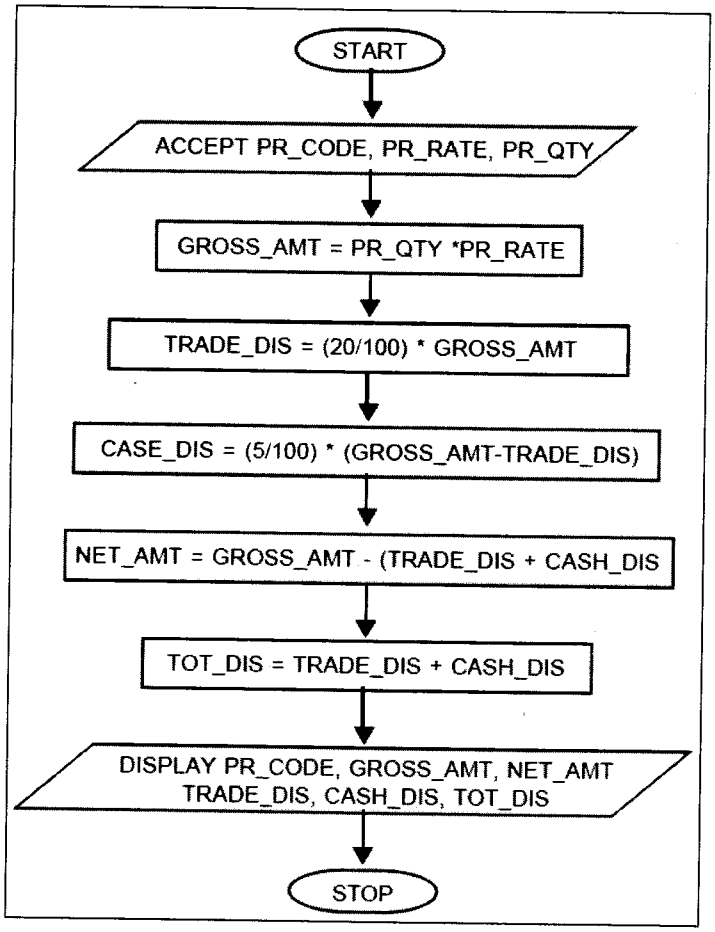


Figure 8.12

Examples of Pseudocodes

Example 1:

A student appears for a test in 3 subjects. Each test is out of 100 marks. The percentage of each student has to be calculated and depending on the percentage calculated, grades are given as under:

Percentage	Grade
=>90	E+
80-90	E
70-79	A+
60-69	A
50-59	B+
<50	FAIL

The pseudocode for the same is given below:

```
start
accept m1, m2, m3
pr = (m1 + m2 + m3) /3
if per > = 90
            grad = 'E+'
else
            if          per >= 80 and per < 90
                        grade = 'e'
            else
                  if    per > = 60 and per < 70
                              grade = 'a'
                  else
                              if    per > = 50 and per < 50
                                    grade = 'b+'
                              else
                                    grade = 'fail'
                              endif
                  endif
endif
      endif
endif
display grade
end
```

Example 2:

The pseudocode to find the factorial for any given number is given below. Factorial means the product of the positive integers 1, 2, 3 up to and including a given integer. Factorial is derived by applying the following equation:

```
n! (factorial of n) = n* (n-1)* (n-2)* (n-3).....*3*2*1
```

Pseudocode:

```
start
display 'Enter a value for n'
Accept n
fact = 1
while n ! = 0            ** ! = is a symbol for not
equal to
DO
            fact = fact *n
            N = N - 1
enddo
display 'The factorial value is :', FacT
end
```

Example 3:

Generate the first 20 numbers in the Fibonacci series.

Fibonacci series is a series of integers in which each integer is equal to the sum of the two preceding integers in the series. The series is formulated mathematically by Xi = Xi-1 + Xi-2, where X0 = 0 and X1 = 1; that is 0,1,2,3,5,8....

The series looks like:

0+1 = 1, 1+1 = 2, 1+2 =3, 2+3 = 5, 3+5 =8, etc.

If we were required to generate 5 numbers then the series would look like: 0 1 2 3 5...

The pseudocode for the above problem is given below:

Pseudocode:

```
start
display 'Enter how many numbers to generate:'
accept n
twoback = 0
oneback = 1
display twoback, oneback
n = n - 2
current = 1
```

```
while n > 0
do
            current = twoback + onebcak
            twoback = oneback
            oneback = current
            display current
            n = n -1
enddo
end
```

Example 4:

The sum of all odd numbers in the range 1 to 10 (i.e. 1+3+5+7+9=25).

The pseudocode to solve above problem is given below:

Pseudocode:

```
start
n = 1
sum = 0
repeat
            sum = sum + N
            N = N + 2
until N > 10
display 'The sum is : ', SUM
END
```

8.4 DECISION TABLES

A Decision Table is a table of rows and columns that shows conditions and actions. 'Decision Rules', include a decision table, states what procedure to follow when certain conditions exist.

8.4.1 Decision Table Characteristics

The Decision Table is made up of four sections:

- Condition Statements
- Condition Entries (Alternatives)
- Action Statements
- Action Entries.

The condition statement identifies the relevant conditions.

Condition Entries (alternatives) tell which value, if any, applies for a particular condition.

Action Statement lists the set of all steps that can be taken when a certain condition occurs.

Action Entries show what specific actions in a set to take, when selected conditions or combinations of conditions are true.

You can enter a note below the table to help state when to use the table or to distinguish it from other tables. The columns on the right, linking conditions and actions form decision rules.

These state the conditions that must be satisfied for a particular set of actions to be taken. For Decision Tables unlike decision trees, there is no order sequence. The decision rule incorporated 'all' the conditions that must be true, not just one condition at a time.

8.4.2 Building Decision Tables

To develop Decision Tables the following steps should be used:

1. Determine the most relevant factors to be considered in making a decision. This identifies the conditions in the decision. Each condition selected should have the potential to either occur or not occur. Partial occurrence is not possible.
2. Determine the most feasible steps or activities under varying conditions, not just the current conditions. This identifies the actions.
3. Study the combinations of conditions that are possible. For every N number of conditions, there are 2^N combinations to be considered. For example, for three conditions, there are eight possible combinations; $2^3 = 8$. For four, $2^4 = 16$ combinations are possible and can be included in the table.
4. Fill in the table with decision rules. There are two ways to fill in the table.
   (i) In a longer method, here you have a fill in condition rows with a yes or no value for each possible combination of conditions.
   (ii) The other method of completing the table deals with one condition at a time and for each possible combination of conditions.
   (iii) The other method of completing the table deals with one condition at a time and adds to the table for each additional condition but does not add duplicate combination of conditions and actions as discussed.
      (a) State the first condition and permissible actions.
      (b) Add the second condition by duplicating the first half of the matrix and filing in the different Y and N values from the new condition in both halves of the expanded matrix.
      (c) Repeat set b for each additional condition.
5. Mark action entries with X to signal action(s) to take; leave cells blank or mark with a dash to show that no action applies to that row.
6. Examine the table for redundant rules or for contradictions within rules (discussed below).

   Following the above simple guidelines will help to:

   (i) save time in building a decision table from collected information

   (ii) point out where information is missing

(iii) show where conditions do not matter in a process

(iv) indicate where there are important relations or results that others were not aware of, in other words, not considered.

Thus, using decision tables can produce more complete and accurate analysis.

8.4.3 Checking Decision Tables

After constructing a table, analysts verify it for correctness and completeness to ensure that the table includes all the conditions, along with the decision rules that relate them to actions. Analysts should also examine the table for redundancy and contradictions.

Eliminating Redundancy

Decision Tables are likely to get too large if allowed to grow in an uncontrolled way. Removing redundant entries help to manage table size. Redundancy occurs when:

1. Two decision rules are identical except for one condition row
2. The actions for two rules are identical

Removing Contradictions

Decision rules contradict each other when, two or more rules have the same set of conditions and the actions are different. This could occur when either there is an error in constructing the table or when the analysts receive discrepant information from different individuals about how decisions are made.

Impossible Situations

When building decision tables, it is possible to set up impossible situations. Now let us see an example for impossible situations. One thing we have to make sure is to see that accuracy is maintained and redundancy is avoided. In Table 8.1 below, rule 1 is not possible as a person who is earning more than ₹ 50, 000 per year cannot earn less than ₹ 2,000 per month at the same time.

Table 8.1 Decision table checking for impossible situations

Conditions and Actions	Rules			
	1	2	3	4
Salary > ₹50, 000 per year	Y	Y	N	N
Salary < 2,000 per month	Y	N	Y	N
Action 1				
Action 2				

8.4.4 Developing Decision Tables

In order to build decision tables, the analyst needs to determine the maximum size of table, eliminate any impossible situations, inconsistencies or redundancies and simplify the table as much as possible.

1. Determine the number of conditions that may affect the decision. Combine the rows that overlap. For example, conditions that are mutually exclusive. The number of conditions becomes the number of rows in the top half of the decision table.

2. Determine the number of possible actions that can be taken. This becomes the number of rows in the lower half of the decision table.
3. Determine the number of conditions, alternatives for each condition (rules).
4. Calculate the maximum number of columns in the decision table by multiplying the number of alternatives for each condition.
5. Fill in the condition alternatives. Start with the first condition and divide the number of columns by the number of alternatives for that condition. For e.g., for 2 conditions, there are 8 rules. Divide 8 by 2 (2..). So write Y in four columns and N in remaining 4 columns as follows:

Condition 1:	Y	Y	Y	Y	N	N	N	N
Condition 2:	Y	Y	N	N	Y	Y	N	N
Condition 3:	Y	N	Y	N	Y	N	Y	N

According to the decision tree example let us see the decision table given below:

Conditions and Actions	Rules							
	1	2	3	4	5	6	7	8
Grade between 10-19	Y	Y	Y	Y	N	N	N	N
Grade between 20-29	Y	Y	N	N	Y	Y	N	N
Grade between 30-19	Y	N	Y	N	Y	N	Y	N
Actions								

Note: It is assumed that all employees exist.

6. Complete the table by inserting an X where rules suggest certain actions.
7. Combine rules where it is apparent that an alternative does not make a difference in the outcome. For e.g.,

Condition 1:	Y	Y
Condition 2:	Y	N
Action 1	X	X

This can be expressed as:

Condition 1:	Y
Condition 1:	-
Action 1	X

The dash (-) signifies that condition 2 can be either Y or N and the action will still be taken.

8. Check the table for any impossible situations, contradictions and any redundancies. According to the earlier example, there are certain impossible situations and redundancies. Take rule 1, 2, 3

and 5. In all these rules, there are two Y's, actually speaking a person cannot have two grades, he has to be in a particular grade, so these rules have to be eliminated as it is an impossible situation.

The last rule says that an employee exists but then there is no grade given to that employee, here there is a contradiction, so again this has to be eliminated.

9. Rearrange the conditions and actions (or even rules) if this makes the decision table more understandable.

Note: The decision table for the example above has been explained in the limited entry table after eliminating all impossible situations.

8.4.5 Type of Table Entries

There are three types of table forms:

- Limited-Entry Form
- Extended-Entry Form
- Mixed-Entry Form

Limited-Entry Form

The basic table structure consists only of 'Y', 'N', and blank entries. This is a limited-entry form. It is the most commonly used format.

This decision table is made according to the example given in decision tree. We have arrived at this table after eliminating all the impossible situations, contradictions and redundancies. This has already been explained.

Decision Table with Limited Entry form:

Table 8.2

Conditions and Actions	Rules				
	1	2	3	4	5
Employee Exists	Y	Y	Y	Y	N
Grade between 10-19	Y	N	N	N	-
Grade between 20-29	N	Y	N	N	-
Grade between 30-39	N	N	Y	N	-
Grade greater than 39	N	N	N	Y	-
DA 20% of Basic, HRA = 600	X				
DA 20% of Basic, HRA = 400		X			
DA 0, HRA = 40% of Basic			X		
DA 0 HRA = 50% of Basic				X	
No Calculation					X

In a limited entry decision table, the conditions are expressed as simple Yes or No questions, whereas in an Extended Entry table, conditions have more than two possible states.

Extended-Entry Form

The extended-entry form replaces Y and N with action entries, telling the reader how to decide. Here, the condition and action statements themselves are not complete, therefore, the entries contain more than one Yes and No.

In the extended entry form the condition is that, if an employee exists and if he is in the marketing department ("MKT"), he is eligible for commission. The commission is based on the total sales made for the month and is calculated as follows:

If sales > = 10000 then commission is 20%

If sales > = 5000 and < 10000 then commission is 10%

If sales < 5000, commission is 0

The table is shown in Table 8.3.

Table 8.3

Conditions and Actions	Rules				
	1	2	3	4	5
Employee Exists	Y	Y	Y	Y	N
Department	MKT	MKT	ADM	MKT	-
Total sales	10,000	2000	-	5000	-
Commission applicable	20%	0	n.a.	10%	n.a.

The decision table above, explains how the commission is given according to the sales per month. According to these conditions, the actions have to be taken. In the first rule, the person exists, he is in the "MKT" department and his total sales is upto 10000, so he is eligible for commission, he will get commission 20% of sales amount.

In rule 3 the person is in "ADM" department, so he does not have any sales figure, so he is not eligible for any commission as such.

Mixed-Entry Form

This form consists of combined features of limited and extended-entry forms. Generally, only one form should be used in each section of the table, but between the condition and action sections, either form may be used.

Now, let us see an example of the mixed entry form, where there are various combination of conditions and actions taken.

According to the company's rules and policies, following are the conditions and actions to be taken:

For all employee in the grade 10-19 and those in marketing department commission is calculated as follows:

If sales > = 10000 then commission is 20%

If sales > = 5000 and < 10000 then commission is 10%

If sales < 5000, commission is 0

If the person is in any other department then he is not entitled for commission

Second condition is to check, if the employee is supposed to pay income tax. This is done as follows:

If the employee's salary > = 50,000 - Tax applicable

If salary < 50000 - Tax not applicable

Tax is calculated for employees of all departments.

Table 8.4

Conditions and Actions	Rules				
	1	2	3	4	5
Department	MKT	MKT	FIN	FIN	MKT
Grade 10 -19	Y	Y	Y	Y	Y
Salary 50, 000 p.a.	Y	N	Y	N	N
Salary 50, 000 p.a.	-	Y	N	Y	Y
Total sales	20000	10000	-	-	3000
Commission applicable	20%	10%	n.a.	n.a.	0
Income Tax applicable	X	-	X	-	-
Income Tax exempted	-	X	-	X	X
INSERT table 6.4 FROM PAGE-127- PF-CRC-U-6					

The decision table in Table 8.4 explains the following:

In the first rule, the person is in "MKT" department, the grade is between 10 and 19 and the salary per annum is more than 50,000 and sales is ₹ 20,000. The action taken for this rule is - 20% commission is given because he is in the marketing department and is applicable for income tax.

In rule 4, the employee is in Finance ("FIN") department, grade is between 10-19 and salary less than 50,000 per annum, so he is exempted from income tax and he is not given any commission as he is not in the marketing department.

In rule 5, he is in the marketing department, his grade is between 10-19, he need not pay income tax as salary is less than 50,000, he is not given commission as his sales is only ₹ 3000.

All these tables are made while design specifications are made. Each table is made according to a required module and you can see how each table is designed and how actions are taken.

ELSE Form

This form aims at omitting repetition by using ELSE rules.

To build an ELSE form decision table:

- Specify the rules with condition entries, to cover all sets of actions except for one.
- This will be the rule to follow when none of the other explicit conditions are true.
- This rule is the final column on the right, the ELSE column.
- If none of the other conditions are true, then the ELSE decision rule is followed.

The ELSE rule eliminates the need to repeat condition that lead to the same actions.

The conditions for the ELSE form is that the employees of marketing department are given commission according to the total sales done.

The commission is given as follows:

If Sales > = 10000 then 20% commission

If sales > = 8000 and sales < 10000 then 15% commission

If sales > = 5000 and sales < 8000 then 10% commission

Else

If sales < 5000 then 2% commission

Table 8.5

Conditions and Actions	Rules			
	1	2	3	4
Sales > = 10000	N	Y	N	E
Sales > = 8000 and < 10000	Y	-	N	L
Sales > = 5000 and < 8000	N	-	Y	S
				E
Commission	15%	20%	10%	2%

The Else form decision table shown below, has an extra ELSE column. The ELSE is applicable for all the marketing department employees who have made sales less than 5000.

The first rule tells us that the employee is in the marketing department, his sales amount is between 8000 and less than 10000, so he gets 15% commission.

8.5 STRUCTURE CHARTS

Another structured programming tool is structure Charts that is discussed below.

A structure chart is a diagram consisting of rectangular boxes representing modules and connecting arrows. Structure charts encourage top-down structured design and modularity. Top-down structured design deals with the size and complexity of an application by breaking it up into a hierarchy of modules that result in an application that is easier to implement and maintain.

Top-down design allows the systems analyst to judge the overall organizational objectives and how they can be met in an overall system. Then, the analyst moves to dividing that system into subsystems and their requirements. The modular programming concept is useful for the top-down approach: once the top-down approach is taken, the whole system is broken into logical, manageable-sized modules, or subprograms.

Modular design is the decomposition of a program or application into modules. A module is a group of executable instructions (code) with a single point of entry and a single point of exit. A module could be a subroutine, subprogram, or main program. It also could be a smaller unit of measure such as a paragraph in a COBOL program.

Data passed between structure chart modules has either Data Coupling where only the data required by the module is passed or Stamp Coupling where more data than necessary is passed between modules.

The modules in a structure chart fall into three categories:

- Control modules, determining the overall program logic
- Transformational modules, changing input into output
- Specialized modules, performing detailed, functional work

A lower level module should not be required to perform any function of the calling, higher level module. This would be "improper subordination."

Modules are represented by rectangles or boxes that include the name of the module. The highest level module is called the system, root, supervisor, or executive module. It calls the modules directly beneath it which in turn call the modules beneath them.

A connection is represented by an arrow and denotes the calling of one module by another. The arrow points from the calling (higher) module to the called (subordinate) module.

Note: The structure charts drawn in the Kendall and Kendall text book do not include the arrowhead on the connections between modules. Kendall and Kendall draw plain lines between module boxes.

A data couple indicates that a data field is passed from one module to another for operation and is depicted by an arrow with an open circle at the end.

A flag, or control couple, is a data field (message, control parameter) that is passed from one module to another and tested to determine some condition. Control flags are depicted by an arrow with a darkened circle at the end. Sometimes a distinction is made between a control switch (which may have two values, e.g., yes-no, on-off, one-zero, etc.) and a control flag (which may have more than two values).

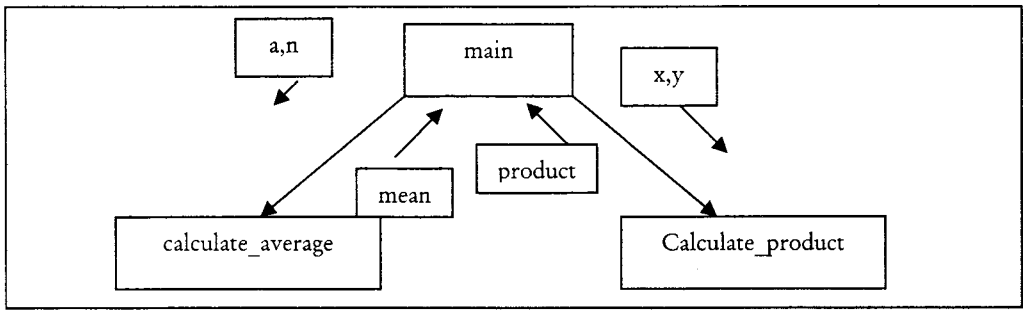
Modules that perform input are referred to as afferent while those that produce output are called efferent.

A structure chart is a graphic tool that shows the hierarchy of program modules and interfaces between them. Structure charts include annotations for data flowing between modules. An arrow from a module P to module Q represents that module P invokes module Q. Q is called the subordinate of P and P is called the super ordinate of Q. The arrow is labeled by the parameters receives as inputs by Q and the parameters returned as output by Q with the appropriate direction of flow. E.g.

```
void main()
{
      avg=calculate_average(a,n);
      product=calculate_product(x,y);
}
float calculate_average( float *x, int size)
{
```

```
        float sum=0.0;
        int I;
        for (i=0;i<size;i++)
        sum = sum + x[i];
        return (sum/size);
}
int calculate_product( int a, int b)
{
        return(a * b);
}
```

Structure Chart of the Program Above is as follow:



Check Your Progress

Fill in the blanks:

(a) A program flowchart is a .....

(b) An alternative method of representing program logic is a .....

(c) Mixed entry form consist of .....

8.6 LET US SUM UP

A program flowchart is a pictorial representation of the logic required to accomplish a task. It includes all necessary steps of a program. It is called a flowchart because it charts the flow of a program. It is a symbolic representation of each input, output and processing step. An alternative method of representing program logic is a pseudocode Instead of using symbols to represent the program logic steps, a pseudocode uses statements which are a bridge between actual programming and ordinary English. In a pseudocode, each step is written using a simple English phrase which is also called a construct. A Decision Table is a table of rows and columns that shows conditions and actions. 'Decision Rules', include a decision table, states what procedure to follow when certain conditions exist. After constructing a table, analysts verify it for correctness and completeness to ensure that the table includes all the conditions, along with the decision rules that relate them to actions. A structure chart is a diagram consisting of rectangular boxes representing modules and connecting arrows.

Structure charts encourage top-down structured design and modularity. Top-down structured design deals with the size and complexity of an application by breaking it up into a hierarchy of modules that result in an application that is easier to implement and maintain.

8.7 KEYWORDS

Flowchart: A program flowchart is a pictorial representation of the logic required to accomplish a task

Pseudocode : An alternative method of representing program logic is a pseudocode.

Decision Table: A Decision Table is a table of rows and columns that shows conditions and actions.

Structure Chart : A structure chart is a diagram consisting of rectangular boxes representing modules and connecting arrows.

8.8 QUESTIONS FOR DISCUSSION

1. Discuss the role of various programming tools with suitable examples.
2. A student appears for a test in 5 subjects. Each test is out of 100 marks. You have to calculate the percentage of each student. Finally, issue grades depending upon the student's percentage. The grades are given as under:

Percentage	Grade
> =90	E
80 - 90	A+
70 - 79	A
60 - 69	B+
50 - 59	B
<50	Fail

   Write down the pseudo code and draw the flow chart for the above problem.
3. Write a pseudo code that generates the first 20 numbers in the Fibonacci series.
4. Draw the flowchart to find the factorial of a number given by the user.
5. Write the pseudo code to calculate sum of all odd numbers in the range 20 to 50.
6. What are decision tables? Draw the decision table to calculate commission on sales.

   The commission is given as follows:

   If sales > =50000 then 25% commission

   If sales > =30000 and sales <50000 then 20% commission

   If sales > =10000 and sales <30000 then 10% commission

   else

   If sales <10000 then 5% commission.
7. What are structure charts? Discuss the modules available in structure chart.

Check Your Progress: Model Answers

(a) pictorial representation of the logic required to accomplish a task.

(b) pseudocode

(c) combined features of limited and extended-entry forms.

8.9 SUGGESTED READINGS

Kenneth Leroy Busbee, *Programming Fundamentals: A Modular Structured Approach*, University Press of Florida

Byron S. Gottfried, *Schaum's outline of theory and problems of programming with Basic*, McGraw-Hill

C. Joseph Sass, *BASIC programming and applications*, Allyn and Bacon

LESSON

9

STRUCTURED PROGRAMMING ISSUES

CONTENTS

9.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss structured programming issues
- Discuss correctness, reliability, etc.

9.1 INTRODUCTION

Structured programming techniques imply, that the program development and coding must follow a top-down approach and structured programming constructs.

In this lesson we will discuss some criteria used for a good program.

9.2 CRITERIA FOR A GOOD PROGRAM

Following are some of the questions which can be asked to judge if a program is good:

- Does it solve the problem stated in the specification?
- Does it work correctly under all conditions?
- Does it have proper documentation?
- Is it written using a modular approach?
- Does it make efficient use of computer time and memory?

These criteria's can be met automatically if sufficient thought and effort is invested in every stage of program design.

Following are the objectives that must be keep in mind while writing a program:

9.2.1 Correctness

A program should produce output as per specification. It should conform to end user requirements i.e. the user must be satisfied with the output that is produced from the program. Hence correctness is a very subjective term.

9.2.2 Reliability

A program should function accurately for a long period of time. It must also functions correctly over all ranges and combination of data.

9.2.3 Robustness

A program must be developed in such a way that it can handle out of range data or exceptional data without crashing. That is to say that if a program finds some piece of data which is not being handled by the application than such data should be recognised and an error message should be flashed.

9.2.4 Execution Efficiency

A program should execute in small core memory and should process data fast.

9.2.5 Ease of Use

A program should be user friendly i.e. supported by menu options, good user documentation and on-line help facility.

9.2.6 Maintainability

There should be a sensible program structure and use of standard language features so that it is easy to understand, make changes and enhance.

9.2.7 Portability

A program should use standard features of an operating system, language compliers and separate components dealing with data, peripherals and processing, so that the program can be executed on different machines. However, some minor changes may have to be made, which are essentially language dependent and cannot be avoided.

9.2.8 Verification with Validation

Like testing, verification is also intended to find errors. Executing a program in a simulated environment to find errors performs it.

When commercial systems are developed with the main aim of distributing them to dealers for sale purposes, they first go through verification, sometimes called alpha testing. The feedback from the validation phase generally brings some changes in the software to deal with errors and failure that are uncovered. Then a set of user sites is selected for putting the system into use on a live basis.

9.2.9 Modularity

Modularity refers to the division of software into separate modules which are differently named and addressed and are integrated later on in order to obtain the completely functional software. It is the only property that allows a program to be intellectually manageable. Single large programs are difficult to understand and read due to large number of reference variables, control paths, global variables, etc. The desirable properties of a modular system are:

- Each module is a well defined system that can be used with other applications.
- Each module has a single specific purpose.
- Modules can be separately compiled and stored in a library.
- Modules can use other modules.
- Modules should be easier to use than to build.
- Modules are simpler from outside than inside.

Modularity thus enhances the clarity of design which in turn eases coding, testing, debugging, documenting and maintenance of the software product.

It might seem to you that on dividing a problem into sub problems indefinitely, the effort required to develop it becomes negligibly small. However, the fact is that on dividing the program into numerous small modules, the effort associated with the integration of these modules grows. Thus, there is a number N of modules that result in the minimum development cost. However, there is no defined way to predict the value of this N.

In order to define a proper size of the modules, we need to define an effective design method to develop a modular system. Following are some of the criteria defined by Meyer for the same:

- ***Modular Decomposability:*** The overall complexity of the program will get reduced if the design provides a systematic mechanism to decompose the problem into sub-problems and will also lead to an efficient modular design.

- ***Modular Composability:*** If a design method involves using the existing design components while creating a new system it will lead to a solution that does not re-invent the wheel.
- ***Modular Understandability:*** If a module can be understood as a separate standalone unit without referring to other modules it will be easier to build and edit.
- ***Modular Continuity:*** If a change made in one module does not require changing all the modules involved in the system, the impact of change-induced side effects gets minimized.
- ***Modular Protection:*** If an unusual event occurs affecting a module and it does not affect other modules, the impact of error-induced side effects will be minimized.

One of the key concepts in the application of programming discipline is the design of a program as a set of units referred to as blocks or modules. A program module is defined as the part of a program that performs a separate function, e.g., input, input validation, processing of one type of input. A program module may be quite large, so that it may be further divided into logical sub modules. The process of subdivision continues until all modules are of manageable size in terms of complexity of logic and numbers of instructions.

Programs can be logically separated into the following functional modules:

1. Initialization
2. Input
3. Input data validation
4. Processing
5. Output
6. Error handling
7. Closing procedure

The modules reflect a logical flow for a computer program. After initialization, processing proceeds logically with input, input validation, various processing modules, and output. Error handling may be required during execution of any module.

Basic Attributes

A module is a collection of program statements with five basic attributes:

- Input
- Output
- Function
- Mechanism
- Internal data

Control Relationship between Modules

The structure charts show the interrelationships of modules by arranging them at different levels and connecting modules in those levels by arrows. An arrow between two modules means the program

control is passed from one module to the other at execution time. The first module is said to call or invoke the lower level modules.

There are three rules for controlling the relationship between modules:

1. There is only one module at the top of the structure. This is called the root or boss module.
2. The root passes control down the structure chart to the lower level modules. However, control is always returned to the invoking module and a finished module should always terminate at the root.
3. There can be no more than one control relationship between any two modules on the structure chart, thus, if module A invokes module B, then B cannot invoke module A.

Communication between Modules

Two types of informations are passed between modules:

1. Data : Shown by an arrow with empty circle at its tail.
2. Control : Shown by a filled-in circle at the end of the tail of arrow.

Module Design Requirements

A hierarchical (or modular) structure should present many advantages in management, developing, testing, and maintenance. However, such advantages will occur only if modules fulfill the following requirements.

1. ***Coupling:*** Coupling means the strength of relation between the modules in a system so that change in one module has limited effect on any other module. It means that coupling should be minimized.
2. ***Cohesion:*** Cohesion means strength of relations within the module. It means cohesion should be maximized.
3. ***Span of Control:*** It means number of modules subordinate to calling module. Limit of span of control may vary from 5 to 7 modules.
4. ***Size:*** The number of instructions contained in a module should be limited as that module size is small.
5. ***Shared Use:*** Functions should not be duplicated in separated modules, but established in a single module that can be invoked by any other module when needed.

Check Your Progress

1. Define communications between modules.
2. Define verification.

9.3 LET US SUM UP

Structured programming techniques imply, that the program development and coding must follow a top-down approach and structured programming constructs. Sequence indicates that if there is no intervening control structure, the flow of control normally passes from one instruction to the next, in

sequence. Selection indicates a decision, from among several options Repetition or looping constructs occur, when a set of instructions are to be repeated several times.

A program should produce output as per specification. A program should function accurately for a long period of time. There should be a sensible program structure and use of standard language features so that it is easy to understand, make changes and enhance. A program should use standard features of an operating system, language compliers and separate components dealing with data, peripherals and processing, so that the program can be executed on different machines. Like testing, verification is also intended to find errors. Executing a program in a simulated environment to find errors performs it. Modularity refers to the division of software into separate modules which are differently named and addressed and are integrated later on in order to obtain the completely functional software. It is the only property that allows a program to be intellectually manageable.

9.4 KEYWORDS

Modularity: Modularity refers to the division of software into separate modules which are differently named and addressed and are integrated later on in order to obtain the completely functional software.

Verification: Verification is intended to find errors.

Coupling: Coupling means the strength of relation between the modules in a system so that change in one module has limited effect on any other module. It means that coupling should be minimized.

Cohesion: Cohesion means strength of relations within the module. It means cohesion should be maximized.

9.5 QUESTIONS FOR DISCUSSION

1. Discuss the criteria for a good program.
2. What are the different types of objectives considered while writing the program?
3. What is modularity? Discuss some criteria's related to modularity.
4. What are Module Design Requirements? Discuss.
5. Differentiate between cohesion and coupling.

Check Your Progress: Model Answers

1. Two types of informations are passed between modules:
   - i.. Data : Shown by an arrow with empty circle at its tail.
   - ii. Control : Shown by a filled-in circle at the end of the tail of arrow.
2. Like testing, verification is also intended to find errors. Executing a program in a simulated environment to find errors performs it. When commercial systems are developed with the main aim of distributing them to dealers for sale purposes, they first go through verification, sometimes called alpha testing.

9.6 SUGGESTED READINGS

Kenneth Leroy Busbee, *Programming Fundamentals: A Modular Structured Approach*, University Press of Florida

Byron S. Gottfried, *Schaum's outline of theory and problems of programming with Basic*, McGraw-Hill

C. Joseph Sass, *BASIC programming and applications*, Allyn and Bacon

10.1 INTRODUCTION

To solve a computation problem, its solution must be specified in terms of sequence of computational steps such that they are effectively solved by a human agent or by a digital computer. Computer understandable notation for the specification of such a sequence of computational steps is referred to as programming language.

Software testing is a critical element of software quality assurance and resents the final review of specification, design, and code. The source-code once generated needs to be tested for bugs and defects to get rid of maximum errors before the software is sent to the customer. Test cases are designed with an aim to find errors.

During initial stages of testing, a software engineer performs all the tests. However, at later stages a specialist may be involved. Tests must be conducted to find the highest possible number of errors, must be done systematically and in a disciplined way. Testing involves checking both the internal program logic and the software requirements.

10.2 PROBLEM SOLVING APPROACHES

The specification of the sequence of computational steps in a particular programming language is termed as a program. The task of developing programs is called programming and the person engaged in programming activity is called programmer.

Algorithm

Every problem has some given information and some results are to be obtained. But the question is that what procedure should be applied to the given information so that we can get the required result.

The development of a proper procedure is called Algorithm. The following examples would explain algorithm more effectively.

e.g.: Obtain the percentage of marks obtained by a student in an examination.

Solution: In the problem maximum marks and marks obtained is given. The required result is percentage of marks and the formula used is

% of marks =

An approximate algorithm for the problem can be as follows:

Step 1 : Read name, marks obtained, and maximum marks.

Step 2 : Divide marks obtained by maximum marks and store it in Per.

Step 3 : Multiply Per. by 100 to get percentage.

Step 4 : Write name and percentage.

Step 5 : Stop.

Rules for Developing Algorithm

There are no set rules for developing algorithms. While writing an efficient algorithm, the following points must be kept in mind:

1. Every procedure should carefully specify the input and output requirements.
2. Meaning of variables should be clearly defined.
3. The flow of program should generally be forward except for normal looping and unavoidable instance.

10.2.1 Top-down Design

The top-down design approach is based on the fact that large problems become more manageable if they are divided into a number of smaller and simpler tasks which can be tackled separately. What is really required is that each of these parts have the properties of a module.

Top-down design approach is performed in a special way. The main program is written first. It is tested before sub programs are written. To do this, the actual sub programs are replaced with stubs. The stubs simply test to see if the data is passed correctly. After the main program is written and checked, each module is written and tested in turn. This should first be done without the main program in order to isolate a stub if an error occurs. A simple main program is written to test the sub programs. If the modules run properly, then it is tested with the main program. If the module and the main program run properly, then the next module is written and checked and so on. To describe the program at its highest level, we use something called the universal program. The process of *stepwise refinement* work out the details of each part of the program.

Advantages

1. At each stage, the sub programs are tested by themselves and then the main program is tested. Whenever modules are added, they are tested with the main program, hence, if any error occurs it will probably be in a module only and this will be easy to debug.
2. It is desirable for modules to be kept small in general. As far as possible a module should be less than 100 lines long.

10.2.2 Bottom-up Design

A bottom-up approach would be to write the most basic subroutines in the hierarchy first and then use them to make more sophisticated subroutines. The pure bottom-up approach is generally not recommended because it is difficult to anticipate which low level subroutines will be needed for any particular program. It can often be a useful first step to produce a library of basic functions and procedures before embarking on a major project.

In the bottom-up approach it is usually assumed that the basic routines created will be general enough to be used more than once. Using the subroutines, to construct a program, save yourself repeating the same lines of code by reusing it. A routine that is used many times has a very difficult status to those higher in the hierarchy. It is more like a basic instruction in the programming language than a large scale program component.

10.2.3 Brute Force Approach

Brute force approach is a straight forward approach to solve the problem. It is directly based on the problem statement and the concepts.

- It is one of the simplest algorithm design to implement and covers a wide range of problems under its gamut
- Brute force is a simple but a very costly technique
- Example: Breaking Password.

One approach in optimization is straightforward and requires considerable computation power: brute force methods which try to calculate all possible solutions and decide afterwards which one is the best. These methods are feasible only for small problems (in terms of the dimensionality of the phase space), since the number of possible states of the system increases exponentially with the number of dimensions. In the case of continuous predictor variables, the number of states is infinite. Despite these drawbacks, brute force methods do have a few benefits: they are simple to implement, and in the case of discrete systems, all possible states are checked.

As a consequence, brute force methods are often seen as reference methods for calculating the number of states, or the number of calculations necessary to find the optimum with a probability of 100%. Hence, it can be used for the estimation of the effort to solve a problem.

The implementation of brute force algorithms is rather simple. In fact, one only has to try out all possible states of a system. If this is not possible, because the system is described by continuous variables, one has to try all possibilities according to a certain definition of precision for each continuous variable.

Example:

Consider a system which is controlled by 3 continuous predictor variables: x_1 to x_3: if x_1, and x_3 need a resolution of 200 intervals, and x_2 has to be discretized using 1,000 steps, the resulting number of calculations in a brute force approach will be 200 times 1,000 times 200 = 40,000,000.

10.3 TESTING FUNDAMENTALS

Testing software can be considered as the only destructive (psychologically) step in the entire life cycle of software production. Although all the initial activities aimed at building a product, the testing is done to find errors in software.

10.3.1 Objectives of Testing

- Executing a program in order to find errors.
- A good test case is the one that has a high probability of finding an undiscovered error.
- A successful test is the one that reports an as-yet undiscovered error.
- The objective while designing tests is to come up with the test cases that systematically uncover different classes of errors in minimum possible time and effort.

10.3.2 Benefits of Testing

- It reveals the errors in the software.
- It ensures that software is functioning as per specifications and it meets all the behavioral requirements as well.

- The data obtained during testing is indicative of software reliability and quality as a whole.
- It indicates presence of errors and not absence of errors.

10.3.3 Testing Principles

Before coming up with ways to design efficient test cases, one must understand the basic principles of testing:

1. ***Test cases must be traceable to requirements.*** Because software testing reveals errors, so, the severe defects will be those that prevent the program from acting as per the customer's expectations and requirements.
2. ***Test planning must be done before beginning testing.*** Test planning can begin soon after the requirements specification is complete and the detailed test cases can be developed after the design has been fixed.
3. ***Pareto principle applies to software testing.*** Pareto principle states that 80 percent of the uncovered errors during testing will likely be traceable to 20 percent of all the program components. Thus, the main aim is to thoroughly test these 20 percent components after identifying them.
4. ***Testing should begin with small and end in large.*** The initial tests planned and carried out are usually individual components and as testing progresses the aim shifts to find errors in integrated components of the system as whole rather than individual components.
5. ***Exhaustive testing is impossible.*** For a normal sized program the number of permutations of execution paths is very huge. Thus, it is impossible to execute all the combinations possible. Thus, while designing a test case it should be kept in minds that it must cover the maximum logic and the component-level design.
6. ***Efficient testing can be conducted only by a third party.*** The highest probability of finding errors exists when the testing is not carried by the party which develops the system.

A program developed should be testable i.e. it should be possible to test the program. The testability of a program can be measured in terms of few properties like: operability, observability, controllability, decomposability, simplicity, stability, understandability, etc.

The test also, must also comply with the characteristics of a good test case. These characteristics are mentioned here:

- The probability of finding error should be high. In order to achieve this, tester must understand the actual functionality of the software and think of a suitable condition that can lead to failure of the software.
- A test case must be non-redundant. Because the testing time and resources are limited, it will waste a lot of time to conduct the same test again and again. Every test must have a distinct purpose.
- A test case must be of the best quality. In a group of similar test cases, the one that covers the maximum scenarios to uncover maximum errors should only be used.
- A good test case should neither be too simple nor too complex. A complex test that includes several other tests can lead to masking of errors. Thus, each test case should be executed separately.

10.4 TEST CASE DESIGN

The designing of tests for software can be as challenging as the initial design of the product itself. Thus, software engineers must design test cases that have highest probability of finding defects within minimum time and effort.

A lot of methods exist to design test cases. These methods ensure systematic development of test cases, completeness and highest likelihood of finding errors.

A product can be tested in two ways: (1) Knowing the specified function that a product has been designed to perform; tests can be done to ensure that these functions are existing and that too without errors; (2) Knowing the internal workings of a product the tests can be performed to ensure that all the internal operations are being performed as per specifications and all internal components have been adequately built. The first approach is called black-box testing and the second, white-box testing.

Black-box testing of software refers to tests that are conducted at the software interfaces. These kinds of tests ensure that software functions are operational, input is accepted properly and output is correctly produced and that the external information e.g., database is maintained. It does not deal in depth with the internal logic of the program.

White-box testing works on the principle of closely monitoring the procedural details of the software. All the logical paths of the software are tested using test cases for specific conditions and loops. The actual results are compared with the expected results to ensure that the correct results are produced as a result of the functions being performed.

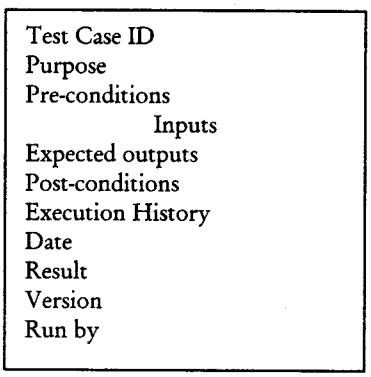


Figure 10.1: Typical Test Case Information

10.5 WHITE-BOX TESTING

White-box testing or glass-box testing is a test case design method that uses the control structure of the procedural design to obtain test cases. Using this methodology, a software engineer can come up with test cases that:

1. Guarantee that all independent paths within a module have been exercised at least once;
2. Exercise all the logical decisions based on their true and false sides;
3. Executing all loops at their boundaries and within them; and
4. Exercise internal data structures to ensure their validity.

The reason of conducting white-box testing largely depends upon the nature of defects in software:

- Logic errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed.
- We often believe that a logical path is not likely to be executed when, in fact, it may be executed on a regular basis.
- Typographical errors are random.

Each of these reasons provides an argument for conducting white-box tests. Black-box testing might miss out these kinds of errors.

10.6 BLACK-BOX TESTING

Black-box testing or behavioral testing, mainly focuses on the functional requirements of the software. It enables the software engineer to develop a set of inputs that can fully involve all the functional requirements for a program. It is not an alternative approach to white-box testing. Rather, it completes the testing by uncovering different types of errors than the white-box approach.

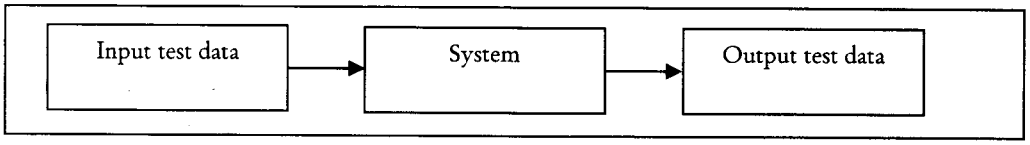


Figure 10.2: Black Box Testing

This approach attempts to find errors in the following categories:

1. Incorrect or missing functions
2. Interface related errors
3. Errors in data structures or database accesses
4. Performance related errors
5. Initialization and termination errors.

This technique is applied at a later stage unlikely to the white-box testing approach. Using this technique, we can arrive at test cases that satisfy the following criteria:

1. test cases that reduce the total test cases by a large quantity to achieve reasonable testing and
2. test cases that tell us something about the presence or absence of a variety of errors rather than a particular kind of error.

The techniques are as below:

10.6.1 Boundary Value Analysis

The art of testing is to come up with a small set of test cases such that the chances of detecting an error are maximized while minimizing the chances of creating redundant test cases that uncover similar errors. It has been observed that the probability of finding errors increases if the test cases are designed to check boundary values.

Consider a function F with x and y as two input variables. These inputs will have some boundaries:

$a <= x <= b$

$p <= y <= q$

Hence, inputs x and y are bounded by two intervals [a,b] and [p,q] respectively. For x, we can design test cases with values a and b, just above a and b and just below a and b which will have higher chances to detect errors. Similar is the case for y. This is represented in the Figure 10.3.

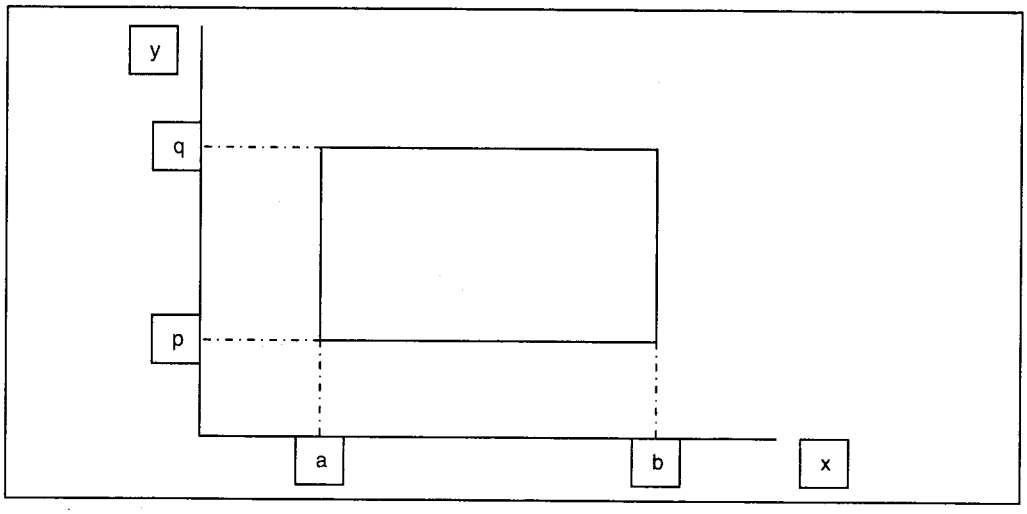


Figure 10.3: Input Domain of a Function of Two Variables

Boundary value test cases are obtained by keeping one variable at its extreme and the other at its nominal value. At times we might want to check the behavior when the value of one variable exceeds its maximum. This is called robustness testing. We may also include test cases to check what happens when more than one variable have maximum values. This is called worst case analysis. Using prior testing knowledge and experience to test similar programs is called *ad-hoc testing.*

10.6.2 Equivalence Class Testing

In this method of testing, input domain of a program is divided into a finite number of equivalence classes such that the test of a representative value of each class is equivalent to a test of any other value. i.e. if a test in a class detects one error all other test cases belonging to the same class must detect the same error. Also, if a test case in a class did not detect an error the other test cases of the same class also should not detect the error. This method of testing is implemented using the following two steps:

1. The equivalence class is identified by taking each input condition and dividing it into valid and invalid classes.
2. Developing test cases using the classes identified. This is done by writing test cases covering all the valid equivalence classes and a single case for invalid equivalence class.

Again good test cases will be those that check for boundary value condition.

10.6.3 Decision Table based Testing

Decision tables are useful for describing situations in which a number of combinations of actions are taken under varying sets of conditions. There are four parts of a decision table namely, Condition stub, Action stub, Condition entries and Action entries. These are described in Figure 10.4.

Condition Stub	Entry						
C1	True				False		
C2	True		False		True		False
C3	True	False	True	False	True	False	-
Action Stub A1	X	X			X		
A2	X		X			X	
A3		X			X		
A4				X		X	X

Figure 10.4: Decision Table Terminology

To develop test cases from decision tables, we treat conditions as input and actions as outputs. Sometimes conditions end up referring to equivalence classes of inputs, and actions refers to major functional processing portions of the item being tested.

10.6.4 Cause Effect Graphing Technique

One drawback of boundary value analysis and equivalence partitioning is that these they do not explore combinations of input circumstances which may result in interesting conditions. These situations must be tested. If we consider all possible valid combinations of equivalence classes, then it results in a large number of test cases, many of which may not uncover any undiscovered errors.

This technique helps in selecting, in a systematic approach, a high-yield set of test cases. It is also useful in pointing out incompleteness and ambiguities in the specifications. The following steps are used to derive test cases:

1. The causes and effects are identified. A cause is a distinct input condition and effects are output conditions or a system transformation. These are identified by reading the specification and identifying the words or phrases that describe causes and effects. Each cause and effect is assigned a unique number.
2. The semantic content of specification is studied and transformed into a Boolean graph linking the causes and effects. This is the cause effect graph.
3. The graph is annotated with constraints describing combinations of causes and/or effects that are impossible because of syntactic or environmental constraints.
4. By methodically tracing state conditions in the graph, the graph is converted into a limited entry decision table. Each column in the graph represents a test case.
5. The columns in the decision table are converted into test cases. The basic notation for the graph is shown in Figure 10.5.

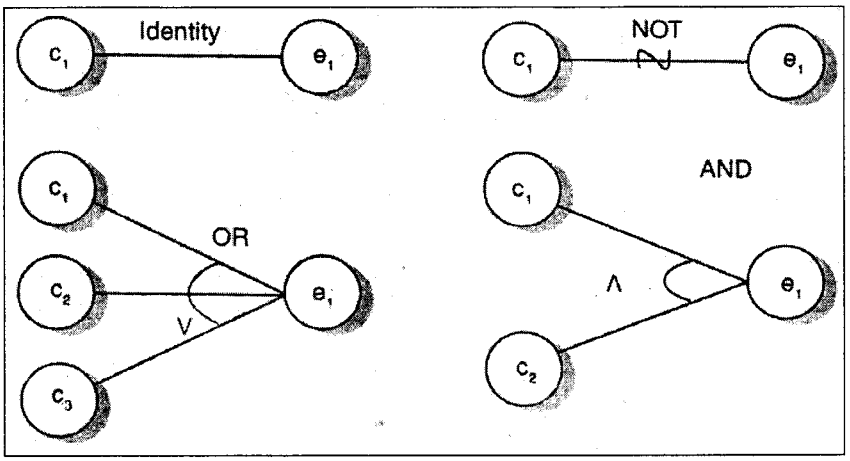


Figure 10.5: Basic Cause Effect Graph Symbols

Think of each node as having values either 0 or 1, where 0 represents the 'absent state' and 1 represents the 'present state'. The identity function states that if c_1 is 1, e_1 is 1; else e_1 is 0. The NOT function states that if c_1 is 1, e_1 is 0 and vice versa. The OR function states that if c_1 or c_2 or c_3 is 1, e_1 is 1 else e_1 is 0. The AND function states that if both c_1 and c_2 are 1, e_1 is 1 else e_1 is 0. The AND and OR functions can have any number of inputs.

Check Your Progress

1. What is Cause Effect Graphing Technique?
2. Define Decision Table based Testing.

10.7 LET US SUM UP

Every problem has some given information and some results are to be obtained. But the question is that what procedure should be applied to the given information so that we can get the required result. The development of a proper procedure is called Algorithm. The top-down design approach is based on the fact that large problems become more manageable if they are divided into a number of smaller and simpler tasks which can be tackled separately. What is really required is that each of these parts have the properties of a module. A bottom-up approach would be to write the most basic subroutines in the hierarchy first and then use them to make more sophisticated subroutines. The pure bottom-up approach is generally not recommended because it is difficult to anticipate which low level subroutines will be needed for any particular program. Brute force approach is a straight forward approach to solve the problem. One approach in optimization is straightforward and requires considerable computation power: brute force methods which try to calculate all possible solutions and decide afterwards which one is the best.

Testing software can be considered as the only destructive (psychologically) step in the entire life cycle of software production. Although all the initial activities aimed at building a product, the testing is done to find errors in software. White-box testing or glass-box testing is a test case design method that uses the control structure of the procedural design to obtain test cases. Black-box testing or behavioral testing, mainly focuses on the functional requirements of the software. It enables the software engineer to develop a set of inputs that can fully involve all the functional requirements for a program.

10.8 KEYWORDS

White-box Testing: White-box testing or glass-box testing is a test case design method that uses the control structure of the procedural design to obtain test cases.

Black-box Testing: Black-box testing or behavioral testing, mainly focuses on the functional requirements of the software.

Top-down Design: The top-down design approach is based on the fact that large problems become more manageable if they are divided into a number of smaller and simpler tasks which can be tackled separately.

BVA: The art of testing is to come up with a small set of test cases such that the chances of detecting an error are maximized while minimizing the chances of creating redundant test cases that uncover similar errors.

Brute Force Approach: It is a straight forward approach to solve the problem.

Equivalence Class Testing: In this method of testing, input domain of a program is divided into a finite number of equivalence classes such that the test of a representative value of each class is equivalent to a test of any other value.

Decision Tables: These are useful for describing situations in which a number of combinations of actions are taken under varying sets of conditions.

10.9 QUESTIONS FOR DISCUSSION

1. What is an algorithm? Give example. Discuss the rules for developing an algorithm.
2. What is top design approach? Give its advantages.
3. What is brute force approach? How it is used as a problem solving approach?
4. What is testing? Discuss the objectives of testing.
5. Discuss the basic principles of testing used to design efficient test cases.
6. Differentiate between white box testing and black box testing.
7. Differentiate between Boundary Value Analysis and Equivalence class testing.

Check Your Progress: Model Answers

1. This technique helps in selecting, in a systematic approach, a high-yield set of test cases. It is also useful in pointing out incompleteness and ambiguities in the specifications.
2. Decision tables are useful for describing situations in which a number of combinations of actions are taken under varying sets of conditions. There are four parts of a decision table namely, Condition stub, Action stub, Condition entries and Action entries.

10.10 SUGGESTED READINGS

Kenneth Leroy Busbee, *Programming Fundamentals: A Modular Structured Approach*, University Press of Florida

Byron S. Gottfried, *Schaum's outline of theory and problems of programming with Basic*, McGraw-Hill

C. Joseph Sass, *BASIC programming and applications*, Allyn and Bacon.

UNIT IV

LESSON

11

CLIENT-SIDE AND SERVER-SIDE PROGRAMMING LANGUAGES

CONTENTS

Contd...

11.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss the concept of Client-side and server-side programming languages
- Discuss declaring variables
- Understand working with date and time
- Discuss using mathematical operators and functions
- Discuss using conditional statements

11.1 INTRODUCTION

In the computer networking environment client server model is used. Server computer (also called a host computer) is a computer on the network that provides information and services to client computers. Server also manages network resources. For example, a file server is a computer and storage device dedicated to storing files. Any user on the network can store files on the server. A print server is a computer that manages one or more printers, and a network server is a computer that manages network traffic. A database server is a computer system that processes database queries.

The client computers are the systems who needs the services of server computer. The term "client" was first applied to devices that were not capable of running their own stand-alone programs, but could interact with remote computers via a network. These "dumb" terminals were clients of the time-sharing mainframe computer. The client-server model distinguishes client systems from server systems, which communicate over a computer network. A client-server application is a distributed system comprising both client and server software. A client software process may initiate a communication session, while the server waits for requests from any client.

The client/server model has become one of the central ideas of network computing. Most business applications being written today use the client/server model. So does the Internet's main program, TCP/IP. In marketing, the term has been used to distinguish distributed computing by smaller dispersed computers from the "monolithic" centralized computing of mainframe computers. But this distinction has largely disappeared as mainframes and their applications have also turned to the client/server model and become part of network computing.

In the usual networking environment, one server, sometimes called a daemon, is activated and awaits client requests. Typically, multiple client programs share the services of a common server program. Both client programs and server programs are often part of a larger program or application.

We propose that all client/server systems have the following distinguishing characteristics:

- ***Service:*** Client/server is primarily a relationship between processes running on separate machines. The server process is a provider of services. The client is a consumer of services. In essence, client/server provides a clean separation of function based on the idea of service.
- ***Shared resources:*** A server can service many clients at the same, time and regulate their access to shared resources.
- ***Asymmetrical protocol:*** There is a many-to-one relationship between clients and server. Clients always initiate the dialog by requesting a, service. Servers are passively awaiting requests from the clients.
- ***Transparency of location:*** The server is a process that can reside on the same machine as the client or on a different machine across a network. Client/server software usually masks the location of the server from the clients by redirecting the service calls when needed. A program can be a client, a server, or both.
- ***Mix-and-match:*** The ideal client/server software is independent of hardware or operating system software platforms. You should be able to mix-and-match client and server platforms.
- ***Message-based exchanges:*** Clients and servers-are loosely coupled systems that interact through a message-passing mechanism. The message is the delivery mechanism for the service requests and replies.
- ***Encapsulation of services:*** The server is a "specialist." A message tells a server what service is requested; it is then up to the server to determine how to get the job done. Servers can be upgraded without affecting the clients as long as the published message interface is not changed.
- ***Scalability:*** Client-server systems can be scaled horizontally or vertically. Horizontal scaling means adding or removing client workstations with only a slight performance impact. Vertical scaling means migrating to a larger and faster server machine or multi servers.
- ***Integrity:*** The server code and server data is centrally maintained, which results in cheaper maintenance and tile guarding of shared data integrity. At the same time, the clients remain personal and independent.

11.2 CLIENT-SIDE SCRIPTING LANGUAGES AND SERVER-SIDE SCRIPTING LANGUAGES

When a user requests a website by typing URL in address bar of browser, the request is admitted by the web server which hosts that site. Web Server performs some code to complete client request and produce output. The code which is implemented at server end is called server-side script and computer languages used to write this script are called server-side scripting languages. Examples of server-side scripting languages are CGI, ASP and PHP.

The reaction or output of web server is send to requesting client. This response consists of HTML and some other script. The HTML is provided by browser. Browser also performs the script received from server. This script or small program implanted in HTML is called Client-side script and we can write

this script in client-side scripting languages. Client-side script is used for client-side validation and other operations such as drop down menus, moving fonts. For example when user fills in an entry form,. to ensure that user has typed all necessary fields we require to write code to verify this before saving it in database. This validation code can be written to execute on server-side or client-side. If we write this code in client-side scripting language this will save time and traffic load between server and client. Examples of client-side scripting language are JavaScript and VB Script.

11.2.1 Client-side Languages

All web pages (well, almost) are printed with HTML. HTML is a markup language, which fundamentally permits you to format content, place images, tables, etc. In modern years, new languages are coming up, which harmonize HTML:

- *CSS:* CSS is a language that assists you separate content and formatting. It's not a programming language, but it helps you classify text and imaging formatting for your content more simply than HTML does.
- ***Javascript:*** A script language that assists you make your pages more vibrant. Simple uses involve dynamic menus and effects. You can write complicated software using Javascript, see the 'Asynchronous web' part below.
- ***Other scripting languages:*** There are more options for client-side scripting, such as VBScript. But these are less common then Javascript

All of these languages are actually part of the HTML page, they just make it better. We'll discuss them later on.

11.2.2 Server-side Languages

There are many server-side languages. Most of them are used to the similar purpose - to generate software (an executable, or script) that runs on the server-side, and produce a HTML page (perhaps with CSS and Javascript, too). Common languages include:

- *PHP:* An open source language, which seems to be the most popular server-side scripting language today.
- *ASP:* Active Server Pages, Microsoft own language.
- *ASP.NET:* The new generation of ASP. It will run as a Dot Net runtime, on the server machine.
- *JSP:* Java server pages - like ASP, but written in Java.

These are the type of languages that permit you to build complicated web sites, such as Amazon.com. Typically they will interact with a Database to save and retrieve data. Common databases are MySQL for the open source world, SQLServer from Microsoft, and of course Oracle and all the rest.

11.3 VARIABLE DECLARATION AND "DIM" STATEMENT

This section offers a quick foreword of what is a variable and how variable can be declared explicitly with a Dim statement and implicitly with an assignment statement.

Like many other programming languages, VBScript uses variables to set aside memory to store data and to name that memory location.

A variable must be declared with a name and a particular data type. Since VBScript supports only one data type, Variant, all variables will be declared as Variant by default. Here are some rules on variable declaration:

1. ***Explicit Declaration*** – A variable is declared with the "Dim" statement in the following syntax:

   Dim variable_name, ...

   where "variable_name" is a text label to recognize this variable. Multiple variables can be declared with single "Dim" statement. The data type of this variable will be Variant.

   Note that, in Visual Basic, you are permitted to identify a particular data type in the Dim statement, because multiple data types are supported in Visual Basic. So the subsequent statement is valid in Visual Basic:

   Dim author As String ' Allowed in Visual Basic

   But in VBScript, you are not permitted to identify the data type in a Dim statement. So statements below are not valid in VBScript:

   Dim author As String ' Not allowed in VBScript

   Dim buffer As Variant ' Not allowed in VBScript

2. ***Implicit Declaration*** – A variable name is used on the left side of an assignment statement without being declared before. The data type of an implicitly declared variable is "Variant". Here are some examples of variable implicit declaration:

   author = "Herong" ' Implicit declaration of "author"

   buffer = 1/3 ' Implicit declaration of "buffer"

   Just in case you want to know, "Dim" stands for "Dimension".

Some rules about variable names:

- A variable name must begin with an alphabetic character. For example, "price" is a valid variable name. But "$price" is not a valid variable name.
- The dot character (.) can not be used in a variable name. For example, "my.price" is not a valid variable name.
- Variable names are case insensitive. For example, "author" and "Author" are the same variable name.
- VBScript reserved keywords can not be used as variable names. For example, "empty" can not be used as a variable name.

11.4 WORKING WITH DATE AND TIME

Applications need to be able to access and work with date and time values. Many applications are written for business and scientific purposes, where recording the date and time of the program run is vital to the success of the project. Visual Basic includes the date and time functions described below.

VBScript supports the following normally used date and time functions:

- CDate(vVariant) - Converts a valid date and time expression to the variant of subtype Date
- Date() - Returns the current system date

- DateAdd(sType, iInterval, tDate) - Returns a date to which a specified time interval has been added
- DateDiff(tDate1, tDate2) - Returns the number of intervals between two dates
- DatePart(sType, tDate) - Returns the specified part of a given date
- DateSerial(iYear, iMonth, iDate) - Returns the date for a specified year, month, and day
- DateValue(vVariant) - Returns a date
- Day(tDate) - Returns a number that represents the day of the month (between 1 and 31, inclusive)
- FormatDateTime(tDate, iType) - Returns an expression formatted as a date or time
- Hour(tDate) - Returns a number that represents the hour of the day (between 0 and 23, inclusive)
- IsDate(vVariant) - Returns a Boolean value that indicates if the evaluated expression can be converted to a date
- Minute(tDate) - Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
- Month(tDate) - Returns a number that represents the month of the year (between 1 and 12, inclusive)
- MonthName(iMonth) - Returns the name of a specified month
- Now() - Returns the current system date and time
- Second(tDate0 - Returns a number that represents the second of the minute (between 0 and 59, inclusive)
- Time() - Returns the current system time
- Timer() - Returns the number of seconds since 12:00 AM
- TimeSerial(iHour, iMinute, iSecond) - Returns the time for a specific hour, minute, and second
- TimeValue(tDate) Returns a time
- Weekday(tDate) - Returns a number that represents the day of the week (between 1 and 7, inclusive)
- WeekdayName(iWeekDay) - Returns the weekday name of a specified day of the week
- Year(tDate) - Returns a number that represents the year

Date data type not only provides a place to store a date but also performs many tasks on a Date variable, such as adding or subtracting time, comparing dates and obtaining the System date and time.

11.4.1 Creating and Initializing a Date

A Visual Basic Date can be created in the same way any other variable is created. For example the following Visual Basic code excerpt creates a Date variable:

```
Dim dteAppointment As Date
```

The Date value is initialized using a date string in the from mm/dd/yyyy encapsulated in hash (#) characters. For example to set the date to August 2, 2007:

Dim dteAppointment As Date = #8/2/2007#

A Date variable may be set to a specific time using the format hh:mm:ss AM/PM. For example, to set the time to 1:02 PM:

Dim dteAppointment As Date = #1:02:00 PM#

To set both the date and time:

Dim dteAppointment As Date = #8/2/2007 1:02:00 PM#

11.4.2 Accessing the Date and Time in a Date Variable

The simplest way to obtain the current setting of a Date variable is simply to reference the Date as you would any other variable, using the variable name. For example, the following Visual Basic code excerpt will display the current setting of the Date variable in a MessageBox:

Dim dteAppointment As Date = #8/2/2007 1:02:00 PM#

MsgBox (dteAppointment)

11.4.3 Formatting Dates and Times

The default format of the Date and Time in Visual Basic is not always acceptable. For this reason, Visual Basic provides the Format() function to allow the format of the date to be controlled. The Format() function is actually a diverse function that can be used to format various types of data (such as string, monetary values and numbers). The Format() function uses the following syntax:

Format(value, style)

The value represents the data to be formatted. The style parameter defines how the data is to be formatted. In terms of formatting dates, a number of pre-defined styles are available.

The month can be formatted using sequences of 'M' characters. For example:

Format(#8/2/2007#, "MM") - returns Month as 2 digits (i.e. 08)

Format(#8/2/2007#, "MMM") - returns abbreviated Month name (i.e. Aug)

Format(#8/2/2007#, "MMMM") - returns full Month name (i.e. August)

Similarly the day of the month may be formatted using the 'd' character:

Format(#8/2/2007#, "d") - returns day of month as 1 (day < 10) or 2 digits (day > 9) (i.e. 2 or 12)

Format(#8/2/2007#, "dd") - returns day of month as 2 digits (i.e. 02 or 12)

Format(#8/2/2007#, "ddd") - returns abbreviated day of week (i.e Tue)

Format(#8/2/2007#, "dddd") - returns unabbreviated day of week (i.e Tuesday)

The 'y' character is used to control the formatting of the year:

Format(#8/2/2007#, "y") - returns year as 1 digit (i.e. 7)

Format(#8/2/2007#, "yy") - returns year as 2 digits (i.e. 07)

Format(#8/2/2007#, "yyyy") - returns year as 4 digits (i.e. 2007)

The format of the time may be formatted using 'h', 'm', 's' and 'tt' characters:

Format(#1:02:00 PM#, "hh:mm:ss tt") - returns as 01:02:00 PM

Format(#1:02:00 PM#, "hh:mm:ss") - returns as 01:02:00

Format(#1:02:00 PM#, "hh:mm") - returns as 01:02

Format(#1:02:00 PM#, "h:mm") - returns as 1:02

The format styles may be combined in any order to create the desired format. For example:

Format (#8/2/2007 1:02:00 PM#, "MMMM, d, yyyy, hh:mm") - Returns August, 2, 2007, 01:02

11.4.4 Adjusting a Date or Time

The value of a Date variable can be adjusted either backwards or forwards using the Visual Basic DateAdd() function. The Syntax of DateAdd() is as follows:

DateAdd(interval, number, date) As Date

The interval parameter specifies the unit of time to be added or subtracted (such as day, year, hour, minute etc). The allowable values for this parameter are as follows:

Value Unit of Time to Add/Subtract

DateInterval.Day Day

DateInterval.DayofYear Day

DateInterval.Hour Hour

DateInterval.Minute Minute

DateInterval.Month Month

DateInterval.Quarter Quarter

DateInterval.Second Second

DateInterval.Weekday Day

DateInterval.WeekOfyear Week

DateInterval.Year Year

For example, the following code will return a date one month into the future from the specified date:

DateAdd(DateInterval.Month, 1, #8/2/2006#)

The following code excerpt returns a date one week previous to the specified date:

DateAdd(DateInterval.Week, -1, #8/2/2006#)

11.4.5 Retrieving Parts of a Date or Time

The interval types listed in the DateAdd() section above can be used with the Visual Basic DatePart() function to extract parts of a date or time. The syntax for the DatePart() function is as follows:

DatePart(interval, date) As Integer

For example:

DatePart(DateInterval.Month, #8/2/2007#) ' Returns 8

DatePart(DateInterval.Day, #8/2/2007#) ' Returns 2

11.4.6 Finding the Interval between Two Dates or Times

The difference between to dates or times can be determined using the Visual Basic DateDiff() function, the syntax for which is:

DateDiff(interval, Date1, Date2)

The interval parameter specifies the unit of time to use when measuring the difference between two date and using the same values as those outlined for the DateAdd() function covered previously in this lesson. The two date values the represent the dates to compared. The function returns the difference as the number of specified units. For example, if the interval is defined as DateInterval.Day and the two dates are a week apart, DateDiff() will return 7.

The following examples demonstrate possible uses of the DateDiff() function:

DateDiff(DateInterval.Year, #08/02/2006#, #08/02/2007#) ' one year difference so returns 1

DateDiff(DateInterval.Month, #08/02/2007#, #09/02/2007#) ' one month difference so returns 1

DateDiff(DateInterval.Month, #08/02/2006#, #08/02/2007#) ' one year difference so returns 12 (months)

11.4.7 Accessing the System Date and Time

One of the most common tasks related to dates and times in a Visual Basic application involves getting the current system date and time. This is achieved using the Visual basic DateTime object. The DateTime object contains a number of properties, the most important of which are Now and Today. The Today property returns the current date held by the system. For example:

Dim objCurrentDate As Date = DateTime.Today

The Now property of the Visual Basic DateTime object contains both the date and the current system time. For example:

Dim objCurrentDate As Date = DateTime.Now

11.4.8 Checking if a Value is a Valid Date

Visual Basic provides the IsDate() function to ascertain whether a value is a valid date or not. This can be useful for checking that a user has entered a valid date into TextBox control. The IsDate() function takes a value as a parameter and returns True or False depending on whether the value is a valid date or not.

11.5 USING MATHEMATICAL OPERATORS

Arithmetic operators work on numeric operands and produce numeric result. Following are the arithmetic operators.

11.5.1 Addition

Addition is performed in Visual Basic using the plus (+) operator.

Purpose: To add two numeric type data values.

Syntax: result = expression1 + expression2

The + operator syntax has these parts:

Part	Description
Result	A required numeric variable. It cannot be constant.
expression1	A required expression evaluating to a numeric value.
expression2	A required expression evaluating to a numeric value.

For example:

```
Dim intValue As Integer
intValue = 10 + 5 + 6
```

Variables may also be used as operands:

```
Dim intValue, intOperator1, intOperator2 As Integer
intOperator1 = 10
intOperator2 = 5
intValue = 10 + intOperator1 + intOperator2
```

Symbol: +

11.5.2 Subtraction and Negation

The subtraction (-) is used to perform subtraction in Visual Basic:

Purpose: To find the difference between two numbers or to indicate the negative value of a numeric expression.

Syntax1: result = number1 - number2

Syntax2: - number

The – operator syntax has these parts:

Part	Description
Result	A required numeric variable.
number	A required numeric expression.
Number1	A required numeric expression.
Number2	A required numeric expression.

For example:

```
Dim intValue, intOperator1, intOperator2 As Integer
intOperator1 = 10
intOperator2 = 5
intValue = 10 - intOperator1 – intOperator2
```

A negative number is denoted by prefixing the number with the (-) sign:

intOperator2 = - 67

11.5.3 Multiplication

Visual Basic, like most other programming languages, use the (*) operator (rather than the x we all use when writing mathematical expressions) to perform multiplications.

Purpose: To multiply two numbers.

Syntax: result = number1 * number2

The * operator syntax has these parts:

Part	Description
Result	A required numeric variable.
Number1	A required numeric expression.
Number2	A required numeric expression.

For example:

```
Dim intValue, intOperator1, intOperator2 As Integer
intOperator1 = 10
intOperator2 = 5
intValue = intOperator1 * intOperator2
```

11.5.4 Division

Division is achieved in Visual Basic using the / operator. In division, the left hand operand is divided by the right hand operand. For example, the following example returns 5 (20 divided by 4):

```
Dim intValue, intOperator1, intOperator2 As Integer
intOperator1 = 20
intOperator2 = 5
intValue = intOperator1 * intOperator2
```

Division-1

Symbol: /

Purpose: To divide two numbers and return a floating-point result.

Syntax: result = number1/ number2

The / operator syntax has these parts:

Part	Description
Result	A required numeric variable.
Number1	A required numeric expression.
Number2	A required numeric expression.

Division-2

Symbol: \

Purpose: To divide two numbers and return an integer result.

Syntax: result = number1 \ number2

The \ operator syntax has these parts:

Part	Description
Result	Required; any numeric variable.
Number1	Required; any numeric expression.
Number2	Required; any numeric expression.

11.5.5 Exponentiation

Exponentiation involves raising a number to a particular power. For example 103 which would evaluate to 1000. The carat (^) character is used to represent exponentiation in Visual Basic:

```
Dim intValue As Integer
intValue = 10 ^ 3
```

Purpose: To raise a number to the power of an exponent.

Syntax: result = number ^ exponent

The ^ operator syntax has these parts:

Part	Description
Result	Required; any numeric variable.
Number	Required; any numeric expression.
Exponent	Required; any numeric expression.

11.5.6 Remainder Operator

Modulus involves the division of two numbers and obtaining the remainder of that division. Modulus calculations are performed in Visual Basic using the Mod keyword.

For example:

10 Mod 2 - returns 0

10 Mod 4 - returns 2

12 Mod 5 - returns 2

Symbol: Mod

Purpose: To obtain remainder by divide two numbers.

Syntax: result = number1 **Mod** number2

The Mod operator syntax has these parts:

Part	Description
Result	A required numeric variable.
Number1	A required numeric expression.
Number2	A required numeric expression.

11.5.7 Assignment Operator

Purpose: To assign/store a value to a variable (or property).

Syntax: variable = value

The = operator syntax has two parts:

Part	Description
Variable	Any variable or any writable property.
Value	Any numeric or string literal, constant, or expression.

The name on the left side of the equal sign can be a simple scalar variable or an element of an array. Properties on the left side of the equal sign can only be those properties that are writable at runtime.

11.6 USING MATHEMATICAL FUNCTIONS

Similar to other programming languages, VBScript offers a rich library of incorporated functions to deal with including strings, dates, typecasting, basic math, etc. When you consider generating functions in any programming language, it is useful to publicize yourself with the obtainable built-in functions of that language. The thought is if the programming language already provides a built-in function for a specific task, as a programmer, you do not have to create your own function for that task. The incorporated function is likely to be more efficient so shun reinventing-the-wheel!

In this section, we will discuss some of the built-in mathematical functions available in VBScript. Most of those functions are very simple to use. Table 11.1 provides a reference to mathematical functions available in VBScript.

Table11.1: Built-in Math Functions

Built-in math function	Meaning
Abs (number)	Absolute value
Exp (number)	e ^ number; e = 2.71828182845905
Fix (number)	Returns integer portion
Int (number)	Returns integer portion
Hex (number)	Coverts base 10 to hexadecimal
Oct (number)	Coverts base 10 to octal
Round (number)	Rounds to an integer
Round (number, dec)	Rounds to dec decimal places.
Sgn (number)	Sign
Sqr(number) (number > 0)	Square root

Here is a swift review of some of the incorporated math function available in VBScript:

- Abs (number) returns the absolute value of a number. In other words, if the number is greater than or equal to zero, the number is returned. If, however, the number is less than zero, the negation of the number is returned. For instance, Abs (5) would return 5 and Abs (-1) would return 1.
- Exp (number) raises e (approximately 2.71828) to the power number. For example, Exp (5) says e^5, which is approximately 148.41315.
- Fix (number) returns integer portion, greater than or equal to the number. For instance, Fix (-5.80) returns -5.
- Int (number) is different from Fix(number) because Int (number) returns integer portion less than or equal to number. For example, Int (-5.80) returns -6.
- Hex (number) returns strings consisting of the number converted to hexadecimal base. For example, Hex (10) returns A, Hex (11) returns B, Hex (12) returns C, etc.
- Oct (number) returns strings consisting of the number converted to octal base. For instance Oct (8) returns 10, Oct (9) returns 11, Oct (10) returns 12, etc.
- Round (number) rounds a number to the nearest integer; for example, Round (5.45) rounds 5.45 to 6 and Round (5.01) returns 5.
- Round (number, dec) rounds a number to dec decimal places. Round (5.45, 1) rounds 5.45 to 1 decimal place: 5.4. Round (100.67899, 3) returns 100.679.
- Sgn (number) returns -1 if number is less than 0, 0 if number is 0, and 1 if number is greater than 0. Consider the following as examples: Sgn(-5.9) returns -1, Sgn(0) returns 0; and Sgn(5) returns 1.
- Sqr (number) (number > 0) provides a square root of number, provided the number is positive. For example, sqr (4) returns 2, sqr (100) returns 10.

11.7 USING CONDITIONAL STATEMENTS

11.7.1 If - Then

Life is full of choices (isn't it so nice??). We look around and what do we observe? We are constantly making decisions every day. We leave home and if it is raining, we take our raincoats along. Our mother might say, go to the market and buy 5 kg potatoes if potatoes cost less than 10 rupees a kilo.

In programming terms, we can say this: 'if' it is raining, 'then' put on the raincoat. Similarly, 'if' potatoes are < Rs. 10/kg, 'then' buy 5 kgs. The underlying concept is this: 'if' a condition is 'true' 'then' we will take some action. The same approach can also be applied to programming. We can specify a condition to VB. 'If' the condition given by us is true, 'then' all of the statements following the 'if' will be executed, otherwise VB will jump over those statements. For example, the following lines of code

Code Listing if.1

```
IF text1.TEXT  =  "Shreyas" THEN
MSGBOX " Hello "  &  " Shreyas "
END IF
```

will display a message box with the message "Hello Shreyas" printed on it, 'if' Text1 contains the text 'Shreyas'. If Text1 does not contain the text Shreyas, the message box will NOT be displayed. Take another example. Suppose there are two integer variables, named int_a and int_b. Analyze the code segment that follows.

Code Listing if.2

```
IF int_a > int_b THEN
Text2.TEXT = int_a
END IF
```

The value of variable int_a will be printed in Text2, only if the value of variable int_a is greater than variable int_b. Otherwise, Text 2 will stay as it is.

General Syntax of If-Then

This brings us to the general syntax of the If - Then statement, which is as follows:

If condition Then

.....

'execute all of these statements if the condition is true

.....

.....

'if condition is false none of these statements will be executed

.....

End If

11.7.2 If - Then - Else

In our discussion of if-then, we have so far considered only one option. That is,

if int_a > int_b, then print int_a

But what if we want to print int_b if 'int_a > int_b' is not true? The simple if-then does not offer another option, i.e., it does something if the condition is true, but does nothing if the condition is false. This extra option is offered by if-then-else, a modification of the simple if-then we have just seen. To see this in action, we will modify the Code List if.1 as follows:

Code Listing if.3

```
IF text1.Text = "Shreyas" THEN
MSGBOX  " Hello "  &   " Shreyas"
ELSE
MSGBOX   " Hello "  &  " Mr/Miss No Name"
END IF
```

In this code, VB will check for the contents of Text1. If the text entered in "Text1" is "Shreyas", the message box "Hello Shreyas" will be displayed. If the text entered is not "Shreyas" but anything else, the message box "Hello Mr/Miss No Name" will be displayed. Modifying the Code List if.2,

Code Listing if.4

```
IF  int_a  >  int_b  THEN
text1.TEXT  =  int_a
ELSE
text1.TEXT  =  int_b
END IF
```

this will display value of the variable int_a in Text1 if the condition [int_a > int_b] is true. Otherwise, it will display value of the variable int_b in Text1.

General Syntax and Working of If-Then-Else

This code segment shown below will work this way, depending on whether the condition following 'if' is true or false:

- ***Condition is true:*** All of the statements following 'then' will be executed, until VB comes across the keyword 'else'. Once 'else' is reached, VB will 'jump over' all of the statements between 'else' and 'end if' and come out of the 'if - then - else' block. It will then continue the normal program execution after the 'end if' statement.
- ***Condition is false:*** None of the statements following 'then' will be executed. However, all of the statements following 'else' will be executed, until the time VB comes across the keywords 'end if'. After 'end if' is read, the 'if – then – else' block will terminate. VB will then continue the normal execution of the program after the 'end if' statement.

  The general syntax of if-then-else statement is as follows:

  If condition Then

  .....

  ..... .

  'execute all of these statements

  'starting from "then"

  'ending at "else"

  'if the condition is true

  .....

  Else ----------

  .....

  .....

  'execute all of these statements

'starting from "else"

'ending at "end if"

'if the condition is false

.....

End If

11.7.3 Multiple (Nested) If - Then Statements

The code segment we have used in discussing if - then - else works fine if one of the numbers is greater than the other one. But what if both the numbers are equal? Suppose we gave the value of

int_a = 10

and

int_b = 10

what will be the output of Code List if.4? The result will be ..... you guessed it right, 10! The Code List if.4 checks for [int_a > int_b] , i.e. 10 > 10. Since this condition is not true, i.e. false, the statements following 'else' will be executed, which print the value of int_b which, in the present case, is 10. As is obvious, if..then..else is most effective when there are only two ways to go. In the Code Listing if.4, the possible outcomes are -

int_a > int_b or int_a < int_b or int_a = int_b

As we can clearly see now, a simple if..then..else will be ineffective in the case just mentioned above, since, this situation has three possible outcomes. We therefore need three blocks to effectively deal with the three different outcomes. Hence, we need another block in addition to the 'then' and 'else' block that 'if..then..else' offers. We can get this option by adding another if..then block 'inside' the existing if..then..else block., like this:

Code Listing if.5

```
IF  int_a > int_b THEN
text1.TEXT = int_a       'printed when a > b --- 1
ELSE
IF  int_a < int_b THEN   '--- 2
text1.TEXT = B     'printed when a < b --- 3
ELSE
text1.TEXT = "both are equal" '--- 4
END IF
END IF
```

Observe from Code Listing if.5 that for every if, a corresponding end if has to be used. The 'inner' if..then is represented by block 'I', the 'outer' if..then by 'O'. The code runs as follows:

- This code will first check for [int_a > int_b]. If the condition is true, the program goes to line number '1', prints the value of 'int_a' in Text1, and skips over all of the other lines. It then comes out of the if-then blocks.
- If int_a > int_b is false, the program goes to line '2' and checks for [int_a < int_b]. If this condition is true, the program goes to line '3' and prints the value of 'int_b' in Text1. It then comes out of the if-then blocks.
- However, if this condition [int_a < int_b] is also false, then it is obvious that both the numbers are equal. Hence, the program switches to line number 4 and prints the message 'both are equal' in Text1.

It is important to realize the fact that the inner block 'I' must be completely inside the outer block 'O'. If this is not done, we will get wrong results.

General Syntax of Multiple If-Then-Else

The general syntax for multiple if-then-else statements is as follows:-

```
IF      condition1  THEN
        ......
        ......
        'execute all of these statements, ending at  "else"
......
ELSE
IF condition2  THEN
'execute all of these lines, ending at  "else"
        'if the condition2 is true
        ......
ELSE
        ......
        'execute all of these statements
        'ending at  "end if"
        'if the condition2 is false
END  IF  'inner  if-then-else block ends
END  IF         'outer if-then-else block ends
```

As explained above, the inner block must be completely contained by the outer block. In well-designed nested if-then blocks, if we draw lines showing the boundaries of the various blocks, it would look like the left-side figure below:

Name of Establishment and Post	Name of the Incumbent	Duty pay (Personal pay or Special Pay if any should be shown in this column as a separate entry below pay)	EARNINGS										
			Dearness Pay / Gr. Pay Leave Salary	Dearness Allowance	House Rent Allowance	Medical Allowance	Other Compensatory Allowance	Fixed Travelling Allowance		Gross Amount	Festival Advance	SPF / FBF	Subscription
(1)	(2)	(3)	(4) Rs.	(5) Rs.	(6) Rs.	(7) Rs.	(8) Rs.	(9) Rs.	(10) Rs.	(11) Rs.	(12) Rs.	(13) Rs.	(14) Rs.

G.P.F. Contribution			DEDUCATION												
	Recovery of advance and instalments P.L.I.Premium	G.P.F./P.L.I. Account C.P.F. Number	House Rent	House Building Advance	H.B.A./F.B.F.	Other deductions M.C.A./O.C.A.	Interest on MCA/OCA	Marriage Advance	Interest on M.A.	Income Tax	Khadi/Handloom Advance	Other deductions / HF (Specify deduction)	Total Deductions	Net amount payable in month	Acquittance
4) s.	(15)(16) Rs.	(17) Rs.	(18) Rs.	(19) Rs.	(20) Rs.	(21) Rs.	(22) Rs.	(23) Rs.	(24) Rs.	(25) Rs.	(26) Rs.	(27) Rs.	(28) Rs.	(29) Rs.	(30) Rs.

T.N.T.C. FORM 47

(See Subsidiary Rule 7 under Treasury Rule 16)

Token No :
Date :

Payable at .....

DETAILED PAY BILL OF PERMANENT / TEMPORARY ESTABLISHMENT OF THE .....

FOR THE MONTH OF .....

VOUCHER No.
MONTH / YEAR
SUB - ACCOUNT No.

SUB - TREASURY
TREASURY / PAO / | | | |

Drawing Officer

D.O.Code | | | | | | |

(Space for Treasury Officer's Enfacement)

D.P.Code

HEAD OF SERVICE
MAJOR HEAD
MINOR HEAD
SUB - HEAD

		DEBIT
Pay	01	1
D.P	-	-
DA	03	0
HRA	-	-
MA	01	3
OA	01	4
FTA	-	-
Grand Total		

only)

1) The total pay of a Government servent Officiating in a certain scales should be drawn for the period that he officiates in that section.

2) The following abbreviations should be used in this bill and in all other documents submitted with pay bills.

Earned Leave	EL
Un-earned leave on private affairs	UEL (P)
Un-earned leave on medical certificate	UEL (M)
Extraordinary leave with medical certificate	EOL (MC)
Extraordinary leave without medical certificate	EOL
On Other duty	OD
Leave Salary	LS
Subsistance Allowance	Sub Allow
Transit Pay	TP
Conveyance Allowance	CA

	CREDIT	
FA	77	-
FBF	01	-
SPF	02	-
HBA/FBF	03	-
GPF	04	-
TPF	05	-
PLI	06	-
IT	07	-
HR. PWD/HB	08	-
HBA/HDFC	09	-
HBA	10	-
MCA	11	-
OCA	12	-

(Under Rupees .....

Conveyance Allowance	CA
Fixed Travelling Allowance	FTA
Under Suspension	SP
On foreign Service	FS
Vacant	VAC
Last Pay certificate	LPC
General Provident Fund	GPF
Postal Life Insurance	PLI
House Rent	HR
House Building Advance	HBA
Other Recoveries	OR
Khadi Advance	KA
Recovery of Over Payment	ROP
Dearness Allowance	DA
Other Compensatory Allowance	OCA
City Compensatory Allowance	CCA
Handloom Advance	HLA

ABSTRACT

Total Column (12) .....

Deduct - Total General Provident Fund column (14 & 15) .....

Deduct Other Funds .....

Deduction - Column .....

Deduct-Income Tax Column (23 and 24) .....

Deduct - In adjustment of Advances recoverable as detailed in the attached statement column.

Deduct - Recoveries on account of House Rent , as detailed in the attached statement column (17)

Total - Deductions

Net sum required for Payment .....
(Rupees in words)

OCA	12	-
HLA	13	-
KA	14	-
MA	15	-
TANSI	16	-
WCA	17	-
TB	18	-
H.EDN	19	-
INT.HBA	20	-
INT.MCA	21	-
INT.MA	22	-
INT.OCA	23	-
RCA	24	-
OR	25	-
HF	30	-

Total - Deductions : _____ _____

Total - Net Amount : _____

PAY ORDER

PAY RUPEES (In FIGURES) .....

(IN WORDS) .....

.....

.....

.....

IN CASH AND RUPEES .....

(IN WORDS) .....

.....

.....

BY TRANSFER

Asst. / Pay and Accounts Officer

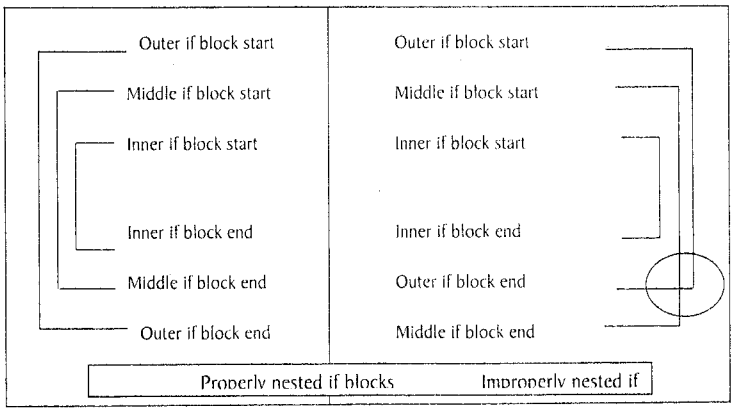
Sub. Asst. Treasury Officer

Details of pay of Absentees Refunded				
Serial Number	DESIGNATION OF THE POST	Name of the Incumbent	Period	Amount Rs. P.
(1)	(2)	(3)	(4)	(5)

1. Contents Received
2. Certified that the claim made by me is in accordance with Government rules and orders pertinent hereto and that all the facts stated in this claim are true to the best of my knowledge and belief. I am awared that I am fully responsible for any error in respect of facts, rules or Government orders as applicable to this bill.

Signature

Designation HEAD OF OFFICE



As seen in the figure above, all of the block lines are complete and none of the block lines are intersecting one another. However, in the figure on the right, the outer and middle block lines are intersecting towards the bottom-right of the figure. In this situation, we get run-time errors that are difficult to handle.

11.7.4 If - Then - Elseif Statement

If - then performs fine, but as the levels of nesting increase, the number of *if* keywords and the associated *end if* keywords also increases. While the increase in if..*end if* keywords is not a problem for the computer, it is a bit tough for us as programmers. We have to keep a close rack of the number of times we have used *if* so that we close with an equal number of *end if* statements. VB offers a useful feature to overcome this problem. Instead of adopting this approach where we have three *'end - if'* commands (since we have three *'if'* commands too), we can combine the two statements of block 'a' and also of block 'b' into a single statement, like this

```
If condition1 Then
...
Elseif condition2 Then        'Else and If of block "a" combined
...
Elseif condition3 Then        'Else and If of block "b" combined
...
Else
...
End If
```

The advantages of this approach are obvious. It makes our code look neater. We can more easily follow the flow of logic in the program. At the same time, we have the benefit of typing 'end..if' only once, even though we have checked for multiple conditions in the code. This saves time spent on typing.

Having come this far, we will now modify our earlier Code Listing if.5 as follows, using the Elseif keyword:

Code Listing if.6

```
IF int_a > int_b THEN    '-  -  -  -  I
text1.TEXT = int_a       'printed when int_a > int_b
```

```
ELSEIF int_a < int_b THEN      '- - - - II
text1.TEXT = int_b       'printed when int_a < int_b
ELSE
text1.TEXT = "BOTH ARE EQUAL" 'printed when both numbers are equal
END IF
```

You can see in the code that we have used 'If' twice (on lines marked as I and II), but 'End If' has been used only once.

Let us make another piece of code. We will accept the name of a colour in a textbox (Txt_col) and then click on a command button (Cmd_col) to change the background colour of the VB form as follows:

- If the user enters 'green', we change the colour to green.
- If the user enters 'red' we change the colour to red.
- If there is any other value, we change the colour to blue.

Code Listing if.7

```
PRIVATE SUB  cmd_col_CLICK( )
IF  txt_col.TEXT  =  "green"  THEN
form1.BACKCOLOR  =  VBGREEN
ELSEIF   txt_col.TEXT  =  "red" THEN
form1.BACKCOLOR  =  VBRED
ELSE
form1.BACKCOLOR  =  VBBLUE
END IF
END SUB
```

11.7.5 Select..Case Statement

Apart from If..then statements, another programming construct performs the task of conditional branching as good - maybe even better. This is the select..case statement. We'll now re-make the Code listing if.3 using the same logic, but with select..case. We will write the code as follows:

Code Listing select.1

```
2. CASE  =  "Shreyas"      'IF text1 IS "Shreyas"
3. MSGBOX  " Hello " &  "Shreyas"
4. CASE ELSE       'IF text1 IS NOT "Shreyas"
5. MSGBOX  " Hello  Mr/Miss No Name"
6. END SELECT
```

This code starts by reading the value of text1.Text, as per the line no.1. It then checks the value of text1, matching it with "Shreyas", as per line no. 2. The program will now take two different routes, depending on whether [Text1 = "Shreyas"] or not:

(a) If the value matches "Shreyas", line no.3 is executed, displaying a message box with the given message. The program then skips over line nos. 4 and 5, and comes to line no. 6. This line terminates the select..case block. Thus, line no. 3 acts as the then option of if..then..else.

(b) If the value does NOT match "Shreyas", line no.3 is skipped and program continues at line no. 4. This line acts as the else of if..then..else. Line no. 5 tells what to do when the else part - case else - is to be executed.

Select..case, thus, offers a choice of multiple branching, depending on which condition happens to be true. The condition is checked whenever the 'Case = ..' statement is encountered. If the condition is found to be true, the program executes all of the statements following the current 'Case = ..' upto the next 'Case' statement.

General Syntax of Select-Case

A general syntax for the Select - Case statement, thus, is as follows:

```
Select Case variable name/control name
Case = value1  'check for condition1
....
....                execute these statements
....        if condition1 is true
....
Case = value2  'check for condition2
....
....        execute these statements
....         if condition2 is true
....
Case  =  value3        'check for condition3
....
....      execute these statements
....      if condition3 is true
....
Case Else
....
....      execute these statements when
....      all other conditions are false
....
End  Select
```

Select..Case is the same as 'Switch' in 'C++'. Unlike C++, VB has no 'break' keyword. There is no 'default' in VB: 'Case Else' does same job.

We will now use Select..Case to modify our Code Listing if.7 we have made earlier.

Code Listing select.2

```
1.  SELECT CASE var1      ' read the value of the variable var1
2.  CASE = "green" '  if var1 is green
3.  form1.BACKCOLOR  =  VBGREEN
4.  MSGBOX  "Green Colour"
5.  CASE = "red"   '  if var1 is red
6.  form1.BACKCOLOR = VBRED
7.  MSGBOX  "Red Colour"
8.  CASE ELSE      ' if var1 is neither green nor red
9.  form1.BACKCOLOR = VBBLUE
10. MSGBOX  "DEFAULT COLOUR Blue"
11. END SELECT
```

This code selects the value of a variable 'var1' at line 1 [Select Case var1] and then checks it at the following line numbers:

- line 2 [Case = "green"]
- line 5 [Case = "red"] and
- line 8 [Case Else]

The program then executes one of the three paths depending on the value of var1:

(a) 'Var1' is "green": This value becomes true at line no. 2 resulting in execution of line nos. 3 and 4 only, i.e. all lines between [Case = "green"] and [Case = "red"]. The form's backcolor changes to green. An appropriate message box is also displayed. The program then skips over all lines from line no. 5 up to line no. 10. Line no. 11 ends the Select..Case block.

(b) 'Var1' is "red": This value becomes true at line no. 5 resulting in execution of line nos. 6 and 7 only, i.e., all lines between [Case = "red"] and [Case Else]. The form's backcolor changes to red. An appropriate message box is also displayed. The program then skips over all lines from line no. 8 up to line no. 10. Line no. 11 ends the Select..Case block.

(c) 'Var1' is NOT "red" or "green": This value becomes true at line no. 8 resulting in execution of line nos. 9 and 10 only, i.e., all the lines between [Case Else] and [End Select]. The form's backcolor changes to blue. An appropriate message box is also displayed. The program then encounters Line no. 11, which ends the Select..Case block.

Compared to nested if..then code, a select..case code looks neat. Also, select..case runs faster than multiple if..then..else statements.

11.7.6 Nested Select-case

Similar to the concept of nesting we have seen in if..then..else, Select..Case can also be nested. An example will make this clear to us. Suppose we have a program which accepts a student's category and status. The program checks the category and then mentions the government grant for the students based on the following criteria:

- General Category - No grant
- OBC category - ₹ 1,500
- SC/ST category - ₹ 2,500 if status = 'bpl'
- SC/ST category - ₹ 2,000 if status = 'ric'

Code Listing select.3

```
1.  SELECT CASE  txt_categ.TEXT
2.  CASE IS = "GEN"
3.  MSGBOX "Sorry, no grant for you."      'if the code is gen
4.  CASE IS = "OBC"
5.  MSGBOX "Your government grant is Rs. 1500."  'if the code is obc
6.  CASE IS = "SC", "ST" 'if case is either sc or st
7.  SELECT CASE  txt_status.TEXT
8.  CASE IS = "bpl"
9.  MSGBOX "Your grant is Rs. 2500."       'if the values are sc/stand bpl
10. CASE IS  =  "ric"
11. MSGBOX  " Your grant is Rs. 2000."     'if the values are sc/st and ric
12. END SELECT
13. CASE ELSE
14. MSGBOX  "Sorry, the category code is invalid"
15. END SELECT
```

Here, the code runs as follows:

1. The line nos. from 7 to 12 mark the 'nested block'.
2. Line no. 1 selects the text in Txt_categ text box.
3. Line no. 2 checks if the text is 'gen'. If true, line no. 3 is executed, and the remaining lines are all skipped.
4. Line no. 4 checks if the text is 'obc'. If true, line no. 5 is executed, and the remaining lines are all skipped.
5. The line no. 6 checks if the text is 'sc' or 'st'. If true, the 'nested' Select-Case block is started.
   (a) Line no. 7 selects the text in Txt_status textbox.
   (b) Line no. 8 checks if 'text' in Txt_status is 'bpl'. If true, line no. 9 is executed. Remaining lines of inner and outer block are skipped.
   (c) Line no. 10 checks if 'text' in Txt_status is 'ric'. If true, line no. 11 is executed, and the remaining lines are all skipped.
   (d) Line no. 12 terminates the inner select...case block.

6. Line no. 13 is reached if the user has entered a wrong category, i.e., the condition at line no. 2, 4, and 6 is 'false'.
7. Line no. 14 prints a message for wrong category entry.
8. Line no. 15 terminates the outer select...case block.

To save time, just type the line [Case = 'po'] or [Case ='sc'] etc., and press enter VB will automatically insert the [Is] keyword.

That is, we just type

Case = 'po'

and press enter. VB will automatically convert this line to

Case Is = 'po'

This way, we will type less and avoid any possible spelling errors.

11.7.7 Select-case With Multiple Options (OR)

We will see another aspect of the 'Select-Case' statement, if we observe the line no. 6 above, i.e.

CASE IS = 'sc','st'

this line checks for two possible values of Txt_Categ, 'sc' or 'st'. This line is just like saying

IF txt_categ.TEXT = 'sc' or txt_categ.TEXT = 'st' THEN

Thus, if we have to check for one out of two or more values, we just type (Case Is =) and then put all the values one after the other, separated by a comma. For example, the code list given below checks for city names given in a text box (Txt_city) and then prints the state to which the city belongs.

Code List select.4

```
1.  SELECT CASE  txt_city.TEXT
2.  CASE IS  =  "BHOPAL" , "ujjain", "gwalior"
3.  MSGBOX  "M.P. STATE"
4.  CASE IS  =  "MUMBAI" , "thane" , "nasik"
5.  MSGBOX  "MAHARASHTRA STATE"
6.  CASE IS  =  "LUCKNOW" , "ALLAHABAD" , "MEERUT"
7.  MSGBOX  "U.P. STATE"
8.  CASE ELSE
9.  MSGBOX  "The state is NOT known"
10. END SELECT
```

Here, the code runs as follows:

1. Line no. 1 selects the text in txt_city textbox.
2. Line no. 2 checks text of txt_city. If it is either 'bhopal' or 'ujjain' or 'gwalior', line no. 3 is executed. The other lines are by-passed.

3. Line no. 4 checks the text of txt_city. If the value is 'mumbai' or 'thane' or 'nasik', line no. 5 is executed, by-passing all other lines of code.
4. Line no. 6 checks text of txt_city. If it is 'lucknow' or 'allahabad' or 'meerut', line no. 7 is executed. Remaining code lines are by-passed.
5. Line no. 8 is reached only if the line nos. 2, 4, and 6 don't match the value of txt_city, and then line no. 9 is executed.
6. Line no. 9 prints an error message for the user.
7. Line no. 10, of course, terminates the Select..Case block.

11.7.8 Select-case with Multiple Options (AND)

We will now see another aspect of the 'Select-Case' statement. Consider the situation where we are preparing a marksheet and we need to calculate the division. As is expected, the criteria will be:

- 80% upto 100% - honours
- 60% upto 79.9% - 1st division
- 45% upto 59.9% - 2nd division
- 33% upto 44.9% - 3rd division
- any other value - fail

The code for the problem will be as follows:

Code List select.5

```
1. SELECT CASE  CINT(txt_perc.TEXT) 'convert into nearest integer value
2. CASE 80 to 100 'if perc is between 80 and 100
3. MSGBOX  "HONOURS DIVISION"
4. CASE 60 TO 79  'if perc is between 60 and 79
5. MSGBOX  "1ST DIVISION"
6. CASE 45 TO 59  'if perc is between 45 and 59
7. MSGBOX  "2ND DIVISION"
8. CASE 33 TO 44  'if perc is between 33 and 44
9. MSGBOX "3RD DIVISION"
10. CASE ELSE 'if perc is not in any range mentioned above
11. MSGBOX  "FAIL"
12. END SELECT
```

Here, the code runs as follows:

1. Line no. 1 selects the text in txt_perc textbox and converts it to the nearest integer value (e.g. 79.6 to 80, 79.4 to 79, and so on.).
2. Line no. 2 checks text of txt_perc. If between 80 and 100, line no. 3 is executed. The other lines are by-passed.

3. Line no. 4 checks text of txt_perc. If between 60 and 79, line no. 5 is executed, by-passing all other lines of code.
4. Line no. 6 checks text of txt_perc. If between 45 and 59, line no. 7 is executed. Remaining code lines are by-passed.
5. Line no. 8 checks text of txt_perc. If between 33 and 44, line no. 9 is executed. Other lines are by-passed.
6. Line no. 10 is reached if all other criteria of 'percentage' are not fulfilled.
7. Line no. 11 prints a messagebox with the message 'Fail'.
8. Line no. 12 terminates the Select..Case block.

This is a situation similar to the "and" in "if..then..else", i.e,

- If the percentage is between 80 and 100, then give the message "honours".
- If the percentage is between 60 and 79, then give the message "1st division", and so on.

The "to" option in select..case should be in ascending order (80 to 100, 60 to 79). It won't work in descending order (100 to 80, 79 to 60).

Check Your Progress

1. Define variables.
2. Define Creating and initializing Date.

11.8 LET US SUM UP

When a user requests a website by typing URL in address bar of browser, the request is accept by the web server which hosts that site. Web Server executes some code to fulfill client request and generate output. The code which is executed at server end is called server-side script and computer languages used to write this script are called server-side scripting languages. Examples of server-side scripting languages are CGI, ASP and PHP. Client-side script is used for client-side validation and other operations like drop down menus, moving fonts If we write the code in client-side scripting language this will save time and traffic load between server and client. Examples of client-side scripting language are JavaScript and VB Script.

A variable must be declared with a name and a specific data type. Since VBScript supports only one data type, Variant, all variables will be declared as Variant by default. Applications need to be able to access and work with date and time values. Many applications are written for business and scientific purposes, where recording the date and time of the program run is vital to the success of the project.. Arithmetic operators work on numeric operands and produce numeric result. Just like other programming languages, VBScript provides a rich library of built-in functions to deal with including strings, dates, typecasting, basic math, etc. When you consider creating functions in any programming language, it is useful to familiarize yourself with the available built-in functions of that language.

11.9 KEYWORDS

Serversidescript: The code which is executed at server end is called server-side script.

Clientsidescript: The code which is executed at client end is called client-side script.

CSS: CSS is a language that helps you separate content and formatting.

Javascript: A script language that helps you make your pages more 'dynamic'.

Date(): Returns the current system date.

DateAdd(sType, iInterval, tDate): Returns a date to which a specified time interval has been added.

DateDiff(tDate1, tDate2): Returns the number of intervals between two dates.

DatePart(sType, tDate): Returns the specified part of a given date.

DateSerial(iYear iMonth,iDate): Returns the date for a specified year, month, and day.

DateValue(vVariant): Returns a date.

Day (tDate): Returns a number that represents the day of the month (between 1 and 31, inclusive).

FormatDateTime(tDate, iType): Returns an expression formatted as a date or time.

Hour(tDate): Returns a number that represents the hour of the day (between 0 and 23, inclusive).

IsDate(vVariant): Returns a Boolean value that indicates if the evaluated expression can be converted to a date.

Minute(tDate): Returns a number that represents the minute of the hour (between 0 and 59, inclusive).

Month(tDate): Returns a number that represents the month of the year (between 1 and 12, inclusive).

MonthName (iMonth): Returns the name of a specified month.

Now(): Returns the current system date and time.

Time(): Returns the current system time.

Timer(): Returns the number of seconds since 12:00 AM.

TimeSerial(iHour,iMinute,iSecond): Returns the time for a specific hour, minute, and second.

TimeValue (tDate): Returns a time.

Weekday(tDate): Returns a number that represents the day of the week (between 1 and 7, inclusive).

WeekdayName (iWeekDay): Returns the weekday name of a specified day of the week.

Year(tDate): Returns a number that represents the year.

Abs (number): It returns the absolute value of a number. In other words, if the number is greater than or equal to zero, the number is returned. If, however, the number is less than zero, the negation of the number is returned. For instance, Abs (5) would return 5 and Abs (-1) would return 1.

Exp (number): It raises e (approximately 2.71828) to the power number. For example, Exp (5) says e^5, which is approximately 148.41315.

Fix (number): It returns integer portion, greater than or equal to the number. For instance, Fix (-5.80) returns -5.

Int (number): It is different from Fix(number) because Int (number) returns integer portion less than or equal to number. For example, Int (-5.80) returns -6.

Hex (number): It returns strings consisting of the number converted to hexadecimal base. For example, Hex (10) returns A, Hex (11) returns B, Hex (12) returns C, etc.

Oct (number): It returns strings consisting of the number converted to octal base. For instance Oct (8) returns 10, Oct (9) returns 11, Oct (10) returns 12, etc.

Round (number): It rounds a number to the nearest integer; for example, Round (5.45) rounds 5.45 to 6 and Round (5.01) returns 5.

Round (number, dec): It rounds a number to dec decimal places. Round (5.45, 1) rounds 5.45 to 1 decimal place: 5.4. Round (100.67899, 3) returns 100.679.

Sgn (number): It returns1 if number is less than 0, 0 if number is 0, and 1 if number is greater than 0. Consider the following as examples: Sgn(-5.9) returns -1, Sgn(0) returns 0; and Sgn(5) returns 1.

Sqr(number)(number > 0): It provides a square root of number, provided the number is positive. For example, sqr (4) returns 2, sqr (100) returns 10.

11.10 QUESTIONS FOR DISCUSSION

1. Differentiate between client-side languages and server-side languages.
2. Discuss the declaration of variables. Differentiate between explicit variables and implicit variables.
3. Discuss how date and time works in scripts.
4. Discuss adding date and time in scripts.
5. Discuss the mathematical functions available in vbscript.
6. Differentiate between If-then and If-then-else statements.

Check Your Progress: Model Answers

1. VBScript uses variables to reserve memory to store data and to name that memory location.
2. A Visual Basic Date can be created in the same way any other variable is created. For example the following Visual Basic code excerpt creates a Date variable:

   Dim dteAppointment As Date

   The Date value is initialized using a date string in the from mm/dd/yyyy encapsulated in hash (#) characters. For example to set the date to August 2, 2007:

   Dim dteAppointment As Date = #8/2/2007#

11.11 SUGGESTED READINGS

Pratt, *Programming Languages*, Pearson Education India

K. L. James, *The Internet: A User's Guide*, PHI Learning Pvt. Ltd

Steven M. Schafer, *HTML, XHTML, and CSS Bible*, John Wiley and Sons,

UNIT V

LESSON

12

PROCEDURES, LOGICAL OPERATORS AND LOOPS

CONTENTS

12.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss creating functions and string functions
- Discuss creating sub-routines
- Understand logical connectives and operators
- Discuss using loops

12.1 INTRODUCTION

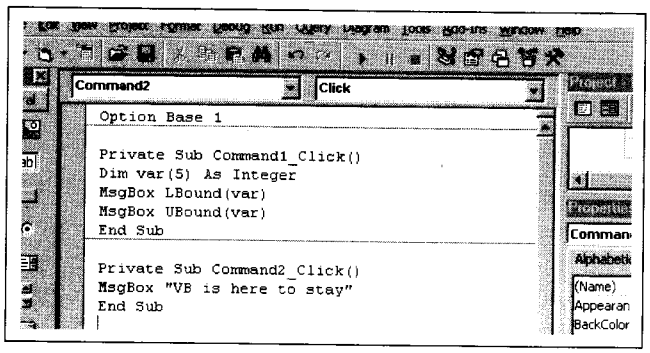
As we have been observing for some time now, a project in VB is not a 'monolithic', i.e., single-stone structure. It is made up of a number of small code segments that are linked in some way. For example, the code for the click event of a Command Button command1 is written within the stub:

```
PRIVATE SUB command1_CLICK()
    ...
END SUB
```

As is very obvious, we might be having another command button 'command2' and the code for its click event will be written within

```
PRIVATE SUB command2_CLICK()
    ...
END SUB
```

Here is a sample picture of the code window of a VB project.



We can clearly see that we have code written for the click event of different command buttons. The different code segments are self-contained and are separated from one another by the lines running across the code window. These code segments perform a particular task. These are what we call procedures. Thus, a procedure is a code segment that performs a particular task and is self-contained.

A procedure offers the advantage of better management of code. For example, a code might be required at multiple locations in a project. It can be pasted at multiple locations, and the output will be the same. Now, any change in the code means all the copies have to be updated. A procedure will help avoid this repetitive work. We can write the code for the procedure at only one place and we can call on this procedure through code. Thus, any changes to be made will be limited to just one location.

The kind of procedures we have used so far are known as 'event procedures'. This is because they are procedures that are related to an event, and will be executed when a certain event is raised. Thus, we primarily code through event procedures in VB.

In addition to event procedures, VB offers the following types of procedures:

- Sub-procedures
- Functions
- Property

12.2 SUB-ROUTINES

A sub-routine is a procedure that performs a task but does not return a value. The general syntax of a sub-procedure is as follows:

PUBLIC/PRIVATE procedure name ( )

END SUB

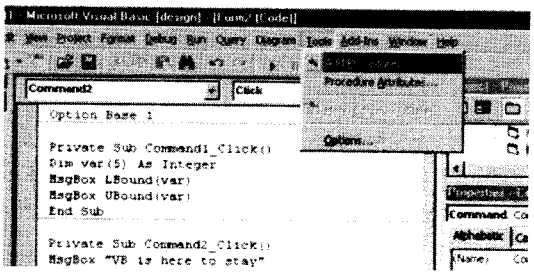
An alternative method for the syntax involves the use of parameters. The parameters are an optional, though a very powerful feature of procedures. Multiple parameters are separated by commas. The syntax is shown here:

PUBLIC/PRIVATE procedure name (parameter list)

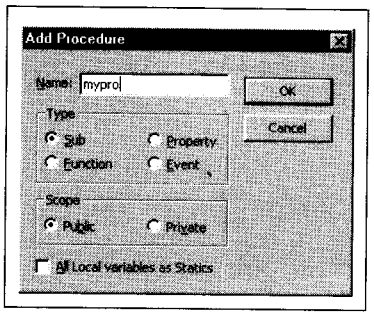
END SUB

VB offers a number of in-built procedures and also allows us to create our own procedures. This is how we create our own procedure in VB:

1. Switch to the code window.
2. Click on the tools menu.
3. From the sub-menus, click on add procedure, as shown in the picture.



4. From the dialog box that appears now, type a name for the procedure in the textbox appearing at the top (we have typed 'mypro').The dialog box is shown in the figure that follows:

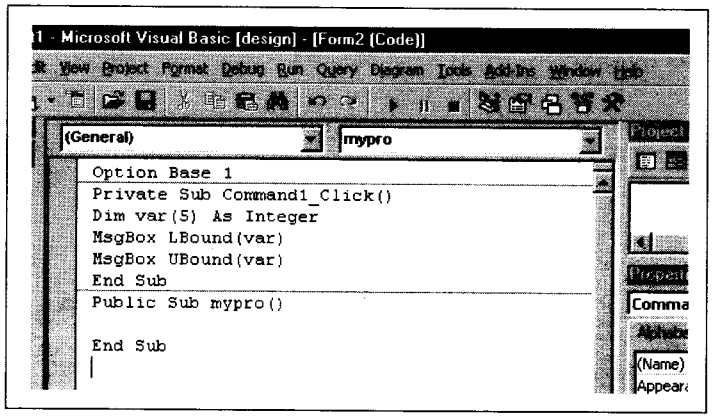


5. Click on the 'Sub' option (it is the default selection, anyway).
6. Click on the 'OK' button. The dialog box will disappear and the code window will be fully visible once again.
7. Observe the appearance of our own procedure's stub in the code window ('mypro') as

   PUBLIC SUB mypro()

   END SUB

   The code window has been shown in the picture that follows



8. We can now type some code into our own sub-procedure. Our procedure is ready to be used.

   Suppose we have typed the following code for our sub procedure.

Code Listing sub.1

```
PUBLIC SUB mypro()
PRINT "HELLO everyone !! I have come from the mypro procedure"
END SUB
```

This procedure can now be called from anywhere in our Form. To see it in action, draw two command buttons on the form (with default names of command1 and command2). Now come to the code window and code for the event procedure of both Command Buttons, as follows:

Code Listing sub.2

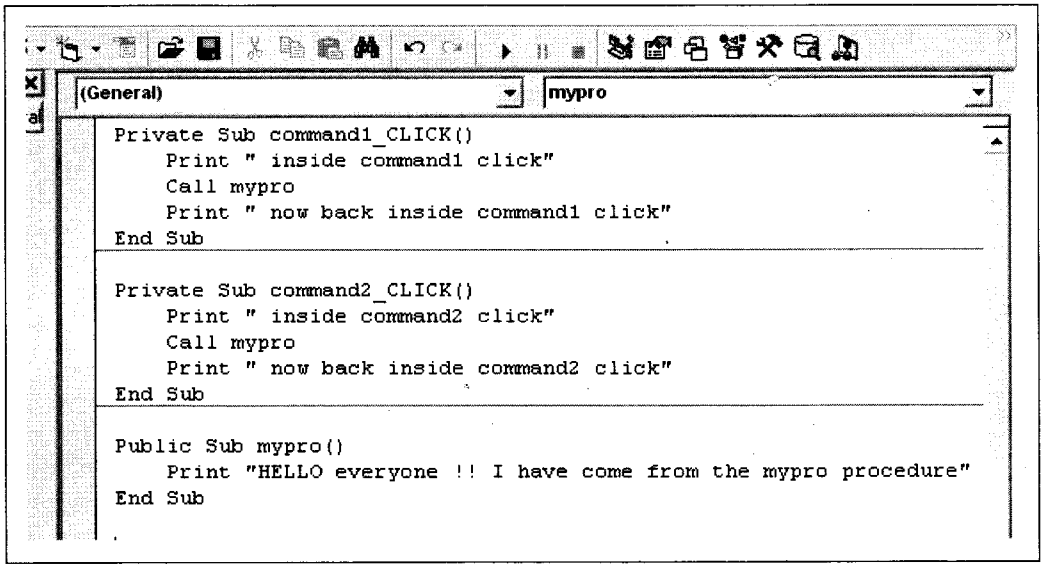
```
PRIVATE  SUB  command1_CLICK( )
PRINT  " inside command1 click"
CALL  mypro
PRINT  " now back inside command1 click"
END  SUB
```

Code Listing sub.3

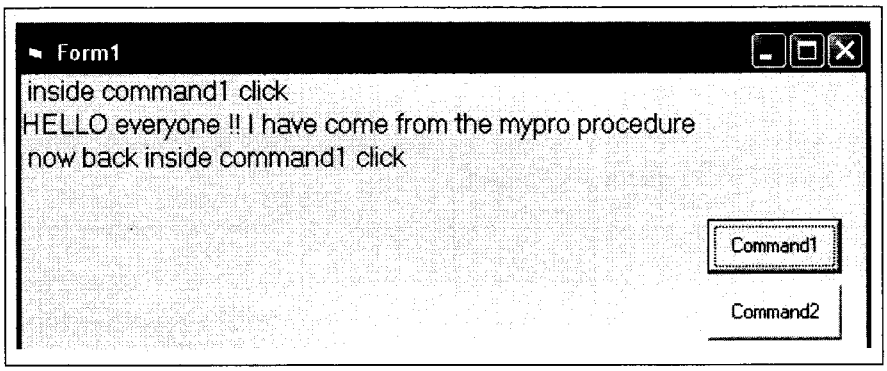
```
PRIVATE  SUB  command2_CLICK( )
PRINT  " inside command2 click"
CALL  mypro
```

```
PRINT  " now back inside command2 click"
END  SUB
```

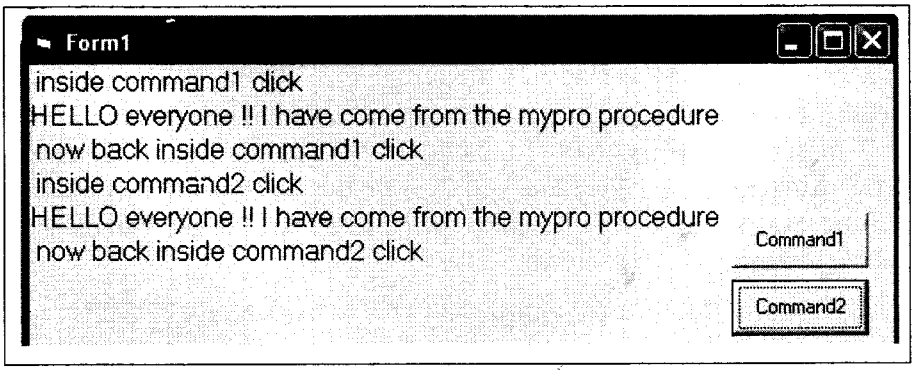
Once we have coded everything, our code window might appear like this:



When we run the program and click on the command button 'command1', we see an output as follows:



A similar output will be displayed when we click next on the command button command2. Note that command1 in the output message has been replaced with command2. Everything else is the same.



The line no. 2 of code listing sub.2 is 'calling' the sub-procedure mypro. Thus, the program control 'shifts to' the procedure mypro and the procedure is executed from the 1st line to the last. Once the procedure is completely executed, the program control shifts back to the calling sub.

Now, if a change is required in the procedure mypro, it will be confined to the procedure itself. The code that is calling mypro, need not necessarily be changed. In fact, if the method of calling the procedure does not change, the calling block will never know that the internal working of the procedure has changed. For example, if we change the declaration of mypro from

```
PUBLIC SUB mypro()
```

To

```
PUBLIC SUB mypro (int_par1 AS INTEGER)
```

We will have to change the way of calling mypro from CALL mypro

to

```
CALL mypro (int_var1)
```

where int_var1 refers to an integer variable. Since we have changed the declaration of the procedure from no-parameter mode to a single-parameter mode, we have to change the way of calling it. Anytime we change the number of parameters or the type of parameters in a procedure declaration, we have to change the style of calling on the procedure.

If we don't make the change in the mode of calling from the upper statement to the lower statement, we will get an error of 'type mismatch' at runtime.

A procedure call should match in 'quality and quantity'. The number and datatype of arguments must match the parameters in the procedure.

As long as the number of parameters or the type of parameters for calling the procedure do not change, the procedure call will remain unchanged.

12.3 FUNCTIONS

Functions are a type of procedure, which return a value. This is what separates a function from a sub-procedure. Otherwise, the function and sub-procedure are the same. When we want a self-contained code, executing a particular task and also 'returning a value' to the calling program, we use a function. If a value is not to be returned, we use a procedure.

The general syntax of a function is:

PUBLIC/PRIVATE function name (parameter list) AS DATATYPE

As the syntax shows, a function declaration is not much different from that of a procedure. The only difference is the presence of a return datatype, at the end of the declaration. Since a function has to return a value, the declaration of a function will include the datatype as the last two words of the function declaration. For example, if a function area accepts an integer parameter and returns an integer to the calling program, the syntax will be:

PUBLIC/PRIVATE FUNCTION area (int_par1 AS INTEGER) AS INTEGER

For a function join_name that accepts two strings as parameters and returns a string, the syntax could be:

PUBLIC/PRIVATE FUNCTION join_name (str_par1 AS STRING,str_par2

AS STRING) AS STRING

Once again, the return datatype of the function has been given as the last word.

Here is an example of a function. This function will accept the length of a side of a square and calculate the area of the square based on that number. The code will be as follows:

Code Listing function.1

```
PRIVATE FUNCTION square_area (int_p AS INTEGER) AS INTEGER
square_area = int_p ^ 2        'the ^ function (exponentiation) raises
a no. to any power.
END SUB
```

Here, the procedure has only one executable line of code. This line of the procedure does two things:

- Raises the parameter for the function int_p to the 2nd power through the use of the 'exponentiation' function [^].
- Assigns this value to the function name square_area. This value is then returned to the calling program.

Here is some code to use the function created above. For this, create a form with 2 textboxes text1 and text2 and a single Command Button command1. The textbox text1 will be used to type the value of the argument for the function square_area. Textbox text2 will display the result of the function call. Now come to the code window and type the following code:

Code Listing function.2

```
PRIVATE SUB command1_CLICK( )
1. DIM int_var1 AS INTEGER
2. DIM int_return AS INTEGER
3. int_var1 = CINT (text1.TEXT)
4. int_return = square_area (int_var1)
5. text2.TEXT = int_return
   END SUB
```

In the code given above, the function square_area has already been created under the code listing function.1. Analyzing the code further,

1. Line nos. 1 and 2 declare 2 integer variables by the name int_var1 and int_return, respectively. This int_var1 will be the input (argument) we give to the function square_area. The int_return is the return value we will get from the function square_area.
2. Line no. 3 converts the contents of text1 into an integer, through the use of the in-built 'cint' function [cint(text1.text)]. This value is stored in the variable int_var1.

3. Line no. 4 shifts the program control to the function square_area. It calls this function, sending it int_var1 as an argument. Inside square_area, the argument int_var1 becomes the parameter int_p. Once the function completes execution, the return value of the function is stored in the variable int_return.
4. Line no. 5 displays the value of int_return in text2. This is the value returned from the function square_area.

As we can see, a function call takes the program from one code segment to another. The program control shifts from the currently executing procedure or function to the function called by us through code. Once the called procedure or function is executed, the program control shifts to the line after the line which had called the code. To represent this clearly, consider our code listing function.1 and code listing function.2

Code Listing function.1

```
PRIVATE FUNCTION square_area (int_p AS INTEGER)AS INTEGER
square_area = int_p ^ 2
'the '^'function(exponentiation)raises a number to any power
END SUB
```

Code Listing function.2

```
PRIVATE SUB command1_CLICK( )
1. DIM int_var1 AS INTEGER
2. DIM int_return AS INTEGER
3. int_var1 = CINT (text1.TEXT)
4. int_return = square_area (int_var1)
5. 'program shifts from line no. 4 to code listing function.1
6. 'program resumes from line no. 7
7. text2.TEXT = int_return
   END SUB
```

In this code, line no. 4 of code listing function.2 shifts the program control to square_area by calling it. Once square_area is completely executed, the program control shifts back to line no. 7 of code listing function.2. This line is the first executable line of code listing function.2, after the function has been called.

This scheme of calling a function can be represented inside the computer as:

PRIVATE SUB command1_CLICK( )

```
1. DIM int_var1 AS INTEGER
2. DIM int_return AS INTEGER
3. int_var1 = CINT(text1.TEXT)
4. int_return = square_area (int_var1)
5. PRIVATE FUNCTION square_area (int_p AS INTEGER) AS INTEGER
6. square_area = int_p ^ 2
```

```
7.  'the '^' function(exponentiation) raises a number to any power
8.  END SUB
9.  text2.TEXT =  int_return
END SUB
```

Thus, the called function behaves as if it is a part of the normal execution of the calling block, though the two procedures are physically different.

The call to a procedure is a complete statement by itself. A function call is always a part of an expression.

Here is another function. This function accepts the first and last name and joins them to give the full name. The code is as follows:

Code Listing function.3

```
PRIVATE FUNCTION join_name(str_p1 AS STRING,str_p2 AS STRING)AS  STRING
join_name  =  str_p1  &  "  "  &  str_p2
END FUNCTION
```

To make this code work, make a form with three textboxes text1, text2 and text3 and a command button command1. The user will enter the first name and last name in text1 and text2, respectively. When the user clicks on command1, the function join_name will be called and result will be displayed in text3. The code is:

Code Listing function.4

PUBLIC SUB command1_CLICK()

```
1.  DIM str_fname AS STRING
2.  DIM str_lname AS STRING
3.  DIM str_fullname AS STRING
4.  str_fname = text1.TEXT
5.  str_lname = text2.TEXT
6.  str_fullname = join_name(str_fname, str_lname)
7.  text3.TEXT = str_fullname
END SUB
```

Now we run the program, type the first name in text1 and the last name in text2, and then click on command1. The following happens:

1. Line nos. 1 to 3 declare three string type of variables to hold our data and the results.
2. Line no. 4 assigns the contents of text1 to str_fname.
3. Line no. 5 assigns the contents of text2 to str_lname.
4. Line no. 6 now calls the function join_name with 2 arguments. The program control shifts to the first line of join_name. Once the function is fully executed, the program control shifts back to code listing function.4. The return value of join_name is stored in the variable str_fullname.
5. Line no. 7 displays the contents of str_fullname in text3. This text, as we know, shows the complete (joined) name typed by the user.

12.3.1 String Functions

As the name suggests, these functions work on string type of data. They help perform a number of useful activities that are of immense help in validating and manipulating our data.

Suppose we have a string variable named str_name with the data 'Mamta Puri', we can perform the following operations on it. Observe the syntax and the output given to exactly understand the working of these functions.

The following string functions return a numeric output:

- TRIM can be used to remove leading and trailing spaces from a string data before saving it into the database.
- TRIM can be used to remove leading and trailing spaces when accepting an input from the user for searching a record in the database.

Of these functions, srtreverse, replace and instr are new to VB 6. Instr returns a value of '0' if the string to be searched is not found in the given string.

12.4 USING LOGICAL CONNECTIVES AND OPERATORS

Logical operators work on logical type of operands to produce a logical type result (True or False). *And*, *Or* and *Not* are three logical operators.

12.4.1 And

Purpose: To perform a logical conjunction on two logical expressions.

Syntax: result = expression1 And expression2

The And operator syntax has these parts:

Part	Description
Result	A required boolean variable.
Expression1	A required boolean expression.
Expression2	A required boolean expression.

If both expressions evaluate to True, result is True. If either expression evaluates to False, result is False. The following table illustrates how *result* is determined:

expression1	expression2	Expression1 And Expression2
True	True	True
True	False	False
True	Null	Null
False	True	False
False	False	False
False	Null	False
Null	True	Null
Null	False	False
Null	Null	Null

The And operator also performs a bitwise comparison of identically positioned bits in two numeric expressions and sets the corresponding bit in *result* according to the following table:

bit in *expression1*	Bit *expression2*	And
0	0	0
0	1	0
1	0	0
1	1	1

12.4.2 Or

Purpose: To perform a logical disjunction on two expressions.

Syntax: result = expression1 **Or** expression2

The **Or** operator syntax has these parts:

Part	Description
Result	A required boolean variable.
Expression1	A required boolean expression.
Expression2	A required boolean expression.

If either or both expressions evaluate to True, result is True. The following table illustrates how *result* is determined:

expression1	expression2	expression1 Or expression2
True	True	True
True	False	True
True	Null	True
False	True	True
False	False	False
False	Null	Null
Null	True	True
Null	False	Null
Null	Null	Null

The Or operator also performs a bitwise comparison of identically positioned bits in two numeric expressions and sets the corresponding bit in *result* according to the following table:

bit in *expression1*	bit in *expression2*	Or
0	0	0
0	1	1
1	0	1
1	1	1

12.4.3 Not

Purpose: To perform logical negation on an expression.

Syntax: result = Not expression

The Not operator syntax has these parts:

Part	Description
Result	Required; any boolean variable.
Expression	Required; any boolean expression.

The following table illustrates how *result* is determined:

expression	NOT
True	False
False	True
Null	Null

In addition, the **Not** operator inverts the bit values of any variable and sets the corresponding bit in *result* according to the following table:

Bit in *expression*	NOT
0	1
1	0

12.4.4 XOR (Exclusive Or)

Purpose: To perform logical negation on an expressions.

Syntax: result =expression1 **Xor** expression2

The **Xor** operator syntax has these parts:

Part	Description
Result	An optional boolean variable.
Expression1	A Required boolean expression.
Expression2	A required boolean expression.

If one, and only one, of the expressions evaluates to **True**, *result* is **True**. However, if either expression is Null, *result* is also **Null**. When neither expression is **Null**, *result* is determined according to the following table:

expression1	expression2	Xor
True	True	False
True	False	True
False	True	True
False	False	False

The **Xor** operator performs both a logical and bitwise operations. A bit-wise comparison of two expressions using exclusive-or logic to form the result, as shown in the following table:

Bit in *expression1*	bit in *expression2*	Xor
0	0	0
0	1	1
1	0	1
1	1	0

12.5 LOOP STRUCTURE

Visual Basic provides loop structures to perform iterative operations. Repeating actions can be implemented in two ways

1. ***Determinate Loops:*** Repeats the operation a fixed number of time.
2. ***Indeterminate Loops:*** Continues until you reach a specific predetermined goal, or continue until certain initial conditions have changed.

At times, we might need to execute a group of statements repeatedly. That is, we want to execute them over and over again. To consider some simple examples:

- Preparing the results of an exam: we start from the first student and continue with the same set of rules upto the last student.
- Preparation of bills in a telephone company: we take the first customer and continue with the billing process upto the last customer in our list.
- Preparing a bill in a shop: we start processing the order from the first product to the last.
- On the popular TV game show KBC: questions are repeatedly asked till the time the participant opts out or gives a wrong answer.

As is obvious, we are repeating a group of statements, applying similar conditions and rules at every stage. This repeated execution of statements can be achieved through what we call as 'looping' or 'iterative' statements. We will now see all of these statements.

12.5.1 For-next Loop

The general syntax of the For-Next Loop is as follows:

```
FOR i = initial value  TO  final value   'line 1
        'All of  these statements
        'will be repeated as long
        'as the numeric value
        'of the variable 'i' is
        'between the initial value
        'and the final value
NEXT i          'line 8
```

where i can be any integer variable. In For-Next terminology, 'i' is known as an index variable. The code starts by initializing the value of i, as given in the line [For i = initial value to final value]. All of the statements upto line no. 8 (Next i) are then executed.

At line number 8, two important things happen:

1. The value of i is incremented by 1.
2. "Next i" then checks if the value of i falls in the given range on line1 (initial value and final value)

If the incremented value of i falls within the range, all the lines between 1 and 8 are executed once again. Once again, the program comes to line 8. Once again the process happening at line number 8 is repeated, as we have just discussed.

If the incremented value falls between the range, all of the lines between 1 and 8 are again executed. This process continues till the time the value of i is between the initial and the final value. That is, the lines between 1 and 8 are repeatedly executed. This is what we call as the 'loop'.

Once incremented value of 'i' does NOT fall in the given range, execution of lines between 1 and 8 is stopped. Thus, the For-Next loop is terminated. The program will then start processing statements written after "Next i" statement.

Let us say we want to print a sequence of numbers from 1 to 3. Our program will look like this:

Code Listing for.1

```
1. FOR i = 1 TO 3 'define initial and final values
2. PRINT i
3. NEXT i   'check if loop is to be repeated
4. MSGBOX  " For - Next loop has ended"
```

Here, the code runs as follows:

1. Line no. 1 starts the loop, assigning initial of 1 for i and a final value of 3.
2. Line no. 2 prints the value of i on the screen, i.e., 1.
3. Line no. 3 "Next i" increments the value of 'i' by 1: 'i' becomes 2.
4. The line "Next i" checks if 'i' falls in the range 1 to 3 as specified in line no. 1. The current value of 'i', i.e., 2, is between 1 and 3. Hence, line no. 2 is repeated.
5. Line no. 2 again prints the value of 'i' on the screen, i.e., 2.
6. Line no. 3 is reached again. "Next i" again increments 'i' by 1. So, 'i' now becomes 3. The current value is then checked at line no.1. Since the current value of i is between 1 and 3, line no. 2 is repeated.
7. Line no. 2 again prints the value of 'i' on the screen, i.e., 3.
8. Line no. 3 is reached again. "Next i" again increments 'i' by 1. So, 'i' now becomes 4. The current value of 'i' is 4, but 4 is NOT between 1 and 3. Thus, line no. 2 is not repeated. The For..Next loop is terminated, and further printing of the value of 'i' is stopped. Now the program comes to line 4.
9. Line no. 4 gives a message box indicating that the loop has terminated. The program stops.

Take another example. This time we will print a name on the screen four times. The code segment is as follows:

Code Listing for.2

```
1. FOR  i = 1 TO 4      'define initial and final values
2. PRINT  "Mamta Puri"
3. NEXT  i  'check if loop is to be repeated
4. MSGBOX  "Name printing task is over"
```

Here, the code runs as follows:

1. Line no. 1 starts the for loop. It assigns an initial of 1 for the index variable i and a final value of 4.
2. Line no. 2 prints 'Mamta Puri' on the screen (1st time).
3. Line no. 3 "Next i" increments the value of 'i' by 1: 'i' becomes 2. The line "Next i" then checks if 'i' falls in the range 1 to 4 as given in line no. 1. The current value of i, i.e., 2, is between 1 and 4. Hence, line no. 2 is repeated.
4. Line no. 2 again prints 'Mamta Puri' on the screen (2nd time).
5. Line no. 3 is reached again. "Next i" again increments 'i' by 1. So, 'i' now becomes 3. The current value is then checked at line no.1. Since the current value of 'i' is between 1 and 4, line no. 2 is repeated.
6. Line no. 2 again prints 'Mamta Puri' on the screen (3rd time).
7. Line no. 3 is reached again. "Next i" again increments 'i' by 1. So, 'i' now becomes 4. The current value of 'i' is 4 and 4 is between 1 and 4. Thus, line no. 2 is repeated.
8. Line no. 2 again prints 'Mamta Puri' on the screen (4th time).
9. Line no. 3 is reached again. "Next i" again increments 'i' by 1. So, 'i' now becomes 5. The current value of 'i' is 5, but 5 is NOT between 1 and 4. Thus, line no. 2 is not repeated. The loop is terminated. The program shifts to line no. 4.
10. Line no. 4 gives a message box on the screen. The program then stops.

For-Next with Step

So far, we have always increased the value of the index variable by 1. But what if we want to increase it by 2, or by 5? What if we want to decrease it by 1, or by 5? This option is provided by the Step keyword. The general syntax of For..Next now becomes

```
FOR i = initial value TO final value      STEP  j
        'everytime the statement
        'NEXT I is reached
        'i will change by
        'the value given in
        'STEP j
NEXT i
```

where 'j' is an integer variable, either positive or negative. If we want to decrease the value of " i " at every step, 'j' should be a negative integer.

Let us now use Step to make a program that generates the multiples of 2 upto the number 10, i.e., the output should be: 2,4,6,8,10.

Code Listing for.3

```
1. FOR i = 2 TO 10 STEP 2     'increase i by 2 at every iteration
2. PRINT i
3. NEXT i   'check if loop is to be repeated
4. MSGBOX "loop is over"
```

Here, the code runs as follows:

1. Line no. 1 gives 'i' the initial value of 2.
2. Line no. 2 prints '2' on the screen.
3. Line no. 3 increases the value of 'i' by 2, because the value of Step is 2. This changes the value of i to 4 . Line no. 3 then checks if the current value of i, i.e., 4 is between 2 and 10. Since it is, the program shifts to line no. 2 and prints the value of i.
4. As long as the value of i is between 2 and 10, the program will keep shifting to line no.2 from line no. 3. Once 'i' comes out of the range (2 to 10) the for-next loop will terminate.
5. Now line no. 4 will run, informing the user that the loop has terminated.

Thus, after giving the output as a sequence, i.e.,

2 4 6 8 10

the loop terminates.

We can make the current code do the reverse thing - printing even numbers from 10 to 2, in descending order. The output expected, thus, becomes - 10,8,6,4,2. The code is going to be as follows:

Code Listing for.4

```
1. FOR i = 10 TO 2 STEP -2    ' decrease i by 2 at every iteration
   'compare line above with line no. 1 of Code Listing for.3
2. PRINT i
3. NEXT i   'check if loop is to be repeated
4. MSGBOX "loop is over"
```

Here, the code runs as follows:

1. Line no. 1 gives 'i' the initial value of 10.
2. Line no. 2 prints '10' on the screen.
3. Line no. 3 decreases the value of 'i' by 2, because the value of Step is [-2]. This changes the value of 'i' to 8. Line no. 3 then checks if the current value of 'i' [8] is between 10 and 2. Since it is, the program shifts to line no. 2 and prints the value of 'i'.

As long as the value of 'i' is between 10 and 2, the program will keep shifting to line no.2 from line no. Once 'i' comes out of the range [10 to 2] the for-next loop will terminate. Then line no. 4 will be executed.

If we compare Code Listing for.3 and Code Listing for.4, we will notice that only line 1 has been changed. Thus, after giving the output

10 8 6 4 2

the loop terminates.

Here is another program, which accepts five names, one by one, and displays them on a message box. The code is like this:

Code Listing for.5

```
1. DIM nam AS STRING
2. FOR i = 1 TO 5 'define initial and final values
3. nam = INPUTBOX ("Please enter a name")
4. MSGBOX nam & "is a nice name indeed"
5. NEXT i   'check if loop is to be repeated
6. MSGBOX "loop is over"
```

Here, the code runs as follows:

1. Line no. 1 declares a String type of variable 'nam' at line 1.
2. Line no. 2 begins the for-next loop, initializing 'i' to the value 1.
3. Line no. 3 accepts a name and stores it in the variable 'nam'.
4. Line no. 4 displays this name in a message box alongwith a message that should please any user!
5. Line no. 5 increases 'i' by 1 and then checks if 'i' falls between 1 and 5. Since the present value of 'i' does fall in the range, line nos. 3 and 4 are re-executed, again asking for a name and displaying it in a message box.

As long as the value of 'i' stays between 1 and 5, the For..Next loop will continue to accept a name and display it in a message box. Once the value of 'i' goes beyond 5, the loop is ended and line no. 6 is executed.

12.5.2 Nested Loops

For..Next loops can be 'nested', with the same rules for nesting as for others: no crossing-over of the inner and outer loops. Thus, the following loop is valid as it does not have any intersection of the outer and inner For..Next loop.

```
FOR i = 1 TO 5
        FOR j = 1 TO 5
                'some processing
                'done here
        NEXT j
```

```
NEXT i
```

Whereas, the following loop is invalid, since two loops intersect:

```
FOR i = 1 TO 5
        FOR j = 1 TO 5
                'some processing
                'done here
        NEXT i
NEXT j
```

To minimize risk of wrong nesting, just type 'Next' instead of 'Next i', 'Next j', etc. This way, VB will follow a proper nesting by itself.

```
FOR i = 1 TO 4
        FOR j = 1 TO 4
                'some processing
                'done here
        NEXT
NEXT
```

By the way, can you guess how many times the inner 'j' loop will run? A total of 16 times! This is because of the following reason:

1. 'i' is 1: While 'i' stays at 1, 'j' will change in value from 1 to 4. Thus, the 'j' loop will run a total of 4 times when [i=1]. The variable 'i' does not change in value at present because the 'i' loop's [Next I] statement is not reached in this duration. (The program keeps looping between 'For j-Next j' statements.) Once 'j' becomes 5, the 'j' loop ends and the program comes to [Next I] statement. Now, the value of 'i' becomes 2 and the control shifts back to the [For j] statement.
2. 'i' is 2: The 'j' loop will again change in value from 1 to 4. Thus, the 'j' loop will run a total of 4 times while [i=2]. The variable 'i' does not change in value at present because the 'i' loop's [Next I] statement is not reached in this duration. (The program keeps looping between 'For j-Next j' statements.) Once 'j' becomes 5, the 'j' loop ends and the program comes to [Next I] statement. Now, the value of 'i' becomes 3 and the control shifts back to the [For j] statement.
3. 'i' is 3: The 'j' loop will again change in value from 1 to 4. The 'j' loop will run a total of four times while [i=3]. Once 'j' becomes 5, the 'j' loop ends and the program comes to [Next I] statement. Now, the value of 'i' becomes 4 and the control shifts back to the [For j] statement.
4. 'i' is 4: The 'j' loop will again change in value from 1 to 4. The 'j' loop will run a total of four times while [i=4]. Once 'j' becomes 5, the 'j' loop ends and the program comes to [Next I] statement. Now, the value of 'i' becomes 5. Since the value of 'i' is no longer between the 'i' loop condition [1 to 4], the 'i' loop is terminated. The program comes out of the 'i' loop and stops after [Next I].

As just explained, for every single value of i, j changes value from 1 to 4. Thus, the j loop runs '4' times for every single run of the i loop. The i loop will change from 1 to 4 and then terminate. This means

that the i loop will run a total of '4' times. Thus, the program will run a total of 4 * 4 = 16 times. (No. of iterations of 'i' loop * No. of iterations of 'j' loop for each single value of 'i').

12.5.3 Do Loops

Apart from the For-Next loop, VB provides the Do loop as another iterative mechanism. Needless to say, the need of this loop is the same as that of the For-Next loop. This loop is, broadly speaking, of two types:

- Do-While-Loop
- Do-Until-Loop

In both the types, we provide a condition after the keyword "Do-While" or "Do-Until". The repetition of the loop depends on whether the condition is true or false at any given point of time.

Do-While-Loop

In this loop, the loop lines are repeated as long as the given condition is true. Once the condition becomes false, the loop is terminated and the program continues with the statements that appear after the Do-While loop.

The general syntax for the Do-While loop is

```
DO WHILE some condition
        'execute some
        'statements repeatedly
        'while the condition
        'is true
LOOP
```

All of the statements written between 'Do-While' and 'Loop' are repeatedly executed till the time the condition mentioned after Do-While is true. That is, the 'Loop' statement shifts the control back to 'Do-While' and the condition is then checked. If the condition is true, the loop is re-executed. Once the condition becomes false, the program stops the repeated loop execution and begins to execute the statements after 'Loop'.

As an example, we will remake our Code Listing for.1 using Do-While along with a command button's click event:

Code Listing do.1

```
1.  i = 1
2.  DO WHILE i < 4 'check if i is less than 4
3.  PRINT i
4.  i = i + 1
5.  LOOP     'check if loop is to be repeated
6.  MSGBOX  " Do - While loop over"
```

Here, the code runs as follows:

1. Line no. 1 assigns the initial value for the variable 'i'.
2. Line no. 2 will check the condition [i < 4]. If the condition is true, the program will execute all lines upto line no. 5.
3. Line no. 3 will print the value of 'i' [1].
4. Line no. 4 increases the value of 'i' by 1: 'i' now becomes 2.
5. Line no. 5 shifts the control back to line 2, where the condition [i < 4] will be checked again. With the current value of 'i' [2], this condition is true. Thus, line no. 3 is executed, printing the value of 'i' [2] on the screen. Line no. 4 again increases value of 'i' by 1. So, 'i' is now 3. Line no. 5 shifts the control back to line 1, where the condition [i < 4] will be checked again. As long as the value of 'i' is less than 4, the program will keep on printing and incrementing the value of 'i' and checking whether the loop should be continued or not.

Once 'i' becomes equal to 4, the loop is terminated. The program shifts to line no. 6, displaying a message box on the screen and then stops.

Take another example. As done in Code Listing for.2, we will print a name on the screen four times. This time we'll use Do-While to get the same result, using a command button's Click event:

Code Listing do.2

```
1. i = 1
2. DO WHILE  i < 5      'check if i is less than 5
3. PRINT  "Taranya Gupta"
4. i = i + 1
5. LOOP     'check if loop is to be repeated
6. MSGBOX  " Name printing is over"
```

Here, the code runs as follows:

1. Line no. 1 assigns a value of 1 to the variable 'i'.
2. Line no. 2 compares the value of 'i' with 5. If it is less than 5, the remaining lines are executed. Otherwise, the loop is not executed.
3. Line no. 3 prints 'Taranya Gupta' on the screen.
4. Line no. 4 increases 'i' by 1.
5. Line 5 takes the control back to line 2 and here the condition [i < 5] is checked again. If it is true, line nos. 3 and 4 are repeatedly executed. Once the condition [i < 5] becomes false, further execution of the loop is stopped. The program now shifts to line no. 6.
6. Line no. 6 displays a message box on the screen. The program stops.

We'll now get the same results as from Code Listing for.3 seen earlier in the chapter. This time, though, we'll be using Do-While. Again, use a command button's Click as the event for the code. The code is as follows:

Code Listing do.3

```
1. i = 2
2. DO WHILE i < 11      'check if i is less than 11
3. PRINT i
4. i = i + 2
5. LOOP     'check if loop is to be repeated
6. MSGBOX "do while ended"
```

Here, the code runs as follows:

1. Line no. 1 assigns the number 2 to the variable 'i'.
2. Line no. 2 checks if [i < 11]. If it is true, line 3 is executed. If it is not true, all the loop lines are skipped over, and program control shifts to line no. 6.
3. Line no. 3 prints the value of 'i' on the screen.
4. Line no. 4 increases the value of 'i' by 2.
5. Line no. 5 takes the program back to line 2, where the condition [i < 11] is again checked. If the condition is true, line nos. 3 and 4 are re-executed. This means that the program continues with the repetition of the loop as long as the condition [i < 11] is true. Once the condition [i < 11] becomes false, the loop is terminated, and control shifts to the line following line 5.
6. Line no. 6 displays a messagebox and the program ends.

Thus, after giving the output

2 4 6 8 10

the loop terminates.

Now we will generate the same output in reverse order, i.e., 10,8,6,4,2, with the following code

Code Listing do.4

```
1. i = 10
2. DO WHILE i > 1 'check if i is greater than 1
3. PRINT i
4. i = i - 2
5. LOOP     'check if loop is to be repeated
6. MSGBOX "loop over"
```

Here, the code runs as follows:

1. Line no. 1 assigns 10 to 'i'.
2. Line 2 checks the condition [i > 1]. Since i is initially 10, this condition is true. Thus, line nos. 3 and 4 are run. After line 4, 'i' decreases by 2, i.e., 'i' drops to 8.
3. Line no. 5 takes the control back to line 2 and again checks for the condition [i > 1]. Since 'i' is still greater than 1, lines 3 and 4 are again executed. This process of repetition continues while 'i'

is greater than 1. Once 'i' becomes 0, the condition [i > 1] becomes false, and the repetition of the loop is stopped. The program then continues with any statements written after line 5 (LOOP).

4. Line no. 6 terminates the program after displaying a messagebox.

The output produced by Code List do.4 is

10

8

6

4

2

Do-Until Loop

With the Do-While loop, the repetitions continue as far as the given condition is "true". That is, the loop will continue "while" the condition is "true". The Do-Until loop is just the reverse of this. In a Do-Until loop, the repetitions take place as far as the given condition is "false". In other words, the loop will continue "until" the condition becomes "true". Once the condition becomes true, the loop is terminated and the program continues with the statements mentioned after Do-Until loop. The general syntax for the Do-Until loop is:

```
DO UNTIL some condition
        'execute some
        'statements repeatedly
        'until the condition
'becomes true
LOOP
```

All of the statements written between Do-Until and Loop are repeatedly executed as far as the condition mentioned after Do-Until is not true. That is, the Loop statement shifts the control back to Do-until and the condition is then checked. If the condition is not true, the loop is re-executed. Once the condition becomes true, the program stops the repeated loop execution and begins to execute the statements after Loop.

As an example, we will remake our Code Listing do.1 using Do-Until, to get the same result as from code listing do.1 using a button's click event:

Code Listing do.5

```
1. DO UNTIL i > 4 'continue if i is NOT greater than 4
2. PRINT i
3. i = i + 1
4. LOOP     'check if loop is to be repeated
5. MSGBOX  " Do - Until loop over"
```

Here, the code runs as follows:

1. Line no 1 will check the condition [i > 4]. If the condition is not true, the program will execute all lines upto line no. 4. If the condition is true, the loop will not be executed.
2. Line no. 2 will print the value of 'i'.
3. Line no. 3 will increase the value of 'i' by 1, which now becomes 2.
4. Line no. 4 will shift control back to line 1, where the condition [i > 4] will be evaluated again. With the current value of 'i' [2], this condition is still not true. Thus, line no. 2 is re-executed, printing the value of 'i' [2] on the screen. Line no. 3 increases value of 'i' by 1 again. This process of repeating the statements (from line no. 1 upto line no. 4) will continue until [i > 4]. Once 'i' becomes greater than 4, the loop will terminate.
5. Line no. 5 displays a message box and the program terminates.

Take another example. This time we will print a name on the screen four times, exactly as we have done in Code Listing for.2 and Code Listing do.2. Now we will use Do-UNTIL to get the same result.

Code Listing do.6

```
1.  i = 1
2.  DO UNTIL i > 4 'continue if i is NOT greater than 5
3.  PRINT "Sam"
4.  i = i + 1
5.  LOOP     'check if loop is to be repeated
6.  MSGBOX    " Name printing is over"
```

Here, the code runs as follows:

1. Line no. 1 initializes 'i' to 1.
2. Line no. 2 checks if the current value of 'i' is greater than 5. As the condition is not true, line no. 3 is executed.
3. Line no. 4 now increases 'i' by 1.
4. Line 5 takes the control back to line 2 and here the condition [i > 5] is checked again. If it is not true, the process of repeating the loop continues. Once the condition [i > 5] becomes true, further execution of the loop is stopped. The program now shifts to line no. 6.
5. Line no. 6 displays a message box on the screen and the program stops.

Now we will generate a series of numbers - 2,4,6,8,10. You might recollect that we have already done this using For-Next and Do-While. Now we will be using Do-Until to get the same result. Use the click event of a button for the code:

Code Listing do.7

```
1.  i = 2
2.  DO UNTIL i > 11      'continue loop if i is NOT greater than 11
3.  PRINT i
4.  i = i + 2
5.  LOOP     'check if loop is to be repeated
```

Here, the code runs as follows:

1. At line no. 1 'i' is assigned value of 2.
2. Line 2 checks whether [i > 11]. Since it is not, Line 3 is executed, printing the value of 'i' on the screen.
3. Line 4 increases the value of 'i' by 2.
4. Line 5 takes the program back to line 2, where the condition [i > 11] is again evaluated. If the condition is true, line nos. 2 and 3 are re-executed. This means that the program continues with the repetition of the loop as long as the condition [i > 11] is not true. Once the condition becomes true, the loop is terminated, and control shifts to the line following line 5.

   Thus, after giving the output

   2
   4
   6
   8
   10

   the loop terminates.

Here is another program, which accepts five names, one by one, and displays them in a message box, on the click of a button. The code, using Do-Until, is:

Code List do.8

```
1. DIM  nam  AS  STRING
2. i = 1
3. DO UNTIL i > 5 'continue loop if i is NOT greater than 5
4. nam = INPUTBOX ("Please enter a name")
5. MSGBOX nam & "is a very nice name"
6. i = i + 1
7. LOOP     'check if loop is to be repeated
```

Here, the code runs as follows:

1. Line 1 declares a string type of a variable.
2. Line 2 initializes 'i' to 1.
3. Line 3 checks the condition [i > 5]. Since the current value of 'i' [1] does not satisfy this condition, line 4 is executed.
4. Line no. 4 asks for a name through an Input box.
5. Line no. 5 displays the name in a message box.
6. Line no. 6 increments 'i' by 1 (i now becomes 2).

7. Line 7 transfers control back to line 3. Line 3 again checks for [i > 5]. This process of continuing with the loop will go on until [i > 5], i.e., 'i' becomes greater than 5. Once [i > 5] becomes true, further re-execution of the loop is stopped.

Check Your Progress

1. Define indeterminate loops.
2. Define Do-while loop.

12.6 LET US SUM UP

A sub-routine is a procedure that performs a task but does not return a value. Functions are a type of procedure, which return a value. This is what separates a function from a sub-procedure. As the name suggests, string functions work on string type of data. They help perform a number of useful activities that are of immense help in validating and manipulating our data.

A logical expression is a combination of constants, variables and operators, which evaluates to either TRUE or FALSE values. Logical expressions are formed using logical and relational operators. A logical operator compares two values (constants, expressions, variables etc.) and produces either true or false when used in an expression. Visual Basic provides many looping constructs such as do-while, while-wend and their variants.

12.7 KEYWORDS

Looping: Repeated execution of statements.

Logical Expression: A combination of constants, variables and operators, which evaluates to either TRUE or FALSE values.

Logical Operator: A logical operator compares two values (constants, expressions, variables etc.) and produces either true or false when used in an expression.

Functions: Functions are a type of procedure, which return a value.

String Functions: String functions work on string type of data

Sub Routines: A sub-routine is a procedure that performs a task but does not return a value.

12.8 QUESTIONS FOR DISCUSSION

1. Discuss the creation of sub routines.
2. How are functions created?
3. What are sting functions? Discuss.
4. What are logical operators? Discuss different logical operators with examples.
5. Differentiate between determinate loops and indeterminate loops.
6. What are do loops. Discuss with example.

7. What will be the output of the following code?

```
For j = -5 to 5
    MsgBox Str(j + 1)
    If (j > 4) Then
        Exit For
    End If
Next j
```

8. How many times will the statement1 execute in the following loop?

```
For j = 1 to 10
    For k = j to 10
        Statement1
    Next k
Next k
```

Check Your Progress: Model Answers

1. ***Indeterminate Loops:*** Continues until you reach a specific predetermined goal, or continue until certain initial conditions have changed.
2. In this loop, the loop lines are repeated as long as the given condition is true. Once the condition becomes false, the loop is terminated and the program continues with the statements that appear after the Do-While loop.

12.9 SUGGESTED READINGS

Paul Lomax, *Learning VB* Script, Volume 1, O'Reilly Media, Inc.

Jerry Lee Ford, *Learn VB Script in a weekend*, Premier Press

Adrian Kingsley-Hughes, Kathie Kingsley-Hughes, Daniel Read, *VB Script programmer's reference*, Wrox/Wiley Pub.

LESSON

13

VB SCRIPT AND FORMS

CONTENTS

13.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Understand vbscript
- Discuss working of vbscript with forms.
- Understand error hiding

13.1 INTRODUCTION

VB Script is defined as a scripting language developed by Microsoft. You can make your web pages more vibrant and interactive with the help of this scripting language. VB Script is a light edition of Visual basic and it has a simple syntax.

VB Script is extensively used and most accepted as a client side scripting language. In html language you use < and > around the tags. But you can use many tags inside one pair of < % and % >. For Printing a variable you can use < % = % >.

Scripting languages like Javascript and Vbscript are intended as an expansion to html language. The Web browsers such as Microsoft Internet Explorer obtain the scripts along with the remaining web page document. It is the browser accountability to parse and process the scripts. These scripts are extensively used as a client side scripting languages. Now we will start learning how the Script language works and we will show you step by step . So just focus on the further topics.

13.2 A SIMPLE PAGE

When you want to insert scripts in your web pages you are organizing you have to use < script > starting tag and < / script > ending tag where the script is going to close. The following example exemplifies how to add functionality to a Web page by counting VBScript code. The example of this simple page is shown below:

```
< html >
< head >
< title >vbscript example< /title >
< script language="vbscript" >
Msgbox "Hello"
< /script >
< /head >
< body >
< /body >
< /html >
```

We have used the example with a simple going "Hello" as there is a inclination that when a new learner is learning a new language we always welcome him.

Well, in the above example you have seen the language quality. We have used VbScript as the scripting language. This quarrel is needed as there are more than one scripting language. Without the language argument, browser would not able to understand if the text between the tags was vbscript or javascript or any other scripting language.

As the Script is fully dependent on the browser so next question that occur is " How to handle Non Supporting browsers". If the client browser has hindered the script than your web page might now function properly. So for your benefit you can also the server side language

13.3 USING VB SCRIPT WITH FORMS

As the reputation of web page forms raises, so does the requirement to be able to legalize data before the client browser proposes it to the web server. As a scripting language, VBScript is well matched for this task. After the validation of the form, the same script can be used to forward the data on to the server. In this lesson we will look at both the process of validating and submitting forms.

13.3.1 Validating Your Forms

The process of validating forms entails checking the form to see if:

- All of the required data is proved
- The data provided is valid

Meticulous data validation scripts can be monotonous to code but are well worth their return in validating the quality of the data.

The validation example that we will be probing does not enclose anything new in the way of VBScript Before reading any further you may find if favorable to consider how you would legalize an HTML form using the VBScript techniques.

Let's see an example to give you a thought of what is possible when it comes to validating forms.

13.3.2 Checking Form Input

This example is quite easy. It has a single field in which the user can enter their age and a single command button that is used to submit their age to the server. <HTML>

```
<HEAD>
<TITLE>Working With VBScript: Example 5a</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!-- Instruct non-IE browsers to skip over VBScript modules.
Option Explicit
Sub cmdSubmit_OnClick
' Check to see if the user entered anything.
 If (Len(document.frmExample5a.txtAge.value) = 0) Then
  MsgBox "You must enter your age before submitting."
  Exit Sub
 End If
' Check to see if the user entered a number.
 If (Not(IsNumeric(document.frmExample5a.txtAge.value))) Then
  MsgBox "You must enter a number for your age."
  Exit Sub
 End If
' Check to see if the age entered is valid.
 If (document.frmExample5a.txtAge.value < 0) Or _
   (document.frmExample5a.txtAge.value > 100) Then
  MsgBox "The age you entered is invalid."
  Exit Sub
 End If
' Data looks okay so submit it.
 MsgBox "Thanks for providing your age."
 document.frmExample5a.submit
End Sub
-->
```

```
</SCRIPT>
</HEAD>
<BODY>
<H1>A VBScript Example on Variables</H1>
<P> This example demonstrates validation techniques in VBScript. </P>
<FORM NAME="frmExample5a">
 <TABLE>
  <TR>
   <TD>Enter your age:</TD>
   <TD><INPUT TYPE="Text" NAME="txtAge" SIZE="2">
  <TR>
   <TD><INPUT TYPE="Button" NAME="cmdSubmit" VALUE="Submit"></TD>
   <TD></TD>
  </TR>
 </TABLE>
</FORM>
</BODY>
</HTML>
```

13.3.3 How It Works

The empathy of this validation script is established in the click event procedure for the cmdSubmit command button. We begin by inspecting if the user entered anything at all into the field using VBScript's *Len* function. This function returns the length of a string. If the length is 0, the data is invalid. We inform the user and exit the submit procedure via the Exit Sub statement:

```
' Check to see if the user entered anything.
 If (Len(document.frmExample5a.txtAge.value) = 0) Then
  MsgBox "You must enter your age before submitting."
  Exit Sub
 End If
```

Next we test out to see if what the user entered is a numeric value. The VBScript function IsNumeric returns a true value when it is a number. If not, we tell the user and exit:

```
'Check to see if the user entered a number.
 If (Not(IsNumeric(document.frmExample5a.txtAge.value))) Then
  MsgBox "You must enter a number for your age."
  Exit Sub
 End If
```

Our concluding check involves verifying that the age they entered appears to be sensible for our environment. I have determined that no age less than 0 or greater than 100 is acceptable. Using an *If..Then* statement we can check the value of the input field in opposition to this criteria:

```
'Check to see if the age entered is valid.
 If (document.frmExample5a.txtAge.value < 0) Or _
   (document.frmExample5a.txtAge.value > 100) Then
  MsgBox "The age you entered is invalid."
  Exit Sub
 End If
```

That's it. While this example is by no means the most thorough validation script you will come across it provides you with a basis of what is possible with VBScript.

13.3.4 Submitting Your Forms

Judged against to validation, the procedure of submitting a form is simple. In our example we've used a normal HTML button with the *Submit* caption that is joined to an event procedure that both validates and at the same time submits the form. The code that we would have to add to our previous example to submit the form is shown below:

```
'Data looks okay so submit it.
 MsgBox "Thanks for providing your age."
 document.frmExample5a.submit
```

The *MsgBox* statement lets the user know that their data has been processed. The form is then submitted by invoking the *Submit* method of the form object. As we saw in lesson 3 on objects, methods cause an object to perform a task. Here we are using the *submit* method of our form to cause the form to submit its data, just as if we had used a *submit* control.

13.4 ERROR HIDING

Error hiding is defined as an anti-pattern, in computer programming. The programmer conceals error messages by overriding them with exception handling. Consequently the root error message is concealed from the user (hence 'error hiding') and so they will not be told what the real error is. Error hiding is a curse of support engineers' jobs as it frequently delays the declaration of the problem by hiding information needed to identify what is going wrong.

A common argument for error hiding is the wish to hide intricacy from the user. Often best practice is to lift an exception to the user to conceal a complex error message but to save the full error message to an error log which a support engineer can access to resolve the problem.

Example:

```
try
ImportFile(filename);
except
```

```
// an exception with almost no information
raise Exception.Create('import failed');
end;
```

This code fragment tries to open a file and read it into memory. If it fails (for whatever reason) the user only gets a message telling them that the import failed, not why or indeed which file failed to import.

```
// better approach
try
ImportFile(filename);
except
on E:Exception do
begin
// build an informative message
E.Message := 'Import of file <'+filename+'> failed.'+#13#10 +
E.Message;
// re-raise the exception
raise;
end;
end;
```

In this example the user at least gets a message that tells them which file was unsuccessful to import; this gives them a start at identifying the problem. A more absolute solution would comprise additional information on why the import failed and write the information to a log file, perhaps (for very complex or enterprise level applications) also producing extra 'trace' files containing detailed records of the state of the application when the error appeared.

Sometimes error hiding is a applicable activity, for example accessing the contents of a file that does not exist in Java version 1.3 or older would result in an IOException message without any reference to the misplaced file. In this case is would be sensible to veil the error and raise an exception based on what the application was trying to do at the time, giving what extra information can be obtained.

Another reason for Error Hiding is to shun component crashing in case of failure. In spite of the error, the component carry on its job. The user (and the testers) never see the crash, even if some anomalies *can* be discovered, as the results are not those predictable.

This is completed either with a try/catch with an empty catch clause, or by executing code depending on the returning error status:

Example of try/catch error hiding (C++ code):

```
try
{
doSomething() ;
doSomethingElse() ;
```

```
anotherDoSomething() ;
}
catch(...)
{
}
```

As it is this code is tremendously difficult to correct (and is even more, in case of nested "empty try/catch" Error Hiding code), and any variance is extremely difficult to trace to its origin, increasing maintenance costs, only to keep up an appearance of robustness.

Example of error returning error hiding (VBScript code):

```
If doSomething() Then
If doSomethingElse() Then
If anotherDoSomething() Then
anotherOtherDoSomething()
End If
End If
End If
```

The outcome is that when some error happens, it is secreted by the code (because it's error prone or simply verbose to add an Else clause) until someone notices something is muddled.

A tricky outcome is that when some similar code is written, but without the If/End If clauses, and is executed after the first code, the second code will fail, but no one will know the failure happened before and was hidden.

Thus one can aggravate the appearance of a bug months after the bug was first introduced (but hidden and thus, never discovered).

Check Your Progress

Define Checking Form Input.

13.5 LET US SUM UP

There are scripting languages like Javascript and Vbscript and they are designed as an extension to html language. The Web browsers like Microsoft Internet Explorer receives the scripts along with the rest of the web page document. When ever you want to add scripts in your web pages you are preparing you have to use < script > starting tag and < / script > ending tag where the script is going to close. As the popularity of web page forms increase, so does the need to be able to validate data before the client browser submits it to the web server. As a scripting language, VBScript is well suited for this task. Error hiding is an anti-pattern, in computer programming. The programmer hides error messages by overriding them with exception handling. As a result of this the root error message is hidden from the user (hence 'error hiding') and so they will not be told what the actual error is.

13.6 KEYWORDS

VB Script: It is widely used and most popular as a client side scripting language.

Error Hiding: It is an anti-pattern, in computer programming. The programmer hides error messages by overriding them with exception handling.

13.7 QUESTIONS FOR DISCUSSION

1. What is vbscript? Discuss how a simple page is created using vbscript.
2. Discuss how vbscript is used with forms.
3. What is error hiding? Discuss it with example.
4. Write the output of the following code:

(a)
```
Dim z As Integer
Do Until z > 10
    z = z + 1
    Print z
Loop
```

(b)
```
Dim y
Do
y = y + 2
Print y
Loop While y >= 20
```

(c)
```
Dim d As Form
For Each d In Forms
d.Hide
Next
```

Check Your Progress: Model Answers

It has a single field in which the user can enter their age and a single command button that is used to submit their age to the server.

13.8 SUGGESTED READINGS

SAMS Teach Yourself Visual Basic 6 in 24 Hours; Greg M. Perry, Sanjaya Hettihewa; SAMS Publishing; 1988

Visual Basic 2005 in a Nutshell; Paul Lomax, Steven Roman, Ron Petrusha, Tim Patrick; O'Reilly; 2006

Beginning Visual Basic 6; Peter Wright; Wrox Press; 1998