install. Right-click on the font file and click on the Install option in the menu. Make sure we are installing the desktop fonts and not web fonts. Zipped folders MUST be unzipped to extract the font files.

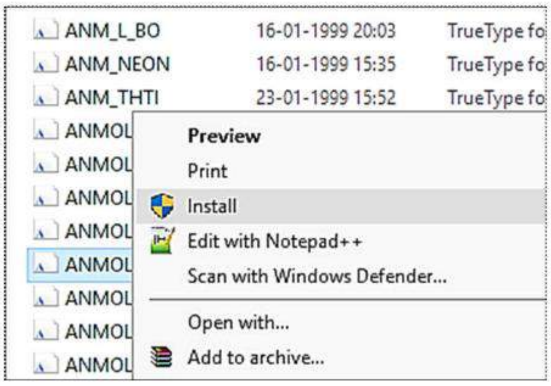

ANM_L_BO 16-01-1999 20:03 TrueType fo
ANM_NEON 16-01-1999 15:35 TrueType fo
ANM_THTI 23-01-1999 15:52 TrueType fo
ANMOL
Preview
Print
Install
Edit with Notepad++
Scan with Windows Defender...
Open with...
Add to archive...

Fig. 4.22 Fonts

4.14 UTILITY PROGRAMS

A program that performs a specific task related to the management of computer functions, resources, or files, as password protection, memory management, virus protection, and file compression is called Utility Program. In other words, Software that plays a supporting role for users and developers, are called Utility Programs. Basic utility programs include file/folder management (copy, move, etc.), file search and compare, disk format and partition, as well as diagnostic routines to check performance and the health of the hardware. Let's discuss some of the commonly used Utility program:

4.14.1 File Compression tools

File compression is a process of "packaging" a file (or files) to use less disk space. Compression software allows you to take many files and compress them into one file, which is smaller than the combined size of the originals.A file compression utility can be helpful in situations where hard drive space needs to be saved or large files have to be sent via email. Almost every computer in the world has a file compression utility.

There is plenty of file compression software available in the market. However, every software supports different file archive formats and offers different features. Winzip, WinRAR, 7-zip are the example of commonly used file compression tools.

The above software are required to download from Internet, but if we want to compress our files/folders then Windows gives us an option for this. Following steps should be used to compress files/folders using built option of window are as follows:

- Select the files/folders to compress
- Right click on it/them
- Click Send to → Compressed (Zipped) Folder, as shown in figure 4.23.

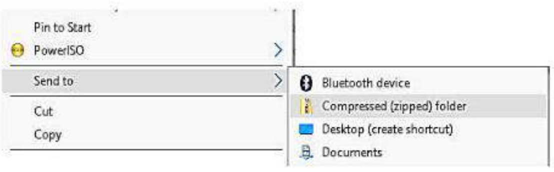


Fig. 4.23

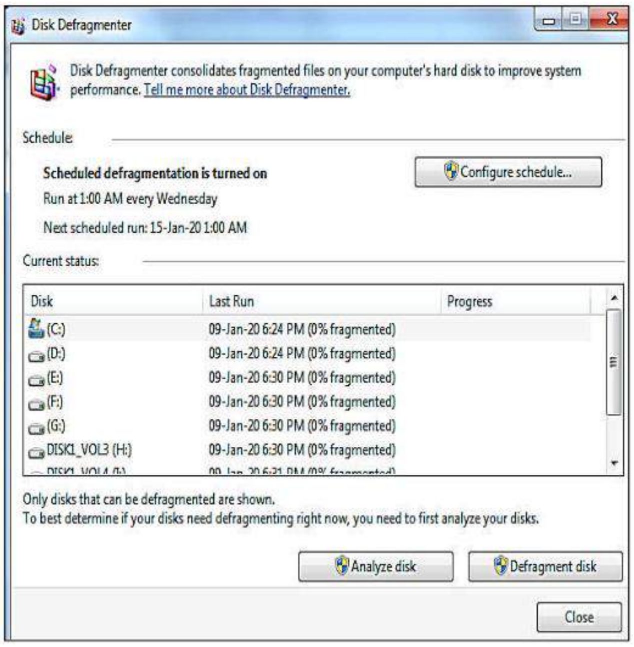
In this way, our required files/folders will be compressed to some extent.

4.13.2 Disk Defragmentation

It is another important inbuilt utility of Windows operating system. Defragmentation is like cleaning space for our PC.It picks up all of the pieces of data that are spread across our hard drive and puts them back together again. It is very important because every computer is affected by the constant scattering growth of data in disk. If we don't clean space in disk then our PC start giving problems.

Disk fragmentation occurs when a file is broken up into pieces to fit on the disk. Because files are constantly being written, deleted and resized, fragmentation is a natural occurrence.

When a file is spread out over several locations, it takes longer to read and write. But the effects of fragmentation are far more widespread: Slow PC performance, long boot-times, random crashes etc.



Disk Defragmenter

Disk Defragmenter consolidates fragmented files on your computer's hard disk to improve system performance. Tell me more about Disk Defragmenter.

Schedule:

Scheduled defragmentation is turned on
Run at 1:00 AM every Wednesday
Next scheduled run: 15-Jan-20 1:00 AM

Current status:

Disk	Last Run	Progress
(C:)	09-Jan-20 6:24 PM (0% fragmented)	
(D:)	09-Jan-20 6:24 PM (0% fragmented)	
(E:)	09-Jan-20 6:30 PM (0% fragmented)	
(F:)	09-Jan-20 6:30 PM (0% fragmented)	
(G:)	09-Jan-20 6:30 PM (0% fragmented)	
DISK1_VOL3 (H:)	09-Jan-20 6:30 PM (0% fragmented)	
DISK1_VOL4 (I:)	09-Jan-20 6:31 PM (0% fragmented)	

Only disks that can be defragmented are shown.
To best determine if your disks need defragmenting right now, you need to first analyze your disks.

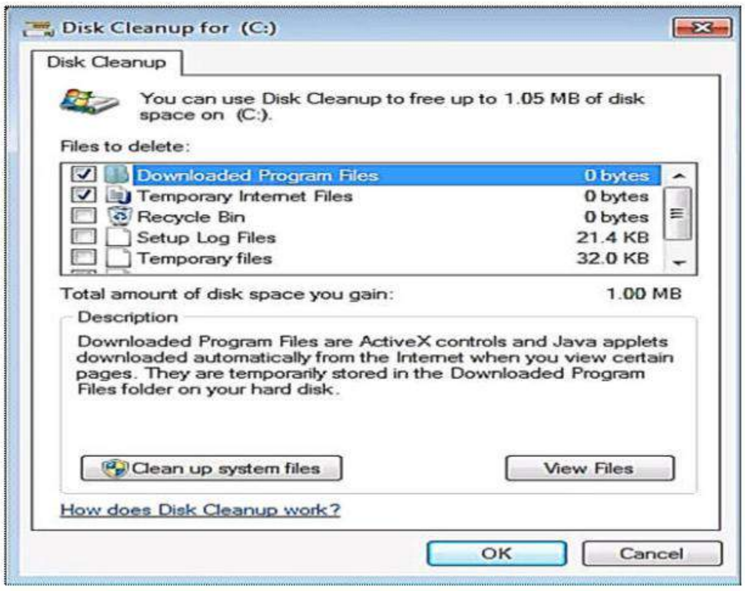
Fig. 4.24 Disk Defragmenter

To defrag a Hard disk in computer in Windows, perform the following steps:

- Open the Windows Start menu by pressing the Window key from the keyboard.
- Type **'defragment'** and a search result called **'Defragment and Optimise Drives'** will show up.
- Click on it to open the **Disk Defragmenter** window.
- In the opened window, select the disk (from the list shown) that we wish to defragment.
- Now click on the Defragment disk to start defragmentation process.

4.14.3 Disk Clean-up:

Disk Clean-up is computer maintenance utility that is included in the Microsoft Windows operating system and it is designed to free up space on the hard drive. The cleanup process involves searching and analysing the hard drive for files that are no longer needed. Then it proceeds to remove them and thus freeing up disk space on the hard drive.


Disk Cleanup for (C:)

Disk Cleanup

You can use Disk Cleanup to free up to 1.05 MB of disk space on (C:).

Files to delete:

☑	Downloaded Program Files	0 bytes
☑	Temporary Internet Files	0 bytes
☐	Recycle Bin	0 bytes
☐	Setup Log Files	21.4 KB
☐	Temporary files	32.0 KB

Total amount of disk space you gain: 1.00 MB

Description

Downloaded Program Files are ActiveX controls and Java applets downloaded automatically from the Internet when you view certain pages. They are temporarily stored in the Downloaded Program Files folder on your hard disk.

Fig. 4.25 Disk Cleanup

To defrag a computer in Windows, perform the following steps:

- Open the Windows Start menu by pressing the Window key from the keyboard.
- Type **'disk clean'** and a search result called **'Disk Clean-up'** will show up.
- Click on it to open the Disk Clean-upwindow.
- In the opened window, choose what type of files and folders to delete.
- Click OK
- To delete system files that are no longer needed, click Clean up system files. We may be prompted by UAC (User Account Control) to confirm the action.
- Click Delete Files
- To free more space, go to the More options tab.
- Click Clean up at the Programs and Features section to remove program files that are no longer needed.
- Click Clean up at the System Restore and Shadow Copies section to remove restore points, except the last one.

4.14.4 Backup and Restore

A **backup** is a copy of a file or a set of multiple files that is stored in a separate location from the original, such as a DVD, an external drive or someplace else on the Internet. A backup helps protect our files from being permanently lost or damaged during an accidental deletion, a virus attack or a failure of our system. Typically, people make backups of files including pictures, videos, music, projects and financial records. However, programs and software do not normally need to be copied into a backup as they typically take up a lot of space and we can re-use the original product to reinstall them if necessary.

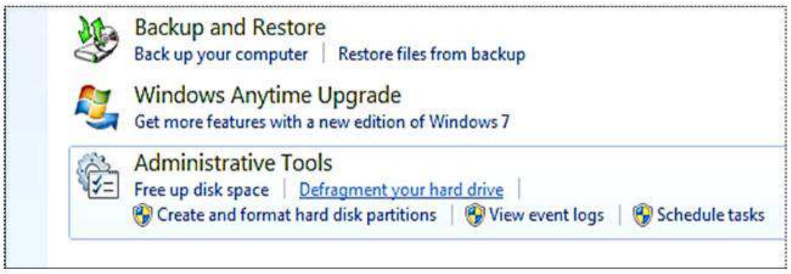

Backup and Restore
Back up your computer | Restore files from backup
Windows Anytime Upgrade
Get more features with a new edition of Windows 7
Administrative Tools
Free up disk space | Defragment your hard drive |
Create and format hard disk partitions | View event logs | Schedule tasks

Fig 4.26 Backup and Restore

Whereas, a **system restore** is a process that generally happens automatically on our computer's operating system. At various points in time, our computer will create restoring points where it "remembers" some of the information we are working on. We will use a restore when our computer has a problem. For example, if a program is freezing but we have not had time to save the document we were working on, a restore will allow us to go back to a previous point, which can be as far back as a couple of days ago, or just a few minutes before the problem occurred. This doesn't mean we will get the original document in its entirety, but it will allow us to get back some past data without damaging our system's integrity.

A backup is not automatic, while restore point are created automatically by our computer. Backup and Restore option is available in the Control Panel of the window.

4.15 SHUTTING DOWN OPTIONS

Windows supports several states for when we're not at our computer, and they're not all the same. Some methods help us shut down our computer completely, while another methods makes it look like our PC is turned off but it's actually ready to jump into action at a moment's notice.

The key to shutting down our Windows computer is in the Start menu. Click the Start button and we'll see, among other items, the shutdown button. Next to that button is a triangle; click the triangle to bring up the other shut down options such as: switch user, Log off, Lock, Restart, Sleep, Hibernate etc. The Switch User, Log Off, Lock, and Sleep options don't turn off the computer. The Restart, Hibernate, and Shut Down options do turn off the system. All these options are explained below:

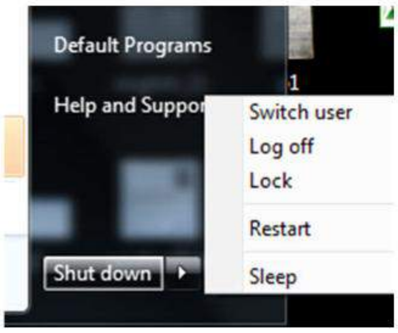


Fig. 4.27

- **Switch User :** We remain logged in to the computer and our programs continue to be open, but choosing this option allows another user to access the computer.
- **Log Off :** We can end our Windows session, save our stuff, and quit programs, but Windows remains on and ready for other people to use the computer.

- **Lock :** Not a complete logout, this option helps you protect your stuff by displaying the Windows logon screen. We must type our password or log in as another user to get into the computer.
- **Restart :** The computer is shut down and then started again when this option is chosen from the Shut Down menu. It's also called a reset or a warm boot.
- **Sleep :** The computer is put into a low-power consumption mode, saving energy. Also known as Stand By, this mode may put the entire computer, or only the monitor or hard drives, into low-power mode. In this power-saving mode, the computer comes back to life quickly, usually with the press of a key or jiggle of the mouse.
- **Hibernate :** Choosing this option, the best power-saving mode, shuts down the computer and turns it off. But information in memory is saved so that when the computer turns on again, we simply resume our former activities (after logging in). Hibernation saves the most power, but it takes longer to restart the computer than either Sleep or Standby mode, because we're literally turning it on again.
- **Shut Down or Turn Off :** When this option is selected, the computer is shut down: We're logged out of our account, which closes our programs and allows us to save our data. Windows then shuts itself down, and eventually the computer turns itself off.

Points To Remember

1. Computer hardware maintenance involves taking care of the computer's physical components, such as its keyboard, hard drive and internal CD or DVD drives.
2. Whereas software maintenance is a process by which a computer program is altered or updated after it has been released.
3. Preventive Maintenance is the process of inspecting hardware on a regular basis to ensure it stays in good running order.
4. Safemode is a diagnostic mode of a computer operating system (OS). Safe Mode starts our PC with a minimal set of drivers.
5. A driver is software that a device uses to work with our PC.
6. PnP is a term used to describe that the devices will start work with a computer system as soon as they are connected.
7. A port is a physical docking point using which an external device can be connected to the computer.
8. A software update includes bug fixes, and other small improvements, while a software upgrade changes the version of software.
9. A font is the combination of typeface and other qualities, such as size, pitch, and spacing.
10. File compression is a process of "packaging" a file (or files) to use less disk space.

11. A backup is a copy of a file or a set of multiple files that is stored in a separate location from the original, such as a DVD, an external drive or someplace else on the Internet.

12. Using Log Off,we can end our Windows session, save our stuff, and quit programs, but Windows remains on and ready for other people to use the computer.

EXERCISE

Part-A

1. Objective Type Questions

I. ______________ is a process by which a computer program is altered or updated after it has been released.

a.	Software maintenance	b.	Hardware maintenance
c.	Corrective maintenance	d.	Preventive Maintenance

II. In computing, ___________ is the process of starting a computer.

a.	Safe mode	b.	booting
c.	starting	d.	login

III. Which of the following is not a type of computer port?

a.	Ethernet	b.	PS/2 Port
c.	VGA	d.	Printer

IV. ___________________ security tool is built into the latest versions of Windows and helps guard our PC against viruses and other malware.

a.	Antivirus	b.	Malware
c.	Windows Defender	d.	Defragmenter

V. ______________ is a software which acts as an interface between the end user and computer hardware.

a.	Windows Defender	b.	File Compression Utility
c.	Operating system	d.	Security Tools

2. Fill in the Blanks:

I. _____________ is the process of inspecting hardware on a regular basis to ensure it stays in good running order.

II. A _______________ is software that a device uses to work with our PC.

III. A ____________ is a physical docking point using which an external device can be connected to the computer.

IV. A __________ is the combination of typeface and other qualities, such as size, pitch, and spacing.

V. Using ___________,we can end our Windows session, save our stuff, and quit programs, but Windows remains on and ready for other people to use the computer.

3. **Write the Full form of following:**

   I. PnP

   II. USB

   III. VGA

   IV. UAC

   V. OS

   VI. NAP

Part-B

4. **Short Answer Type Questions. (Write the answers in 4-5 lines)**

   I. What is Preventive Maintenance?

   II. What do you mean by Plug and Play Devices?

   III. Write about the PC Security tools.

   IV. What do you know about Windows Operating Systems?

   V. What is Control Panel in Windows operating system?

Part-C

5. **Long Answer Type Questions. (Write the answers in 10-15 lines)**

   I. Write the difference between software update and upgrade.

   II. Explain the basic guidelines for preventive maintenance.

   III. What are ports? Explain any two types of computer ports.

   IV. What are Utility Programs? Explain any one utility program in detail.

   V. What are various Shutting Down options? Explain.

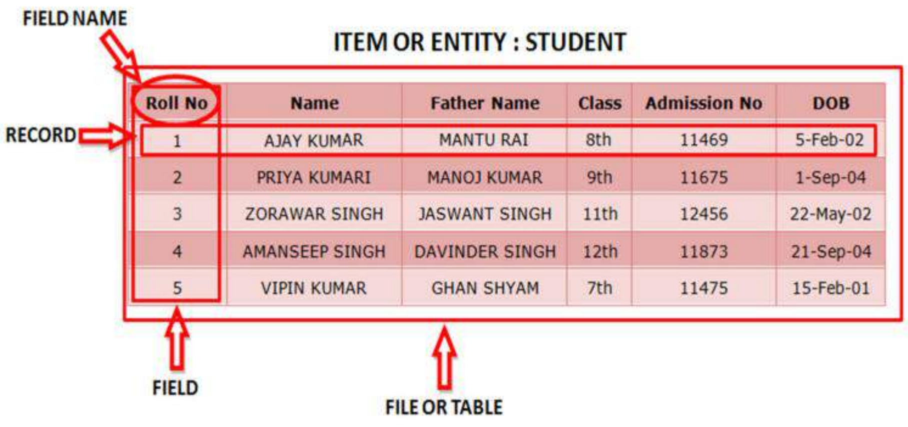
CHAPTER - 5 DATABASE MANAGEMENT SYSTEM

OBJECTIVES OF THIS CHAPTER

5.1 Introduction
5.2 Database
5.3 DBMS
5.4 Application Areas of DBMS
5.5 Data Base Life Cycle
5.6 DBMS Working
5.7 Architecture of DBMS
5.8 Features/ Characteristics of DBMS
5.9 Limitations of DBMS
5.10 Cloud Database

5.1 INTRODUCTION

In previous classes, we have studied about data and information. We all know that data is similar type of facts or figures and processed form of data is known as information. For example: number '24' may be treated as data, but Age 24, Street No. 24 or Sector 24 is processed form of data which provides some information.



Roll No	Name	Father Name	Class	Admission No	DOB
1	AJAY KUMAR	MANTU RAI	8th	11469	5-Feb-02
2	PRIYA KUMARI	MANOJ KUMAR	9th	11675	1-Sep-04
3	ZORAWAR SINGH	JASWANT SINGH	11th	12456	22-May-02
4	AMANSEEP SINGH	DAVINDER SINGH	12th	11873	21-Sep-04
5	VIPIN KUMAR	GHAN SHYAM	7th	11475	15-Feb-01

Fig: 5.1 Table of entity Student

Related information of a particular item may be treated as Record. For example: A student is an item. Student's Name, Roll No., Class, Admission No., etc. is Record of the student.

Now, Collection of related record of Number of different students can be treated as File.

The Collection of such Files or Tables is known as database.

Now we can understand the hierarchy of Database and can be represented as under:

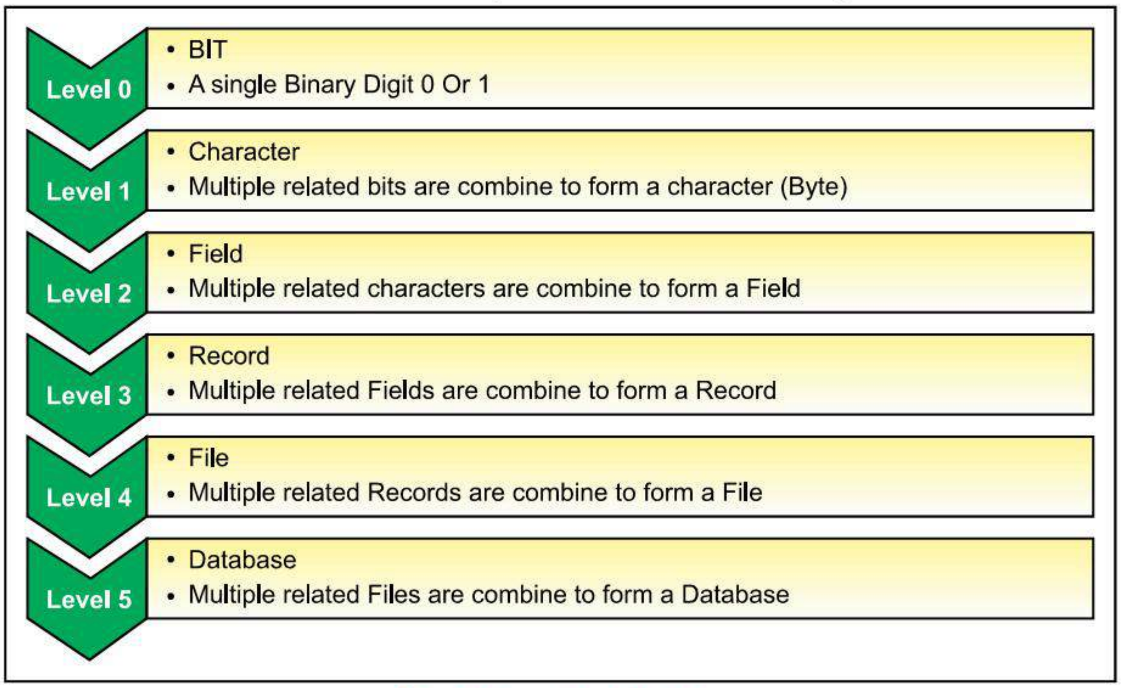

Level 0
• BIT
• A single Binary Digit 0 Or 1
Level 1
• Character
• Multiple related bits are combine to form a character (Byte)
Level 2
• Field
• Multiple related characters are combine to form a Field
Level 3
• Record
• Multiple related Fields are combine to form a Record
Level 4
• File
• Multiple related Records are combine to form a File
Level 5
• Database
• Multiple related Files are combine to form a Database

Fig: 5.2Hierarchy of Database

5.2 DATABASE

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc. The basic purpose of database is to combine data from all different sources. So that the useful information available to many users for their different purposes.

5.3 DBMS

DBMSstands for Database Management System. Traditionally, data was organized in file formats. DBMS is a computerized record keeping system that allows us to electronically organise and manipulate data using computer in a fast and secure way. It allows us to store and maintain data using the software and computer. In simple words, Database management system is software that is used to manage the database. For examples the school Database organizes the data for the admin, staff and students etc.

Examples of some popular DBMS are MySQL, Microsoft Access, dBASE, FoxPro, Oracle, etc.

DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and many more.It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

5.4 APPLICATION AREAS OF DBMS

Some of application areas of DBMS are shown in table below:

Area	Uses
Educational Institutes	For student information, registration, courses and grades.
Banks	For customer information, account activities, payments, deposits, loans, etc.
Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.
Manufacturing	It is used for the management of supply and for tracking production of items. Inventories status in warehouses etc.
Tour &Travelling	For reservations and schedule information.
Sales	Use for storing customer, product & sales information.

5.5 DATA BASE LIFE CYCLE

A software development life cycle model (SDLC) consists of a set of processes defined to accomplish the task of developing a software application that is functionally correct and satisfies the user's needs. Various processes in developing any software are:

1. Planning
2. Requirements
3. Design
4. Development
5. Implementation
6. Testing
7. Installation and Maintenance

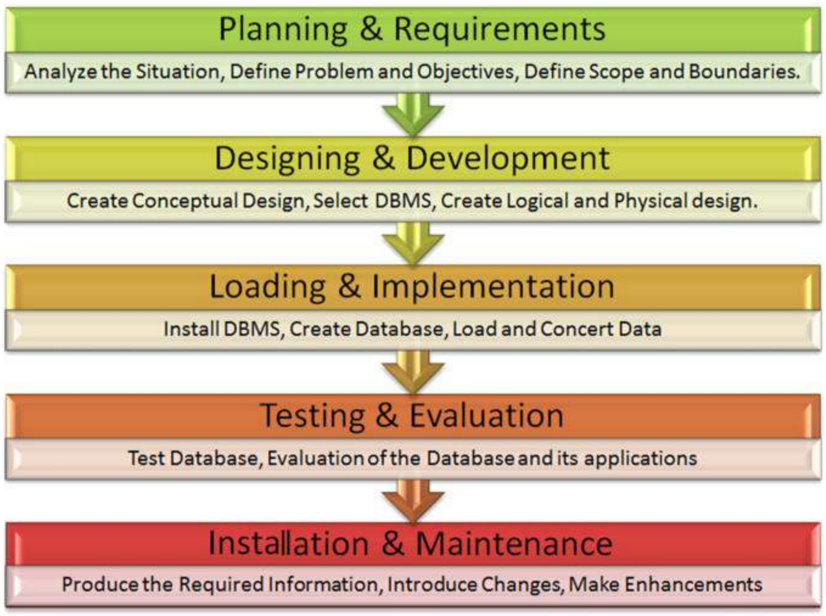

Planning & Requirements
Analyze the Situation, Define Problem and Objectives, Define Scope and Boundaries.
Designing & Development
Create Conceptual Design, Select DBMS, Create Logical and Physical design.
Loading & Implementation
Install DBMS, Create Database, Load and Concert Data
Testing & Evaluation
Test Database, Evaluation of the Database and its applications
Installation & Maintenance
Produce the Required Information, Introduce Changes, Make Enhancements

Fig 5.3Data Base Life Cycle

5.6 DBMS WORKING

A DBMS stores data in a way that it becomes easier to retrieve, manipulate, and produce information. It is basically a computerized record keeping system.

A typical DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows:

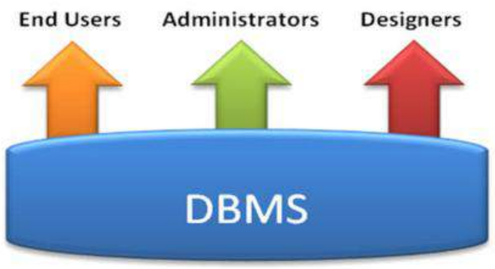


Fig: 5.4

- **Designers –** Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in which format. They identify and design the whole set of entities, relations, constraints, and views.
- **Administrators –** Administrators maintain the DBMS and are responsible for administrating the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.
- **End Users –** End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the market rates to sophisticated users such as business analysts.

5.7 ARCHITECTURE OF DBMS

Database Management system is composed of many inter-related components. DBMS is not always directly available for users and applications to access and store data in it. These components are organized to achieve the goals of system, is referred to as architecture of the system. A Database Management system can be centralised (all the data stored at one location), decentralised (multiple copies of database at different locations) or hierarchical, depending upon its architecture.

Database system also can be designed in different architectures of a DBMS. The major purpose of this system is to provide users with an abstract view of data. The system hides certain detail of how the data is stored and maintained. The goal of the architecture is to separate the user applications and the physical database.

Database Architecture is of following types:

1. **1-tier DBMS architecture :** In this, the database is directly available to the user for using it to store data. Generally such a setup is used for local application development, where programmers communicate directly with the database for quick response.

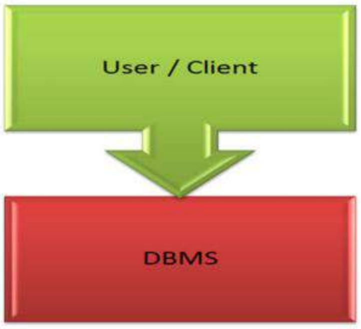


Fig: 5.5

2. **2-tier DBMS architecture :** In 2-tier DBMS architecture, an Application layer is included in between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.

   In the 2-tier architecture, we have an application layer which can be accessed programmatically to perform various operations on the DBMS. The application generally understands the Database Access Language and processes end users requests to the DBMS.

   An application interface known as ODBC (Open Database Connectivity) provides an API that allows client side program to call the DBMS. Most DBMS vendors provide ODBC drivers for their DBMS.

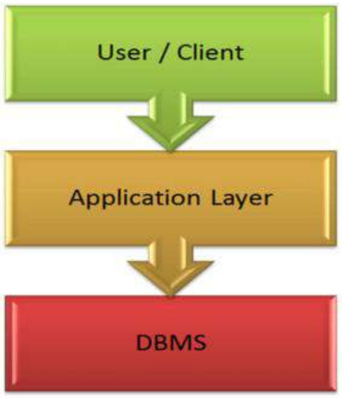


Fig: 5.6

Such architecture provides the DBMS extra security as it is not exposed to the End User directly. Also, security can be improved by adding security and authentication checks in the Application layer too.

3. **3-tier DBMS architecture :** 3-tier DBMS architecture is the most commonly used architecture for web applications.

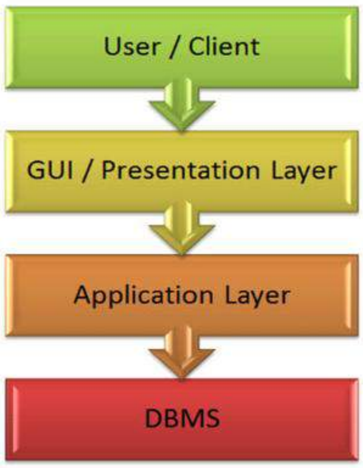


Fig: 5.7

It is an extension of the 2-tier architecture. In 3-tier architecture, an additional Presentation or GUI Layer is added, which provides a graphical user interface for the End user to interact with the DBMS. For the end user, the GUI layer is the Database System, and the end user has no idea about the application layer and the DBMS system.

5.8 FEATURES/CHARACTERISTICS OF DBMS

If we Compare DBMS to the File Based Data Management System, It has many advantages. Some of these advantages are given below:

- **Reducing Data Redundancy :** In file-based data management systems, multiple files were stored at different locations in a system or even across multiple systems. Multiple copies of the same file leads to data duplicity or data redundancy. This is prevented with DBMS as there is a single database and any change in it is reflected immediately. So, there is no chance of encountering duplicate data.
- **Sharing of Data :** The various users of the database can share the data among themselves. Here are various levels of authorisation to access the data, and consequently the data can only be shared based on the correct authorisation protocols being followed. Many remote users can also access the database simultaneously and share the data between themselves.
- **Data Integrity :** It means that the data is accurate and consistent in the database. Data Integrity is very important as there are multiple databases in a DBMS. All of these databases contain data that is visible to multiple users. So it is necessary to ensure that the data is correct and consistent in all the databases and for all the users.
- **Data Security :** Data Security is very important concept in a database. Only authorised can access the database and their identity should be authenticated using a username and password. Unauthorised users should not be allowed to access the database under any circumstances as it violates the integrity constraints.

- **Privacy :** The privacy in a database means only the authorized users can access a database according to its privacy constraints. There are levels of database access and a user can only view the data he is allowed to. For example - In social networking sites, access constraints are different for different accounts a user may want to access.
- **Backup and Recovery :** Database Management System automatically takes care of backup and recovery. The users need not to backup data periodically because this is taken care of by the DBMS. Moreover, it also restores the database after a crash or system failure to its previous condition.
- **Data Consistency :** Data consistency is ensured in a database because there is no data redundancy. All data appears consistently across the database and the data is same for all the users viewing the database. Moreover, any changes made to the database are immediately reflected to all the users and there is no data inconsistency.

5.9 LIMITATIONS OF DATABASE MANAGEMENT SYSTEM

The following are the disadvantages of Database Systems

- **Setup Cost :** DBMS requires huge amount of investment, needed to setup the required hardware and the softwares. High initial investment based upon size and functionality of organization is required.
- **Database Complexity :** The designing of the database system is complex, difficult and is very time consuming task to perform. The Developer, designer, DBA and End user of database must have complete skills, if they don't understand this complex system then it may cause loss of data or database failure.
- **Technical staff requirement :** Initial training is required for organization's employees, all programmers and user. Large amount of human efforts, the time and cost is needed to train the end users and application programmers in order to get used to the database systems.
- **Database Failure :** As we know that in DBMS, all the files are stored in single database so chances of database failure become more. Any accidental failure of component may cause loss of valuable data. This is really a big question mark for big firms.
- **Performance :** Traditional files system was very good for small organizations as they give splendid performance. But DBMS gives poor performance for small scale firms as its speed is slow.

5.10 CLOUD DATABASE

A cloud database is the database that typically runs on cloud computing platform. Users can choose from two types of methods to run their database in the cloud. The first method is cloud platforms allow users to install and maintain their own databases for a limited time. That is, users can purchase or maintain a database from a third party and use other services provided by cloud computing platforms. For example, users can use Oracle Database 11g Enterprise Edition provided by Oracle on Microsoft Azure (which is PaaSPlatform-as-a-service). The second method is cloud platforms are responsible for installing and maintaining the databases and users pay for these parts of services. This method is called Database-as-a-service (DBaaS)

- A database service built and accessed through a cloud platform
- Enables enterprise users to host databases without buying dedicated hardware
- Can be managed by the user or offered as a service and managed by a provider
- Can support relational databases and NoSQL databases
- Accessed through a web interface or vendor-provided API

Points To Remember

1. Related information of a particular item may be treated as Record
2. Collection of related record of Number of different students can be treated as File
3. The Collection of such Files or Tables is known as database
4. The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently
5. DBMS stands for Database Management System
6. DBMS is a computerized record keeping system that allows us to electronically organise and manipulate data using computer in a fast and secure way
7. Database management system is software that is used to manage the database.
8. Designers are the group of people who actually work on the designing part of the database
9. Administrators maintain the DBMS and are responsible for administrating the database.
10. End users are those who actually reap the benefits of having a DBMS
11. In 1-tier DBMS architecture the database is directly available to the user for using it to store data
12. In 2-tier DBMS architecture, an Application layer is included in between the user and the DBMS
13. In 3-tier architecture, an additional Presentation or GUI Layer is added, which provides a graphical user interface for the End user to interact with the DBMS

EXERCISE

Part-A

1. **Write True or False**

   I. Collection of related record of Number of different students can be treated as Database. 🟩

   II. The database is used to retrieve, insert and delete the data efficiently. 🟩

III. DBMS is a computerized record keeping system.

IV. Database management system is Hardware.

V. Database management system is used to manage the database.

2. **Very Short Answer Type Questions**

I. Related information of a particular item may be treated as?

II. The Collection of Files or Tables is known as?

III. DBMS stands for?

IV. Who are the group of people who actually work on the designing part of the database?

V. Who maintains the DBMS and are responsible for administrating the database?

Part-B

3. **Short Answer Type Questions. (Write the answers in 4-5 lines)**

I. Explain the Application Areas of DBMS?

II. Define about DBMS Working?

III. What do you mean by End User?

IV. Define the 2-tier DBMS architecture in DBMS?

V. What is Cloud database?

Part-C

4. **Long Answer Type Questions. (Write the answers in 10-15 lines)**

I. Explain the Features of Database Management System?

II. Explain the Limitations of Database Management System?

III. Explain about Architecture of DBMS?

IV. Explain about Data Base Life Cycle?

V. Make the chart about hierarchy of Database?

Lab Activity

- Make a chart to represent the life cycle of a database

CHAPTER - 6
CONCEPT OF PROGRAMMING AND PROGRAMMING LANGUAGES

OBJECTIVES OF THIS CHAPTER

6.1 INTRODUCTION

In this chapter, we are going to learn about the concept of program, programming, programming process and different categories of programming languages used for computer systems. A program is basically a set of instructions to be executed by computer to perform some task. The process of writing a program is called programming. The person who writes the program is called programmer. When a programmer writes a program, he or she goes through a particular process. This process of developing a program is called programming process. The programmer can use any language from hundreds of available programming languages for program development.

6.2 CONCEPT OF PROGRAM AND PROGRAMMING

We know that a computer system basically consists of two parts: hardware and software. Without software, computer-hardware cannot do anything hence computer is nothing but a piece of metal without software. To make the computer-hardware to do something, we must install and use software in our computer system. Now the question arises what is software?

Software is a set of computer programs which are designed and developed to perform desired tasks on computer. It is the software which makes a computer capable of data processing, storing and retrieval. Basically, softwares are categorised into two types: system software and application software. **System software** are designed and developed to control the functionality & to operate computer system hardware while the application software are designed and developed to perform specific tasks using computer system. System softwares are more complex as compared to application softwares. The development (programming) of system softwares require more skills as compared to application software development.

Software is usually not a single entity. It is a set (collection) of programs. A programmer must write instructions in a prescribed sequence of the programming language being used for development so that the computer system becomes capable of successfully performing the desired task. Thus, we can say that a **Program** is a set of Instructions that the computer executes.

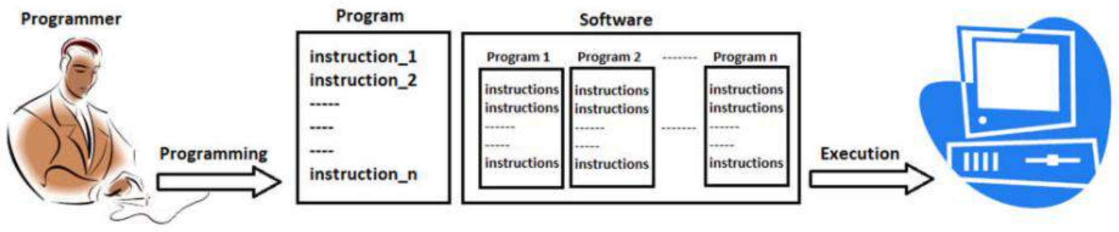


Fig. 6.1 Program, Programmer, Programming and Software

The process of writing system program is known as **System programming** and the programmer for the same is called System Programmer whereas writing application program is known as **application programming** and the programmer for the same is called application programmer.

6.3 PROGRAMMING LANGUAGES

The programming languages are similar to natural languages which are used in our daily life such as Punjabi, Hindi and English etc. As we use natural languages for communication purposes, similarly computer programming languages are used to communicate with the computer systems and to make computers work as desired through softwares.

System programs (Example: Operating Systems) are designed to control & operate the input/output devices, memory, processor etc. To write system program, such as operating system, programmer needs to control the hardware components of computer system. It is possible only if the programmer knows the internal architecture of hardware components. Therefore, system programming is the task of skilled programmers that have a detailed knowledge of the hardware components of the computer system. Machine, Assembly and C languages are widely used to develop system programs.

Application programs are developed to perform a particular task or to solve a particular problem. For example: student management system, library management system, payroll system, inventory control system, word processors, spread sheets, graphics software etc are application programs. Application programmer does not need to possess in-depth hardware knowledge. The most popular application programming languages are PYTHON, COBOL, FORTRAN, BASIC, PASCAL, C, C++, JAVA etc.

6.3.1 Types of Programming Languages

In this era, hundreds of programming languages are available. These languages are categorised on the basis of their ability to develop different kinds of software. Some of the programming languages are best for writing system software while some others are best suitable for writing application software or mobile applications:

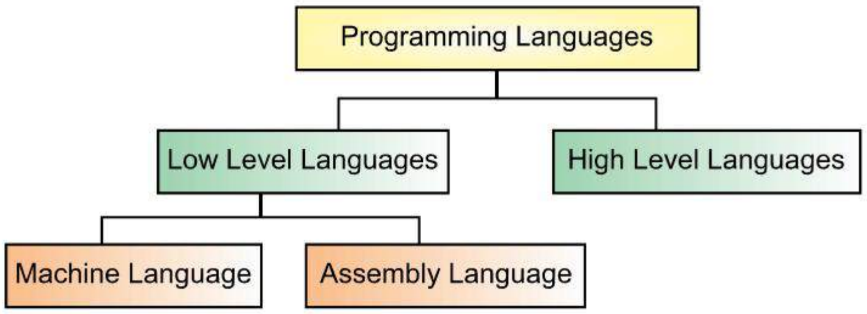


Fig. 6.2 Types of Programming Languages

For designing and developing system programs, low level programming languages are used while for developing application-programs, many high level programming languages have been designed. Let's know more about different type of programming languages:

A. Low Level Languages : Machine language and assembly languages are called low level languages. These programming languages are close to computer hardware and have more direct access to the features of the hardware. These are used to develop drivers, high performance code, kernels for operating systems etc. A detailed explanation of low-level languages is given below:

a. **Machine Language :** Machine language is also known as **Binary Language**. It is considered as the first generation of computer programming languages. Machine language is the fundamental language for computer systems because this language is directly understood by the computer hardware. Unlike high level programming languages, there is no need for translation of machine language code to make it understandable by the computer. This language consists of only two binary digits - 0 and 1.

Every instruction in machine language consists of two parts: Opcode and operand, as shown in the diagram below:

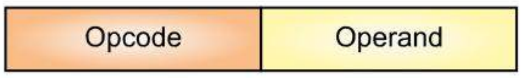


Fig. 6.3 Instruction Format in Machine Language

Here, Opcode is the Operation Code and Operand is the Operation Address. The first part - **Operation code** is the command which tells the computer what operation is going to be performed. The second part **- Operation Address** is the memory address which tells the computer where to find the data for the operation to be performed.

Because the instruction codes are written using binary digits 0 and 1 only, so it becomes difficult to remember the machine instruction codes in binary format. There are many advantages and disadvantages of using machine language some of which are explained below:

Advantages of Machine Language:

- Binary format instructions are **directly understood** by computer without any translation.
- Machine instructions are **executed fast** because these are executed directly without any translation.

Disadvantages of Machine Language:

- It is **difficult to remember** the machine instruction codes as they are made up of complex combinations of binary digits 0 and 1.
- It is the most **difficult process to find errors** in the machine instruction codes.
- Highest level of knowledge of **low-level internal details** of hardware is required for programming in machine language.
- Programs developed in machine language are **machine dependent** because machine instructions are written according to the underlying architecture of computer system. So, these instructions are machine specific which cannot be executed on the computer systems having different architecture.

b. **Assembly Language :** This language is also known as **Symbolic Language** because symbolic names of instructions are used instead of binary codes. This language is considered as second generation of computer programming languages. The major benefit of assembly language as compared to machine language is that it reduces coding time and the amount of information the programmer has to remember. The symbolic names of instructions can be easily remembered therefore it also becomes easy to find errors in the program and to modify it as compared to machine language.

Despite of these benefits, programming in assembly language still requires in-depth technical knowledge of hardware. So, programmer must be aware with the machine architecture for programming in assembly language. Due to this hurdle, programs written in assembly language are still machine-dependent. These programs cannot be executed on other machines having different architectures.

Symbolic names used for operation codes in Assembly Language are called **Mnemonic Codes.** For example: the codes for addition, subtraction, multiplication, and division operation are ADD, SUB, MUL and DIV respectively in Assembly language. These codes are the examples of Mnemonic codes.

Now the question arises how do computers execute the assembly language code because computers can execute instructions only in binary format. In order to execute an assembly language code on a computer, it must be translated into equivalent machine understandable code. For this translation, a translator program, named Assembler, is used. Assembler is a language translator program which translates the assembly language code into equivalent machine code. In the following sections, we will study in detail about the assemblers.

Advantages of Assembly Language:

- It is easy to learn and remember the codes of assembly language because it uses English like codes instead of binary digits as compared to machine language.
- Finding and correcting errors in the assembly language program is easy when compared to machine (binary) language.
- Assembly language programs have the equivalent efficiency of the machine language programs.

Disadvantages of Assembly Language:

- Knowledge of low level internal details of hardware is required for programming in assembly language. Therefore hardware technical skills are required for the programmer to do programming in assembly language.
- Programs developed in Assemblylanguage are machine dependent because assembly instructions are written according to the underlying architecture of computer system. So these instructions are machine specific which cannot be executed on the computer systems having different architecture.

B. High Level Languages : The primary objective of developing high level languages is that these languages facilitate a large number of people to build programs or software without the need to know the internal low level details of computer system hardware. These languages are designed to be machine-independent.

High level languages are English like languages. These languages use simple & special characters and numbers for programming. Therefore, these languages make it easy for common people to learn and write computer programs. An instruction written in high level language is usually called a **Statement**. Each high level language has its own rules for writing program instructions. These rules are called Syntax of the language. Some of the commonly used High Level Languages are: PYTHON, BASIC, COBOL, FORTRAN, PASCAL, C, C++, JAVA, C SHARP etc.

Similar to Assembly Language, high level languages cannot be directly understood by computer systems. Language translators are required for translating them into machine understandable format. There are two approaches for translating high level languages into machine code: first is via **Compiler** and other is **via Interpreter**. Each high level language has its own translator program. We cannot translate a program written in one specific high level language with the compiler of some other specific language. For example, we cannot compile C program using COBOL compiler or vice-versa.

Some of the common categories of high level languages are discussed below:

- **Procedural or Procedure Oriented Languages :** Procedural languages are considered as the **Third Generation of Programming Languages** (3GLs). In procedural languages, a program can be written by dividing it into small procedures or subroutines.

Each procedure contains a series of instructions for performing a specific task. Procedures can be re-used in the program at different places as required.These languages are designed to express the logic of a problem to be solved. The order of program instructions is very important in these languages.Some popular Procedural languages are FORTRAN, COBOL, Pascal, C language etc.

- **Problem-Oriented or Non-Procedural Languages :** Problem oriented languages are also known as Non-Procedural languages. These languages are considered as the **Fourth Generation of Programming Languages (4GL)**. These languages have simple, English-like syntax rules and they are commonly used to access databases. It allows the users to specify what the output should be instead of specifying each step one after another to perform a task. It means there is no need to describe all the details of how the data should be manipulated to produce the result. This is one step ahead from third generation programming languages. These languages provide the user-friendly program development tools to write instructions.Using these languages, user writes the program using application generator that allows data to be entered into the database. The program prompts the user to enter the needed data and then it checks the data for its validity. Examples of problem oriented languages are: SQL (Structure Query Languages), Visual Basic, C# etc. The objectives of these languages are to increase the speed of developing programs and reduce errors while writing programs.
- **Object-Oriented Programming Languages :** The Object-Oriented programming concept was introduced in the late 1960s, but now it has become the most popular approach to develop software.In these programming languages, a problem can be solved by dividing it into a number of objects. Object-Oriented languages support the concept of object, class, encapsulation, data hiding, inheritance and polymorphism etc.Now-a-days, most popular and commonly used Object-Oriented programming (OOPs) languages are C++ and Java.
- **Logic-Oriented languages :** These languages use logic programming paradigms as the design approach for solving various computational problems. Any program written in a logic programming language is a set of sentences in logical form. These sentences express facts and rules about some problem domain. Major logic programming language families include Prolog, Answer Set Programming (ASP) and Datalog. In all of these languages, rules are written in the form of clauses. Such languages are very beneficial in the field of Artificial Intelligence and Robotics.

Advantages of High Level Languages : Some of the common advantages of high level languages are given below:

- High Level languages are easy to learn and understand as compared to low level languages. It is because the programs written in these languages are similar to English-Like statements.

- The errors in a high level language program can be easily detected and removed. All the syntax errors are detected and removed during the compilation process of the program.
- These languages provide a large number of built-in functions that can be used to perform specific task during programming which results in huge time saving i.e. much faster development.
- Programs written in high level language are machine independent. A program written for one type of computer architecture can be executed on another type of computer architecture with little or no changes.

Disadvantages of High Level Languages : Some of the common disadvantages of high level languages are given below:

- A program written in high level languages has lower efficiency & speed as compared to equivalent programs written in low level languages.
- Programs written in high level languages require more time and memory space for execution.
- High level languages are less flexible than low level languages because normally these languages do not have direct interaction with computer's hardware such as CPU, memory and registers.

6.4 LANGUAGE TRANSLATORS

Language translators are also called Language Processors. These are the system programs which are helpful to develop programs. Language translators are designed primarily to perform two main functions as described below:

- These are designed to translate source programs into machine's object code. **Source programs** may be written in Assembly Language or High Level languages while **object code** is a code that a computer CPU can understand without any translation.
- These translator programs are also designed to detect any syntax errors in the source program. Successful translation of source program into object program takes place only if the source program does not have any syntax errors in it.

Each language has its own translator program which can translate the program written only in that specific language. Assembler is a translator program which can translate the source program written in assembly language only. Similarly each high level language has its own translator program, known as Compiler and Interpreter. Some High level languages use Compiler (for example: C/C++ Language) while some other uses Interpreter (for example: BASIC language). But there exists also some languages which have both compiler and interpreter for different levels of translation, for example: JAVA is a language which has both compiler and interpreter. All these types of translator programs are discussed below:

6.4.1 Assembler

It is a language translator which converts assembly language program into machine-understandable format. The program written in assembly language is called Source Program. This source program cannot be directly understood by the computer system. That is why it must be translated into machine understandable format for the execution. It is the assembler which translates this assembly language source program into machine understandable program. The source program after translation (in machine understandable form) is called Object Program (Code).

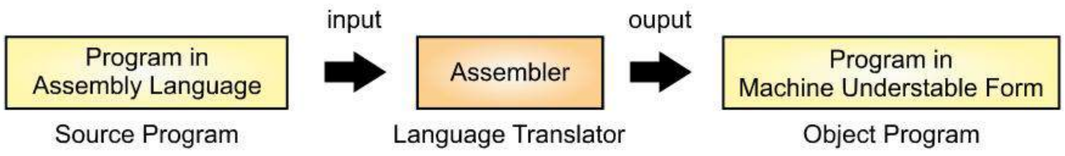


Fig. 6.4 Working of Assembler

6.4.2 Interpreter and Compiler

There are two types of language translators for High level languages i.e. 1. Interpreters 2. Compilers. These translators are used for translating source programs written in High Level languages into machine understandable form.

In the first approach, i.e. Interpreter, one statement of high level language program is taken at a time and it is translated into machine instruction which is executed immediately by the processor. It means no object program is saved in this approach of translation. Whenever we want to execute the program we have to translate the source program every time.

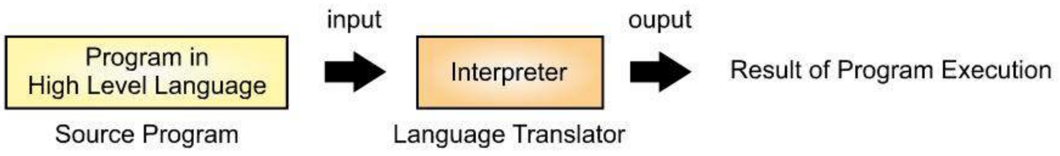


Fig. 6.5 Working of Interpreter

Interpreters do not require large memory space to translate and execute programs. The main disadvantage of interpreters is that they require same amount of time whenever we execute programs on a computer system because every statement of source program must be translated every time.

In the other approach, i.e. Compiler, all statements of the high level language program are taken at a time and they are translated into machine understandable form which is stored as Object Program in Memory. This object program is provided to computer system whenever program is executed. Compilers take more time to translate source program as compared to interpreter. But compiled object program runs much faster than the interpreted program.

Each High Level language has its own compiler. We cannot compile the source code of one language with the compiler of another language. For example FORTRAN compiler cannot compile the source code written in COBOL language and vice-versa.

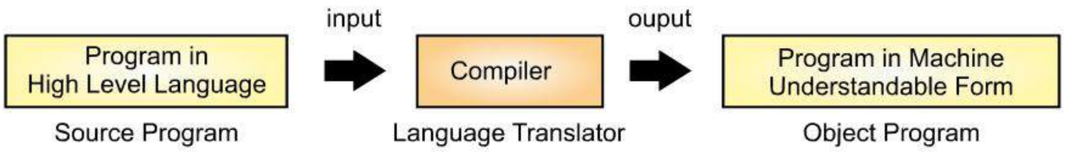

input
ouput
Program in High Level Language
Compiler
Program in Machine Understandable Form
Source Program
Language Translator
Object Program

Fig. 6.6 Working of Compiler

The difference between an interpreter and a compiler may be understood with the help of following analogy. Suppose we want to translate a speech from Tamil to Hindi. We can use two approaches to do this translation. In first approach, translator listens to a sentence in Tamil and immediately translates it into Hindi. In the second approach, the translator listens to the whole passage in Tamil and then gives the equivalent Hindi passage. An interpreter is similar to first approach of translation where sentence-by-sentence translation is carried out whereas compiler is similar to second approach where whole passage is translated in a single step.

6.5 PROGRAMMING PROCESS

We know that a computer needs a program to tell it what to do. Instructions in the program guide the computer how to solve the given problem. But developing a program is not a simple and easy task. A programmer has to go through a specific process for developing a program successfully. The steps involve in the programming process are listed below and are required to be followed in the same sequence:

1. Defining the problem to be solved
2. Plan the solution of the problem
3. Coding the solution in the high level language
4. Compile the program
5. Test and Debug the program
6. Documenting the program

These steps can be explained in detail as follows:

6.5.1 Defining the problem

It is the first step in the programming process. Before a programmer begins his task, he must need to know the extensive details of the problem to be solved through programming. The details of the problem should be provided to the programmer so that he gets a clear understanding of it. Analysis of the problem shows what the required inputs and outputs will be for the solution of problem. After having a clear understanding of the problem, programmer starts thinking about how to solve the given problem.

6.5.2 Planning the Solution

The next step after defining the problem is to prepare a detailed list of steps required to be carried out for solving the problem. We will take an example which shows why planning is required for solving the problem. A teacher asks the student to solve a specific mathematical problem and the student is not familiar with the steps involved in solving the problem. Thus, he would not be able to solve it. The same principle applies to writing computer programs also. A programmer cannot write the instructions for any program unless he understands how to solve the problem manually.

If a programmer knows the steps for solving the problem but while programming, if he applies the steps in the wrong sequence or he forgets to apply any of these steps, he will get a wrong output or even no output at all. Hence to write an effective program, a programmer has to write all instructions in the correct sequence. Therefore, to ensure that the instructions of the program are appropriate and are in the correct sequence, the programmer must plan the program before start writing it. For planning and defining the steps for the programs, programmers normally use Algorithms and Flow-Charts. Both of these approaches are explained below in detail:

6.5.2.1 Algorithms

The development of an algorithm is basic requirement to computer programming. It is a step-by-step description of how to solve a given problem. An algorithm consists of finite steps and a guaranteed result. When these steps are carried out as specified in the algorithm, it produces the required output. The construction of the algorithm requires creative thinking. Before developing a program, a programmer first set out the algorithm so that he can visualize the possible alternatives to solve the given problem. In order to qualify for the sequence of steps, to be called an algorithm, it should have the following features:

1. Each step should be accurate.
2. Each step should be unambiguous, i.e. it should not have dual meaning.
3. The inputs and outputs should be carefully specified.
4. Steps should not be repeated infinitely.
5. After executing the steps, the required output must be produced.

To gain a clear understanding of algorithms, let us consider some simple examples of defining algorithms.

Problem Statement 1 : Calculate and print multiplication of two numbers:

Algorithm-I:

```
Step1: Start
Step2: Read two numbers A and B.
Step3: Multiply A and B and store result in C.
Step4: Print C.
Step5: Stop.
```

Problem Statement 2: Write an algorithm to find whether a given number is Odd or Even.

Algorithm-II:

```
Setp1: Start
Step2: Read a Number A
Step3: Divide the number A by 2 and get remainder value
Step4: If remainder of the division is zero then
       Print "Number is Even"
       Else
       Print "Number is Odd"
Step5: Stop
```

Problem Statement 3: Write an algorithm to print "India is Great" 10 times:

Algorithm-III:

```
Step1: Start
Step2: Count = 1
Step3: While (Count<=10)
Step4: Print "India is great"
Step5: Count=Count + 1
       [End of While loop]
Step6: Stop
```

6.5.2.2 Flow Charts

Flow charts are the common ways to represent algorithms. These are also frequently used by the programmers for planning of the program. Programmers often find them very helpful for developing effective and correct programs.

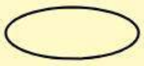
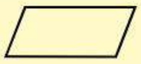
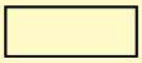
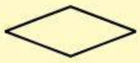
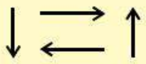
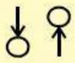
A flowchart is a pictorial representation of the algorithm. Programmers often draw flow charts to visually organize the sequence of steps defined in the algorithm. It uses different types of symbols/shapes to represent different types of instructions. The process of drawing a flowchart for an algorithm is known as flowcharting.

Normally, an algorithm is first represented as flowchart, and the flow chart is then expressed in a programming language to develop a computer program. The main advantage of this two-step approach in program development is that a programmer can more easily detect logical errors in the program logic because a flow chart shows the flow of operations in the pictorial form. Once the flow chart is ready, the programmer can concentrate only on coding of operations. This normally ensures an error-free program.

Experienced programmers, sometimes, write programs without drawing flowcharts. However, beginners should first draw a flowchart to effectively do the programming. Moreover, it is a good practice to have a flow chart along with the computer program. It proves to be very useful during testing of the program as well as during modifications in the program.

Before preparing flowcharts, we have to learn about the different symbols used in the flowcharts. Some of the basic symbols used in the flowcharts are:

Table: 6.1 Symbols used for flowcharts

Symbol	Symbol Name	Name	Description
	Ellipse/Oval	Terminal	It is used to start and terminate the flow chart
	Parallelogram	Input / Output	It is used for taking input data and giving output result
	Rectangle	Processing	It is used for performing computational operations
	Rhombus	Diamond	It is used when we have to choose one path from many paths
	Arrows	Flow Lines	It is used to represent the direction of flow in the flowchart
	Circles	Connectors	It is used to connect different parts of the flowchart.

Now let us consider some examples of flowcharts so that we get familiar with the concept of flowcharts:

a. Draw a flowchart to Calculate and print multiplication of two numbers

(Flowchart for Algorithm-I):

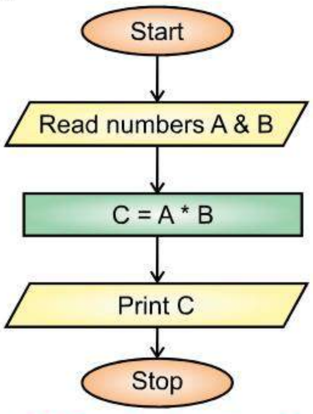


Fig. 6.7 Flowchart for Algorithm-I

b. Draw a flowchart to find whether a given number is odd or even

(Flowchart for Algorithm-II):

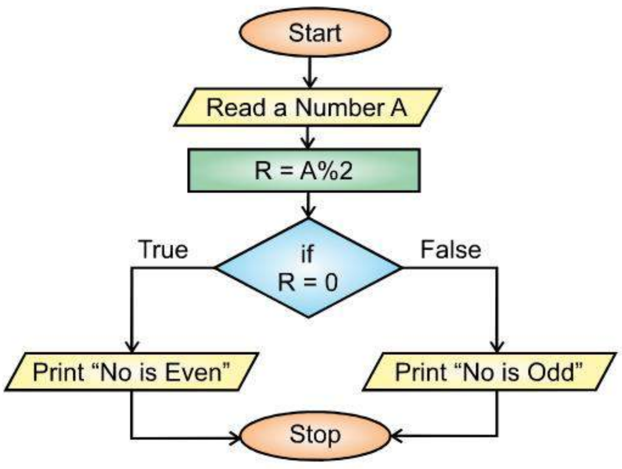


Fig. 6.8 Flowchart for Algorithm-II

c. Draw a flowchart to print "India is great" 10 times

(Flowchart for Algorithm-III):

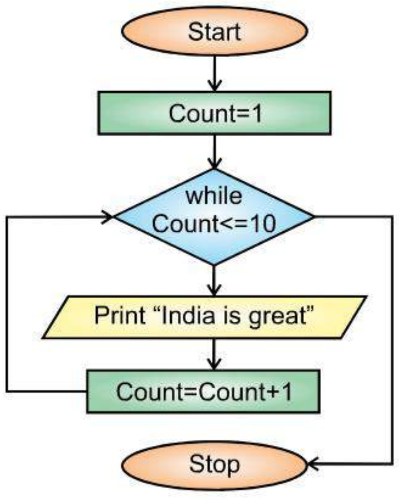


Fig. 6.9 Flowchart for Algorithm-III

Now, we have a clear understanding of how to plan the steps to solve the given problem. Once, inputs, outputs, algorithms and flowcharts have been clearly defined, the next step is to translate these steps into a program using High level programming languages.

6.5.3 Coding the Solution:

The sequence of operations defined in flow-chart can be converted into instruction using some High-Level programming language, such as C, C++, and PASCAL etc. Coding is the process of writing the instructions using programming language to make a program. This program file is known as source program or source code. This code is stored in a disk file. This program contains the logic or steps for solving the problem. For example, let us consider the following source code using C language for the Algorithm-I:

Source Code for Algorithm-I using High Level Language C:

```
#include<stdio.h>
void main( )                    //step1 of the Algorithm-I
{                               //step1 of the Algorithm-I
      int a, b, c;              //step1 of the Algorithm-I
      a=10;                     //step2 of the Algorithm-I
      b=20;                     //step2 of the Algorithm-I
      c=a * b;                  //step3 of the Algorithm-I
      printf("%d", c);          //step4 of the Algorithm-I
}                               //step5 of the Algorithm-I
```

6.5.4 Compile the program

After writing program code in High level language, we have to translate it into such a form that computer can understand & execute it because computer can understand instructions only in binary format.A compiler is a small program that translates the source program into a machine understandable form (i.e. object code). This conversion of source code into object code is known as Compilation. During compilation, compiler also scans the source program for syntax errors. If there are syntax errors in the program, compiler generates error messages. These errors must be corrected to generate the object code. Then the object code will be stored in a disk file. Whenever we want to execute the program, this object code is supplied to computer for execution.

6.5.5 Testing and Debugging the Program

Testing and debugging are important steps in the software development. Testing is a process which makes it sure that program (software) performs the intended task. Testing is a time-consuming task. As long as human beings make programs, the programs will have errors. These program errors are called Bugs. The process of detecting and correcting such bugs is called Debugging. Generally, two types of errors are found in the programs:

- **Syntax Errors :** These errors occur when we do not follow the rules or syntax of programming language being used. These types of errors are automatically detected by compilers during compilation process. A program cannot be successfully compiled until all of the syntax errors in the program are removed. Some examples of syntax errors in C language are: missing semicolon, variable not declared, un-terminated string, compound statement missing etc.
- **Logical Errors :** These errors occur when there are errors in the logic of the program. If our program has logic errors, though it will compile successfully but it may produce wrong result/output. Such types of errors cannot be detected by the compilers.

These errors are either traced out manually by the programmer or some debugging tools may be used to detect such errors. Programmer can detect any faulty logic by examining the output.

6.5.6 Documenting the Program

The final stage in the development process of a program is documentation. The term documentation means specifying the important information regarding the approach and logic applied in the program by the programmer. The documentation enables other programmers to understand the logic and purpose of the program. It is also helpful in the maintenance of the program.

Points To Remember

1. A Program is a set of instructions while a Software is a set of programs.
2. A Programmer is the person who writes the program code.
3. The Process of writing a program is called Programming
4. Machine language is directly understood by computer and consists of binary digits 0 and 1.
5. Assembly Language use mnemonic codes to write instructions in the program.
6. High Level languages use alphanumeric codes to write instructions in the program.
7. Assembler is a language translator which translates Assembly language source program into machine understandable format which is called Object Program Code.
8. Compiler is a language translator which translates High Level language source program into machine understandable format which is called Object code.
9. Algorithm is a set of finite steps to solve some specific problem.
10. Flowchart is the pictorial representation of the algorithm.
11. Writing the instructions to make a program using some computer language is called coding.
12. Finding and correcting errors in the program is called Debugging.

EXERCISE

Part-A

1. **Multiple Choice Questions**

   I. Set of instructions is called ________________.

   a. Group
   b. Software
   c. Program
   d. None of these

II. Which language is directly understood by computer without any translation?
   a. Procedure Oriented Language
   b. Machine Language
   c. Assembly Language
   d. High Level Language

III. Mnemonic codes & symbolic addresses are used in which programming language?
   a. Object Oriented Language   b. Non-Procedural Language
   c. Assembly Language   d. Machine Language

IV. Which translator does not save object code after translation of source program written in high level language?
   a. Translator   b. Compiler
   c. Assembler   d. Interpreter

V. Process of finding and correcting errors in a program is called ____________
   a. Compilation   b. Coding
   c. Debugging   d. Documentation

2. Fill in the Blanks:

I. A person who writes the program is called ____________________

II. Low level internal details of hardware are required for programming in __________

III. _________ is the pictorial representation of algorithm

IV. Process of translating source program written in high level language into object code is called ____________________

V. Those errors which are not detected by the compilers are called ____________ errors.

3. Write the Full form of following:

I. Opcode

II. Operand

III. 4GL

IV. SQL

V. OOP

Part-B

4. Short Answer Type Questions. (Write the answers in 4-5 lines)

I. What is Programming?

II. What are Procedure Oriented Programming Languages?

III. Write the names of different symbols used in flowcharts.

IV. Write the steps used in Programming Process.

V. What are Syntax Errors?

Part-C

5. Long Answer Type Questions. (Write the answers in 10-15 lines)

I. What are low level programming languages? Explain their advantages and disadvantages.

II. What are Language Translators? Explain any one translator in detail.

III. What is algorithm? Explain the different features that an algorithm should have.

IV. Explain different types of errors found in the computer programs.

Lab Activity

- Draw a chart which represents different categories of programming languages.
- Make symbols used in flow charts with the cardboard and label them.

CHAPTER - 7 INTRODUCTION TO C AND BASIC STRUCTURE OF C PROGRAM

OBJECTIVES OF THIS CHAPTER

7.1 INTRODUCTION AND HISTORY OF C

C is a general-purpose programming language. C can be used to develop any type of application programs. We can develop Business applications, Scientific applications etc. using it. It can also be used to develop System programs such as Operating Systems, Language Translators etc. Thus, we can say that it is useful for developing both types of software, i.e. System Software and Application Software.

In 1960, many computer programming languages were emerged like FORTRAN, COBOL etc. But, these languages were used for specific applications. For example: FORTRAN (FORmula TRANslation) was used to develop Scientific Applications only, while COBOL (COmmon Business Oriented Language) was used to develop Business Applications only. Later an international committee was set up to develop a general purpose programming language. As a result, in 1963, Combined Programming Language (CPL) was developed at Cambridge University. It was hard to learn and difficult to implement. So, later in 1967, BCPL (Basic Combined Programming Language) was developed by Martin Richards at Cambridge University. Similarly, B language was developed by Ken Thompson at AT & T's Bell Laboratory in 1970.

But both of these languages, BCPL and B, are type-less languages and too specific. Finally, in 1972, using many important ideas of BCPL and B languages, Language C was developed by Dennis Ritchie at AT & T's Bell Laboratories of USA and it became a general-purpose programming language.

7.2 WHY C-LANGUAGE IS CALLED MIDDLE LEVEL LANGUAGE?

All the programming languages can be divided into two types: Low Level Programming Languages and High Level Programming Languages.

Low Level Languages are known as Machine-Oriented Languages because programmers have to concentrate more on the architecture of underlying hardware (machine) rather than on the logic of the program to be solved. It is difficult to learn & use these programming languages. These languages are used to write machine dependent system programs such as operating system, translators and so-on. Examples of these languages are Assembly Language and Machine Language.

High Level Languages are known as Problem-Oriented Languages because programmers have to concentrate more on the logic of the program rather than the hardware architecture of the machine. It is easy to learn and use these programming languages. These languages are used to write any type of application programs, such as business applications, scientific applications etc. Examples of these languages are FORTRAN, COBOL, Pascal etc.

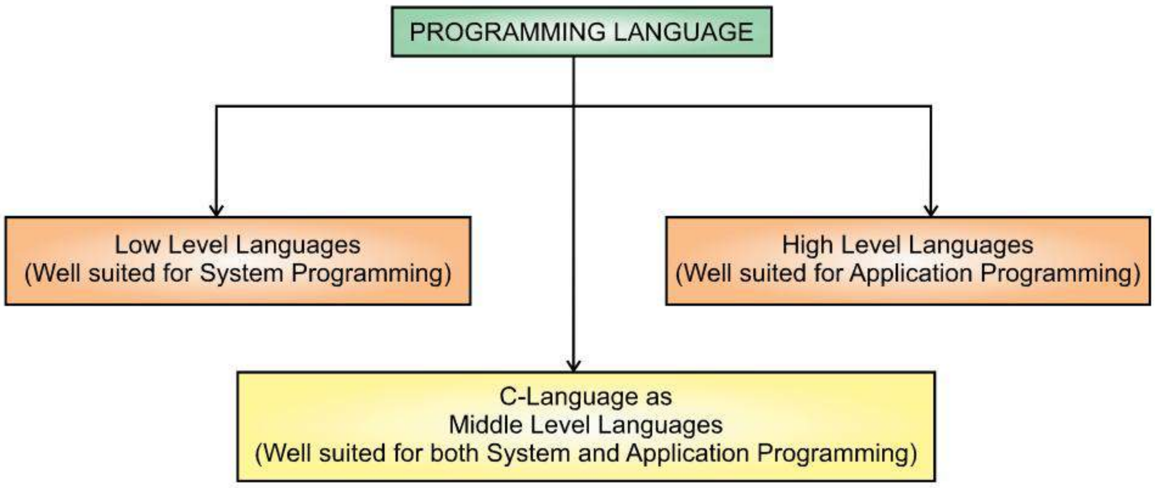

PROGRAMMING LANGUAGE
Low Level Languages (Well suited for System Programming)
High Level Languages (Well suited for Application Programming)
C-Language as Middle Level Languages (Well suited for both System and Application Programming)

Fig. 7.1 Why C is considered specially as a Middle Level Lanaguage

C Language has the capabilities of these both types of programming languages, i.e. Low Level and High Level programming languages. It means C language is well suited for writing both system programs and application programs. Hence, C is a programming language which stands in between the above two types of programming languages. So C is called Middle Level Language.

Though, there is no special category of Middle level Programming language, it is only due to the special features of C language why it is designated as Middle Level Programming Language.

7.3 INTRODUCTION TO C EDITORS & IDEs

Editors are the programs that are useful for writing source codes of languages. There are many programming languages which have their own editors to write programs, such as C, C++ etc. But some programming languages do not have their own editor for writing programs, such as Java. Java Programming can be written in any simple text editor program. After writing source programs in programming languages, they must be compiled in order to execute by the computer systems. These source programs must be compiled by their respective Compilers.

Many programming languages also support the IDEs. IDE stands for Integrated Development Environment. **Integrated Development Environment (IDE)** can be defined as software that gives its users an environment for writing programs (Editor), along with tools for compiling, executing, testing and debugging the programs. Usually,Modern IDE Software are very user-friendly. They provide an easy-to-use interface. These IDE interfaces provide suggestions for syntax to programmers, the graphical user interface having buttons and menus to interact with, editors and plugins and many other features.

There are many IDEs that are available for programming in C. Some of these common IDEs are given below:

- Turbo C
- Code Blocks
- Eclipse
- Code Lite
- Net Beans
- Dev C++ etc….

Turbo C is one of the oldest IDE for programming in the C language. It was developed by Borland and first introduced in 1987. At the time, Turbo C was known for its compact size, comprehensive manual, fast compile speed and low price. But now a days, Turbo C was getting out-dated due to the advancements in the newer Operating Systems, such as Windows 7,

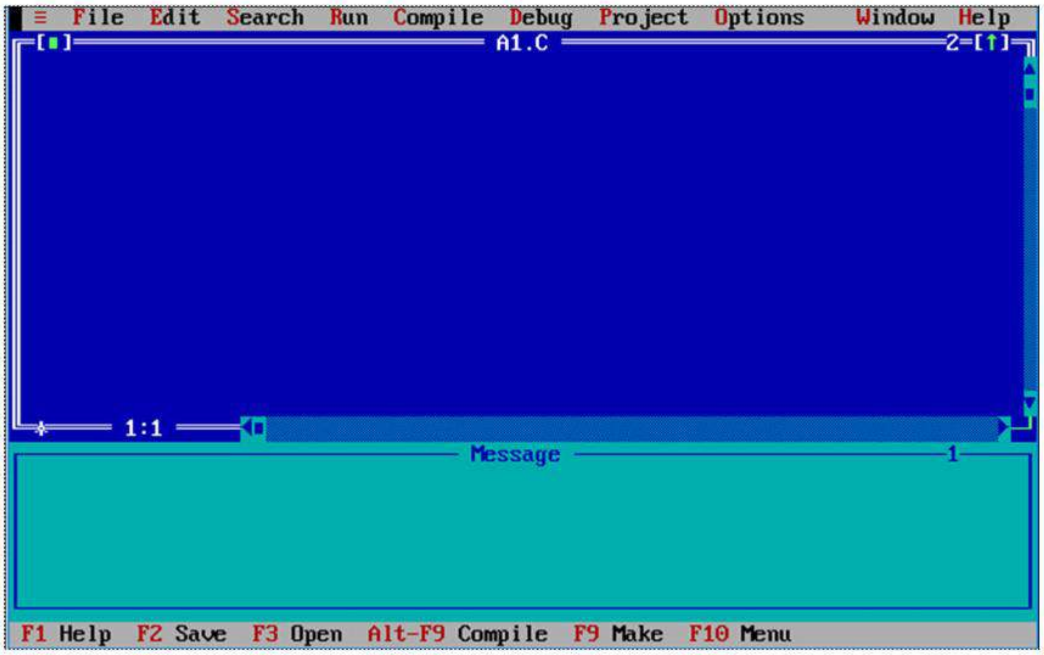

≡ File Edit Search Run Compile Debug Project Options Window Help
A1.C
2
1:1
Message
1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Fig: 7.2 Turbo C/C++ IDE Interface

Windows 8 and Windows 10. In these operating systems, programmers have been facing many issues while they make C programs in Turbo C due to compatibility issues. Many modern IDEs have been developed for programming in C which works well in modern operating systems.

Code::Blocks is also a popular modern IDE for programming in C/C++. It is a free (Open Source), highly extensible and configurable, cross-platform C and C++ IDE. It offers programmers the most demanded and ideal features. It delivers a consistent user interface and feel. Most importantly, we can extend its functionality by using plugins developed by users.

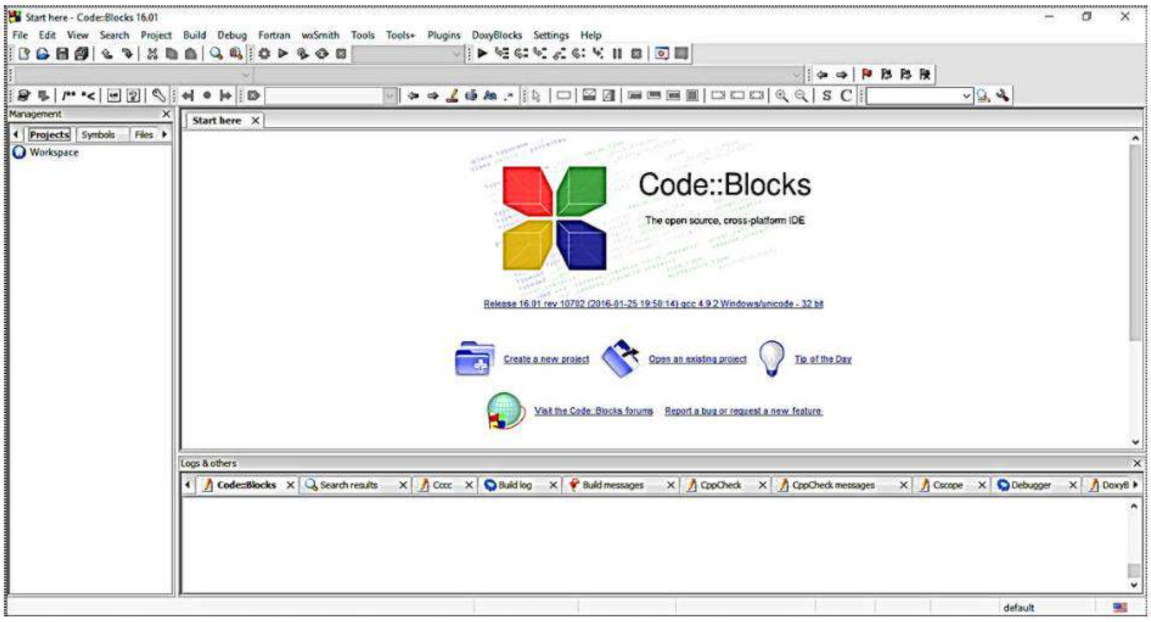

Start here - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
Start here
Code::Blocks
The open source, cross-platform IDE
Release 16.01 rev 10702 (2016-01-25 19:50:14) gcc 4.9.2 Windows/unicode - 32 bit
Create a new project
Open an existing project
Tip of the Day
Visit the Code::Blocks forums
Report a bug or request a new feature
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck
CppCheck messages
Cscope
Debugger
DoxyB
default

Fig: 7.3 Code::Blocks IDE interface

Code::Blocks provides many features to programmers which are given below:

- It works on Windows, Linux and Mac OS X as well.
- It supports Compiling, Debugging, Code Coverage, Profiling, Auto-completion of code
- It provides support for multiple compilers including GCC, mingw, clang, Borland C++ 5.5, etc.
- It is very fast, no need for makefiles
- It provide facilities of debugging by supporting full breakpoints including code breakpoints, data breakpoints, breakpoint conditions plus many moredisplay local functions symbols and arguments
- It is designed to be fully configurable and extensible with its plugins.
- It provides workspace that supports combining of projects
- It supports the feature of Custom memory dump and syntax highlighting
- It also provides support for code analysis, etc.

Code::Blocks IDE can be downloaded from the following link:

http://www.codeblocks.org/downloads

Any one of the C/C++ IDE, that are available in market including modern or old ones, can be used for writing C programs of this book.

7.4 CREATING AND EXECUTING C PROGRAMS

Any C program development involves the following steps:

1. Design the program
2. Write the program using any Text Editor or IDE supporting C Language
3. Save the program by giving filename with extension .c
4. Compile the program
5. It there are any errors in the program, then correct it and repeat the step 4
6. Execute the program.
7. View the Output Window

7.5 GETTING STARTED WITH C

For communication with human beings, we use natural languages like English, Hindi, and Punjabi etc. But to communicate with computers, we can use only those languages which a computer system can understand directly or indirectly. These languages are called Programming Languages. C is a programming language. As we have to learn any natural language before using it, similarly we must also have to learn programming languages like C for communication with the computers. Learning programing languages are very similar to learn any natural language like Hindi, Punjabi etc.

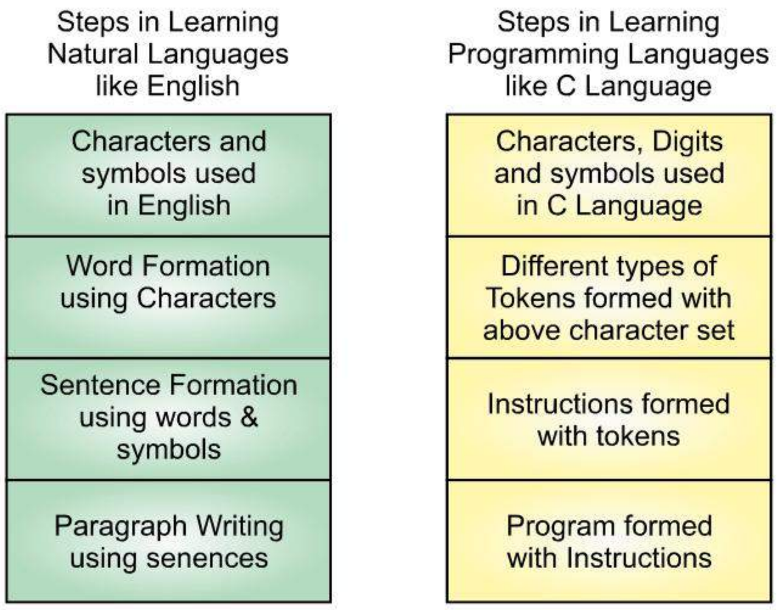

Steps in Learning Natural Languages like English
Characters and symbols used in English
Word Formation using Characters
Sentence Formation using words & symbols
Paragraph Writing using senences
Steps in Learning Programming Languages like C Language
Characters, Digits and symbols used in C Language
Different types of Tokens formed with above character set
Instructions formed with tokens
Program formed with Instructions

Fig. 7.4 Analogy between learning Natural and Computer Languages

There is a very close analogy between learning natural languages (like English, Hindi, Punjabi, etc.) and computer programming languages (like C, C++, JAVA, etc.). The classical method of learning any natural language (for example English) is to learn the alphabets or characters used in the language, then learn to combine these characters to form words, which in turn are combined to form sentences and sentences are combined to form paragraphs. Learning any computer programming language, for example C, is similar and much easier.

Therefore, instead of straight-away learning how to write programs, we must first know what characters, numbers, symbols are used in C, then using these how different tokens are constructed, finally how these tokens are combined to form instructions. A group of instructions would be combined later on to form a program. This analogy of learning natural and computer programming languages can be represented using figure 7.4.

7.6 CHARACTER SET

It is the first step for learning any language whether it is Natural or a Computer Programming Language. We can learn any language only if we know which characters and symbols are allowed in that language. So, before learning C, we must be familiar with the characters and symbols used in the C language. C language supports ASCII (American Standard Code for Information Interchange) character set. The ASCII set of characters includes the following characters and symbols:

- Upper and Lower case Alphabets (A to Z, a to z)
- Digits (0 to 9)
- Special Symbols (all the printable symbols present on the keyboards, For Example: ! @ # $ % ^ & * ( ) - _ + = { } [ ] ; : ' " < , > . ? / | \ etc.
- Some Non-printable characters, For example: new-line, horizontal-tab etc.

The set of these above characters and symbols is called the Character Set of C Language.

7.7 TOKENS

Tokens are like words and punctuation-marks in English language. In any programming language, like C, a program is made up of tokens. Tokens are the smallest individual units in a program. A C-program can have five types of tokens as shown below:

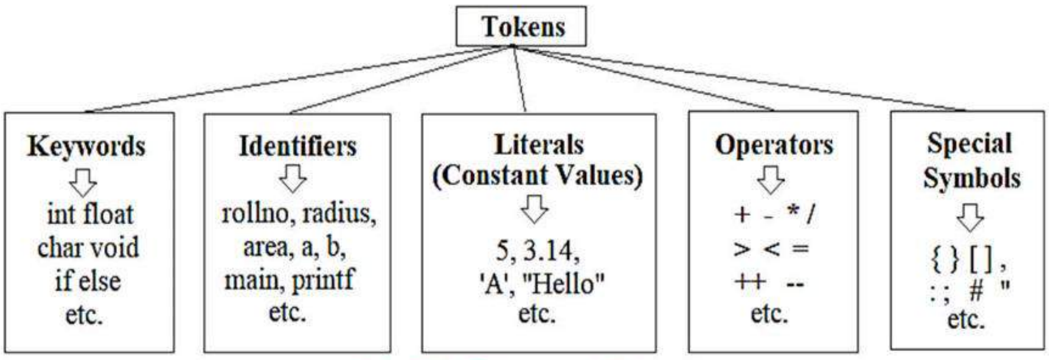

Tokens
Keywords
int float char void if else etc.
Identifiers
rollno, radius, area, a, b, main, printf etc.
Literals (Constant Values)
5, 3.14, 'A', "Hello" etc.
Operators
+ - * / > < = ++ -- etc.
Special Symbols
{ } [ ] , : ; # " etc.

Fig. 7.5 C Tokens with Examples

Introduction of these tokens with suitable examples is given below:

7.7.1 Keywords

Keywords are also called the Reserve Words. These words are predefined in the Compiler of C Language. Meaning of these words is predefined. They are used for special purposes for which they had been defined. We cannot change their meaning. In Turbo C, these words are shown in white colour while in code::blocks these words are shown in blue colour. C language has 32 keywords but some new compilers introduce some more keywords to C Language. Following is the list of 32 keywords that are supported by all the compilers of C language:

Table : 7.1- List of C Keywords

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

These keywords can be used wherever they are required in the program. All the keywords in C programs must be written in lower-case only. As C is case-sensitive language, so if we write these keywords in upper-case in a program, it will display compile errors. (A case sensitive language is a language which considers lower case and upper case alphabets as different elements.)

7.7.2 Identifiers

Identifiers are the name given to elements of program such as variables, constants, arrays, functions, structures etc. Every program element must be named to distinguish it from other elements. The name assigned to the elements should be meaningful because it facilitates easy understanding of the program elements. After naming the program elements, they can be identified by their name. For defining names of program elements, some naming rules must be followed during writing of C programs. These naming rules are given below:

- Identifier name must begin with an alphabet or underscore (_) symbol. It must not begin with a digit. For example: Identifier name 5star will be wrong because it begins with a digit 5 which is not allowed.
- No special character, except underscore (_), is allowed for defining name of program element. For example: if we define a name: roll#, it will be considered wrong and compiler will show an error because we are using # in the name which is a special character.
- Two consecutive underscores are not allowed in the identifier name. For example: Identifier name roll__ no will be wrong because we use two consecutive underscores in the name which is not allowed in c programs.
- In some C language compilers (Turbo C), length of identifiers is restricted to 31 characters. It means identifier name can have maximum 31 characters and minimum

1 character. If we use more than 31 characters in the name than it will not show any error, instead the compiler considers only first 31 characters as identifier name and ignore the remaining characters.

- Keywords cannot be used as identifier names. For example: int cannot be used for defining identifier name because it is a keyword and has a special meaning.
- Identifier names are case-sensitive. It means identifier names in lower-case and upper-case are considered distinct. So, we should take care of lower and upper cases while defining names. For example: roll and ROLL will be considered two different identifiers in C programs.
- Blank spaces in the identifier names are not allowed. For example: identifier name roll no will be wrong because it contains a blank space between the words roll and no.

7.7.3 Literals

Literals are also called constant values. These are the fixed values that are normally assigned to variables in the C-programs. These fixed or constant values can be categorized into two categories: Numeric and Character constant values as shown in the following diagram with suitable examples:

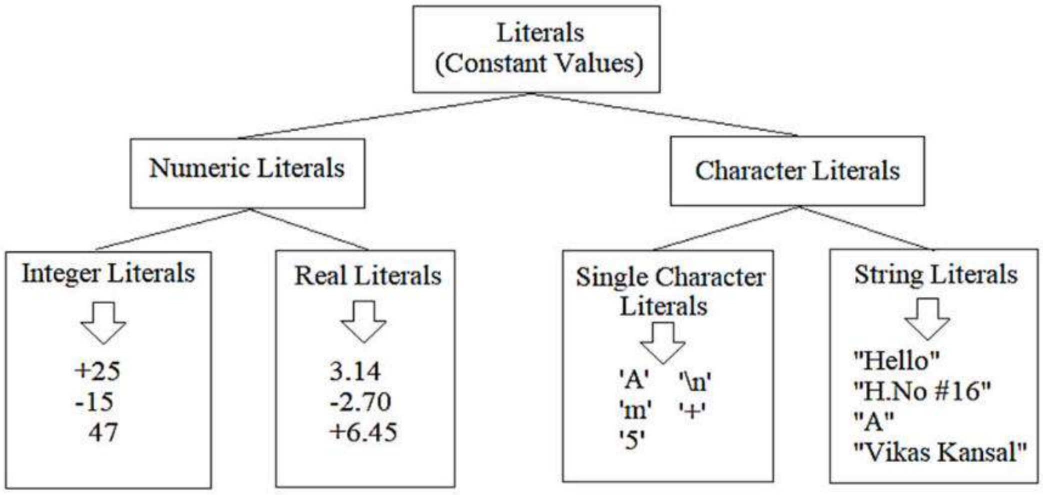


Fig. 7.6 Different Types of Literals (Constant Values)

7.7.3.1 Numeric Literals : These are the numeric values, which can be used for numeric calculations. There are two types of numeric literals:

- **Integer Literals :** These are literals without fractional(decimal) part. It consists of digits from 0 to 9 along with positive (+) or negative (-) sign. If number does not have any sign, it is considered the positive integer literal. For example: 56, +26, -96 etc. are the integer literals.
- **Real Literals :** These are literals with fractional(decimal) part. It consists of digits from 0 to 9 along with positive (+) or negative (-) sign. It also has a decimal point (.) which separates the integral and fractional part of the real number. If integral part

does not have any sign, it is considered the positive real literal. For example: 3.14, +256.5896, -96.14. 36.00 etc. are the real literals.

7.7.3.2 Character Literals : These are the character values, which usually do not involve in the calculations. There are two types of character literals:

- **Single Character Literals :** These are the literals which have a single character enclosed in the single quotes. More than one printable character is not allowed in these literals. Non-printable characters also come in this category of constant literals though two symbols are used in these characters, for example: new line character (\n - backslash and a letter n), but yet they are considered to be a single character. Examples of single character literals are: 'A', 'g', '7', '+', '$', '\n', '\t' etc. Values 'AB', '45' are the invalid examples of single character literals because more than one character is enclosed in single quotes which are not allowed in the single character constant.
- **String Literals :** These are the literals which have one or more characters enclosed in the double quotes. These literals may have the combination of letters, digits, special symbols, and blank space. Examples of these literals are: "V. Kansal", "A", "House#196", "1829" etc.

7.7.4 Operators

Operators are the symbols which are used to perform some mathematical or logical operation. Operators are used to manipulate values/variables in the program. These values/ variables are called operands. C supports a rich set of built-in operators. All these operators can be classified into three broad categories: unary, binary, and ternary. A detailed explanation has been given on operators in the next chapter of this book.

7.7.5 Special Symbols

These are the symbols used as punctuation marks. Each symbol has its different speciality in program. Each symbol is used to denote something special in the program. For example: semicolon (;) is used to terminate the statement, comma (,) is used as a separator, parenthesis ( ) are used to represent the functions, square brackets [ ] are used to denote arrays, Braces { } are used for grouping the statements etc.

7.8 VARIABLES AND CONSTANTS

These both are the important program elements which are used to store values in the program. Both are given a name and type of value to be stored in them. But there is a little difference between them. Variables allow us to change their values during execution time while constant do not. It means constants have fixed values while variables can have changeable values during program execution. C is a **strictly typed language**. It means we must declare variables and constants before using them in the program. If we use variables or constants without declaring them, compiler will generate a syntax error: "Variable not declared". So, each variable or constant must be declared in the program. Following are the syntax rules for declaring variables and constants in the program:

Syntax of variable declaration:

data_type variable_name;

Here, **data_type** tells the compiler what type of value is going to be stored in the variable, and **variable_ name** is the valid identifier which tells the compiler about the name of the variable which will be used to refer to the value stored in the variable. Consider the following example of variable declaration:

int roll_no;

Here, int represents the integer data type while roll_no is the identifier name which is used to refer to the variable. It means the variable roll_no can hold only the integer values. It is not given any value, so a garbage value will be stored in it by default. If we want to assign it a value during its declaration, then it will be called variable initialization.

For example:

int roll_no=5;

This assigned value can be changed later at any time because a variable allows us to change its values at any time during execution.

Syntax of constant declaration:

A variable can be made constant if we put a keyword const before the variable declaration and assign it a fixed value at the time of declaration. Consider the following syntax rule:

const data_type constant_name = value;

Here, const is a keyword which tells the compiler that the given value cannot be changed during program execution. Consider the following example of defining constant:

const float pi=3.14;

In this example, we define a constant value 3.14 which is of real type. It is given a name pi. The keyword float tells that the value will be of real type and const make it a fixed value that cannot be changed during execution time. If we try to change its value, compiler will generate a compile error.

7.9 DATA TYPES

Data type defines which type of data will be stored in the program elements, such as variables, constant, arrays etc. Data types define a specific type or range of values for the variables or other program elements. C is a strongly typed language therefore data type of all the variables must be declared during declaration time.

C supports many different types of data, each of which may be represented differently within the computer's memory. Storage representation of data in memory varies from machine to machine and compiler to compiler. For example: in Turbo C, int data type takes 2 bytes of memory while in Code::Blocks it takes 4 bytes of memory. Following table shows the list of primitive or basic data types available in the Standard C language:

Table 7.2: Primitive Data types available in C

Keyword	Description	Memory Requirement	Range of values	Format
char	Used to store single byte/character data	1 byte	-128 to 127	%c
int	Used to store integer type data	2 byte	-32768 to +32767	%d
float	Used to store single precision floating values	4 byte	3.4×10^{-38} to $3.4\times10^{+38}$	%f
double	Used to store double precision floating values	8 byte	1.7×10^{-308} to $1.7\times10^{+308}$	%lf
void	Used with functions which do not return any value	-	-	-

7.10 HEADER FILES IN C

C provides a huge library of predefined functions to perform various types of tasks in the programs. All these functions are called library functions. To organize these functions, all the functions are logically grouped into separate files. These files are termed as header files. All the header files have **.h** extension. To use any of these functions in our program, we have to include these header files in our program. This inclusion is done by using the **pre-processor directive #include.** In C, pre-processors begin with the # symbol. Consider the following example:

#include<stdio.h>

In this example, stdio.h is a header file and #include is a pre-processor directive. Using this example, we can use any of the function defined in the stdio.h header file in our program.

There are many header files in C. The header files that will be used in a program, depends solely on our requirement in the program. Following are some of the common header files that are used in the C programs:

- **The header file stdio.h :** The full name of this header file is standard input output header file. This file contains the functions that can be used for input and output from the standard input/output devices. For Example: scanf( ) and printf( ) functions
- **The header file conio.h :** The full name of this header file is console input output header file. Console is screen where our program executes. This file contains the functions that are used for console during input/output. For example: clrscr( ) and getch( ) functions
- **The header file math.h :** This file contains mathematical and trigonometric functions that we can use in our programs for various mathematical operations. For example: sqrt( ), pow( ), sin( ), cos() etc.

- **The header file string.h :** This file contains functions that can be used for string manipulation operations. For example: strlen( ), strcpy( ), strupr( ), strlwr( ), strcmp( ) etc.

To use any library function, we must use respective header file with pre-processor directive #include in our program.

7.11 INPUT AND OUTPUT STATEMENTS IN C

Input and Output statements provide interaction between program and users. Using input statements, user provides input to program and using output statements output is displayed by the program to user.

All the input and output operations in C program are carried out using pre-defined functions. Although, there are many formatted and unformatted input/output functions provided by the C library, but normally, this input/output operation is carried out using scanf( ) and printf( ) formatted functions in C programs. Consider the following diagram which shows the purpose of input and output statements in the programs:

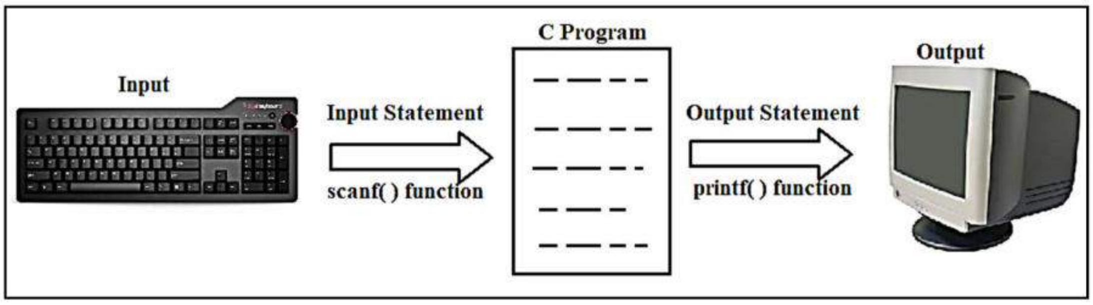


Fig. 7.7 Purpose of Input and Output Statements in a Program

Now, we shall discuss these two commonly used input and output library functions in c programs, i.e. scanf( ) and printf( ) function:

The scanf( ) function:

Input data can be entered into the program from a standard input device by using C library function scanf( ). This function can be used to enter any combination of numerical values, single characters and strings. The general syntax for using functions is as follows:

```
scanf("format string", &arg1, &arg2, …….., &argn);
```

Here, format string refers to a string that contains certain format codes depending on the data type of arguments, and arg1, arg2, …….., argn are the arguments that represent the individual input data items. Normally, arguments are the variables preceded by address operator &. This address operator & specify the address of the variable where the data will be stored. Format string and all the arguments must be separated by the comma operator. To have better understanding of the concept, consider the following example:

```
int a;
float b;
scanf("%d%f",&a,&b);
```

Here, "%d%f" is a format string which specify the type of value to receive from the keyboard. The format string %d is used to input the integer value for variable a, and %f is used to input float value for variable b. Here, a, and b are the two arguments representing variables which are getting values from user. The symbol & before these variables a, and b specifies that the integer and float values will be stored at the memory addresses allotted to a, and b.

The printf( ) function : The printf( ) function is used to display information on the monitor in C programs. This function is normally used to display simple text messages on the monitor (output) screen. The general syntax of this function for displaying simple text is as follows:

```
printf("simple text message");
```

For example:

```
printf("Hello from C Language");
```

This function can also be used to show any numerical, single character and string values stored in variables on the monitor (output) screen. The general syntax for displaying values stored in the variables is as follows:

```
printf("format string", arg1, arg2, ........, argn);
```

Here, format string refers to a string that contains certain format codes depending on the data type of arguments, and arg1, arg2, ..., argn are the arguments that represent the individual output data items. Here, the printf( ) functions read values of arguments from memory and display them on the monitor screen. Format string and all the arguments must be separated by the comma operator. To have better understanding of the concept, consider the following example:

```
int a=56;
float b=3.14;
printf("%d%f",a,b);
```

Here, "%d%f" is a format string which specify the type of value to display at the monitor screen. The format string %d is used to display the integer value stored in variable a, and %f is

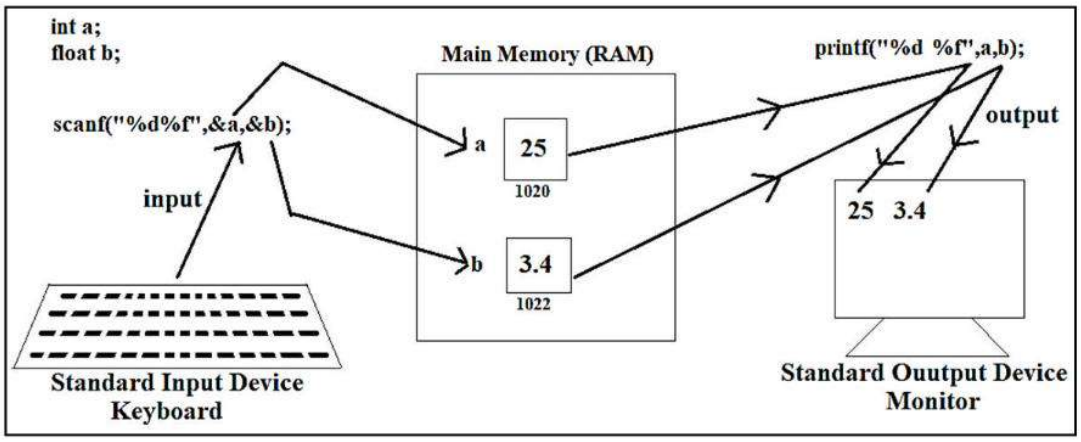


Fig. 7.8 Concept of working with scanf( ) and printf( ) function

used to display float value stored in variable b. Here, a, and b are the two arguments representing variables which are displaying values to user at monitor screen.

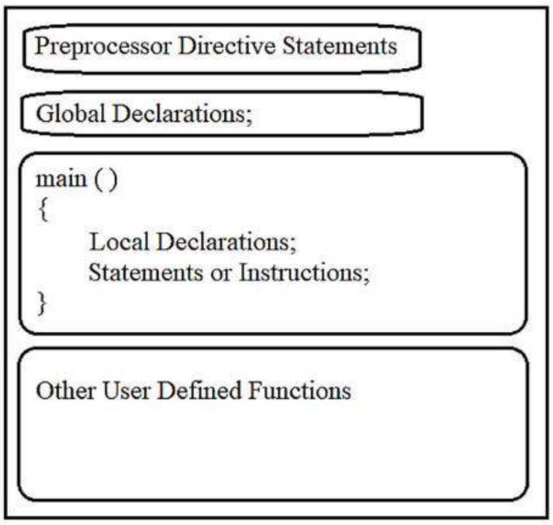
Figure 7.8 tries to illustrates the concept of working with scanf( ) and printf( ) functions in C programs:

Functions scanf( ) and printf( ) are the most important and useful functions that are widely used in any C program. These functions are part of the stdio.h header file. So, the header file stdio.h should be used in each c program.

7.12 STRUCTURE OF C PROGRAM

As we know that a program is a set of instructions written for specific task. Logical grouping of instructions in a program is called a function or block. Each C program is a collection of one or more functions and one of them should be named as main. It is because the function main is the entry point of execution for a C program. It means a computer system starts execution of c program from the main function.

Till now, we have discussed many concepts of learning any programming languages. Now, we have a clear understanding about what is the character-set, different types of tokens, different types of data, input and output statements, and header files. All these concepts are essential to begin programming using c language. Now, we need to know the basic structure of c program where we can put all these stuffs to make simple program. In general, a simple executable C-program can have the structure as shown in figure 7.9.



Preprocessor Directive Statements
Global Declarations;
main ( )
{
Local Declarations;
Statements or Instructions;
}
Other User Defined Functions

Fig. 7.9 Structure of a Simple Executable C Program

Although we have discussed many of concepts used in the above structure, yet again we are going to have a brief discussion of all element of above program-structure:

- **Pre-processor Directive Statements :** The pre-processor statement begins with # symbol. These statements instruct the compiler to perform some operations before compilation. Common use of these directive statements is to include header files or to define symbolic constants etc. Some of the common pre-processor examples are:

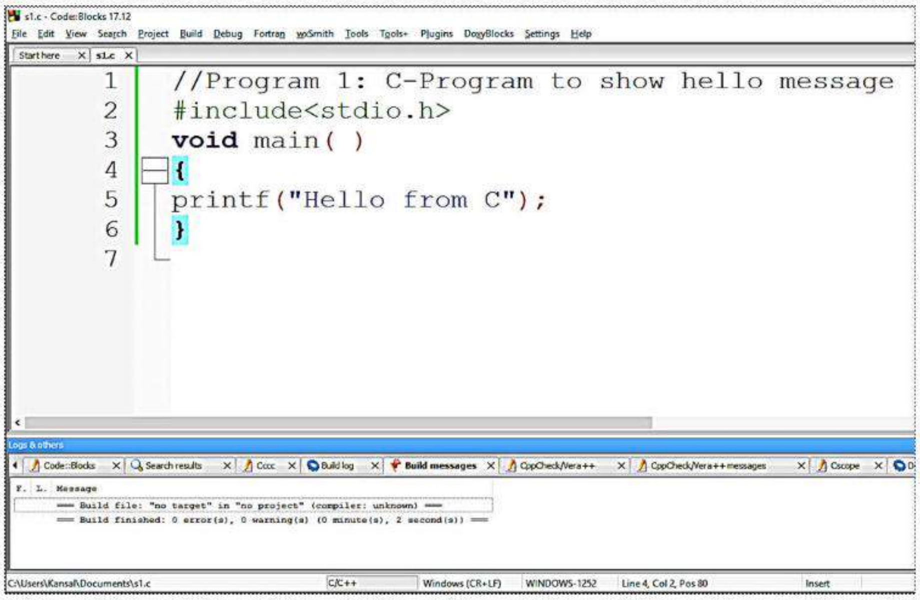
```
#include<stdio.h>
#define PI 3.14
```

- **Global Declarations :** Program elements in these declarations can be used throughout the whole program. These declarations can be variables, functions or any other program element. These declarations are written outside the body of the functions.

- **The main ( ) function :** Execution of C program starts with main ( ) function. C program cannot run without the main function. Every executable C program must contain one main ( ) function.
- **Braces { } :** Braces are used to define the block of statements. The left braces after main( ) indicates the beginning of the main function and the right braces indicates the end of the main function.
- **Local Declarations :** Declarations inside any function are called local declarations. These declarations can be used only within the function in which they are declared.
- **Program statements :** These statements are the instructions of program. They are used to perform a specific task. Each executable statement should be terminated with semicolon.
- **User defined functions :** A function is a logical grouping of statements to perform some specific task. Like main( ) function, we can define more functions according to our requirements. Because these functions are written by the user, therefore these are called user defined functions.

Now let us start C programming by making small C programs using whatever we have learnt in this chapter. Before proceeding, we are assuming that you have downloaded and installed the Turbo C or Code::Blocks for C program. Now, we will proceed with how to develop and execute C programs using Code::Blocks:

1. Open Code::Blocks by double clicking on its icon on the Desktop.
2. Create a New file by Click on **File → New → Empty** File or Using shortcut key **Ctrl+Shift+N**
3. Now type the following program in the file, as shown in the following figure:

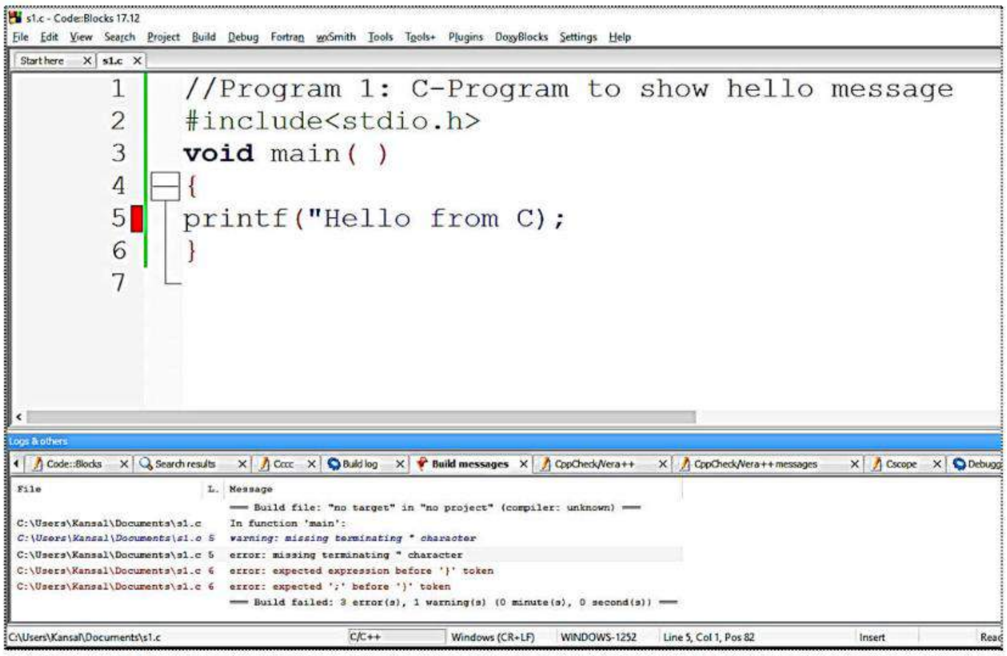


```
//Program 1: C-Program to show hello message
#include<stdio.h>
void main( )
{
printf("Hello from C");
}
```

```
F.  L.  Message
        === Build file: "no target" in "no project" (compiler: unknown) ===
        === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 2 second(s)) ===
```

Program: 7.1

4. After typing the source code of the program, save it by clicking the **File → Save File** or by pressing shortcut key **Ctrl+S**. While saving the file, don't forget to write the file extension .c with the file name, for example: test1.c, where test1 is the file name and .c is the extension name for the C programs.
5. Now, we have to compile and execute this program by clicking on **Build → Build and Run** or by pressing shortcut key **F9** from the keyboard.
6. If the program contains errors, they will be shown in the Logs and Others window which is present at the bottom of this window. If this window is not showing in the interface, we can show it by clicking on **View → Logs** or by pressing shortcut key **F2** from the keyboard. All the errors must be corrected to compile and execute the program. Consider the following program having error at line 5:



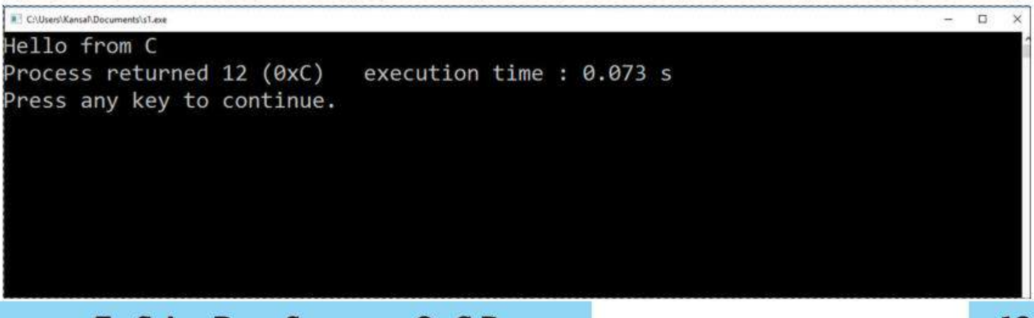
```
1   //Program 1: C-Program to show hello message
2   #include<stdio.h>
3   void main( )
4   {
5   printf("Hello from C);
6   }
7
```

```
File                                  L.  Message
                                          === Build file: "no target" in "no project" (compiler: unknown) ===
C:\Users\Kansal\Documents\s1.c            In function 'main':
C:\Users\Kansal\Documents\s1.c        5   warning: missing terminating " character
C:\Users\Kansal\Documents\s1.c        5   error: missing terminating " character
C:\Users\Kansal\Documents\s1.c        6   error: expected expression before '}' token
C:\Users\Kansal\Documents\s1.c        6   error: expected ';' before '}' token
                                          === Build failed: 3 error(s), 1 warning(s) (0 minute(s), 0 second(s)) ===
```

```
C:\Users\Kansal\Documents\s1.c    C/C++    Windows (CR+LF)    WINDOWS-1252    Line 5, Col 1, Pos 82    Insert    Read
```

In the above figure, red box in the 5th line shows that there is some error in this line. Details of the errors can be seen in the "Build Messages" tab of "Logs and Others" window. All these errors must be removed for proceeding with compilation and execution.

7. When no errors left in the program, it will display the output of the program as shown below:

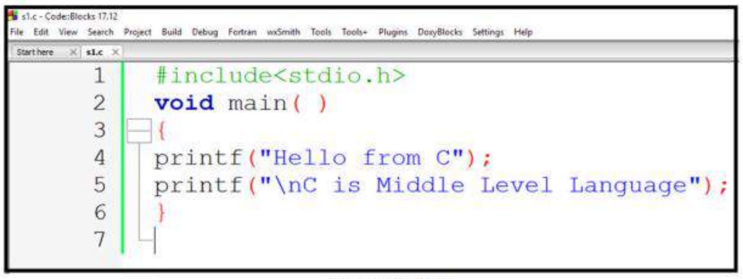


```
Hello from C
Process returned 12 (0xC)   execution time : 0.073 s
Press any key to continue.
```

This output window will also display the time taken to execute the program. Now, to come back to the source code, press any key from the keyboard.

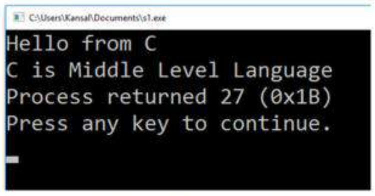
By following the above mentioned steps, we can develop, compile and execute C programs in the Code::Blocks. Following are some more examples of C programs. These programs show the usage of various concepts that has been explained in this chapter.

Program 7.2: C-Program to show multi-line message


s1.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
s1.c

```
#include<stdio.h>
void main( )
{
printf("Hello from C");
printf("\nC is Middle Level Language");
}
```

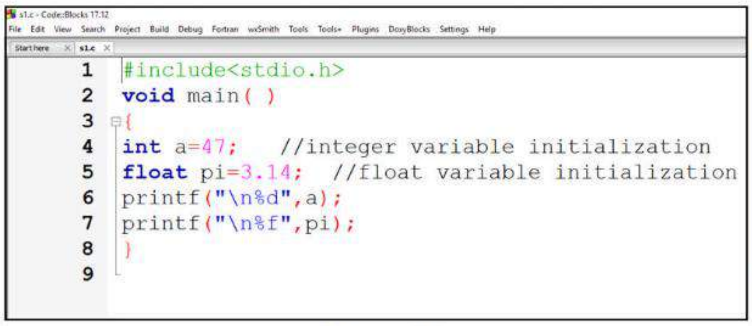
Program 7.2



```
Hello from C
C is Middle Level Language
Process returned 27 (0x1B)
Press any key to continue.
```

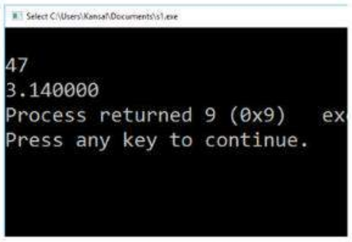
Output of Program 7.2

Program 7.3: C-Program to show Variables Declaration and showing their value


s1.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
s1.c

```
1  #include<stdio.h>
2  void main( )
3  {
4  int a=47;    //integer variable initialization
5  float pi=3.14;  //float variable initialization
6  printf("\n%d",a);
7  printf("\n%f",pi);
8  }
9
```

Program: 7.3



```
47
3.140000
Process returned 9 (0x9)   ex
Press any key to continue.
```

Output of Program: 7.3

In this program (Program 7.3), we have used many types of tokens. Let us discuss these tokens to illustrate whatever we have read in this chapter.

All Tokens in program	:	# include <> stdio.h void main ( ) { int a = 47 ; float pi 3.14 printf " " %d %f pi }
Keywords	:	void, int, float (representing data types)
Identifiers	:	stdio (name of header file), main, printf (name of functions), a, pi (name of variables)
Literals	:	47, 3.14 (fixed values)
Operators	:	= (assignment operator for storing values)
Special Symbols	:	# <> . ( ) { ; " " % }

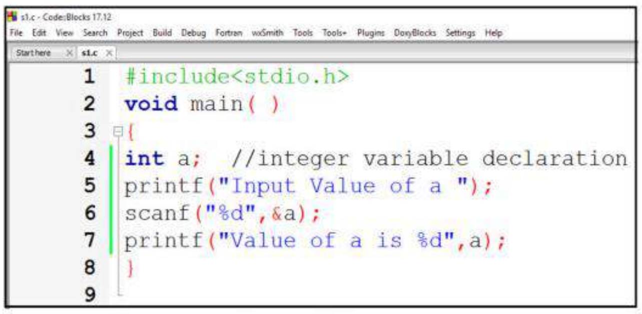
Now we are going to explain the whole program (Program 7.3) line by line so that we have a better understanding of programming:

- **In line 1**of the program, we used pre-processor directive to include header file stdio.h in our program so that we can use printf( ) function in our program.

- **In line 2,** we started the main() function. Our program starts execution from this main( ) function. Any executable program must have the main function. We cannot define more than one main function in a c program.
- **In line 3,** we used opening brace { which shows the beginning of main( ) function.
- **In line 4 and 5,** we declared integer and float type variables: int a; and float pi; and assigned them literal values. These declaration statements are terminated with special symbol semicolon (;).
- **In line 6 and 7,** we used output statement printf( ) function to show the values of integer and float variables a and pi, respective format strings %d and %f are used to represent the values.
- **In line 8,** we use closing brace } which represents the end of the main function.

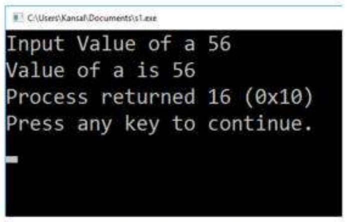
In the line 4 and 5 of above program, we use a special symbol // to explain the code in the program. The lines beginning with // are called Comments. Comments are used to describe our code in the program. They are ignored by compiler during compilation process. The symbol // is used for single line comment. For multiline comments, we can use /* and */ symbols in our program.

Program 7.4: C-Program to show how to use input and output statements with integer variable


s1.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
s1.c

```c
#include<stdio.h>
void main( )
{
int a;   //integer variable declaration
printf("Input Value of a ");
scanf("%d",&a);
printf("Value of a is %d",a);
}
```

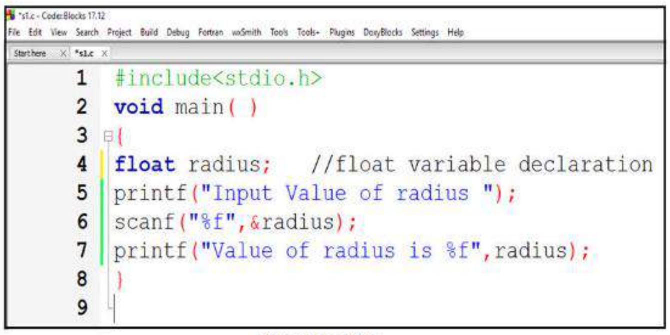
Program 7.4



```
Input Value of a 56
Value of a is 56
Process returned 16 (0x10)
Press any key to continue.
```

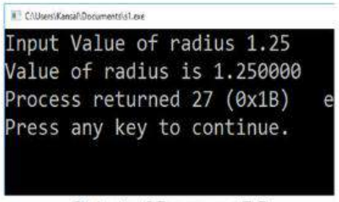
Output of Program 7.4

Program 7.5: C-Program to show how to use input and output statements with float variable


*s1.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
*s1.c

```c
#include<stdio.h>
void main( )
{
float radius;   //float variable declaration
printf("Input Value of radius ");
scanf("%f",&radius);
printf("Value of radius is %f",radius);
}
```

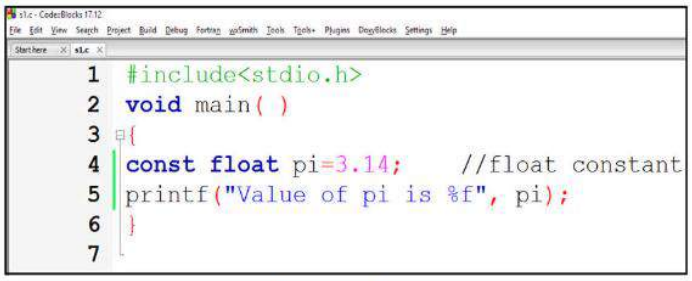
Program: 7.5



```
Input Value of radius 1.25
Value of radius is 1.250000
Process returned 27 (0x1B)
Press any key to continue.
```

Output of Program: 7.5

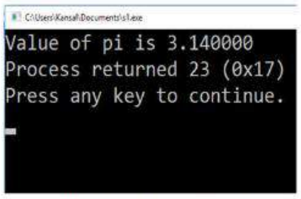
Program 7.6: C-Program to show how to use constants



s1.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
s1.c

```
#include<stdio.h>
void main( )
{
const float pi=3.14;     //float constant
printf("Value of pi is %f", pi);
}
```

Program: 7.6



```
Value of pi is 3.140000
Process returned 23 (0x17)
Press any key to continue.
```

Output of Program: 7.6

If we use Turbo C for programming, then output window will not be display directly after compiling and executing the program. To view the output window, press Alt+F5 or open **Windows** menu and then click **User Screen** to view the output window. But if we use code::blocks for c programming, then output window will automatically appears on our screen after successful execution of the program.

Points To Remember

1. C is a general purpose programming language which is developed by Dennis Ritchie.
2. C is specially designated as a Middle Level language because it has the capabilities of both low and high level languages.
3. IDE is an Integrated Development Environment that provides an environment for writing programs along with tools for compiling, executing, testing and debugging programs.
4. Character set is a set of all allowed characters for making a program.
5. Tokens are smallest individual units in a program. They are like words and punctuation marks in English.
6. Keywords are the reserve words whose meaning are predefined in the C compiler.
7. Identifiers are the names given to program elements such as variables, arrays, functions etc.
8. Operators are the symbols which are used to perform some mathematical or logical operation.
9. Constants do not allow us to change their value during execution time while the value of variables can be changed.
10. Data types define a specific type or range of values for the variables or other program elements that can hold values in the memory.
11. The functions scanf( ) and printf( ) are formatted Input/output functions which can be used to deal with any combination of numerical values, single characters and strings.
12. Comments are used to describe our code in the program.

EXERCISE

Part–A

1. Multiple Choice Questions

I. C is a ______________ purpose programming language.

a.	special	b.	general
c.	objective	d.	None of these

II. Which of the following is invalid example of identifier?

a.	roll_no	b.	%age_marks
c.	rollno	d.	main

III. Which of the followings are the tokens?

a.	keywords	b.	special symbols
c.	Literals	d.	All of these

IV. Which of the following keywords do not represent a data type?

a.	int	b.	float
c.	const	d.	char

V. ______________ are used to describe a code in the program?

a.	Compiler	b.	Comments
c.	Literals	d.	Identifiers

2. Fill in the Blanks:

I. ___________ are the smallest individual units of a program.

II. The names given to program elements, such as variables, constants, arrays, functions etc. are called ________________

III. Those program elements which do not allow changing their value during execution are called ___________________

IV. To work with single precision values, we use _________________ data type.

V. File extension of header files is ___________________

3. Write the Full form of following:

I. FORTRAN

II. BCPL

III. IDE

IV. stdio.h

V. conio.h

VI. ASCII

Part-B

4. Short Answer Type Questions. (Write the answers in 4-5 lines)

I. Why C is called Middle Level Programming Language?

II. What is a character set?

III. What are keywords?

IV. What should be the steps for creating and executing C program?

V. Write the difference between variables and constants.

VI. What are Pre-processor directives?

Part-C

5. Long Answer Type Questions. (Write the answers in 10-15 lines)

I. What are Identifiers? Write the naming rules of identifiers.

II. What are Tokens? What are the different categories of tokens that can be used in a program?

III. What are the data types? Which primitive data types are supported by C language?

IV. Explain the formatted input and output statements used in the C programs.

Lab Activity

- Draw a chart which represents different types of Tokens in C Language with suitable examples
- Write a C Program to Show Your School Name with complete address, each address line must be shown in a separate line

CHAPTER - 8
OPERATORS AND EXPRESSIONS IN C

OBJECTIVES OF THIS CHAPTER

8.1 INTRODUCTION

In the previous chapter, we have studied that variables and constants are used to store values in the program. To manipulate the data stored in these program elements, we have to use operators. In order to perform different types of operations on the data, C provides many types of operators. An operator shows that what type of operation will be performed on the data. C supports a rich set of built-in operators. We have already used some operators such as =, +, - etc. in the previous chapter. In this chapter, we are going to have a detailed discussion about various types of operators that are available in C. We shall also discuss the method of using these operators, their order of evaluation in expressions etc.

8.2 CONCEPT OF OPERATOR AND OPERAND

Operators are the symbols which are used to perform some specific type of operation on data. For example: + symbol is used to perform addition, * is used to perform multiplication, >= is used to perform comparison etc. Here +, * and >= are the operators to perform different types of operations. After performing the operations, all operators produce a value.

To perform any type of operation, we require Operands. **Operands** are the data items on which operators can perform operations. These operands can be either variables or constant values. Consider the following example:

a + 5 * 10

In this example, + and * are the operators which perform the operation on variable 'a' and constant values 5 and 10. Here the variable 'a' and constant values 5 and 10 are called the Operands.

8.3 EXPRESSION

An expression is like a formula in mathematics. An **expression** can be any valid combination of operators and operands. A valid combination is such a combination that confirms to the syntax rules of C language. A valid expression is also known as well-formed-expression. An expression after evaluation always returns a single value. This result can be used in the C programs. Expressions can be as simple as a single value and as complex as a large calculation. Consider the following examples:

```
x= 2.9;
```

It is a simple expressionwhere = operator is used with operands x and 2.9

```
x = 2.9 * y + 3.6 > z - ( 3.4 / z );
```

It is a somewhat complex expression which consists of many operators and operands. Here, =, *, +, >, - and / are the operators and x, 2.9, y, 3.6, 3.4 and z are the operands.

Now, consider the following combination of operators and operands which do not form a valid combination to be an expression:

```
x + y = z;
```

The above combination of operators and operands do not form a valid expression, although we are using valid operators and operands in the above example. But this combination of operators and operands do not follow the syntax rules of C language to be a valid expression. Left side of = operator must represent a valid memory location (identifier) to store value of z. Here, x+y cannot be a valid identifier, because a valid identifier cannot have special character other than underscore (as we learnt in the previous chapter).

All C expressions can be categorized into following two types:

8.3.1 Numerical Expressions

These expressions are used to perform numerical calculations. These expressions always return a numerical value after evaluating operators and operands. Consider the following examples:

```
4 + 3
3.2 - 7.8
```

After evaluation above numerical expression, a numerical value 7 and -4.6 will be produced.

8.3.2 Logical or Conditional Expressions

These expressions are used to perform logical or conditional operations. These expressions always return one of two possible values: either true (1) or false (0). Consider the following examples:

```
14 > 6
15 <= 6
```

After evaluating above conditional expressions, we receive true (1) for the first expression while false (0) for the second expression.

8.4 TYPES OF OPERATORS ACCORDING TO NUMBER OF OPERANDS

C providesa rich set of built-in operators. All these operators can be broadly divided into following three categories **according to number of operands** used by the operators:

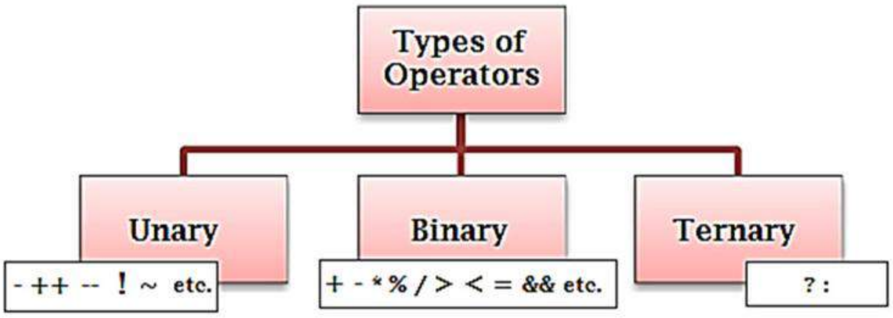


Fig. 8.1 Types of Operators according to number of operands

8.4.1 Unary operators

An operator which requires only one operand to perform its operation is called Unary operator. Common example of unary operator is unary minus. Any positive operand associated with unary minus gets its value changed. Consider the following example:

```
x= 10;
y =15;
z = x+(-y) ;
```

Here, z will be considered as z = 10+ (-15) which produce a result -5.

Since y is initially a positive integer variable, when operated by unary minus, gets its value changed. It will become negative. Some other examples of unary operators are: ++, --, ! and ~ operators.

8.4.2 Binary operators

An operator which requires two operands to perform its operation is calledBinary operator. Most of the operators in C language are of binary type. The syntax for using binary operators is given below:

Operand1 **Operator** *Operand2*

To use any binary operator, it must be put in between the operands. Consider the following examples:

```
a + b
a > b
a = b
```

In these examples, + > and = are the examples of binary operators which are placed in between two operands: a and b.

8.4.3 Ternary Operator

This operator is also known as Conditional Operator. An operator which requires three operands to perform its operation is calledternary operator. There is only one ternary operator in C. Ternary operator in C is represented using ? : symbols. The syntax for using this operator is given below:

exp1 ***?*** *exp2* ***:*** *exp3;*

Here, exp1 must be a conditional expression which produces a result either true (1) or false (0). If the value of exp1 is true then the exp2 will perform its function otherwise exp3 will perform its function. Consider the following example:

```
a=5;
b=10;
c = a > b  ?  a  :  b;
```

Here, the expression a>b will produce false (0) result (Operand1/exp1), therefore value of b (Operand3/exp3) will be stored in variable c. Variable a (Operand2/exp2) will not do any function because it will perform its function only if exp1 is true (1).

8.5 GENERAL CLASSIFICATION OF OPERATORS

All C operators can be generally categorized into following categories:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Bitwise Operators
6. Increment & Decrement Operators
7. Conditional Operators
8. Additional Operators

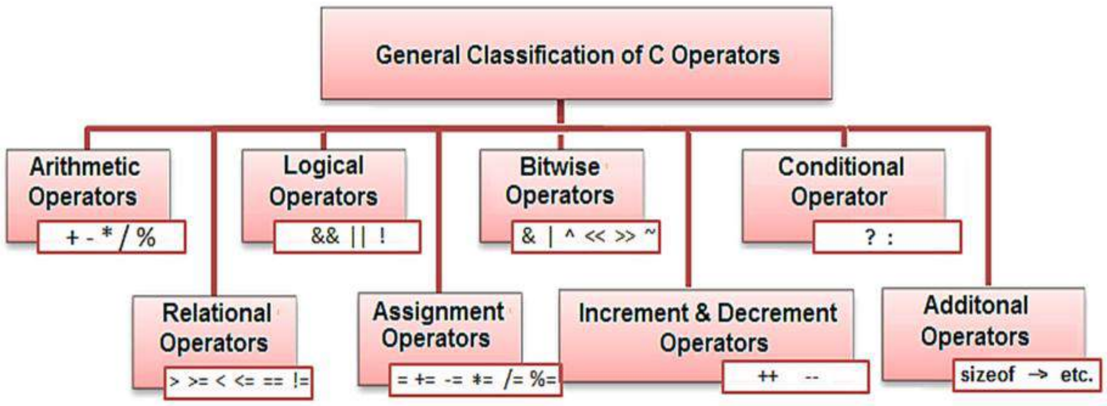


Fig. 8.2 General Classification of Operators in C

Bitwise operators are used for very low level operations i.e. for machine level programming or for performing bit level operations.

8.5.1 Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations such as: addition, subtraction, multiplication, division etc. There are five arithmetic operators in 'C'. All these operators are binary operators because all these operators require two operands to perform their operations. Following table shows the list and working of all these operators:

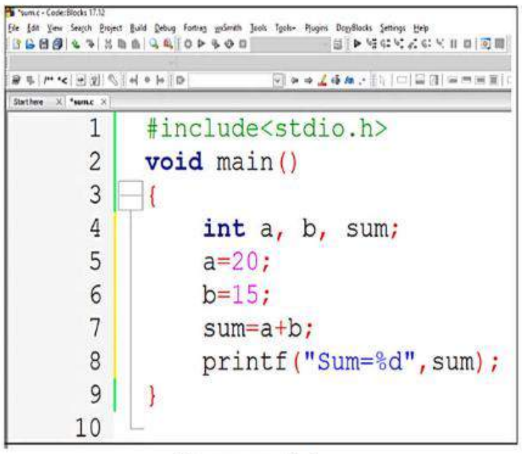
Name	Operator	Description	Examples	
Add	+	Used to perform Addition of numbers.	2+4 → 6	2.0+4.0 → 6.0
Subtract	-	Used to perform subtraction or used as any unary minus.	6-2 → 4	6.0-4.0 → 2.0
Multiply	*	Used to perform multiplication of numbers.	7*2 → 14	7.0*2.0 → 14.0
Divide	/	Used to perform division of numbers.	5/2 → 2 Integer division	5.0/2.0 → 2.5 Real Division
Modulus	%	Used to get remainder value after division of numbers.	7%4 → 3	5.0%2.0 Not allowed

Table: 8.1 Arithmetic Operators

For arithmetic operators, operands can be integer values, floating-point values or character values. The modulus operator requires that both operands be integers & the second operand be nonzero. Similarly, the division operator (/) requires that the second operand be nonzero, though the operands need not to be integers. Division of one integer by another is referred to as integer division. With this division the decimal portion of the quotient will be dropped. If division operation is carried out with two floating- point numbers or with one floating-point number and one integer, the result will be a floating-point quotient.

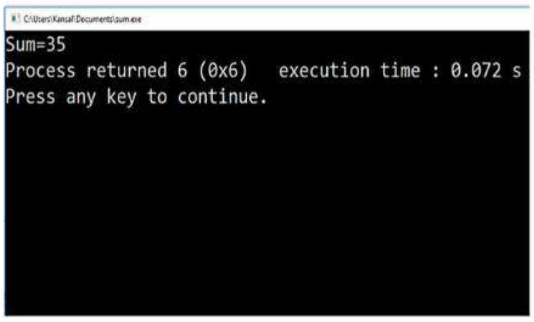
If one or both operands represent negative values, then the addition, subtraction, multiplication and division operations will result in values whose signs are determined by the usual rules of algebra. The interpretation of remainder operation is unclear when one of the operands is negative. Following programs show how to use arithmetic operators in C programming:

Program 8.1: Program in C to find sum of two numbers:



```c
#include<stdio.h>
void main()
{
    int a, b, sum;
    a=20;
    b=15;
    sum=a+b;
    printf("Sum=%d",sum);
}
```

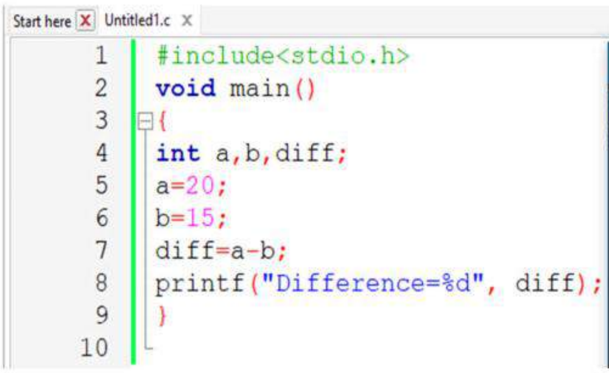
Program: 8.1



```
Sum=35
Process returned 6 (0x6)   execution time : 0.072 s
Press any key to continue.
```

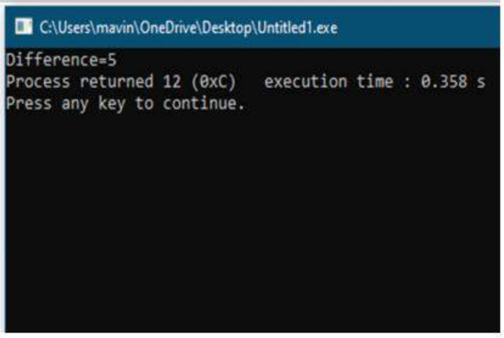
Output of Program 8.1

Program 8.2: A program in C Language to find thedifference of two numbers



```c
#include<stdio.h>
void main()
{
int a,b,diff;
a=20;
b=15;
diff=a-b;
printf("Difference=%d", diff);
}
```

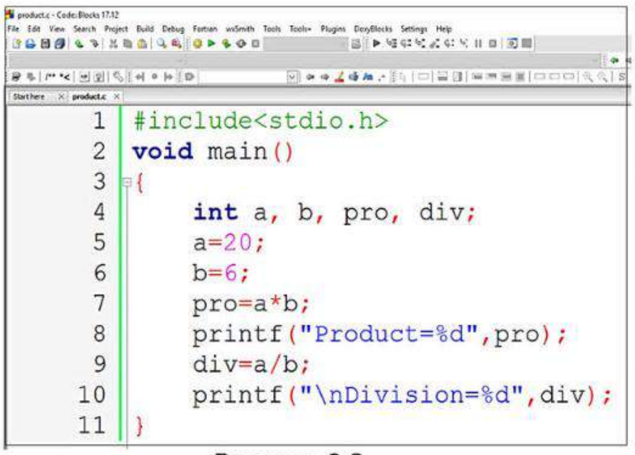
Program: 8.2



```
Difference=5
Process returned 12 (0xC)   execution time : 0.358 s
Press any key to continue.
```

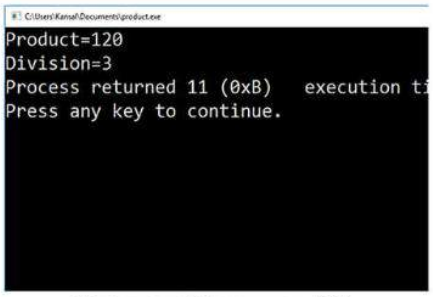
Output of Program 8.2

Program 8.3: A program in C Language to find product and division of two numbers:



```c
#include<stdio.h>
void main()
{
    int a, b, pro, div;
    a=20;
    b=6;
    pro=a*b;
    printf("Product=%d",pro);
    div=a/b;
    printf("\nDivision=%d",div);
}
```

Program 8.3



```
Product=120
Division=3
Process returned 11 (0xB)   execution ti
Press any key to continue.
```

Output of Program 8.3

8.5.2. Relational operators

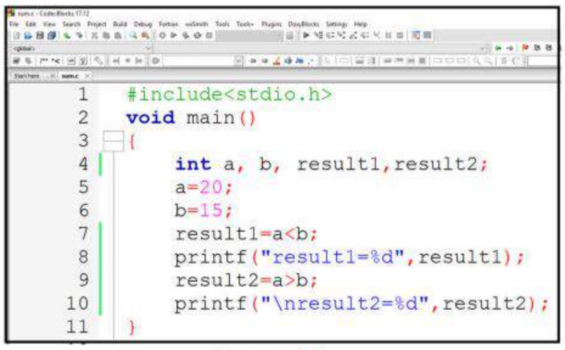
Relational operators are also called comparison operators. These operators are used to test the relationship between operands. In other words, these operators are used to compare values. After comparison, these operators return either true (1) or false (0) value. All the relational operators present in C language are of binary type. It means these operators require two operands to perform their operation. There are 6 relational operators in C which are given below in the table with examples:

Name	Operator	Description	Example	Result
Equals to	==	Used to check whether two values are equal	4==5 5==5	False True
Not Equal to	!=	Used to check whether two values are not equal	4! =5 4! =4	True False
Greater than	>	Used to check whether the first value is greater than second	4>5 5>4	False True
Less than	<	Used to check whether the first value is less than second	4<5 5<4	True False
Greater than or equal to	>=	Used to check whether first value is greater than or equal to second value	5>=5 6>=8 10>=5	True False True
Less than or equal to	<=	Used to check whether first value is lesser than or equal to second value	4<=5 4<=2 4<=4	True False True

Table 8.2 Relational Operators

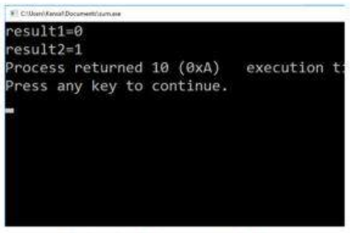
These relational operators are used to form logical expression representing condition. The resulting expression will be of type integer, since true is represented by the integer value 1 and false is represented by the value 0.

Program 8.4: A Program which shows the use of Relational Operators



```
#include<stdio.h>
void main()
{
    int a, b, result1,result2;
    a=20;
    b=15;
    result1=a<b;
    printf("result1=%d",result1);
    result2=a>b;
    printf("\nresult2=%d",result2);
}
```

Program 8.4



```
result1=0
result2=1
Process returned 10 (0xA)   execution t
Press any key to continue.
```

Output of Program 8.4

In the program 8.4 in the line 7, the statement a<b returns 0 as the result is false because 20 < 15. Therefore, in line 8 it shows result1=0. Similarly, in line 9, the statement a>b returns 1 as the result is true because 20>15. Therefore, in line 10 it shows result2=1.

8.5.3 Logical operators

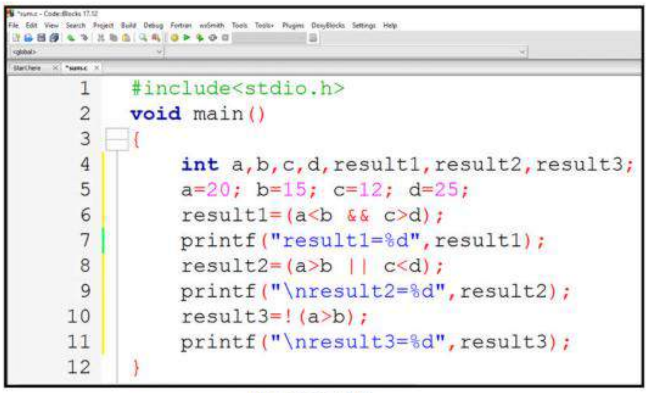
Logical operators are also called Boolean Operators. These operators are used to make compound relational expressions. In other words, we can say that these operators are used when we want to test more than one condition at a time. There are 3 Logical operators in C Language: 'Logical AND', 'Logical OR' and 'Logical NOT'. Here, 'Logical AND' and 'Logical OR' are the binary operators whereas 'Logical NOT' is unary operator. Two operands are required for 'Logical AND' and 'Logical OR' to perform their operations while one operand is required for 'Logical NOT' to perform its operation.

All Logical Operators also return either true or false value. The result of a logical AND operator will return true only if both operands are true, whereas the result of a logical OR operator will be true if either operand is true or if both operands are true. Logical NOT operator returns true only when its operand is false. Following table shows the list and working of all Logical Operators used in C language with suitable examples:

Name	Operator	Description	Example	Explanation and Result
AND	&&	Returns true only if both operands are true otherwise it returns false	3>5 && 4>5 3>5 && 4<5 3<5 && 4>5 3<5 && 4<5	False && False → False False && True → False True && False → False True && True → True
OR	||	Returns true if at least one of its operand is true otherwise it returns false	3>5 || 4>5 3>5 || 4<5 3<5 || 4>5 3<5 || 4<5	False || False → False False || True → True True || False → True True || True → True
NOT	!	Returns true only when its operand is false otherwise it returns false	!(3<5) !(3>5)	!(True) → False !(False) → True

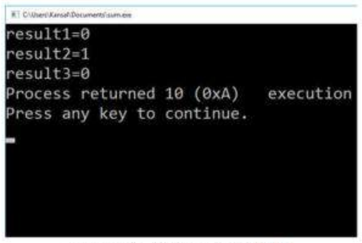
Table 8.3 Logical Operators

Program 8.5 A program in C Language to show the working of logical operators:



```
 1  #include<stdio.h>
 2  void main()
 3  {
 4      int a,b,c,d,result1,result2,result3;
 5      a=20; b=15; c=12; d=25;
 6      result1=(a<b && c>d);
 7      printf("result1=%d",result1);
 8      result2=(a>b || c<d);
 9      printf("\nresult2=%d",result2);
10      result3=!(a>b);
11      printf("\nresult3=%d",result3);
12  }
```

Program 8.5



```
result1=0
result2=1
result3=0
Process returned 10 (0xA)   execution
Press any key to continue.
```

Output of Program 8.5

In the above program:

In line 6 → **result1 = false && false** and it will produce **result1= false (0)**

In line 8 → **result2 = true || true** and it will produce **result2=true (1)**

In line 10 → **result3 = !(true)** and it will produce **result3=false (0)**

8.5.4 Assignment operators

These Operators are used to assign or store values in variables etc. The symbol of assignment operator is =. Consider the following examples which show how to use assignment operator in C programs:

a = - 2;	// assigns -ve value (-2) to the variable.
b = 5;	// assigns value (5) to the variable.
c = a + b;	// assigns the result of expression to the variable.
a = a + 10;	// self-assignment of a variable.

Assignment operators can also be used as shorthand operators. Shorthand assignment operators are useful in self-assignment statements. Following table shows the examples of shorthand assignment operators used in C:

Let's assume int a=5;

Shorthand Operator	Example for Shorthand Assignment	Equivalent Self-Assignment	Result
+=	a+ =2	a = a + 2	a=7
- =	a- =2	a = a - 2	a=3
*=	a* =2	a = a * 2	a=10
/=	a / =2	a = a / 2	a=2
%=	a%=2	a = a % 2	a=1

Table 8.4: Shorthand Assignment Operators

Assignment operator = and the equality (equal to) operator == both are different types of operators. The assignment operator is used to assign a value to an identifier, whereas the equality operator is used to determine if two operands have the same value. These two operators cannot be used in place of one another.

If the two operands in an assignment expression are of different data types, then the value of the expression on the right will automatically be converted to the type of the identifier on the left. For example:

```
int a=3.5;
```

Here, float value 3.5 will automatically be converted to type integer, i.e. value of variable a will become 3

8.5.5 Bitwise Operators

Bitwise operators are used for very low level operations, i.e. for machine level programming or for performing bit level operations. C provides the following operators for handling bitwise operations:

1.	<<	Bit shift left (a specified number or bit positions)
2.	>>	Bit shift right(a specified number of bit positions)
3.	|	Bitwise OR
4.	^	Bitwise XOR
5.	&	Bitwise AND
6.	~	Bitwise one's complement

8.5.6 Increment and Decrement Operators

These are the unary operators. The symbol ++ is used for increment operator while symbol - - used for decrement operator. The increment operator causes its operand to increase by one whereas the decrement operator causes its operand to decrease by one. The operand used with each of these operators must be a single variable. These operators cannot be applied directly on the constant values.

For example:

```
int x=10;     (x is an integer variable that has been assigned a value of 10)
```

Following expression causes the value of x to be increased to 11

```
++x;          (which is equivalent to x= x+1)
```

Similarly, following expressioncauses the original value of x to be decreased to 9:

```
--x,          (which is equivalent to x=x-1)
```

If we use 10++ or --10, it will be a wrong statement as we have already studied that these cannot be applied directly on the constant values.

The increment and decrement operators can each be utilized in two different ways. It depends on whether the operator is written before or after the operand:

- Prefix increment and decrement
- Postfix increment and decrement

If the operator precedes the operand, then the value of operand will be altered before it is used for its intended purpose within the program. This is called **pre increment/decrement.** If, however the operator follows the operand then the value of the operand will be changed after it is used. This is called post increment/ decrement.

For example:

If the value of x is initially 10, it can be increased by two methods:

```
y = ++x;            (pre increment)
```

Here,at first, the value of x will be incremented to 11 and then this incremented value of x will be assigned to variable y, i.e. y will also get value 11 (i.e. x=11 and y=11)

```
y = x++;            (post increment)
```

Here, at first, the value of x will be assigned to variable y (i.e. y will get value 10) then value of x will be incremented to 11 (i.e. y=10 and x=11)

Similarly, decrement operator can be used. For example:

```
y = --x;            (pre decrement)
```

Here,first of all, the value of x will be decremented to 9 and then this decremented value of x will be assigned to variable y, i.e. y will also get value 9 (i.e. x=9 and y=9)

```
y = x--;            (post decrement)
```

Here, first of all, the value of x will be assigned to variable y (i.e. y will get value 10) then value of x will be decremented to 9 (i.e. y=10 and x=9)

8.5.7 Conditional Operator (? :)

It is a ternary operator. There is one and only one ternary operator (? :) in 'C' language. An expression that makes use of the conditional operator is called a conditional expression. This operator has already been defined in the section 8.4.3 of this chapter. Please refer to the specified section for more details of this operator.

8.5.8 Additional Operators:

All the remaining operators that do not come under any above mentioned categories of operators, can be considered as additional operators. Examples of such operators are: sizeof( ) operator, pointer operators, member selection operators etc.

sizeof operator:

It is another unary operator. This operator returns the size of its operand, in bytes. This operator always precedes its operand. The operand may be an expression, or it may be a variable or data type. Consider the following examples:

```
sizeof (x);
sizeof (float);
```

In first example, if x is of char type variable then it returns the result 1, while in example second, we pass keyword for data type float which returns 4 as float data type occupies four bytes in memory.

8.6 TYPE CONVERSION

The value of an expression can be converted to a different data type in C, if desired.When value of one type is converted into some other type, it is called Type Conversion.Operands that differ in type may undergo type conversion before the expression takes on its final value. There are two ways of type conversions in C:

1. Implicit Conversion
2. Explicit Conversion

8.6.1 Implicit Conversion

This type of conversion is automatic. For this type of conversion, we use assignment (=) operator. This type of conversion is used when operand having lower data type is converted into higher data type. There is no loss of information in this type of conversion. Consider the following example for automatic conversion:

```
float n;
n = 5/2;
```

In this example, implicit conversion takes place, as integer type data, after integer division of 5/2 will produce result in the form of integer value, i.e. 2 (but not 2.5). This result will automatically be converted into float type value so that it can be stored in the float variable i.e. value of variable n will become 2.000000

8.6.2 Explicit Conversion

This is forceful conversion. For this type of conversion, we use caste operator. There may or may not be any loss of information in this type of conversion. This type of conversion is used when operand of higher data type is converted into lower data type.

The syntax for this type of casting is:

(Desired data type) Expression

The name of data type into which the conversion is to be made is enclosed in parentheses and placed directly to the left of the value to be converted. The example of type casting is as follows:

For example:

```
float n;
n=(float)5/2;
```

The cast operator converts the value of 5 to its equivalent float representation (i.e. 5.0) before the division by 2. Therefore, it will become float division and the result of division will be 2.500000 which will be stored in the float variable n and hence the value of n will become 2.500000. The cast operator can be used on a constant or expression as well as on a variable.

8.7 PRECEDENCE/ HIERARCHY OF OPERATORS

The operators within C are grouped hierarchically according to their order of evaluation, known as **precedence**. Operators with a higher precedence are carried out before operators having a lower precedence. In simple words, the sequence of evaluation of operators in which they are applied on the operands in an expression is called the **Precedence of Operators**. The natural order can be altered by making use of parentheses.

Precedence of commonly used operators in decreasing order is as follows:

1.	-	++	- - Operators	! ~	sizeof	(datatype)
2.	*	/	%			
3.	+	-				
4.	<	<=	>	> =		
5.	==	!=				
6.	&&					
7.	ǁ					
8.	? :					
9.	=					

Consider the following example which illustrates how arithmetic expressions are evaluated using operators precedence:

a = 5 * 4 / 4 + 8 - 9 / 3;	(* is evaluated)
a = 20 / 4 + 8 - 9 / 3;	(/ is evaluated)
a = 5 + 8 - 9 / 3;	(/ is evaluated)
a = 5 + 8 - 3;	(+ is evaluated)
a = 13 - 3;	(- is evaluated)
a = 10;	result of expression

Points To Remember

1. **Operators** are the symbols which are used to perform some specific type of operation on data.
2. **Operands** are the data items on which operators can perform operations.
3. An **Expression** is like a formula in mathematics which can be any valid combination of operators and operands.
4. An operator which requires only one operand to perform its operation is calledUnary operator.
5. An operator which requires two operands to perform its operation is calledBinary operator.
6. Ternary operator is also known as **Conditional Operator** which requires three operands to perform its operation.

7. '%' operator is also known as **modulus** operator which works only on integer operands.

8. **Relational** operators are symbols that are used to test the relationship between two variables..

9. There are three **Logical** operators in C language, they are **and, or, not.** They are represented by &&, || and ! respectively

10. **Assignment operators** in C assign the value of an expression to an identifier and the most commonly used assignment operator is =.

11. The **increment** operator (++) causes its operand to be increased by one.

12. The **decrement** operator (**- -**) causes its operand to be decreased by one.

13. When value of one type is converted into some other type, it is called **Type Conversion.**

14. The sequence of evaluation of operators in which they are applied on the operands in an expression is called the **Precedence of Operators.**

EXERCISE

Part-A

1. Multiple Choice Questions

I. The symbols which are used to perform some specific type of operation on data are called?

a.	Operands	b.	Operators
c.	Expressions	d.	Formulas

II. Which operator acts only on one operand?

a.	Unary	b.	Binary
c.	Ternary	d.	Conditional

III. Which of the following is not a Logical Operator?

a.	And (&&)	b.	OR (||)
c.	Equality (==)	d.	NOT (!)

IV. Which symbol is used for Ternary Operator?

a.	: ?	b.	; ?
c.	? :	d.	? ;

V. Which of the following cannot be considered as assignment operator?

a.	=	b.	==
c.	+=	d.	%=

2. Fill in the Blanks:

I. ___________are the data items on which operators can perform operations.

II. Unary operator acts on only ____________ operand.

III. __________ arithmetic operator performs only on integer operand.

IV. When value of one type is converted into some other type, it is called _________.

V. Ternary operator is also known as _______________.

3. **Very Short Answer Type Questions**

I. Which operator causes its operand to be increased by one?

II. Which operators are used to test the relationship between two variables?

III. Write all the Arithmetic Operators used in C programming.

IV. Which operator returns the size of its operand, in bytes?

V. Write the name of two ways of type conversion.

VI. How many relational operators are present in C Language?

Part-B

4. **Short Answer Type Questions. (Write the answers in 4-5 lines)**

I. Define Expression?

II. What is Operand?

III. What is Unary operator?

IV. Define Conditional operator?

V. What is Type Conversion?

VI. What is an operator? Write the name of different types of operators?

VII. Write about increment and decrement operators?

Part-C

5. **Long Answer Type Questions. (Write the answers in 10-15 lines)**

I. Explain the Arithmetic Operators? Write any program using Arithmetic Operators?

II. What are Relational operators? Write any program of Relational operator?

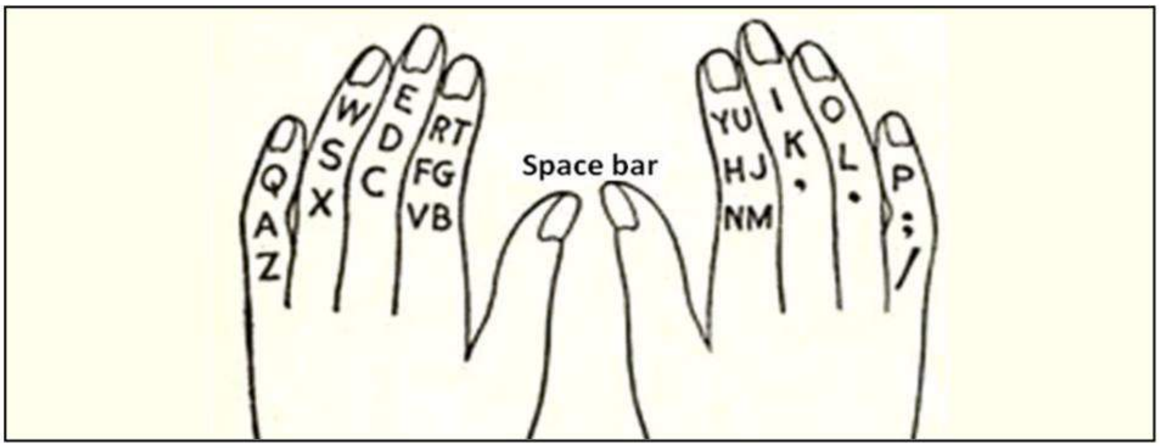
III. Explain Logical operators? Write any program of Logical operator?

Lab Activity

- Write a C program to show the usage of Arithmetic operators
- Write a C program to show the usage of conditional (ternary) operator
- Write the C Programs to solve the following mathematical formulas:
    - z=10(5+1)%2
    - x=(5+7)÷(8+9*2)

APPENDIX – I

Lab Activity for Typing Practice in English



EXERCISE I

asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg
asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg
asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg
asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg
asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg	;lkjh	asdfg

EXERCISE II

Ask	Fad	Alsas	Shad	Lads	Flags	Flask
Jag	Fag	Fall	Hash	Glad	Galls	Salad
Jak	Had	Gaff	Dash	Gall	Flash	Slash
Sad	Lad	Adds	Lash	Hall	Lakhs	Dhalls
Dad	Asks	Alas	Dall	Fall	Glass	Shall

EXERCISE III

qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert
poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy
qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert
poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy
qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert
poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy	qwert	poiuy

EXERCISE IV

awerqfa	;oiupj;	awerqfa	;oiupj;	awerqfa	;oiupj;
awerqfa	;oiupj;	awerqfa	;oiupj;	awerqfa	;oiupj;
awerqfa	;oiupj;	awerqfa	;oiupj;	awerqfa	;oiupj;
awerqfa	;oiupj;	awerqfa	;oiupj;	awerqfa	;oiupj;
awerqfa	;oiupj;	awerqfa	;oiupj;	awerqfa	;oiupj;

EXERCISE V

Fish	Dirks	Oldest	Apple	Grade	Falls	Kodak
Rails	Jaded	Dead	Usual	Sales	Filed	Legal
Lease	Lakes	Agile	Isles	Ahead	Larks	Roses
Forks	Hedge	Skill	Rupee	Grass	Would	Alpine
Jaded	Liked	Equip	Quail	Jokes	Asked	Walks
Fiddle	Saddle	Dead	Filed	Lakes	Lease	Legal

EXERCISE VI

azxcvf	lkmnbj	azxcvf	lkmnbj	azxcvf	lkmnbj
azxcvf	lkmnbj	azxcvf	lkmnbj	azxcvf	lkmnbj
azxcvf	lkmnbj	azxcvf	lkmnbj	azxcvf	lkmnbj
azxcvf	lkmnbj	azxcvf	lkmnbj	azxcvf	lkmnbj
azxcvf	lkmnbj	azxcvf	lkmnbj	azxcvf	lkmnbj

EXERCISE VII

Cat	Jack	Colour	Neither	Enemy	Boat	Calcutta
Not	Have	Joints	Calling	Voted	Very	Vineyard
Met	Wind	Nerves	Enlarge	Money	Move	Material
Men	Verb	Verbal	Someone	Marry	Give	Sterling
Bent	Joint	Jackets	Examine	Thousand	Cylinder	Assessment
King	Carry	Jumbled	Examined	Struggle	Possible	Beginning
Zeal	Night	Booklet	Gracious	Grizzled	Frequent	Meanings
Zero	Tonic	Cutting	Becoming	Zodiacal	Exponent	Doubtless

EXERCISE VIII

12345	098767	12345	098767	12345	098767
12345	098767	12345	098767	12345	098767
12345	098767	12345	098767	12345	098767
12345	098767	12345	098767	12345	098767
12345	098767	12345	098767	12345	098767

EXERCISE IX

Type the following sentences 5 times:

1. Lost time is never regained
2. Get-up early and do your work
3. Today's youth and tomorrow's old
4. Age is a virtue when wisdom is with it.
5. Measure your word before it goes out of you
6. My steps are measured
7. A friend in need is a friend indeed
8. Children are innocent and should be guided rightly.
9. Our Land has great sages who knew the eternal truth.
10. Truth never fails
11. The Quick Brown Fox Jumps Over A Lazy Dog

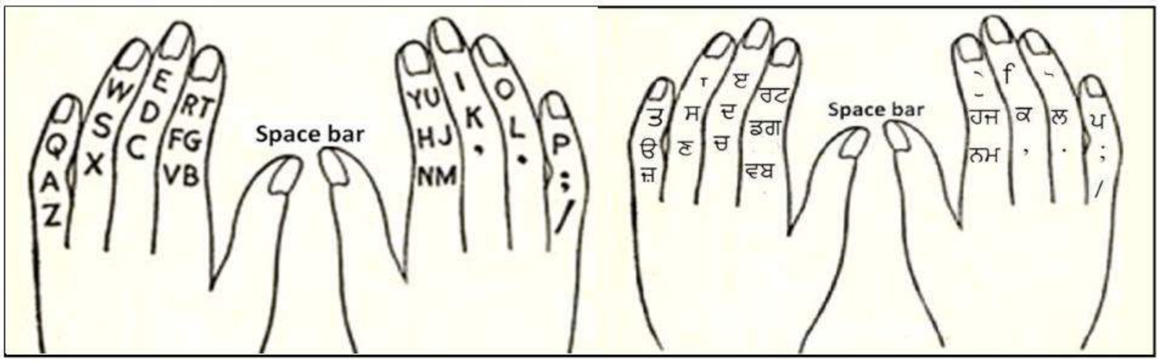
EXERCISE X

Type the following paragraph 10 times:

Our flag is tri-colour. SAFFRON is the symbol of sacrifice and a string mind. WHITE is the symbol of purity, love and peace. GREEN is the symbol of plenty and joy. We hoist and salute our flag. We are ready to make sacrifices for our country. We want peace and progress. We want to be pure.

APPENDIX – II

Lab Activity for Typing Practice in Punjabi (Using Anmol Lipi Font)



HOME ROW EXERCISE - I

ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ
ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ
ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ
ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ
ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ	ੳਸਦਡਗ	;ਲਕਜਹ

HOME ROW EXERCISE - II

ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ
ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ
ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ
ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ
ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ	ਅਸ਼ਧਢਘ	;ਲ਼ਖਝ

SECOND ROW EXERCISE-III

ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ
ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ
ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ
ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ
ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ	ਤ ਾ ੲ ਰ ਟ	ਪ ੋ ਿ ੁ ੇ

SECOND ROW EXERCISE-IV

ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ
ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ
ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ
ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ
ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ	ਥ ਂ ੳ ੍ ਠ	ਫ ੋ ੀ ੂ ੈ

HOME ROW AND SECOND ROW EXERCISE-V

ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;	ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;
ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;	ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;
ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;	ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;
ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;	ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;
ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;	ੳ ਾ ੲ ਰ ਤ ਡ ੳ	; ੇ ਿ ੁ ਪ ਜ ;

HOME ROW AND SECOND ROW EXERCISE-VI

ਦਸਿਹ	ਧਰਿਕਸ	ੌ ਲਦੲਸਟ	ਅਪਪਲੲ	ਘਰੳਦੲ	ਢੳਲਲਸ	ਖੋਦੳਕ
੍ੳਲਿਸ	ਝੳੁਦੲਦ	ਧੲੳਦ	ੂਸੁੳਲ	ਸ਼ੳਲੲਸ	ਢਲਿੲਦ	ਲ਼ੲਗੳਲ
ਲ਼ੲੳਸੲ	ਲ਼ੳਕੲਸ	ਅਗਲਿੲ	ੀਸਲੲਸ	ਅਹੲੳਦ	ਲ਼ੳਰਕਸ	ਸੲਸ
ਢੋਰਕਸ	੍ਰੲਦਗੲ	ਸ਼ਕਲਿਲ	੍ ੁ ਪੲੲ	ਘਰੳਸਸ	ਂ ੇ ੁ ਲਦ	ਅਲਪਨਿੲ
ਝੳਦੲਦ	ਲ਼ਕਿੲਦ	ੳਤੁਪਿ	ਥੁੳਲਿ	ਝੋਕੲਸ	ਅਸਕੲਦ	ਂੳਲਕਸ
ਢਦਿਦਲੲ	ਸ਼ੳਦਦਲੲ	ਧੲੳਦ	ਢਲਿੲਦ	ਲ਼ੳਕੲਸ	ਲ਼ੲੳਸਣ	ਲ਼ੲਗੳਲ

HOME/SECOND/THIRD ROWEXERCISE -VII

ੳ ਜ਼ ਣ ਚ ਵ ਡ	ਲ ਕ ਮ ਨ ਬ ਜ	ਅ ਗ਼ ਯ ਛ ੜ ਢ	ਲ਼ ਖੰ ਃ ਭ ਝ
ੳ ਜ਼ ਣ ਚ ਵ ਡ	ਲ ਕ ਮ ਨ ਬ ਜ	ਅ ਗ਼ ਯ ਛ ੜ ਢ	ਲ਼ ਖੰ ਃ ਭ ਝ
ੳ ਜ਼ ਣ ਚ ਵ ਡ	ਲ ਕ ਮ ਨ ਬ ਜ	ਅ ਗ਼ ਯ ਛ ੜ ਢ	ਲ਼ ਖੰ ਃ ਭ ਝ
ੳ ਜ਼ ਣ ਚ ਵ ਡ	ਲ ਕ ਮ ਨ ਬ ਜ	ਅ ਗ਼ ਯ ਛ ੜ ਢ	ਲ਼ ਖੰ ਃ ਭ ਝ
ੳ ਜ਼ ਣ ਚ ਵ ਡ	ਲ ਕ ਮ ਨ ਬ ਜ	ਅ ਗ਼ ਯ ਛ ੜ ਢ	ਲ਼ ਖੰ ਃ ਭ ਝ

FOURTH ROW EXERCISE-VIII

ੱ12345	098767	ੱ ! ੍ # 4 %	ਸ਼ ਫ਼ * ( )
ੱ12345	098767	ੱ ! ੍ # 4 %	ਸ਼ ਫ਼ * ( )
ੱ12345	098767	ੱ ! ੍ # 4 %	ਸ਼ ਫ਼ * ( )
ੱ12345	098767	ੱ ! ੍ # 4 %	ਸ਼ ਫ਼ * ( )
ੱ12345	098767	ੱ ! ੍ # 4 %	ਸ਼ ਫ਼ * ( )

EXERCISE - IX

ਸਾਇੰਸ	ਲੇਖਕ	ਘਰ	ਛੱਤ	ਇਨਾਮ
ਕੰਪਿਊਟਰ	ਕਿਤਾਬ	ਸਕੂਲ	ਅਧਿਆਪਕ	ਪੈਂਸਿਲ
ਰਸਤਾ	ਜਹਾਜ਼	ਪੰਨ੍ਹਾ	ਮੁਰੰਮਤ	ਜ਼ਿਲ੍ਹਾ
ਸੁਨਾਮ	ਮੋਹਾਲੀ	ਚੰਡੀਗੜ੍ਹ	ਇੰਡੀਆ	ਹਿਮਾਲਿਆ
ਭੂਗੋਲ	ਇਤਿਹਾਸ	ਗਣਿਤ	ਵਿਕਾਸ	ਪੰਜਾਬੀ
ਹਿੰਦੀ	ਪ੍ਰਧਾਨ	ਮੰਤਰੀ	ਮੁੱਖ	ਰੋਜ਼ਗਾਰ
ਯੂਨੀਵਰਸਿਟੀ	ਸੀ.ਪੀ.ਯੂ.	ਮਾਊਸ	ਕੀਅਬੋਰਡ	ਮੋਨੀਟਰ
ਮੈਮਰੀ	ਜੈਨਰੇਸ਼ਨ	ਪ੍ਰੈਜ਼ਨਟੇਸ਼ਨ	ਐਕਸਲ	ਸਪ੍ਰੈਡਸ਼ੀਟ
ਪ੍ਰੋਸੈਸਰ	ਓਪਰੇਟਿੰਗ	ਸਿਸਟਮ	ਵਿੰਡੋ	ਸ਼ਹਿਦ
ਰਾਜਨੀਤੀ	ਅਮੇਰੀਕਾ	ਡਾਟਾ	ਖੁਸ਼ੀ	ਮੱਖਣ
ਰੰਗਾ	ਉੱਠ	ਅੱਖ	ਸਾਂਹ	ਖਿਡਾਰੀ
ਪ੍ਰਿੰਸੀਪਲ	ਪ੍ਰੀਖਿਆ	ਵਿਭਾਗ	ਦਫ਼ਤਰ	ਨਿਰਦੇਸ਼
ਵਿਲੱਖਣ	ਵਿਸ਼ੇਸ਼	ਪੰਜਾਬ	ਸੰਗਰੂਰ	ਅਭਿਆਸ

EXERCISE - X

ਪੰਜਾਬ ਯੂਨੀਵਰਸਿਟੀ ਪਟਿਆਲਾ ਦੇ ਵਾਈਸ ਚਾਂਸਲਰ ਡਾ. ਜਸਪਾਲ ਸਿੰਘ ਦੇ ਦਿਸ਼ਾ ਨਿਰਦੇਸ਼ਾਂ ਅਤੇ ਪੰਜਾਬੀ ਵਿਭਾਗ ਦੇ ਮੁਖੀ ਪ੍ਰੋ. ਲਖਵੀਰ ਸਿੰਘ, ਪ੍ਰੋ. ਬਲਦੇਵ ਸਿੰਘ ਚੀਮਾ, ਡਾ. ਦੇਵਿੰਦਰ ਸਿੰਘ ਦੀ ਅਗਵਾਈ ਵਿੱਚ ਚਲਦਿਆਂ ਡਾ. ਰਾਜਵਿੰਦਰ ਸਿੰਘ ਅਤੇ ਸ. ਚਰਨਜੀਵ ਸਿੰਘ ਨੇ **ਜੀ-ਲਿਪੀਕਾ** ਨਾਮ ਦਾ ਅਜਿਹਾ ਸਾਫ਼ਟਵੇਅਰ ਤਿਆਰ ਕੀਤਾ ਹੈ ਜਿਸ ਰਾਹੀਂ ਦਫ਼ਤਰੀ ਕੰਮਕਾਜ ਤੋਂ ਇਲਾਵਾ ਫੇਸਬੁੱਕ ਸਮੇਤ ਇੰਟਰਨੈੱਟ ਤੇ ਹੋਰ ਕਿਤੇ ਵੀ ਪੰਜਾਬੀ ਵਿੱਚ ਲਿਖਣਾ ਬਹੁਤ ਸੌਖਾ ਹੋ ਗਿਆ ਹੈ। ਇੱਥੋਂ ਤੱਕ ਕਿ ਤੁਸੀ ਆਪਣੀ ਈ-ਮੇਲ ਵੀ ਆਪਣੇ ਕਿਸੇ ਮਿੱਤਰ ਜਾਂ ਕੰਮ ਦੇ ਸਥਾਨ ਤੇ ਪੰਜਾਬੀ ਵਿੱਚ ਭੇਜ ਸਕਦੇ ਹੋ। ਕਿਉਂਕਿ ਇਹ ਸਾਫ਼ਟਵੇਅਰ ਤੁਹਾਨੂੰ ਪੰਜਾਬੀ ਯੂਨੀਕੋਡ ਫੌਂਟ (ਰਾਵੀ) ਵਿੱਚ ਕੰਮ ਕਰਨ ਦੀ ਸਹੂਲਤ ਉਪਲਬਧ ਕਰਵਾਉਂਦਾ ਹੈ।

APPENDIX – III

COMMONLY USED FULL FORMS

ACRONYM		FULL FORM
AI	:	ARTIFICIAL INTELLIGENCE
ARPANET	:	ADVANCED RESEARCH PROJECT AGENCY NETWORK
BMP	:	BITMAP PICTURE
bpi	:	BITS PER INCH
CD	:	COMPACT DISK
CPU	:	CENTRAL PROCESSING UNIT
CSS	:	CASCADING STYLE SHEET
CUI	:	CHARACTER USER INTERFACE
DOS	:	DISK OPERATING SYSTEM
DRAM	:	DYNAMIC RANDOM ACCESS MEMORY
DSL	:	DIGITAL SUBSCRIBER LINE
DVD	:	DIGITAL VIDEO DISK
E COMMERCE	:	ELECTRONIC COMMERCE
EEPROM	:	ELECTRONICALLY ERASABLE PROGRAMMABLE READ ONLY MEMROY
EMAIL	:	ELECTRONIC MAIL
EPROM	:	ERASABLE PROGRAMMABLE READ ONLY MEMROY
FTP	:	FILE TRANSFER PROTOCOL
GB	:	GIGABYTE
GIF	:	GRAPHICS INTERCHANGE FORMAT
GUI	:	GRAPHICAL USER INTERFACE
HTML	:	HYPER TEXT MARKUP LANGUAGE
IAP	:	INTERNET ACCESS PROVIDER
IBM	:	INTERNATIONAL BUSINESS MACHINE
IC	:	INTEGRATED CIRCUIT
ISDN	:	INTEGRATED SERVICE DIGITAL NETWORK

ISP	:	INTERNET SERVICE PROVIDER
IT	:	INFORMATION TECHNOLOGY
JPEG	:	JOINT PHOTOGRAPHIC EXPERT GROUP
KB	:	KILOBYTE
MB	:	MEGABYTE
MIDI	:	MUSICAL INSTRUMENT DIGITAL IDENTIFIER
MODEM	:	MODULATER DEMODULATER
MPEG	:	MOVING PICTURE EXPERT GROUP
MROM	:	MASKED READ ONLY MEMROY
NIC	:	NETWORK INTERFACE CARD
PB	:	PETA BYTE
PC	:	PERSONAL COMPUTER
PNG	:	PORTABLE NETWORK GRAPHICS
POP	:	POST OFFICE PROTOCOL
PROM	:	PROGRAMMABLE READ ONLY MEMROY
RAM	:	RANDOM ACCESS MEMORY
ROM	:	READ ONLY MEMORY
RTF	:	RICH TEXT FORMAT
SERP	:	SEARCH ENGINE RESULT PAGE
SMTP	:	SIMPLE MAIL TRANSFER PROTOCOL
SRAM	:	STATIC RANDOM ACCESS MEMORY
TB	:	TERABYTE
TCP/IP	:	TRANSMITION CONTROL PROTOCOL / INTERNET PROTOCOL
ULSI	:	ULTRA LARGE SCALE INTEGRATED CIRCUIT
UPS	:	UNINTERRUPTED POWER SUPPLY
URL	:	UNIFROM RESOURCE LOCATER
USB	:	UNIVERSAL SERIAL BUS
VLSI	:	VERY LARGE SCALE INTEGRATED CIRCUIT
WWW	:	WORLD WIDE WEB

APPENDIX – IV

COMMONLY USED SHORTCUT KEYS
(MS WORD)

Shortcut Keey	Used for
Ctrl+A	Select All
Ctrl+B	Bold the selected text
Ctrl+C	Copy the selected contents
Ctrl+D	Opens the Font Dialog Box
Ctrl+E	Center Align text
Ctrl+F	Find text
Ctrl+G	Goto line/page no etc.
Ctrl+H	Replace text
Ctrl+I	Italic the selected text
Ctrl+J	Justify paragraph
Ctrl+K	Create Hyperlink for the selected text
Ctrl+L	Left Align the paragraph
Ctrl+M	Increase Indent
Ctrl+N	Create a New File
Ctrl+O	Open Existing File
Ctrl+P	Print File
Ctrl+Q	Clear Indents and Tabs
Ctrl+R	Right Align the text
Ctrl+S	Save File
Ctrl+T	Increase Hanging Indent
Ctrl+U	Underline the selected contents
Ctrl+V	Paste the contents from the clipboard
Ctrl+W	Close File
Ctrl+X	Cut the selected contents
Ctrl+Y	Redo the last action (if possible)
Ctrl+Z	Undo the last operation
Ctrl+1	Single Line Spacing
Ctrl+2	Double Line Spacing
Ctrl+5	1.5 Line Spacing

Ctrl+]	Increase Font Size
Ctrl+[	Decrease Font Size
Ctrl+Shift+C	Copy the Formats of selected text
Ctrl+Shift+V	Paste the copied Formats on selected text
F3	Change Case
F7	Spelling and Grammar Check
Alt+F4	Close Program

COMMONLY USED SHORTCUT KEYS
(MS EXCEL)

Shortcut Key	Used for
Ctrl + A	Select the entire worksheet. If the cursor is currently placed within a table, press once to select the table, press one more time to select the whole worksheet.
Ctrl + B	Bold Text of Selected Cell/Cells
Ctrl + C	Copy the contents of the selected cells to Clipboard.
Ctrl + D	Copy the contents and format of the first cell in the selected range into the cells below. If more than one column is selected, the contents of the topmost cell in each column will be copied downwards.
Ctrl + F	Display the "Find" dialog box.
Ctrl + F1	Show / hide the Excel Ribbon.
Ctrl + G	Open the "Go to" dialog. Pressing F5 displays the same dialog.
Ctrl + N	Create a new workbook.
Ctrl + O	Open an existing workbook.
Ctrl + P	Open the "Print" dialog.
Ctrl + S	Save the active workbook.
Ctrl + T	"Convert selected cells to a table.
Ctrl + V	Pate contents of the Clipboard into the selected cell(s).
Ctrl + W	Close the active workbook.
Ctrl + X	Cut the contents of the selected cells to Clipboard.
Ctrl + Y	Repeat (Redo) the last action, if possible.
Ctrl + Z	Undo last action.
Ctrl + 1	Open the "Format Cells" dialog.
Ctrl + `	Toggle between displaying cell values and formulas.
Ctrl + ;	Enter the current date.
Ctrl + Shift + ;	Enter the current time
F2	Edit the current cell.
F4	Cycle through various combinations of formula reference types. Place the cursor within a cell and hit F4 to get the needed reference type: absolute, relative or mixed
F12	Displays the Save as dialog box.
Home	Return to the 1st cell of the current row in a worksheet.
Tab	Autocomplete the function name.

Ctrl + End	Move to the last used cell of the current worksheet
Ctrl + Home	Move to the beginning of a worksheet
Ctrl + PgDown	Switch to the next worksheet
Ctrl + PgUp	Switch to the previous worksheet
Alt + Enter	In cell editing mode, enter a new line (carriage return) into a cell.
Ctrl + Space	Select the entire column.
Shift + Space	Select the entire row.

WORKSHEET