Computer Fundamentals and Programming Methodology Using 'C'

Master of Computer Applications
TMC 101

Syllabus

- **UNIT-1: Fundamentals of Information Technology:** Introduction of information technology, computer and its characteristics, input/output devices, introduction of software, number system, introduction of communication and network, operating system, database management systems.

- **Basic Programming Concepts:** Basics of problem-solving tools, program design methodology; Pseudo code, Flowcharts, Algorithms, Translators, Programming Paradigms, Programming languages and their characteristics.

- UNIT-2 : **Introduction to C Language:** History of C and its characteristics, Character Set, Identifiers, Variables, Assigning a Name to a Variable, Variable Declarations, Keywords, Tokens, Data Types, Constants, String Constant, Numeric Constant, Structure of C Program, Comment styles in 'C', Storage classes, type conversion.

Syllabus

UNIT-3: Operators and Input/output: Instruction and its Types, Operators and Hierarchy of Operations, Bitwise operators, Expressions in C, Formatted and unformatted Input/output.
Flow Control: Control and Repetitive Statements, break, continue, Local and Global Variables.

UNIT-4:   Function and Arrays: User defined functions and library functions, Recursion and Recursive Function, Call by Value Versus Call by Reference, parameter passing, arrays and functions, **Pointers,** dynamic memory management, A**rrays:** 1D and 2D arrays.

UNIT-5: Strings: Declaring and initialing string variables, reading and writing of string, string handling functions.
Structure and Union: self-referential structures, pointer to structures.
File: Types of files, various modes to open a file. Writing and Reading data, random access of files, command line arguments.
Preprocessor Directives.

Introduction

A computer is an electronic device which basically performs five major operations which includes:

1) accepts data or instructions (input)
2) stores data
3) process data
4) displays results (output) and
5) controls and co-ordinates all operations inside a computer

Computer characteristics

1) **Automatic:** Given a job, computer can work on it automatically without human interventions

2) **Speed:** Computer can perform data processing jobs very fast, usually measured in **microseconds** (10^{-6}), **nanoseconds** (10^{-9}), and **picoseconds** (10^{-12})

3) **Accuracy:** Accuracy of a computer is consistently high and the degree of its accuracy depends upon its design. Computer errors caused due to incorrect input data or unreliable programs are often referred to as *Garbage- In- Garbage-Out* (GIGO)

4)Diligence: Computer is free from monotony, tiredness, and lack of concentration. It can continuously work for   hours without creating any error and without grumbling

5)Versatility: Computer is capable of performing almost any task, if the task can be reduced to a finite series of   logical steps

6)Power of Remembering: Computer can store and   recall any amount of information because of its   secondary storage capability. It forgets or looses certain   information only when it is asked to do so

7)No I.Q.: A computer does only what it is programmed   to do. It cannot take its own *decision* in this regard

8)No Feelings: Computers are devoid of emotions. Their judgement is based on the instructions given to them in   the form of programs that are written by us (human beings)

Data is a collection of unorganized facts & figures and does not provide any further information regarding patterns, context, etc. Hence data means "unstructured facts and figures".

Information is a structured data i.e. organized meaningful and processed data. To process the data and convert into information, a computer is used.

Input Device

- **Parts of Computer, which are used to feed data and commands are called Input Devices**

- Some Input Devices are:
- Keyboard
- Mouse
- Joy Stick
- Scanner
- Voice Input System
- Bar Code Readers
- Optical Mark Reader (OMR)

Output Device

- Output devices receive information from the CPU and present it to the user in the desired Form. Output devices include

- Monitor or Visual Display Unit

- Speakers

- printers

Software Applications

- **<u>System Software:</u>**

- System software: System software is a collection of programs and utilities for providing service to other programs. Other system applications (e.g., operating system components, drivers, telecommunications processors) process largely indeterminate data.

- In either case, the system software area is characterized by heavy interaction with computer hardware; heavy usage by multiple users; concurrent operation that requires scheduling, resource sharing, and sophisticated process management; complex data structures; and multiple external interfaces.

Ex. Compilers, operating system, drivers etc.

Engineering /Scientific software:

Engineering and scientific software: Engineering and scientific software have been characterized by "number crunching" algorithms. Applications range from astronomy to volcano logy, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing. However, modern applications within the engineering/scientific area are moving away from conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take on real-time and even system software characteristics.
Ex. Computer Aided Design (CAD), system stimulation etc.

- **Embedded Software:**

- Embedded software: Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets. Embedded software can perform very limited and esoteric functions (e.g., keypad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems).

- **Personal computer software:**
- The personal computer is the type of computer, which gave revolution to the information technology. The personal computer software market has burgeoned over the past two decades. Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications, external network, and database access are only a few of hundreds of applications.

- **Artificial Intelligence software**
- Artificial intelligence (AI) software is the software, which thinks and behaves like a human. AI software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Expert systems, also called knowledge-based systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing are representative of applications within this category.

Ex. Robotics, expert system, game playing, etc.

- **Web based Software**
- Web-based software: The Web pages processed by the browser are the software that incorporates executable instructions (e.g., CGI, HTML, PERL, or Java), and data (e.g. hypertext and a variety of visual and audio formats). In essence, the network becomes a massive computer providing an almost unlimited software resource that can be accessed by anyone with a modem.

Real-time software: Software for the monitors/analyzes/controls real-world events as they occur is called real time. Elements of real-time software include a data-gathering component that collects and formats information from an external environment, an analysis component that transforms information as required by the application, a control/output component that responds to the external environment, and a monitoring component that coordinates all other components so that real-time response can be maintained.

Business software: Business information processing is the largest single software application area. In a broad sense, **business software is an integrated software and has many components related to a particular field of the business**.

Discrete "systems" for example, payroll, accounts receivable/payable, inventory have evolved into management information system (MIS) software that accesses one or more large databases containing business information. Applications in this area restructure existing data in a way that facilitates business operations or management decision-making. In addition to conventional data processing application, business software applications also encompass interactive computing

Applications
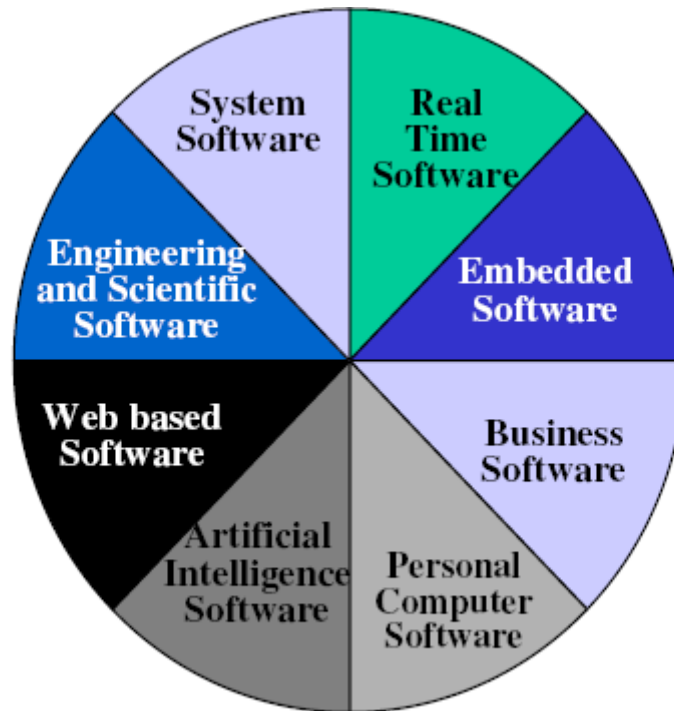
These are various ranges of applications of software.

Database

A database is a collection of inter-related data which helps in the efficient retrieval, insertion, and deletion of data from the database and organizes the data in the form of tables, schemas, reports, etc.

For Example, a university database organizes the data about students, faculty, admin staff, etc. which helps in the efficient retrieval, insertion, and deletion of data from it

Data Definition Language (DDL) - Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

CREATE: to create a database and its objects like (table, index, views, store procedure, function, and triggers)

ALTER: alters the structure of the existing database

DROP: delete objects from the database

RENAME: rename an object

Data Manipulation Language (DML) deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- SELECT: retrieve data from a database
- INSERT: insert data into a table
- UPDATE: updates existing data within a table
- DELETE: Delete all records from a database table
- LOCK TABLE: concurrency Control

Data Control Language (DCL) which acts as an access specifier to the database.(basically to grant and revoke permissions to users in the database

- GRANT: grant permissions to the user for running DML(SELECT, INSERT, DELETE,…) commands on the table

- REVOKE: revoke permissions to the user for running DML(SELECT, INSERT, DELETE,…) command on the specified table

Database Management System

- **Database Management System:** The software which is used to manage databases is called Database Management System (DBMS). For Example, MySQL, Oracle, etc. are popular commercial DBMS used in different applications. DBMS allows users the following tasks:

- **Data Definition:** It helps in the creation, modification, and removal of definitions that define the organization of data in the database.

- **Data Updation:** It helps in the insertion, modification, and deletion of the actual data in the database.

- **Data Retrieval:** It helps in the retrieval of data from the database which can be used by applications for various purposes.

- **User Administration:** It helps in registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information corrupted by unexpected failure.

Shift from File System to DBMS

Disadvantages in File Processing
Example file-based University Management System.

- **Data redundancy** Data is said to be redundant if the same data is copied at many places. If a student wants to change their Phone number, he or has to get it updated in various sections. Similarly, old records must be deleted from all sections representing that student.

- **Inconsistency of Data:** Data is said to be inconsistent if multiple copies of the same data do not match each other. If the Phone number is different in Accounts Section and Academics Section, it will be inconsistent. Inconsistency may be because of typing errors or not updating all copies of the same data.

- **Difficult in accessing data-** A user should know the exact location of the file to access data, so the process is very cumbersome and tedious.

- **Data isolation-** Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult

- **Concurrent access is not possible.-** The access of the same data by multiple users at the same time is known as concurrency. The file system does not allow concurrency as data can be accessed by only one user at a time.

- **Security problems**- Not every user of the database system should be able to access all the data. For example, in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult. *File Systems may lead to unauthorized access to data*

- **No Backup and Recovery-**The file system does not incorporate any backup and recovery of data if a file is lost or corrupted

- *These difficulties, among others, prompted the development of database systems.*

Advantages of DBMS:

- **Controlling of Redundancy:** Data redundancy refers to the duplication of data (i.e storing same data multiple times). In a database system, by having a centralized database and centralized control of data by the DBA the unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space.

- **Improved Data Sharing** : DBMS allows a user to share the data in any number of application programs.

Data Integrity : Integrity means that the data in the database is accurate. Centralized control of the data helps in permitting the administrator to define integrity constraints to the data in the database. For example: in customer database we can enforce an integrity that it must accept the customer only from Dehradun and Delhi city.

Security : Having complete authority over the operational data, enables the DBA in ensuring that the only mean of access to the database is through proper channels. The DBA can define authorization checks to be carried out whenever access to sensitive data is attempted

Data Consistency : By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example: is a customer address is stored only once, we cannot have disagreement on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage that results from redundant data storage.

Enforcements of Standards : With the centralized of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc.

Reduced Application Development and Maintenance Time : DBMS supports many important functions that are common to many applications, accessing data stored in the DBMS, which facilitates the quick development of application.

Disadvantages of DBMS

- **It is bit complex-** Since it supports multiple functionality to give the user the best, the underlying software has become complex. The designers and developers should have thorough knowledge about the software to get the most out of it.

- Because of its complexity and functionality, it uses large amount of memory. It also needs large memory to run efficiently.

- DBMS system works on the centralized system, i.e.; all the users from all over the world access this database. Hence any failure of the DBMS, will impact all the users

Number System

Decimal Number system

The number system that we use in our day-to-day life is the decimal number system. **Decimal number system has base 10 as it uses 10 digits from 0 to 9.** In decimal number system, the successive positions to the left of the decimal point represent units, tens, hundreds, thousands, and so on.

The base of the number system (where the base is defined as the total number of digits available in the number system)

Each position represents a specific power of the base (10).

For example, the decimal number 1234 consists of the digit

4 in the units position, 3 in the tens position, 2 in the hundreds position, and 1 in the thousands position.   Its value can be written as

(1 x 1000)+ (2 x 100)+ (3 x 10)+ (4 x l)
(1 x 103)+ (2 x 102)+ (3 x 101)+ (4 x l00)
1000 + 200 + 30 + 4
1234

Decimal Number System

Example

$$2586_{10} = (2 \times 10^3) + (5 \times 10^2) + (8 \times 10^1) + (6 \times 10^0)$$

$$= 2000 + 500 + 80 + 6$$

S.No.	Number System and Description
1	**Binary Number System** Base 2. Digits used : 0, 1
2	**Octal Number System** Base 8. Digits used : 0 to 7
3	**Hexa Decimal Number System** Base 16. Digits used: 0 to 9, Letters used : A- F

Binary Number System

The base 2 number system is also known as the Binary number system wherein, only two binary digits exist, i.e., 0 and 1.

Binary Number System

- The maximum value of a single digit is 1 (one less than the value of the base)

- A positional number system

- Has only 2 symbols or digits (0 and 1). base = 2

- Each position of a digit represents a specific power of the base (2)

- This number system is used in computers

Example

$10101_2 = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) \times (1 \times 2^0)$

$= 16 + 0 + 4 + 0 + 1$

$= 21_{10}$

Octal Number System

Characteristics

- A positional number system

- Has total 8 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7). Hence, its base = 8

- The maximum value of a single digit is 7 (one less than the value of the base

- Each position of a digit represents a specific power of the base (8)

Octal Number System

Characteristics

- A positional number system

- Has total 8 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7). Hence, its base = 8

- The maximum value of a single digit is 7 (one less than the value of the base

- Each position of a digit represents a specific power of the base (8)

Octal Number System

- Since there are only 8 digits, 3 bits ($2^3 = 8$) are sufficient to represent any octal number in binary

Example

$$2057_8 = (2 \times 8^3) + (0 \times 8^2) + (5 \times 8^1) + (7 \times 8^0)$$

$$= 1024 + 0 + 40 + 7$$

$$= 1071_{10}$$

Hexadecimal Number System

- Each position of a digit represents a specific power of the base (16)

- Since there are only 16 digits, 4 bits ($2^4 = 16$) are sufficient to represent any hexadecimal number in binary

Example

$1AF_{16}$ $=$ $(1 \times 16^2) + (A \times 16^1) + (F \times 16^0)$

$=$ $1 \times 256 + 10 \times 16 + 15 \times 1$

$=$ $256 + 160 + 15$

$=$ 431_{10}

Number System

Decimal Numbers
Base 10 (0-9)

Binary Numbers
Base 2 (0,1)

Octal Numbers
Base 8 (0-7)

Hexadecimal Numbers
Base 16 (0-9,A-F)

Steps for Conversion of Binary to Decimal Number System

To convert a number from the binary to the decimal system, we use the following steps.
Step 1: Multiply each digit of the given number, starting from the rightmost digit, with the exponents of the base.
Step 2: The exponents should start with 0 and increase by 1 every time we move from right to left.
Step 3: Simplify each of the above products and add them.

$$100111_2$$

$1 \times 2^0 = 1 \times 1 = 1$

$1 \times 2^1 = 1 \times 2 = 2$

$1 \times 2^2 = 1 \times 4 = 4$

$0 \times 2^3 = 0 \times 8 = 0$

$0 \times 2^4 = 0 \times 16 = 0$

$1 \times 2^5 = 1 \times 32 = 32$

Sum: 39

$100111 = (1 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$
$= (1 \times 32) + (0 \times 16) + (0 \times 8) + (1 \times 4) + (1 \times 2) + (1 \times 1)$
$= 32 + 0 + 0 + 4 + 2 + 1$
$= 39$
Thus, $100111_2 = 39_{10}$.

Conversion of Decimal Number System to Binary / Octal / Hexadecimal Number System

**Example:** Convert 4320_{10} into the octal system.

Step 1: Identify the base of the required number. Since we have to convert the given number into the octal system, the base of the required number is 8.

Step 2: Divide the given number by the base of the required number and note down the quotient and the remainder in the quotient-remainder form. Repeat this process (dividing the quotient again by the base) until we get the quotient less than the base.

$$8 \mid 4\,3\,2\,0$$

$$8 \mid 5\,4\,0 - 0$$

$$8 \mid 6\,7 - 4$$

$$8 \mid 8 - 3$$

$$1 - 0$$

Step 3: The given number in the octal number system is obtained just by reading all the remainders and the last quotient from bottom to top.

Therefore, $4320_{10} = 10340_8$

$8 \mid 4\,3\,2\,0$

$8 \mid 5\,4\,0 - 0$

$8 \mid 6\,7 - 4$

$8 \mid 8 - 3$

$1 - 0$
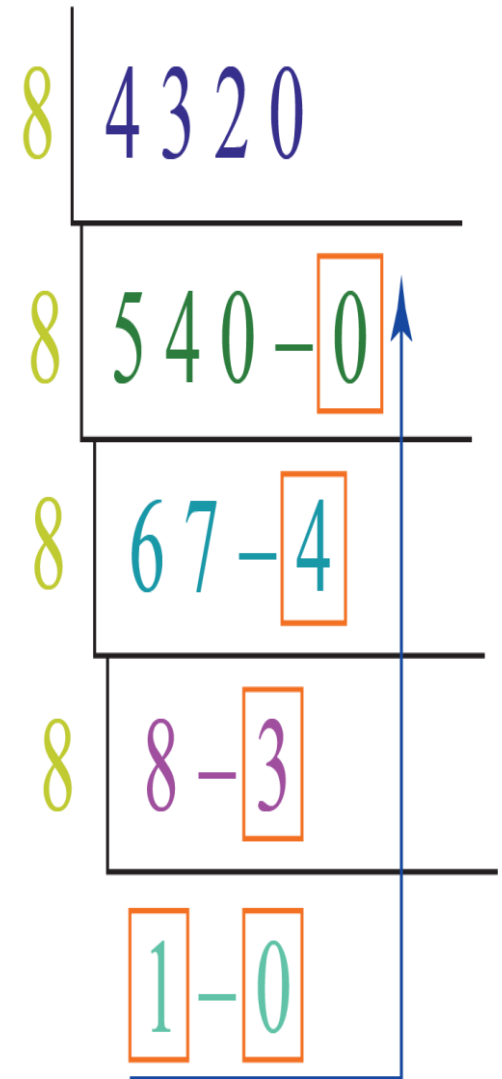
Conversion from One Number System to Another Number System

To convert a number from one of the binary/octal/hexadecimal systems to one of the other systems, we first convert it into the decimal system, and then we convert it to the required systems by using the above-mentioned processes.

Example: Convert 1010111100_2 to the hexadecimal system.
**Step 1:** Convert this number to the decimal number system

$$1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0$$

$$0 \times 2^0 = 0 \times 1 = 0$$
$$0 \times 2^1 = 0 \times 2 = 0$$
$$1 \times 2^2 = 1 \times 4 = 4$$
$$1 \times 2^3 = 1 \times 8 = 8$$
$$1 \times 2^4 = 1 \times 16 = 16$$
$$1 \times 2^5 = 1 \times 32 = 32$$
$$0 \times 2^6 = 0 \times 64 = 0$$
$$1 \times 2^7 = 1 \times 128 = 128$$
$$0 \times 2^8 = 0 \times 256 = 0$$
$$1 \times 2^9 = 1 \times 512 = 512$$

Sum: 700

Thus, $1010111100_2 = 700_{10} \rightarrow (1)$

Step 2: Convert the above number (which is in the decimal system), into the required number system (hexadecimal).
Here, we have to convert 700_{10} into the hexadecimal system using the above-mentioned process. It should be noted that in the hexadecimal system, the numbers 11 and 12 are written as B and C respectively.

$$16 \mid 7\ 0\ 0$$
$$16 \mid 4\ 3 - 12\,(\text{or})\ \boxed{C} \uparrow$$
$$\boxed{2} - 11\ (\text{or})\ \boxed{B}$$
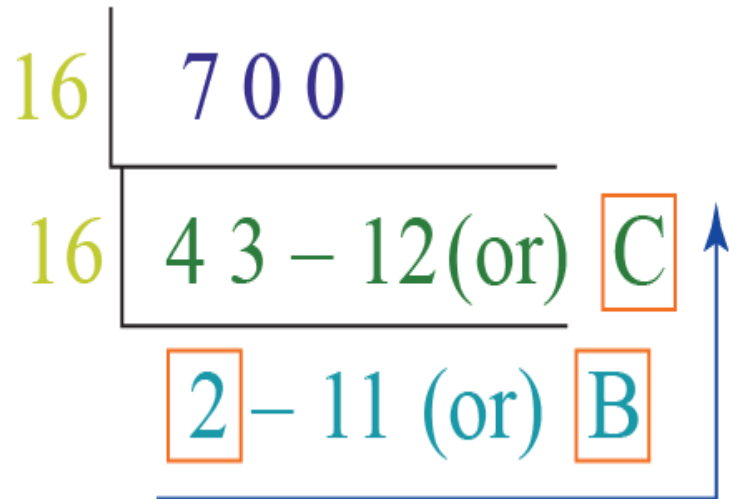
Thus, $700_{10} = 2BC_{16} \rightarrow (2)$

Computer Languages

- Machine language

- Assembly language

- High-level languag

Machine Language

- Only language of a computer understood by it without using a translation program

- Normally written as strings of binary 1s and 0s

Advantage

- Can be executed very fast

Limitations

- Machine Dependent

- Difficult to program

- Error prone

- Difficult to modify

# Assembly/Symbolic Language

- Programming language that overcomes the limitations of machine language programming by:

- Using alphanumeric mnemonic codes instead of numeric codes

- Allowing storage locations to be represented in form of alphanumeric addresses instead of numeric addresses e.g. representing memory locations 1000, 1001, and 1002 as **FRST**, **SCND**, and **ANSR** respectively

```
START  PROGRAM AT 0000
START  DATA AT 1000
SET ASIDE AN ADDRESS FOR FRST
SET ASIDE AN ADDRESS FOR SCND
SET ASIDE AN ADDRESS FOR ANSR
CLA FRST
ADD SCND
STA ANSR
HLT
```

Sample assembly language program for adding two numbers and storing the result

Assembler

- Software that translates as assembly language program into an equivalent machine language program of a computer

Assembly language program	Input →	Assembler	Output →	Machine language program

(Source Program) —— One-to-one correspondence ——→ (Object Program)
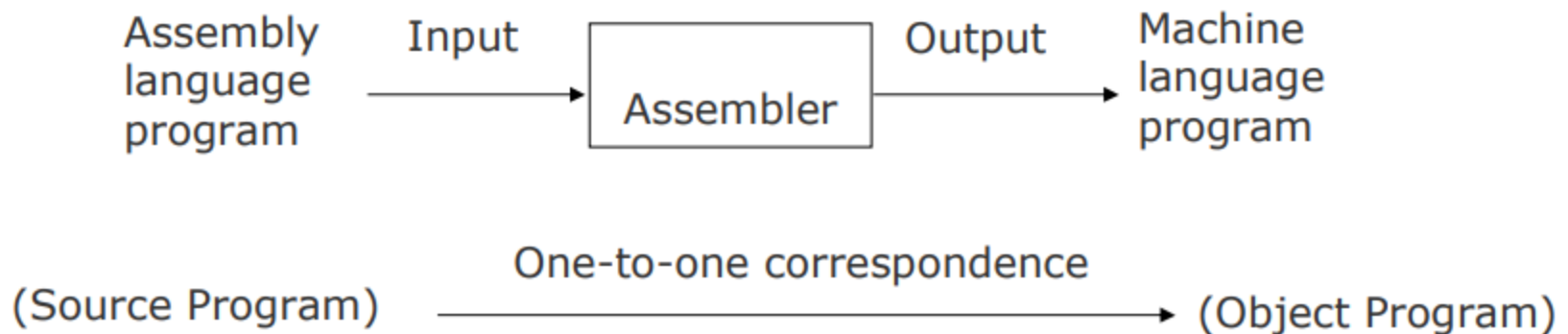
Advantages and disadvantages of Assembly Language Over Machine Language

- Easier to understand and use
- Easier to locate and correct errors
- Easier to modify

Demerits

- Machine dependent
- Knowledge of hardware required
- Machine level coding

High-Level Languages

- Machine independent

- Do not require programmers to know anything about the internal structure of computer on which high-level language programs will be executed

- Deal with high-level coding, enabling the programmers to write instructions using English words and familiar mathematical symbols and expressions
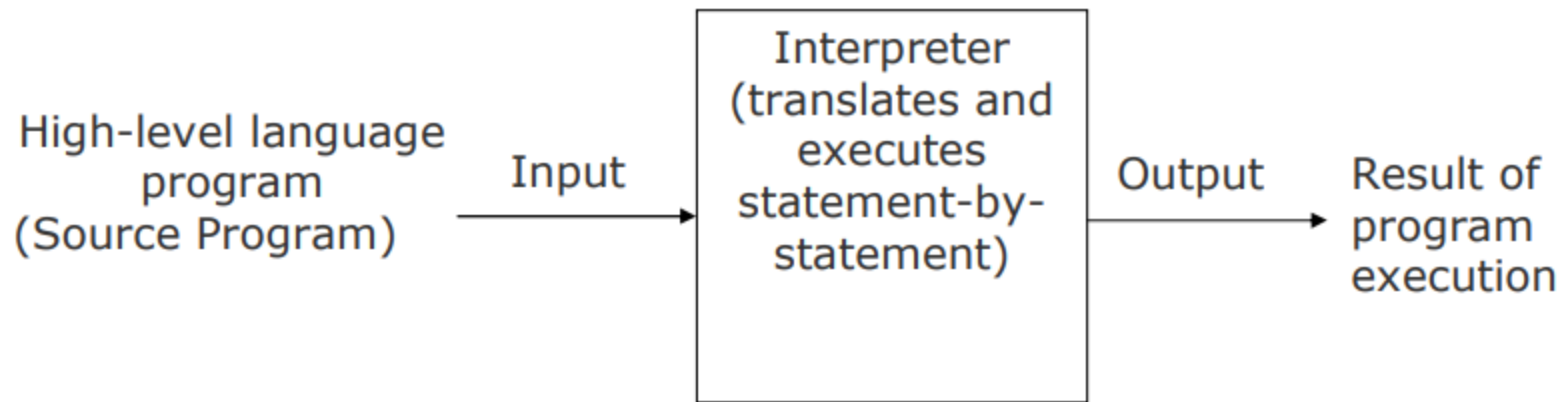
Compiler

- Translator program (software) that translates a high level language program into its equivalent machine language program

- Compiles a set of machine language instructions for every program instruction in a high-level language

High-level language program → Input → Compiler → Output → Machine language program

Interpreter

- Interpreter is a high-level language translator
- Takes one statement of a high-level language program, translates it into machine language instructions
- Immediately executes the resulting machine language instructions
- Compiler simply translates the entire source program into an object program and is not involved in its execution

Interpreter

High-level language program (Source Program) → Input → Interpreter (translates and executes statement-by-statement) → Output → Result of program execution

Merits and demerits

- Machine independent
- Easier to learn and use
- Fewer errors during program development
- Lower program preparation cost
- Better documentation
- Easier to maintain

Demerits
Lower execution efficiency
Less flexibility to control the computer's CPU, memory and registers

ALGORITHM

- An algorithm is a sequence of steps to solve a particular problem or algorithm is an ordered set of unambiguous steps that produces a result and terminates in a finite time

Algorithm

- Algorithm refers to the logic of a program
- It is a step-by-step description of how to arrive solution to a given problem
- Is defined as a sequence of instructions that when executed in the specified sequence, the desired results are obtained

Instructions must possess following characteristics:

- Each instruction should be precise and unambiguous

- Each instruction should be executed in a finite time

  - No instruction should be repeated infinitely. This ensures that the algorithm terminates ultimately.

  - After executing the instructions (when the algorithm terminates), the desired results are obtained

Characteristics of a good algorithm

- Precision — the steps are precisely stated or defined.

- Uniqueness — results of each step are uniquely defined and only depend on the input and the result of the preceding steps.

- Finiteness — the algorithm always stops after a finite number of steps. •

- Input — the algorithm receives some input.

- Output — the algorithm produces some output.

Algorithm to find the sum of two numbers

Step-1 Start

Step-2 Input first numbers say A

Step-3 Input second number say B

Step-4 SUM = A + B

Step-5 Display SUM

Step-6 Stop

Algorithm
Step-1 Start
Step-2 Input two numbers say A & B
Step-3 SUM = A + B
Step-4 Display SUM
Step-5 Stop

Algorithm to find Simple Interest

P : Principle Amount
N : Time in Years
R : % Annual Rate of Interest
SI : Simple Interest

Algorithm
Step-1 Start
Step-2 Input value of P, N, R
Step-3 SI = (P x N x R)/100.0
Step-4 Display SI
Step-6 Stop

Algorithm to convert temperature from Celsius to Fahrenheit

C: temperature in Celsius

F : temperature Fahrenheit

Algorithm

Step-1 Start

Step-2 Input temperature in Celsius say C

Step-3 F = (9/5 x C) + 32

Step-4 Display Temperature in Fahrenheit F

Step-5 Stop

Algorithm to find Area and Perimeter of Square

L : Side Length of Square
AREA : Area of Square
PERIMETER : Perimeter of Square
Algorithm
Step-1 Start
Step-2 Input Side Length of Square say L
Step-3 Area = L x L
Step-4 PERIMETER = 4 x L
Step-5 Display AREA, PERIMETER
Step-6 Stop

Algorithm to find Area and Perimeter of Rectangle

L : Length of Rectangle
B : Breadth of Rectangle
AREA : Area of Rectangle
PERIMETER : Perimeter of Rectangle
Algorithm
Step-1 Start
Step-2 Input Side Length & Breadth say L, B
Step-3 Area = L x B
Step-4 PERIMETER = 2 x (L + B)
Step-5 Display AREA, PERIMETER
Step-6 Stop

Algorithm to find Area and Perimeter of Circle

R : Radius of Circle
AREA : Area of Circle
PERIMETER : Perimeter of Circle
Algorithm
Step-1 Start
Step-2 Input Radius of Circle say R
Step-3 Area = 22/7 x R x R
Step-4 PERIMETER = 2 x 22/7 x R
Step-5 Display AREA, PERIMETER
Step-6 Stop

WRITE AN ALGORITHM TO INPUT A NUMBER AND CHECK NUMBER IS ZERO OR NOT .

Step 1: Start

Step 2: Read a.

Step 3: If a==0 then

      Display "a is Zero".

      else then

      Display "a is not Zero".

Step 4: Stop.

WRITE AN ALGORITHM TO INPUT A NUMBER AND CHECK NUMBER IS ODD OR EVEN .

Step 1: Start

Step 2: Read a .

Step 3: If a %2==0 then

   Display "a is even number".

   else then

   Display "a is odd number".

Step 4: Stop.

WRITE AN ALGORITHM TO PRINT THE LARGER OF TWO NUMBERS

Step 1: Start

Step 2: Read a, b .

Step 3: If a>b then

Display "a is the largest number".

else then

Display "b is the largest number".

Step 4: Stop.

There are 50 students in a class who appeared in their final examination. Their mark sheets have been given to you.

The division column of the mark sheet contains the division (FIRST, SECOND, THIRD or FAIL) obtained by the student.

Write an algorithm to calculate and print the total number of students who passed in FIRST division.

Step 1: Initialize Total_First_Division and Total_Marksheets_Checked to zero.

Step 2: Take the mark sheet of the next student.

Step 3: Check the division column of the mark sheet to see if it is FIRST, if no, go to Step 5.

Step 4: Add 1 to Total_First_Division.

Step 5: Add 1 to Total_Marksheets_Checked.

Step 6: Is Total_Marksheets_Checked = 50, if no, go to Step 2.

 Step 7: Print Total_First_Division.

Step 8: Stop.

There are 100 employees in an organization. The organization wants to distribute annual bonus to the employees based on their performance. The performance of the employees is recorded in their annual appraisal forms.

Every employee's appraisal form contains his/her basic salary and the grade for his/her performance during the year.

The grade is of three categories – 'A' for outstanding performance, 'B' for good performance, and 'C' for average performance.

It has been decided that the bonus of an employee will be 100% of the basic salary for outstanding performance, 70% of the basic salary for good performance, 40% of the basic salary for average performance, and zero for all other cases.

Write an algorithm to calculate and print the total bonus amount to be distributed by the organization.

Step 1: Initialize Total_Bonus and Total_Employees_Checked to zero.

Step 2: Initialize Bonus and Basic_Salary to zero.

Step 3: Take the appraisal form of the next employee.

Step 4: Read the employee's Basic_Salary and Grade.

Step 5: If Grade = A, then Bonus = Basic_Salary. Go to Step 8.

Step 6: If Grade = B, then Bonus = Basic_Salary x 0.7. Go to Step 8.

Step 7: If Grade = C, then Bonus = Basic_Salary x 0.4.

Step 8: Add Bonus to Total_Bonus.

Step 9: Add 1 to Total_Employees_Checked.

Step 10: If Total_Employees_Checked < 100, then go to Step 2.

Step 11: Print Total_Bonus.

Step 12: Stop.

Flowchart

- Flowchart is a pictorial representation of an algorithm

- Flowchart is often considered as a blueprint of a design used for solving a specific problem.

- Uses symbols (boxes of different shapes) that have standardized meanings to denote different types of instruction

- Actual instructions are written within the boxes

- Boxes are connected by solid lines having arrow marks to indicate the exact sequence in which the instructions are to be executed

- Process of drawing a flowchart for an algorithm is called **flowcharting**

Basic Flowchart Symbols
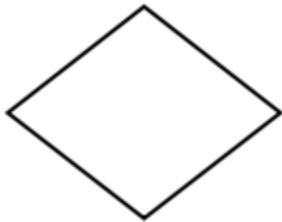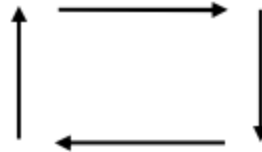

Terminal   Input/Output   Processing

Decision   Flow lines   Connectors
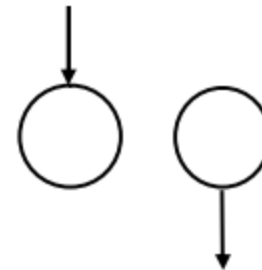
The rounded rectangles, or terminal points, indicate the flowchart's starting and ending points. The parallelograms designate input or output operations. The rectangle depicts a process such as a mathematical computation, or a variable assignment. Arrows that indicate the direction of program flow

Symbol	Symbol Name	Purpose
	Terminal/Terminator (start/stop)	Used at the beginning and end of the algorithm to show start and end of the program.
	Process	Indicates processes like mathematical operations.
	Input/ Output	Used for denoting program inputs and outputs.
	Decision	Stands for decision statements in a program, where answer is usually Yes or No.
	Arrow	Shows relationships between different shapes.
	On-page Connector	Connects two or more parts of a flowchart, which are on the same page.
	Off-page Connector	Connects two parts of a flowchart which are spread over different pages.

Connectors : Sometimes a flowchart is broken into two or more smaller flowcharts. This is usually done when a flowchart does not fit on a single page, or must be divided into sections. A connector symbol, which is a small circle with a letter or number inside it, allows you to connect two flowcharts
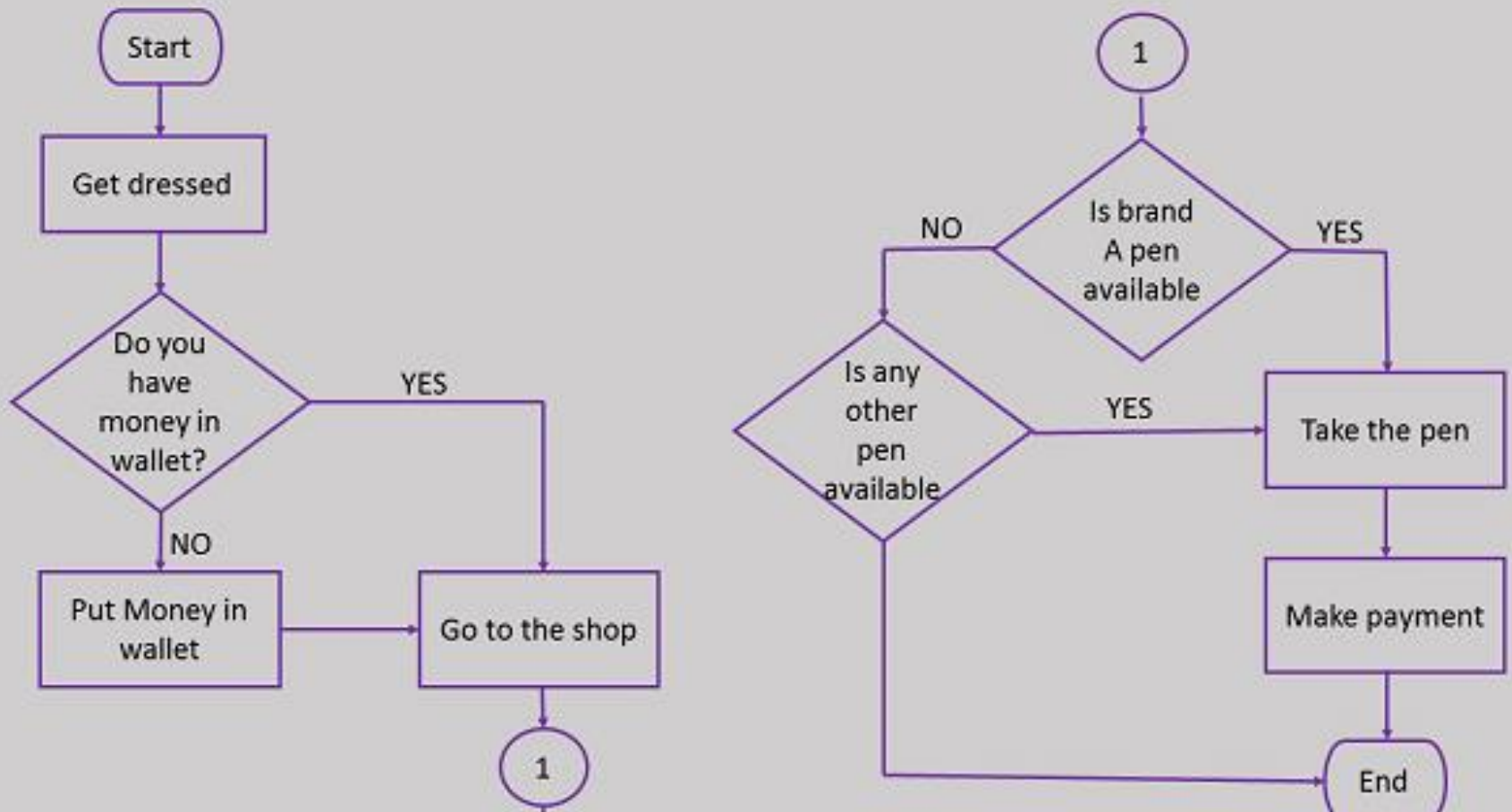
Advantages of Flowchart:
•Flowcharts are a better way of communicating the logic of the system.
•Flowcharts act as a guide for blueprint during program designed.
•Flowcharts help in debugging process.
•With the help of flowcharts programs can be easily analyzed.
•It provides better documentation.
•Flowcharts serve as a good proper documentation.
•Easy to trace errors in the software.
•Easy to understand.
•The flowchart can be reused for inconvenience in the future.
•It helps to provide correct logic.

Limitation of flowchart

- Flowcharts are very time consuming and laborious to draw (especially for large complex programs)
- Redrawing a flowchart for incorporating changes/ modifications is a tedious task
- There are no standards determining the amount of detail that should be included in a flowchart

Flowchart to find the sum of two numbers
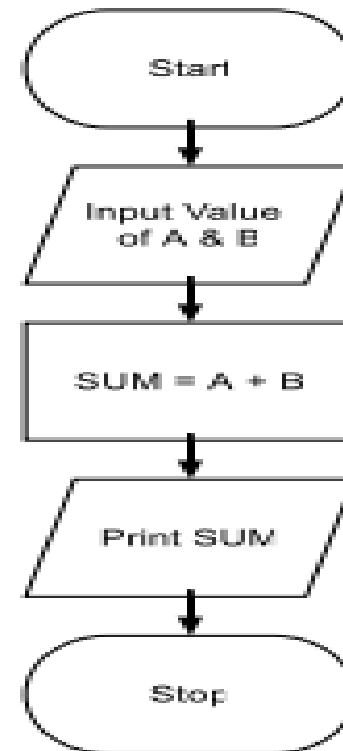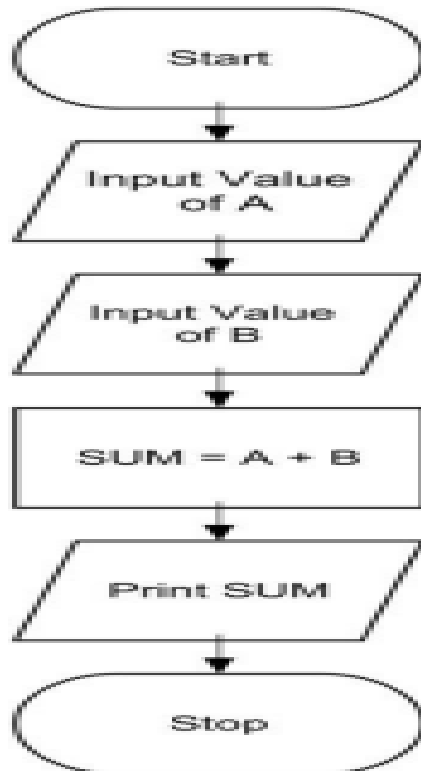
Average of two numbers
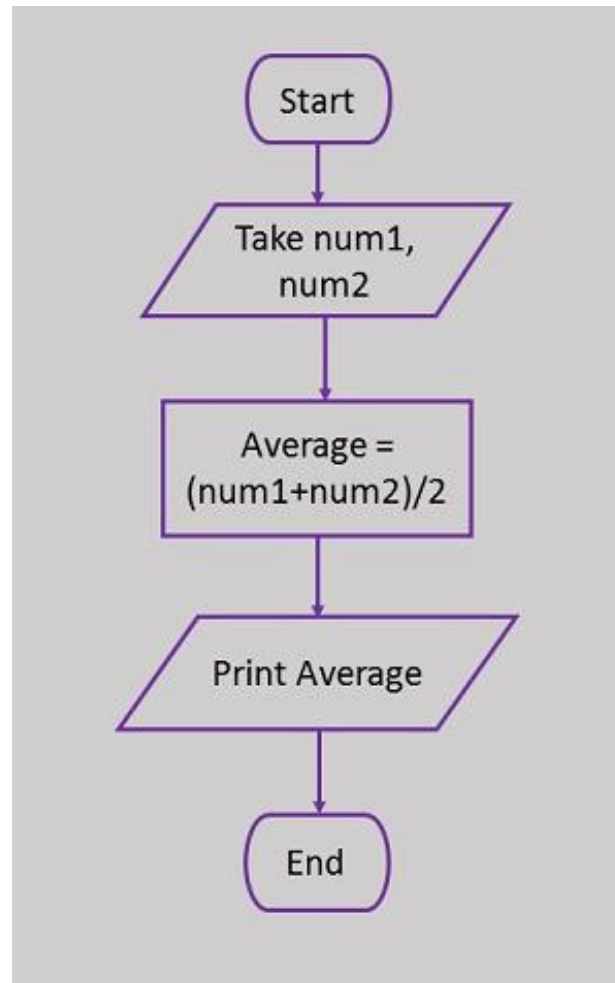
Swap Two Numbers using Temporary Variables

Step-1 Start

Step-2 Input Two Numbers Say NUM1,NUM2

Step-3 Display Before Swap Values NUM1, NUM

Step-4 TEMP = NUM1 Step-5 NUM1 = NUM2

Step-6 NUM2 = NUM1 Step-7 Display After Swa

Values NUM1,NUM

Step-8 Stop

Swap Two Numbers without using temporary variable

Step-1 : Start
Step-2 : Input Two Numbers Say A,B
Step-3 : Display Before Swap Values A, B
Step-4 : A = A + B
 Step-5 : B = A - B
Step-6 : A = A - B
Step-7 : Display After Swap Values A, B
Step-8 : Stop

```
                    ┌─────────────────┐
                   (      Start        )
                    └─────────────────┘
                            │
                            ▼
                    ╱─────────────────╲
                   ╱    Input A, B      ╲
                  ╱───────────────────────╲
                            │
                            ▼
                    ╱─────────────────╲
                   ╱    Write Value     ╲
                  ╱      of A, B         ╲
                 ╱      befoew swap       ╲
                            │
                            ▼
                  ┌───────────────────────┐
                  │      A = A + B         │
                  │      B = A – B         │
                  │      A = A –B          │
                  └───────────────────────┘
                            │
                            ▼
                    ╱─────────────────╲
                   ╱   Write Value of   ╲
                  ╱        A,B           ╲
                 ╱      after Swap        ╲
                            │
                            ▼
                    ┌─────────────────┐
                   (      Stop         )
                    └─────────────────┘
```
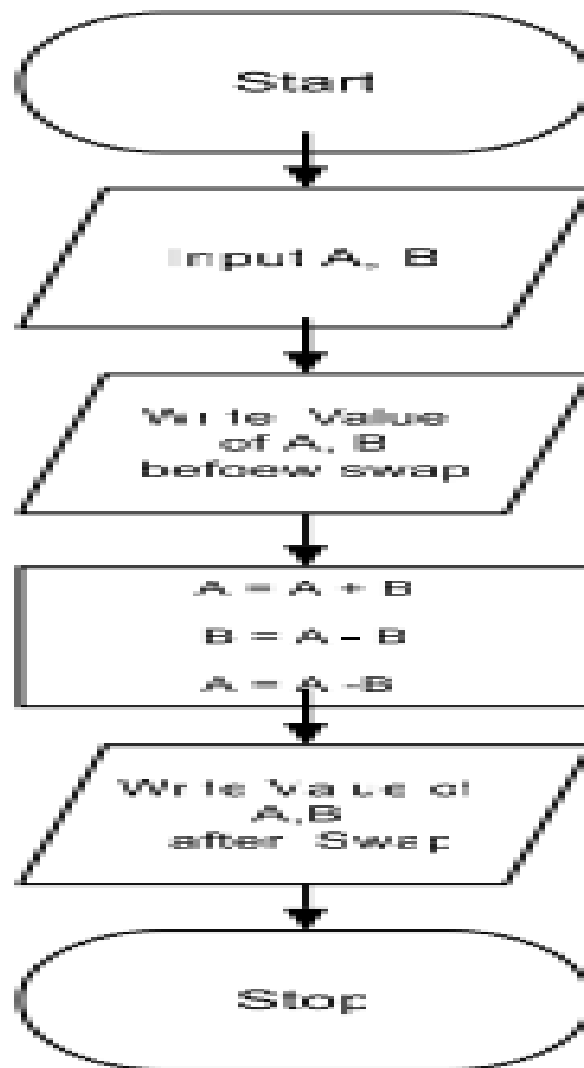
control structures

The algorithm and flowchart include following three types of control structures.

1. **Sequence:** In the sequence structure, statements are placed one after the other and the execution takes place starting from up to down.

2. **Branching (Selection):** In branch control, there is a condition and according to a condition, a decision of either TRUE or FALSE is achieved. In the case of TRUE, one of the two branches is explored; but in the case of FALSE condition, the other alternative is taken. Generally, the 'IF-THEN' is used to represent branch control.

3. **Loop (Repetition):** The Loop or Repetition allows a statement(s) to be executed repeatedly based on certain loop condition e.g. WHILE, FOR loops

Decision Symbols
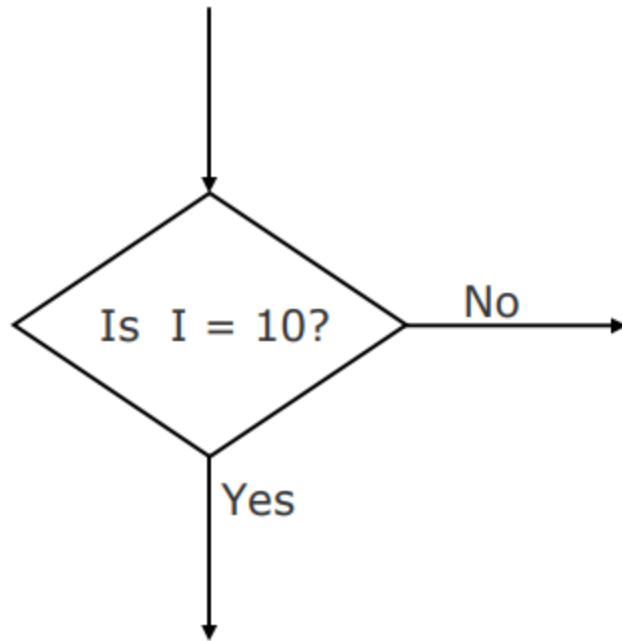
(a) A two-way branch decision.
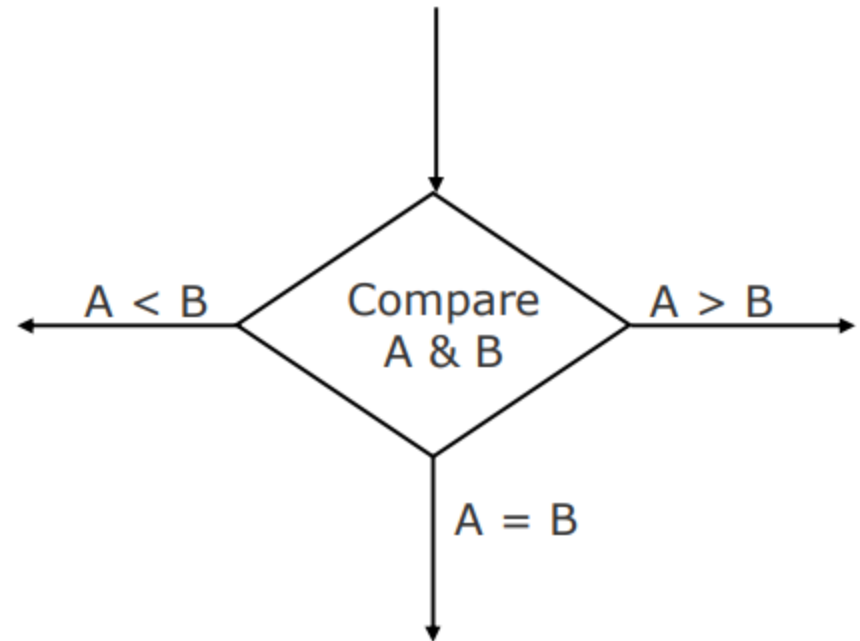
(b) A three-way branch decision.

Decision Symbol
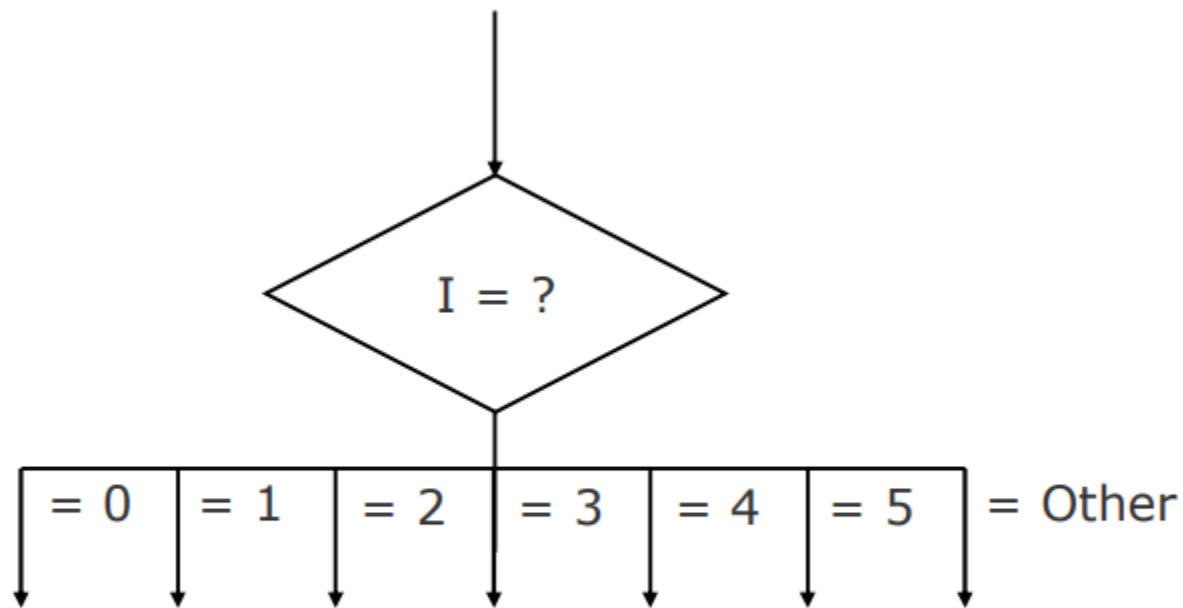
(c) A multiple-way branch decision.

Algorithm & Flowchart to find Even number between 1 to 50

Algorithm

Step-1 : Start

Step-2 : I = 1

Step-3 IF (I >50) THEN
         GO TO Step-7
         ENDIF

Step-4 IF ((I % 2) =0) THEN
         Display I
         ENDIF

Step-5 I = I + 1

Step-6 GO TO Step--3

Step-7 Stop
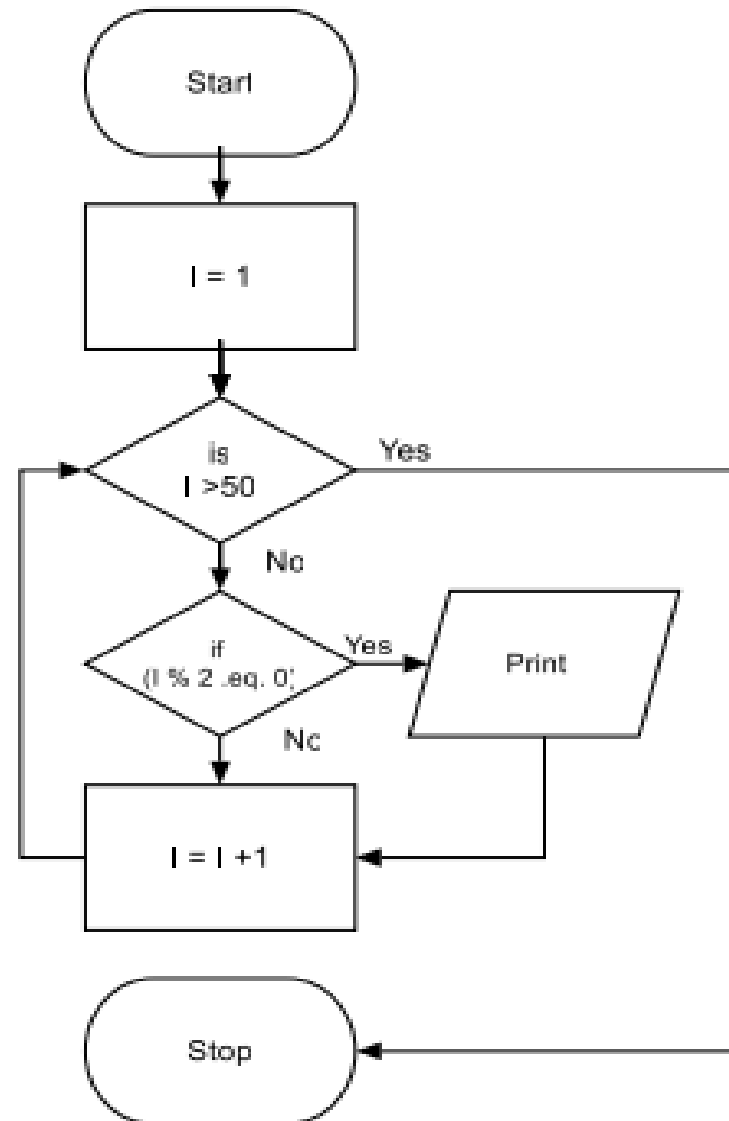
Algorithm & Flowchart to generate first n Fibonacci terms 0,1,1,2,3,5…n (n>2)

Step-1 : Start

Step-2 : Input Value of N

Step-3 : A=0, B=1, COUNT=2

Step-4 : WRITE A, B

Step-5 : IF (COUNT >N) then go to step 12

Step-6 : NEXT= A + B

Step-7 : WRITE NEXT

Step-8 : A=B

Step-9 : B=NEXT

Step-10 : COUNT=COUNT + 1

Step-11 : Go to step-4

Step-12 : Stop

```
            ┌─────────────┐
            │    Start    │
            └──────┬──────┘
                   │
                   ▼
           ╱─────────────────╲
          ╱   Input Value     ╲
          ╲     of N          ╱
           ╲─────────────────╱
                   │
                   ▼
            ┌─────────────┐
            │    A=0      │
            │    B=1      │
            │  COUNT=2    │
            └──────┬──────┘
                   │
                   ▼
           ╱─────────────────╲
          ╱    Write A,B      ╲
          ╲                   ╱
           ╲─────────────────╱
                   │
                   ▼
            ◇─────────────────◇
            │   COUNT > N     │──── Yes
            ◇─────────────────◇
                   │ No
                   ▼
            ┌─────────────┐
            │ NEXT = A + B │
            └──────┬──────┘
                   │
                   ▼
           ╱─────────────────╲
          ╱   Write NEXT      ╲
          ╲                   ╱
           ╲─────────────────╱
                   │
                   ▼
            ┌─────────────┐
            │   A = B     │
            │  B =NEXT    │
            │ COUNT = COUNT│
            │    +1       │
            └──────┬──────┘
                   │
                   ▼
            ┌─────────────┐
            │    Stop     │
            └─────────────┘
```

```
START → Switch Conditional Expression

Case Condition 1 --TRUE--> Statement 1 Break
      |
    FALSE
      ↓
Case Condition 2 --TRUE--> Statement 2 Break
      |
    FALSE
      ↓
Case Condition n --TRUE--> Statement n Break
      |
    FALSE
      ↓
Default --TRUE--> Default Statement

→ Statement just below switch case → END
```
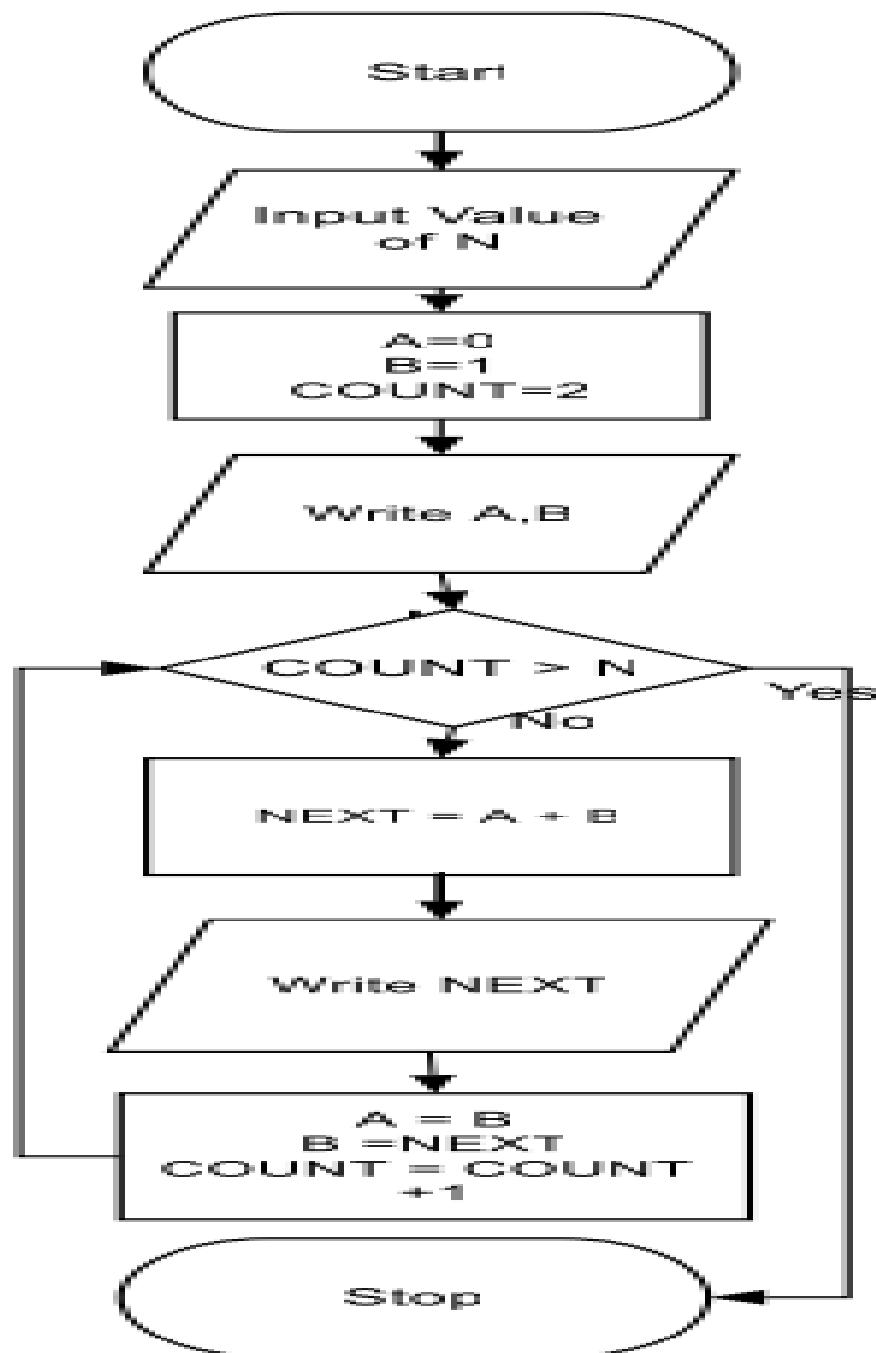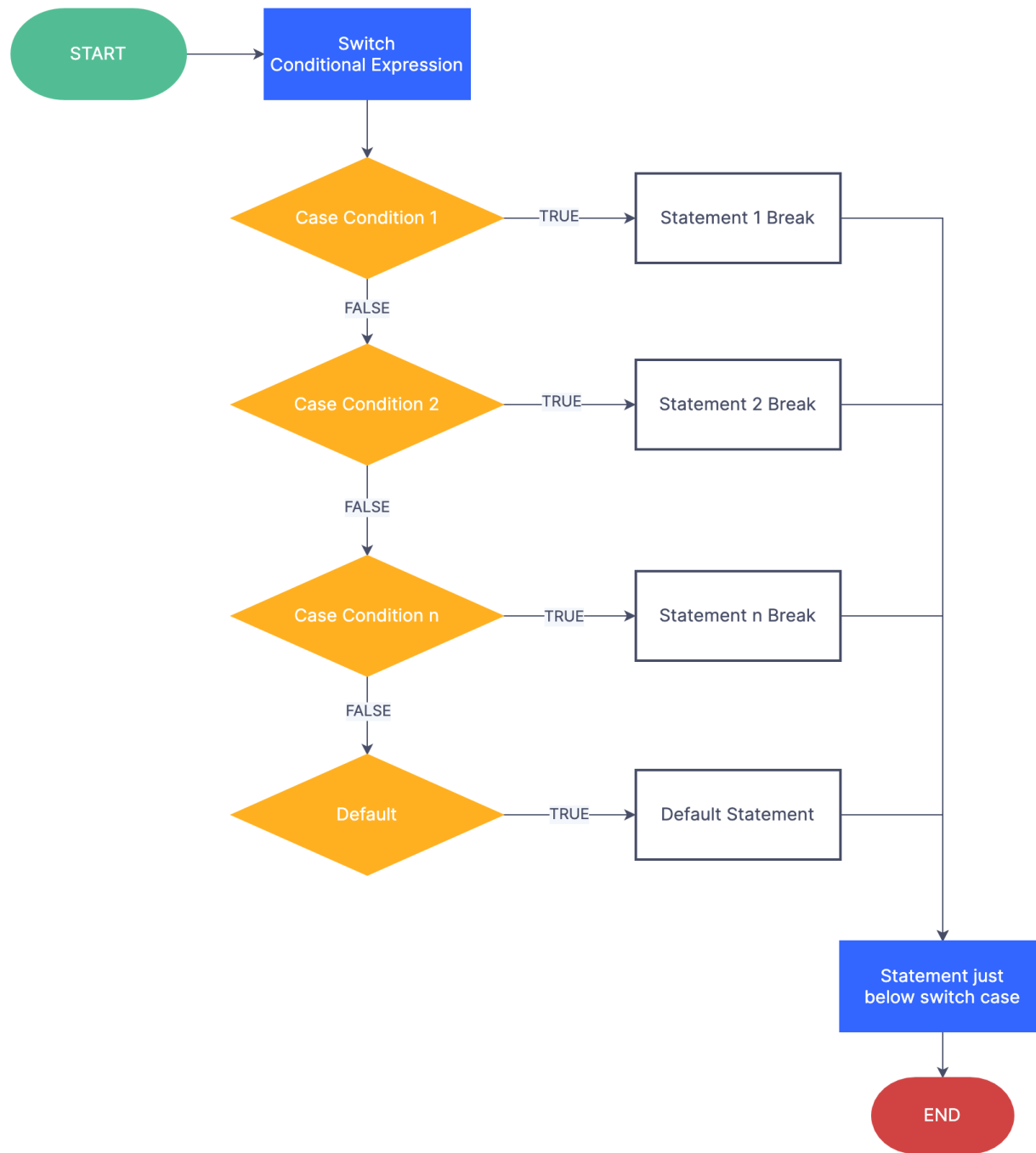
Pseudocode

- A program planning tool where program logic is written in an ordinary natural language using a structure that resembles computer instructions

- " Pseudo" means imitation or false and " Code" refers to the instructions written in a programming language. Hence, pseudocode is an imitation of actual computer instructions

- Because it emphasizes the design of the program, pseudocode is also called Program Design Language (PDL)

Overview of C

- C is a programming language developed at AT & T's Bell Laboratories of USA in 1972.

- It was designed and written by Dennis Ritchie.

- C is a structured programming language

- C supports functions that enables easy maintainability of code, by breaking large file into smaller modules

- Comments in C provides easy readability

- C is a powerful language

Program structure

A sample C Program

```c
#include<stdio.h>
int main( )
{



}
```

Preprocessor Statements:

- These statements begin with # symbol. They are called preprocessor directives. These statements direct the C preprocessor to include header files and also symbolic constants in to C program.

- Some of the preprocessor statements are :
#include: for the standard input/output functions

- **#define** PI 3.141592   symbolic constant

Header files

- The files that are specified in the include section is called as header file.

- These are precompiled files that has some functions defined in them.

- We can call those functions in our program by supplying parameters.

- Header file is given an extension .h

- C Source file is given an extension .c

Main function

- This is the entry point of a program
- When a file is executed, the start point is the main function
- From main function the flow goes as per the programmers choice.
- There may or may not be other functions written by user in a program
- Main function is compulsory for any c program

Writing the first program

#include<stdio.h>

int main()

{

    printf("Hello world");

    getch();

    return 0;

}
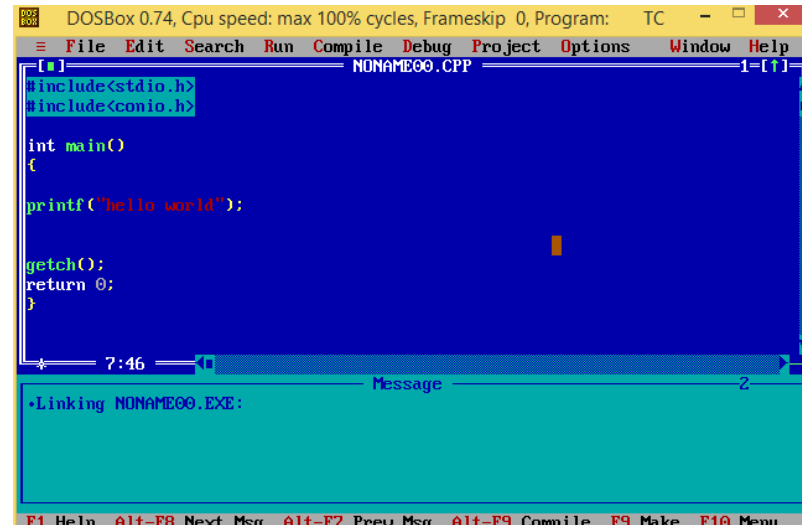
- This program prints Hello on the screen when we execute it
- getch() method pauses the Output Console until a key is pressed

Steps in learning English language:

Alphabets → Words → Sentences → Paragraphs

Steps in learning C:

Alphabets
Digits
Special sy-
mbols

→

Constants
Variables
Keywords

→

Instructions

→

Program
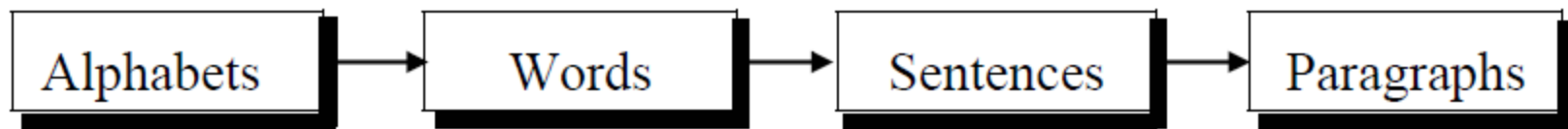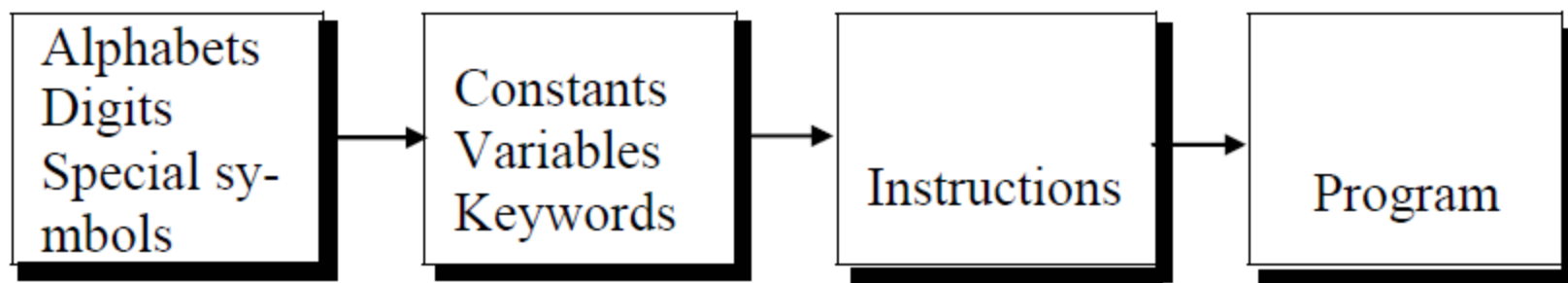
The C Character Set

A character denotes any alphabet, digit or special symbol used to represent information.

Alphabets	A, B, ..., Y, Z a, b, ..., y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ' ! @ # % ^ & * ( ) _ - + =	\ { } [ ] : ; " ' < > , . ? /

Constants, Variables and Keywords

The alphabets, numbers and special symbols when properly combined form constants, variables and keywords. A constant is an entity that doesn't change whereas a variable is an entity that may change.

There are two ways to define constant in C programming.
- const keyword
- #define preprocessor

```c
#include<stdio.h>
int main(){
    const float PI=3.14;
    printf("The value of PI is: %f", PI);
    return 0;  }
```

```c
#include<stdio.h>
int main(){
const float PI=3.14;
PI=4.5;
printf("The value of PI is: %f",PI)
;
    return 0;
}
```

If you try to change the value of PI, it will render compile time error.

The #define preprocessor is also used to define constant.

The #define preprocessor directive is used to define constant or micro substitution. It can use any basic data type.

Syntax: #define token value

```c
#include <stdio.h>
#define PI 3.14
void main()
{
  printf("%f",PI);
}
```

Constants

1. **Numerical constants:**

   — Integer constants

   These are the sequence of numbers from 0 to 9 without decimal points or fractional part or any other symbols. It requires minimum two bytes and maximum four bytes.

   Eg: 10,20, + 30, – 14

   — Real constants

   It is also known as floating point constants.

   Eg: 2.5, 5.342

2. **Character constants:**

   — Single character constants

   A character constant is a single character. Characters are also represented with a single digit or a single special symbol or white space enclosed within a pair of single quote marks

   Eg: 'a', '8', " ".

   — String constants

   String constants are sequence of characters enclosed within double quote marks.

   Eg: "Hello", "india", "444"

```c
#include <stdio.h>
void main()
{
const int height = 100;
const float number = 3.14;
const char letter = 'A;
const char  name[10] = "ABC";
const char back= '\?';

printf("value of height :%d \n", height );
printf("value of number : %f \n", number );
printf("value of letter : %c \n", letter );
printf("value of name: %s \n", name);
printf("value of backslash: %c \n", back);
}
```

Output:
value of height : 100
value of number : 3.140000
value of letter : A
value of name: ABC
value of backslash : ?

Rules for Constructing Integer Constants

- An integer constant must have at least one digit.
- It must not have a decimal point.
- It can be either positive or negative.
- If no sign precedes an integer constant it is assumed to be positive.
- No commas or blanks are allowed within an integer constant.

Ex.: 426
+782
-8000
-7605

Rules for Constructing Real Constants

Real constants are often called Floating Point constants. The real constants could be written in two forms—Fractional form and Exponential form.

- A real constant must have at least one digit.
- It must have a decimal point.
- It could be either positive or negative.
- Default sign is positive.
- No commas or blanks are allowed within a real constant.

Ex.: +325.34

426.0

-32.76

-48.5792

The exponential form of representation of real constants is usually used if the value of the constant is either too small or too large. Following rules must be observed while constructing real constants expressed in exponential form:

•The mantissa part and the exponential part should be separated by a letter e.
•The mantissa part may have a positive or negative sign.
•Default sign of mantissa part is positive.
•The exponent must have at least one digit, which must be a positive or negative integer. Default sign is positive.
•Range of real constants expressed in exponential form is -3.4e38 to 3.4e38.

Ex.: +3.2e-5
4.1e8
-0.2e+3
-3.2e-5

Rules for Constructing Character Constants

• A character constant is a single alphabet, a single digit or a single special symbol enclosed within single inverted commas. Both the inverted commas should point to the left. For example,
• The maximum length of a character constant can be 1 character.

Ex.: 'A'
'I'
'5'
'='

C Variables

An entity that may vary during program execution is called a variable. Variable names are names given to locations in memory.

Rules for Constructing Variable Names

•A variable name is any combination of alphabets, digits or underscores. Some compilers allow variable names whose length could be up to 247 characters.
•The first character in the variable name must be an alphabet or underscore.
•No commas or blanks are allowed within a variable name.
•No special symbol other than an underscore (as in **gross_sal) can be used in a variable name.**

Ex.: si_int
m_hra
pop_e_89

C Keywords

Keywords are the words whose meaning has already been explained to the C compiler (or in a broad sense to the computer). The keywords **cannot be used as variable names because if we do so we are trying to assign a new meaning to the keyword, which is not allowed by the computer.**

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

The general form of **printf( ) function is,**

printf ("<format string>", <list of variables>) ;

<format string> can contain,
%f for printing real values
%d for printing integer values
%c for printing character values

Return type and value of printf function

printf is a library function of stdio.h, it is used to display messages as well as values on the standard output device (monitor). **printf returns an integer value, which is the total number of printed characters**.
new line character ("\n")

```
#include <stdio.h>
void main()
 {     int res;
        res = printf("Hello\n");
        printf("Total printed characters are: %d\n",res);  }
```
Output
Hello
Total printed characters are: 5

Output
Hello
Total printed characters are: 5

Return type and value of scanf function

scanf is a library function of stdio.h, it is used to take input from the standard input device (keyboard). **scanf returns an integer value, which is the total number of inputs**.

```c
#include <stdio.h>
void main()
 { int x,y;
 int res;
 printf("Enter two number: ");
res=scanf("%d%d", &x,&y);
 printf("Total inputs are: %d\n",res);    }
```

```c
#include <stdio.h>
int main()
{    printf("%d", printf("welcome to programming class"));
 return 0;
}
```

```c
8
9   #include <stdio.h>
10
11  int main()
12  {
13      printf("%d", printf("welcome to programming class"));
14      return 0;
15  }
16
```

input

```
welcome to programming class28

...Program finished with exit code 0
Press ENTER to exit console.
```

19

Data Type	Range	Bytes	Format
signed char	-128 to + 127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32768 to +32767	2	%d
short unsigned int	0 to 65535	2	%u
signed int	-32768 to +32767	2	%d
unsigned int	0 to 65535	2	%u
long signed int	-2147483648 to +2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to +3.4e38	4	%f
double	-1.7e308 to +1.7e308	8	%lf
long double	-1.7e4932 to +1.7e4932	10	%Lf

Note: The sizes and ranges of int, short and long are compiler dependent. Sizes in this figure are for 16-bit compiler.

Table 2.8 *Size and Range of Data Types on a 16-bit Machine*

Type	Size (bits)	Range
char or signed char	8	−128 to 127
unsigned char	8	0 to 255
int or signed int	16	−32,768 to 32,767
unsigned int	16	0 to 65535
short int or signed short int	8	−128 to 127
unsigned short int	8	0 to 255
long int or signed long int	32	−2,147,483,648 to 2,147,483,647
unsigned long int	32	0 to 4,294,967,295
float	32	3.4E−38 to 3.4E+38
double	64	1.7E−308 to 1.7E+308
long double	80	3.4E−4932 to 1.1E+4932

By default char (signed char) int is (signed int)

I byte = 8 bits   either 0 or 1

char datatype in C

signed char   unsigned char

Sign Bit
1 bit

Data Bit
7 bits

Data Bit
8 bits
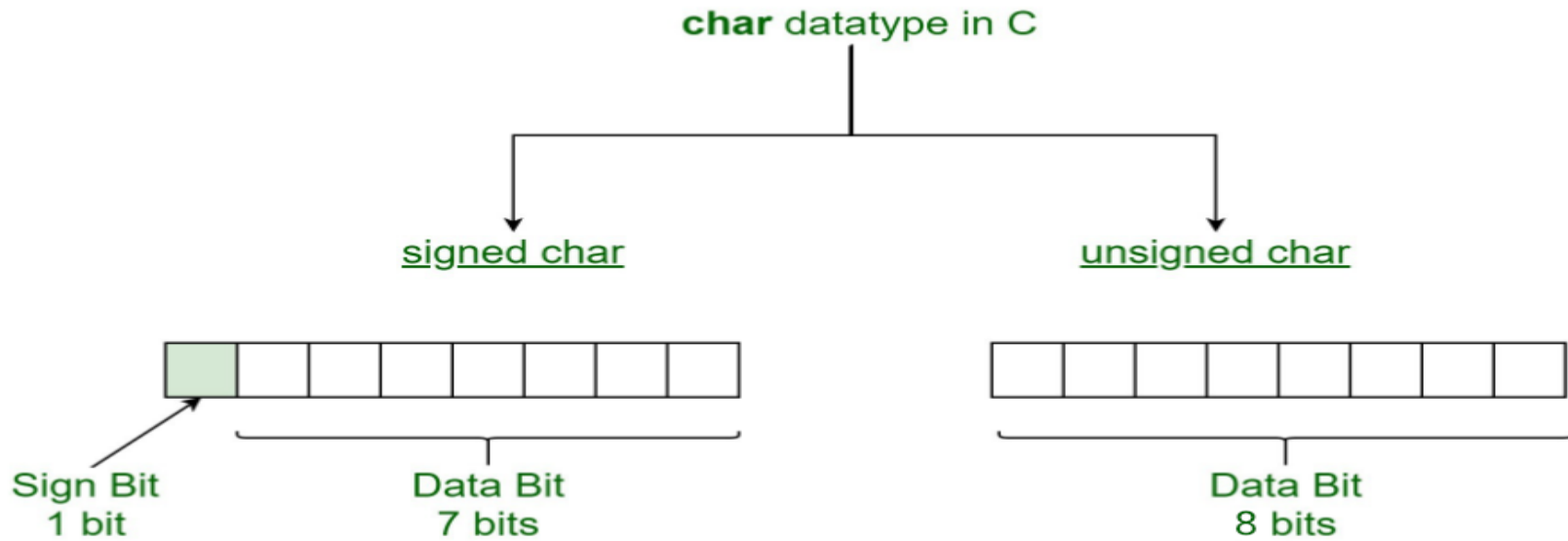
In case of signed one bit is reserved for sign

+/-

-2^{n-1} to $2^{n-1}-1$

-2^7 to 2^7-1

= -128 to 127

unsinged 2^n-1

$2^8-1 = 256-1=255$

So range is *0 to 255* for unsinged char

Unsinged range (0 to 255)

Smallest possible value

0	0	0	0	0	0	0	0

$2^7 \times 0 + 2^6 \times 0 + 2^5 \times 0 + 2^4 \times 0 + 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 0$

$= 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0$

$= 0$

Largest possible value

1	1	1	1	1	1	1	1

$2^7 \times 1 + 2^6 \times 1 + 2^5 \times 1 + 2^4 \times 1 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1$

$= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1$

$= 255$

Singed range -128 to +127

-ve value Smallest possible value

| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$2^7 \times 1 + 2^6 \times 0 + 2^5 \times 0 + 2^4 \times 0 + 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 0$
$= 128 + 0 + 0 + 0 + 0 + 0 + 0 + 0$
$= -128$

+ve value Largest possible value

| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

$2^7 \times 0 + 2^6 \times 1 + 2^5 \times 1 + 2^4 \times 1 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1$
$= 0 + 64 + 32 + 16 + 8 + 4 + 2 + 1$
$= 127$

If the signed bit is 0 it means that number is positive. If signed bit is 1 then number is negative.

Unsinged integer

If int takes 2 bytes   then 2 bytes = 16 bits

Formula is 2^n-1

$2^{16}-1$ = 65536-1=65535

Range is 0 to 65535

singed integer

-2^{n-1} to 2^{n-1} -1

-2^{15} to 2^{15} -1

-32768 to +32767

How to get the sizes:

```
char c;
int i;
printf("%d,%d\n", sizeof(c),
                  sizeof(i) );
```

Output: 1,4

Identifiers : Identifiers are the name of functions, variables and arrays. The identifiers are user-defined words in the C language. These can consist of lowercase letters, uppercase letters, digits, or underscores, but the starting letter should always be either an alphabet or an underscore.

Tokens : Tokens are the smallest elements of a program, which are meaningful to the compiler.

Token is divided into six different types : Keywords, Operators, Strings, Constants, Special Characters, and Identifiers.
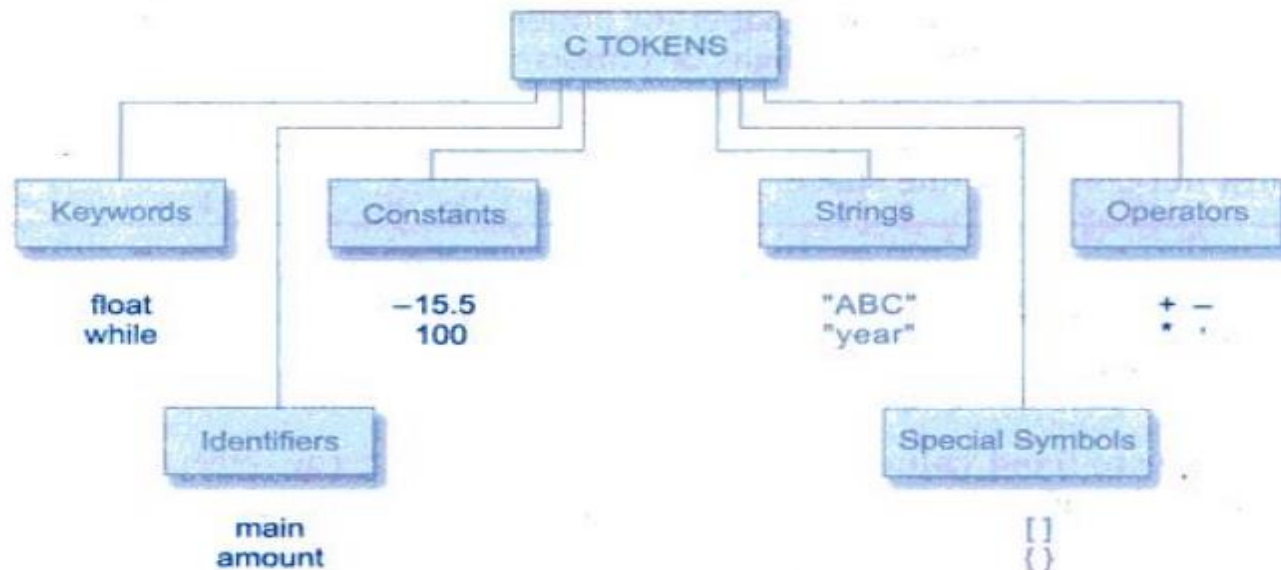
Type Conversion

C provides the facility of mixing different types of variables and constants in an expression. In these types of operations data type of one operand is converted into data type of another operand. This is known as type conversion. Implicit type conversions are done by the compiler while the explicit type conversions are user defined conversions
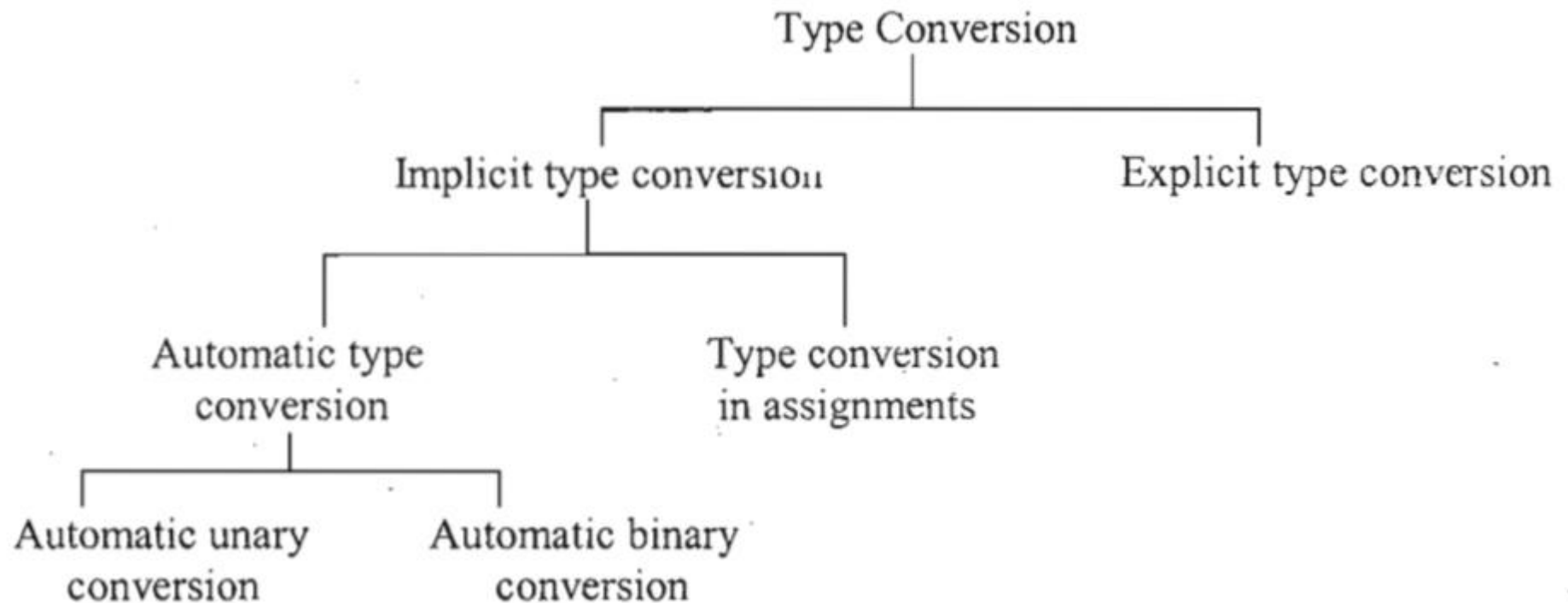
Implicit Type Conversions

These conversions are done by the C compiler according to some pre defined rules of C language. The two types of implicit type conversions are automatic type conversions and type conversion in assignment

Automatic Unary Conversion

All operands of type char and short will be converted to int before any operation. Some compilers convert all float operands to double before any operation.

Automatic binary conversions

The rules for automatic binary conversions are as

• If one operand is long double, then the other will be .converted to long double, and the result will be long double.

•Otherwise if one operand is double, then the other will be converted to double and the result will be double,

•Otherwise if one operand is float, the other will be converted to float and the result will be float.

•Otherwise if one operand is unsigned long int, then other will be converted to. unsigned long int, and the result will be unsigned long int.

•Otherwise if one operand is long int and other is unsigned int

(a) If long int can represent all the values of an unsigned int, then unsigned int will be convert to long int and the result will be long int,

(b) Else both the operands will be converted to unsigned long int and the result will be unsign long int.

Otherwise if one operand is long int, then the other will be converted to long int and the rest will be long int.

•Otherwise if one operand is unsigned int, then the other will be converted to unsigned int and the result will be unsigned int.
•Otherwise both operands will be int and the result will be int.

•we leave aside unsigned variables, then these rules are rather simple and can be summarized by -assigning a rank to each data type. Whenever there are two operands of different data types the operand 'with a lower rank will be converted to the type of higher rank operand. This is called promotion of data type.

```
┌─────────────────┐
│  long double    │      Highest rank
└─────────────────┘
         ▲
         │
┌─────────────────┐
│     double      │
└─────────────────┘
         ▲
         │
┌─────────────────┐
│      float      │
└─────────────────┘
         ▲
         │
┌─────────────────┐
│       int       │
└─────────────────┘
         ▲
         │
┌─────────────────┐
│ char , short int│      Lowest rank
└─────────────────┘
```
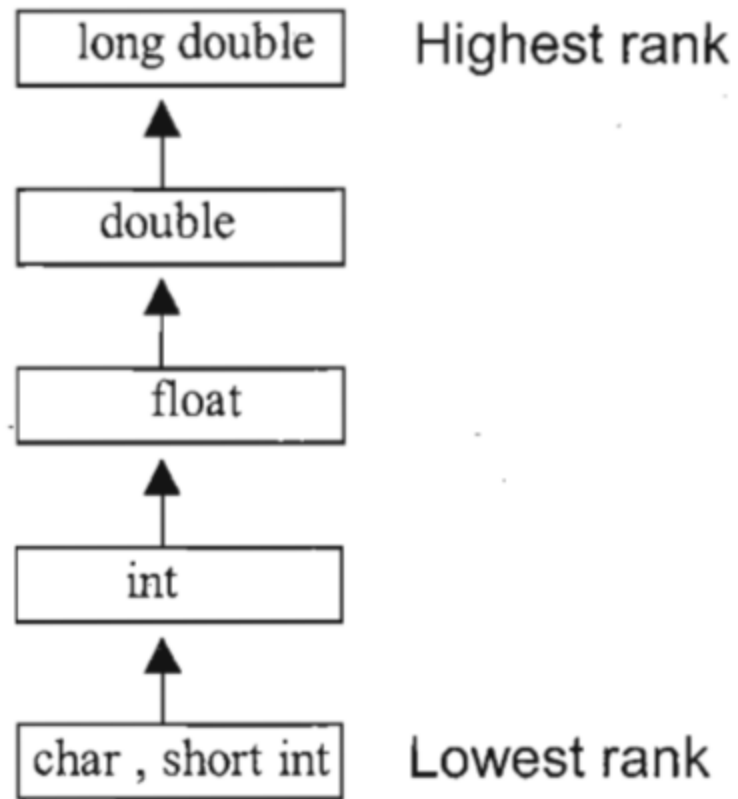
Implicit type conversion, the value of one type is automatically converted to the value of another type.

```c
9  #include <stdio.h>
10
11 int main()
12 {
13     int x;
14     for(x=97;x<=127;x++)
15     printf("%c\t",x); //Implicit casting from int to char |
16     return 0;
17 }
18
```

input

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p |
| q | r | s | t | u | v | w | x | y | z | { | | | } | ~ | | |

Type Conversion In Assignment

If the types of the two operands in an assignment expression are different, then the type of the right hand side operand is converted to the type of left hand operand. Here if the right hand operand is of lower rank then it will be promoted to the rank of left hand operand, and if it is of higher rank then it will demoted to the rank of left hand operand.

Some consequences of these promotion and demotions are

1. Some high order bits may be dropped when long is converted to int, or int is converted to short int or char.
2. Fractional part may be truncated during conversion of float type to Int type.
3. When double type is converted to float type, digits are rounded off.
4. When a signed type is changed to unsigned type, the sign may be dropped. .
5. When an int is converted to float, or float to double there will be no increase in accuracy or precision.

Program to understand the type conversion in assignment.

```c
#include<stdio.h>
main()
{
char  c1,c2;
int    il,i2;
 float  f1, f2;
c1='H';
i1=80. 56;     //Demotion: float converted to int, only 80 assigned to il*/ f1=12.6
c2=i1;      /*Demotion : int converted to character
i2=f1; ;     /*Demotion : float converted to int
```

f2=i1;   /*Promotion : int converted to float

i2=c1   /*Promotion : character converted to int

Explicit Type Conversion Or Type Casting

There may be certain situations where implicit conversions may not solve our purpose. For example

```
float  z;
 int x = 20, y = 3;
 z = x/y;
```
The value of z will be 6.0 instead of 6.66.

In these types of cases we can specify our own conversions known as type casting or coercion. This is done with the help of cast operator. The cast operator is a unary operator that is used for converting an expression to a particular data type temporarily. The expression can be any constant or variable. The syntax of cast operator is   **(datatype) expression**

z = (float)x/y;
#include<stdio.h>
Void main( )
{ int x=5, y=2, float p, q;
p=x/y;
printf("p = %f\n",p);
q=(float)x/y;
printf("q = %f\n",q);

p = 2.000000   q = 2.500000

(int)20.3	constant 20.3 converted to integer type and fractional part is lost(Result 20)
(float)20/3	constant 20 converted to float type, and then divided by 3 (Result 6.66)
(float)(20/3)	First 20 divided by 3 and then result of whole expression converted to float type(Result 6.00)
(double)(x +y -z)	Result of expression x+y-z is converted to double
(double)x+y-z	First x is converted to double and then used in expression

Storage class

A storage class defines the scope (visibility) and life time of variables and/or functions within a C Program.

There are following 4 storage classes which can be used in a C Program:

- auto

- register

- static

- extern

Introduction

Automatic

Register

Static

Extern

Important facts

Introduction

We have a Storage Class attached with every variable that we create. A storage class tells us:

1. Where the variable would be stored?

2. What initial value the variable will have?

3. What is the scope of the variable?

4. What will we be the lifetime for the variable?

S.No.	Storage Class	Keyword Used	Storage Location	Default Initial Value
1	Automatic	auto	Main Memory	Garbage
2	Register	register	CPU Registers	Garbage
3	Static	static	Main Memory	Zero
4	External	extern	Main Memory	Zero

Automatic Storage Class

Storage Location	Main Memory
Default Value	Garbage
Scope	Local to the block where the variable is declared
Lifetime	Till the control remains within the block in which the variable is declared.

Register Storage Class

Storage Location	CPU Registers
Default Value	Garbage
Scope	Local to the block where the variable is declared
Lifetime	Till the control remains within the block in which the variable is declared.

Static Storage Class

Storage Location	Main Memory
Default Value	Zero
Scope	Local to the block where the variable is declared
Lifetime	Value persists between different function calls

Program on Static Storage Class

```c
void main()
{  clrscr();
    incr();
    incr();
   getch();
}
void incr()
{ static int i;
  printf("\n i=%d",i);
  i++;
}
```

Output:

i=0

i=1

External Storage Class

Storage Location	Main Memory
Default Value	Zero
Scope	Global
Lifetime	As long as the program execution does not come to an end.

Comparison between auto & static storage class

- Storage Location is main memory
- Default Value is Garbage
- Scope is Local to the block where the variable is declared
- Lifetime till the control remains within the block in which the variable is declared.

1. Storage Location is main memory
2. Default Value is zero
3. Scope is Local to the block where the variable is declared
4. Value persist between different function calls.

Comparison continued….

```
{  clrscr();

    incr();

    incr();

   getch();

}
void incr()
{ auto int i;
  printf("\n i=%d",i);
  i++;
}
```
Output:

i=26730

i=-3243

```
{  clrscr();

    incr();

    incr();

   getch();

}
void incr()
{ static int i;
  printf("\n i=%d",i);
  i++;
}
```
Output:

i=0

i=1

Operators and Expressions

An operator specifies an operation to be performed that yields a value. The variables, constants can be joined by various operators to form an expression.

An operand is a data item on which an operator acts. Some operators require two operands, while others act upon only one operand. C includes a large number of operators that fall under several different categories, which are as follows

Expression

Expression is a combination of operators, constants, variables and function calls. The expression can be arithmetic, logical or relational

x+y   //arithmetic operation
a = b+c   //uses two operators (=') and (+)
a > b   //relational expression
a== b   //logical expression
func(a, b)   //function call

Type of Operator	Symbolic representation
Arithmetic operators	+, -, *, /, %
Relational operators	>, <, ==, >=, <=, !=
Logical operators	&&, ||, !=
Increment and decrement operator	++ and --
Assignment operator	=
Bitwise operator	&, |, ^, >>, <<, ~
Comma operator	,
Conditional operator	?:

- Arithmetic operators
- Assignment operators .
- Increment and Decrement operators
- Relational operators.
- Logical operators
- Conditional operator
- Comma operator
- sizeof operator
- Bitwise operators
- Other operators

Binary Arithmetic Operators

Operator	Purpose
+	addition
-	subtraction
*	multiplication
/	division
%	gives the remainder in integer division

a= 17, b=4

Expression	Result
a+b	21
a-b	13
a*b	68
a/b	4 (decimal part truncates)
a%b	1 (Remainder after integer division)

Increment And Decrement Operators

C has two useful operators increment (++) and decrement (- -). These are unary operators because they operate on a single operand. The increment operator (++) increments the value of the variable by 1 and decrement operator (- -) decrements the value of the variable by 1.

++x is equivalent to x = x + 1

--x is equivalent to x = x-l

These operators are of two types
•Prefix increment / decrement - operator is written before the operand (e.g. ++x or - -x)

Postfix increment / decrement - operator is written after the operand (e.g. x++ or x - -)

Prefix Increment / Decrement

Here first the value of variable is incremented / decremented then the new value is used in the operation .

X=3   (x whose value is 3.)

The statement y = ++x;   means first increment the value of x by 1, then assign the value of x to y .

equivalent to these two statements   (x = x+1; y = x;)

Hence now value of x is 4 and value of y is 4.

The statement y = - -x ;
 means first decrement the value of x by 1 then assign the value of x to y
This statement is equivalent to these two's statements
x = x -1 ;   y= x;

Hence now value of x is 3 and value of y is 3.

Here first the value of variable is used in the operation and then increment/decrement is perform.

Let us take a variable whose value is 3.

The statement   y = x++;   means first the value of x is assigned to y and then x is incremented.

statement is equivalent to these two statements   y = x;   x = x+1;
Output :

x is 4 and value of y is 3.

y = x- -; means first the value of x is assigned to y and then x is decremented.

This statement is equivalent to these two statements
y = x;
x = x-I;
Output is   **x is 3 and value of y is 4**.

Relational Operators

Relational operators are used to compare values of two expressions depending on their relations. An expression that contains relational operators is called relational expression. If the relation is true "then the value of relational expression is 1 and if the relation is false then the value of expression is 0

Operator	Meaning
<	less than
<=	less than or equal to
==	equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to

a=9, b=5

Expression	Relation	Value of Expression
a < b	False	0
a <= b	False	0
a == b	False	0
a != b	True	1
a > b	True	1
a >= b	True	1
a == 0	False	0
b!=0	True	1
a>8	True	1
2 > 4	False	0

The expression that combines two or more expressions is termed as a logical expression. For combining these expressions we use logical operators. These operators return 0 for false and 1 for true. C has three logical operators.

Operator	Meaning
&&	AND
||	OR
!	NOT

AND (&&) Operation   (This operator gives the net result true if
both the conditions are true, otherwise the result is false.)

Boolean Table

Condition1	Condition2	Result
False	False	False
False	True	False
True	False	False
True	True	True

a = 10, b = 5, c = 0

Expression		Result	Value of expression
(a= =10) && (b>a)	true && false	false	0
(b>=a) && (b= =3)	false && false	false	0
a && b	true && true	true	1
a && c	true && false	false	0

nonzero values are regarded as true and zero value is regarded as false

OR (II) Operator

This operator gives the net result false, if both the conditions have
the value false. otherwise the result is true.

Boolean Table

Condition1	Condition2	Result
False	False	False
False	True	True
True	False	True
True	True	True

a = 10, b = 5, c =0

Consider the logical expression(a >= b) I I (b > 15)
This gives result true because one condition is true

Expression		Result	Value of expression
a || b	true || true	true	1
a || c	true || false	true	1
(a<9) || (b>10)	false|| false	false	0
(b!=7) || c	true || false	true	1

Not (!) Operator

This is a unary operator and it negates the value of the condition. If the value of the condition is false then it gives the result true. If the value of the condition is true then it gives the result false.

Boolean Table

Condition	Result
False	True
True	False

a = 10, b = 5, c = 0

! (a = = 10)

The value of the condition (a= =10) is true.
NOT operator negates the value of the condition. Hence the result is false.

Assignment Operator

x = 8

y = 5

s = x+y-2   /* Value of expression x+y-2 is assigned to s*/

y = x   /* Value of x is assigned to y*/

Operator	Description	Example
=	**Simple assignment operator. Assigns values from right side operands to left side operand**	**C = A + B will assign the value of A + B to C**
+=	**Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.**	**C += A is equivalent to C = C + A**
-=	**Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.**	**C -= A is equivalent to C = C - A**
*=	**Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.**	**C *= A is equivalent to C = C * A**
/=	**Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.**	**C /= A is equivalent to C = C / A**
%=	**Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.**	**C %= A is equivalent to C = C % A**
<<=	**Left shift AND assignment operator.**	**C <<= 2 is same as C = C << 2**
>>=	**Right shift AND assignment operator.**	**C >>= 2 is same as C = C >> 2**
&=	**Bitwise AND assignment operator.**	**C &= 2 is same as C = C & 2**
^=	**Bitwise exclusive OR and assignment operator.**	**C ^= 2 is same as C = C ^ 2**
|=	**Bitwise inclusive OR and assignment operator.**	**C |= 2 is same as C = C | 2**

Conditional Operator

Conditional operator is a ternary operator (? and   :) which requires three expressions as operands. written as

TestExpression   ?   expression1 : expression2

TestExpression is evaluated first

•If TestExpression is true(nonzero), then expression1 is evaluated and it becomes the value of the overall conditional expression.

•IfTestExpression is false(zero), then expression2 is evaluated and it becomes the value of overall conditional expression.

Conditional Operator

.

Conditional operator is a ternary operator (? and :) which requires three expressions as operands. This written as

Test Expression ? expression1 : expression2

a < b ? printf("a is smaller") : printf("b is smaller");

max = a > b ? a : b;

sizeof operator

sizeof is an unary operator. This operator gives the size of its operand in terms of bytes. The operand can be a variable, constant or any datatype (int, float, char etc). For example sizeof(int) gives the bytes occupied by the int datatype .

```
void main( )
{ int var;
printf . ("Size of int %d" ,sizeof (int) ) ;
printf("Size of float %d",sizeof(float));
printf( " Size of var = %d" , sizeo(var) ) ;
printf("Size of an integer constant = %d",sizeof(45));
}
```

Bitwise Operators

C has the ability to support the manipulation of data at the bit level. Bitwise operators are used for operations on individual bits. Bitwise operators operate on integers only. The bitwise operators are as

Operator	Meaning
&	bitwise AND
|	bitwise OR
^	bitwise exclusive OR
<<	shift left
>>	shift right

Bitwise AND Operator (&)

The output of bitwise AND is 1 if the corresponding bits of two operands is 1.
If either bit of an operand is 0, the result of corresponding bit is evaluated to
0.

bitwise AND operation of two integers 12 and 25.

12 = (In Binary)
25 = (In Binary)
_____
8 (In decimal)

	0	0	0	0	1	1	0	0
&	0	0	0	1	1	0	0	1
	0	0	0	0	1	0	0	0

```c
#include <stdio.h>
int main() {
    int a = 12, b = 25;
    printf("Output = %d", a&b);
    return 0;
}
 Output:8
```

Bitwise OR Operator (|)

The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. In C Programming, bitwise OR operator is denoted by |.

12	0	0	0	0	1	1	0	0
25	0	0	0	1	1	0	0	1
	0	0	0	1	1	1	0	1

29

```c
#include <stdio.h>
int main() {
    int a = 12, b = 25;
    printf("Output = %d", a|b);
    return 0;
}
```

Bitwise XOR (exclusive OR) Operator ^

The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite.

12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25
  000011 0 0
^ 000110 0 1

_____

  00010101 = 21 (In decimal)

```c
#include <stdio.h>
int main() {
    int a = 12, b = 25;
    printf("Output = %d", a^b);
    return 0;
}


Output = 21
```

Table 1.13 Truth table of bitwise XOR

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise Complement Operator

Bitwise complement operator is a unary operator (works on only one operand). It changes 1 to 0 and 0 to 1. It is denoted by . **Bitwise complement of any number N is -(N+1).**

0	0	1	0	0	0	1	1
1	1	0	1	1	1	0	0

35

```
#include <stdio.h>
int main() {
    printf("Output = %d\n",~35);
    printf("Output = %d\n",~-12);
    return 0;
}
Output = -36
Output = 11
```

Shift Operators in C programming

There are two shift operators in C programming:

✓ Right shift operator
✓ Left shift operator.

Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.

212 = 11010100 (In binary)

212>>2 = 00110101 (In binary)   [Right shift by two bits]

212>>7 = 00000001 (In binary)

212>>8 = 00000000

212>>0 = 11010100 (No Shift)

Left Shift Operator

Left shift operator shifts all bits towards left by a certain number of specified bits. The bit positions that have been vacated by the left shift operator are filled with 0. The symbol of the left shift operator is <<.

212 = 11010100 (In binary)
212<<1 = 110101000 (In binary) [Left shift by one bit]
212<<0 = 11010100 (Shift by 0)
212<<4 = 110101000000 (In binary) =3392(In decimal)

```c
#include <stdio.h>
int main() {
    int num=212, i;
    for (i=0; i<=2; ++i) {
        printf("Right shift by %d: %d\n", i, num>>I);
    }
    printf("\n");

    for (i=0; i<=2; ++i) {
        printf("Left shift by %d: %d\n", i, num<<i);
    }

    return 0;
}
```

Right Shift by 0: 212
Right Shift by 1: 106
Right Shift by 2: 53

Left Shift by 0: 212
Left Shift by 1: 424
Left Shift by 2: 848

Precedence And Associatively of operators

Consider the following expression

2 + 3 * 5

If addition is performed before multiplication then result will be 25

and if multiplication is performed before addition then the result will be 17.

Priority	Operators	Description
1st	* / %	multiplication, division, modular division
2nd	+ -	addition, subtraction
3rd	=	assignment

Associativity of Operators

When an expression contains two operators of equal priority the tie between them is settled using the associativity of the operators. Associativity can be of two types—Left to Right or Right to Left.

Operator	Description	Precedence level	Associativity
( )	Function call	1	Left to Right
[ ]	Array subscript		
→	Arrow operator		
.	Dot operator		
+	Unary plus	2	Right to Left
-	Unary minus		
++	Increment		
- -	Decrement		
!	Logical NOT		
~	One's complement		
*	Indirection		
&	Address		
(datatype)	Type cast		
sizeof	Size in bytes		
*	Multiplication	3	Left to Right
/	Division		
%	Modulus		
+	Addition	4	Left to Right
-	Subtraction		
<<	Left shift	5	Left to Right
>>	Right shift		
<	Less than	6	Left to Right
<=	Less than or equal to		
>	Greater than		
>=	Greater than or equal to		
= =	Equal to	7	Left to Right
!=	Not equal to		

= = !=	Equal to Not equal to	7	Left to Right
&	Bitwise AND	8	Left to Right
^	Bitwise XOR	9	Left to Right
|	Bitwise OR	10	Left to Right
&&	Logical AND	11	Left to Right
||	Logical OR	12	Left to Right
? :	Conditional operator	13	Right to Left
= *= /= %= += -= &= ^= |= <<= >>=	Assignment operators	14	Right to Left
,	Comma operator	15	Left to Right

x = a+b < c

+ operator has higher precedence than < and =, and < has more precedence than =, so first a+b will be evaluated, then < operator will be evaluated, and at last the whole value will be assigned to x.

If a = 2,b = 6, c = 9   then final value of x will be

x *= a + b

+ operator has higher precedence than *=, so a+b will be evaluated before compound assignment. This is interpreted as x = x*(a+b) and not as x = x*a+b.

x = 5, a = 2, b = 6

Ans=40;

x = a<=b || b==c

Here order of evaluation of operators will be

<=, = =, ||,=. If initial values are

a = 2, b = 3, c = 4,

then final value of x will 'be 1.

.

What if Two operators are of same precedence

For example-

5 + 16 / 2 * 4

Here / and * have higher precedence than + operator, so they will be evaluated before addition. But / and * have same precedence, so which one of them will be evaluated first still remains a problem. *If / is evaluated before *, then the result is 37 otherwise the result is 7.*

Similarly consider this expression
20 - 7 - 5 - 2 -1
Here we have four subtraction operators, which of course have the same precedence level. If we decide to evaluate from left to right then answer will be 5, and if we evaluate from right to left the answer will be 17.

To solve these types of problems, an associativity property is assigned to each operator. Associativity of the operators within same group is same. All the operators either associate from left to right or from right to left

5 + 16 / 2 *4   since / and * operators associate from left to right so / will be evaluated before * and the correct result 37.

20 - 7 - 5 - 2 -1   The subtraction operator associates from left to right so the value of this expression is 5

The assignment operator associates from right to left. Suppose we have a multiple assignment expression like this   x=y=z=5

initially the integer value 5 is assigned to variable z and then value of expression z = 5 is assigned variable y. The value of expression z = 5 is 5, so 5 is assigned to variable y and now the value of expression y = z = 5 becomes 5.

Now value of this expression is assigned to x and the value of hole expression x = y = z = 5 becomes 5.

Role Of Parentheses- In Evaluating Expressions

If we want to change the order of precedence of any operation, we can use parentheses. According to the parentheses rule, all the operations that are enclosed within parentheses are performed first.

For example in expression 24/2+4, division will take place before addition according to precedence rule but if we enclose 2+4 inside parentheses then addition will be performed first. So the value of expression 24/(2+4) is 4.

For evaluation of expression inside parentheses same precedence and associativity rules apply. For example-

(22 - 4) / (2+4*2-1)

Here inside parentheses multiplication will be performed before addition.

Nesting of parentheses

(4* (3+2))/10

In these cases, expressions within innermost parentheses are always evaluated first, and so on, till outermost parentheses. After evaluation of all expressions with parantheses, the remaining expression is evaluated as usual.

In the above expression 3+2 is evaluate first and then 4*5 and then 20/10.