Computer Science

Grade: XI

Programming Concepts and Logics

Reference Note

Bal Krishna Bhusal

Unit Wise Important Questions for Computer Science XI

Unit 5 Programming Concepts and Logics

1.) What is program?

2.) What is programming language? Explain its types with merits and demerits.

3.) Differentiate between compiler and interpreter.

4.) What is error? Briefly explain its types.

5.) Differentiate between syntax and semantics error.

6.) Define algorithm and flowchart with examples.

7.) What is C? Write its Features.

8.) What is a Data type? Explain the type of operator with examples.

9.) What is control structure? Write different between break and continue statement with examples.

10.)What is looping? Write different between while and Do while loop with examples.

11.) Define the term array. What is string? Explain any four string handling function with example.

Unit 5- Programming Concepts and Logics

5.1 Programming Concept

PROGRAMMING CONCEPTS

Program:

Program is a group of instructions given to the computer to perform a certain task. A program is used to solve a problem in computers. The program is written using a series of instruction that consists of all symbols, characters and certain rules.

A computer specialist who is responsible for designing, writing and modifying computer program is known as computer programmer. Programmer had depth knowledge about programming tools, techniques and programming language.

Feature of good program

1) **Integrity:** A program should be complete and must give the desire output.
2) **Clarity:** A program should be clear and should not be ambiguously.
3) **Simplicity:** A method for solving a problem should be very simple and easily understandable.
4) **Efficiency:** A program should take less memory and processing time, as a result it runs faster. Such kind of program automatically increases the efficiency of computer.
5) **Maintainability:** A program should be easy to maintain and update in future.
6) **Reliability:** It should be reliable so that user can depend on it.
7) **Generality:** It should be flexible and easy to operate with a wide range of platform.
8) **Modularity:** A program should be divided into different modules in order to complete a complex problem.
9) **Robustness:** A program should be almost 100 % perfect.
10) **Documented:** Documentation helps for smooth operation for the users and even helps for further modification and maintenance.

Computer Program:

Computer program is a group of instructions given to the computer to perform a certain task. A program is used to solve a problem in computers. The program is written using a series of instruction that consists of all symbols, characters and certain rules.

Programming Language:

The process of writing computer using such rules and symbols is called computer programming or coding. Computer programming is used to prepare computer programs and software.

The language which is used to develop program in computer is called programming language. Being an electronic machine, computer can understand only binary code i.e. 0 and 1 represented by the flow of electricity in the form of 'ON' and 'OFF' in voltage.

Every programming language consists of a set of codes that a computer programmer uses to give instruction to computer to perform some specific task. The set of rules to write the code is called syntax. The computer understand the code after it is complied.

Nowadays there are many programming language for making program in computer. For example: C, C++, VB, .NET, Java etc.

Differences between Program and Software

Program	Software
It is a set of instructions which instructs computer to perform a specific job.	It is a collection of instructions, programs, and data which instruct the computer on how to solve computer problems.
It is independent.	To develop software, collections of programs are needed.
It is a component of software	It is a logical unit driving the computer system.
It defines the computer process.	It defines both data and process.
A programmer creates programs.	Software is created by groups of programmers as a team.
It is not generally licensed for sale.	It is generally licensed under a company.
It cannot be divided in accordance to needs and uses.	It can be divided under various needs and uses like application software, utility software, system software, etc.
Example: SYS, FoRMAT.SYS, interest calculations, etc.	Example: MS Word, Adobe Photoshop, Internet Explorer, etc.

Type

s of computer programming Language

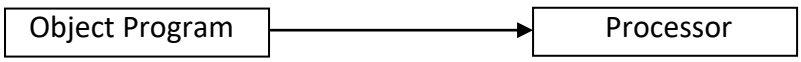
1) Low Level Language
   i) Machine Level Language (First Generation Language)
   ii) Assembly Level Language
2) High Level Language
   i. Procedural language
   ii. Problem oriented language
   iii. Natural Language

Low Level Language:

Low level language is a machine oriented or depended language in which detail knowledge of hardware specification is required to run the program. Statements in Low Level language are written based on the hardware structure of the computer. The low level language is divided into two types:

Machine Level Language:

The machine language which is written in the form of binary 0's and 1's. The machine level language is directly understand by computer because it is written in the form of 0's and 1's. Each instruction of the machine level language tells computer what to perform. The programs written in machine level language are executed very fast than the program written in any other language.



Program in machine Level Language

Advantage

- Program execution is faster than other language.
- It does not require any translating program.

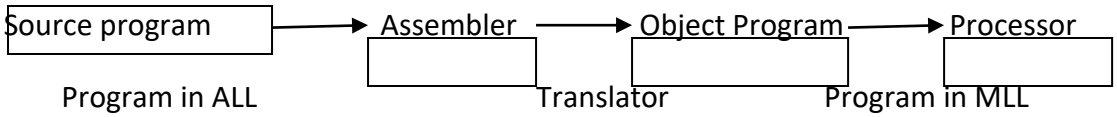
Disadvantages

- Machine Level Language are machine dependent
- It is difficult to perform the machine level language.
- Program development is time consuming.
- There is a greater chance of error in the programming.

- Need to higher level of skill programmer.

Assembly Level Language:

Assembly Level Language is a low level machine dependent language in which the symbolic codes (Mnemonic codes) are used to write assembly programs. With the use of symbolic codes in assembly language, it is easy to write the program for the programmer. For example ADD for addition, SUB for Subtraction, CMP for Comparison. It is easier to understand, find error and correct error. The program written in assembly language needs to be translated into the machine level language. This is done by an assembler an assembler translates the assembly level language into the machine language.



Advantage

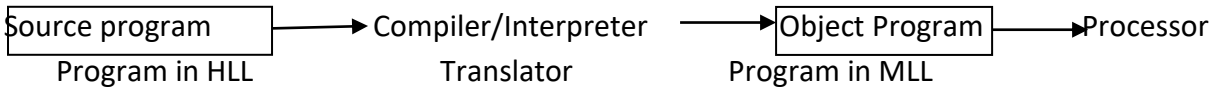
- It is easy to understand
- It is easy to find and correct error.
- It is easier to modify.
- Program Execution is faster than High level language

Disadvantage

- It is machine dependent language.
- Program development and debugging is more difficult and time consuming than HLL.
- It is very hard to remember the mnemonics
- It required translating program.

High Level Language:

High Level languages are the language which uses human language and mathematical notations. High level language uses the English keywords and mathematical symbols to write the program rather than mnemonics codes used in assembly language. High level languages are machine independent. While using high level language the program is not concerned about the machine structure of the computer. The high level language is to be translated into machine level language. This task is done by using the compilers and interpreters. These are also called translating program. The program written using high level language is easier to maintain than low level language. Advanced program can be developed using high level programming language. Some of the common high level language is: BASIC, PASCAL, FORTRAN, C, C++, Java etc.



Advantage

- It is machine independent
- It is easier to learn and develop
- Easily detect and remove errors.
- It is easier to maintain.
- Lower programming cost.
- Better documentation.
- Programmer does not need to remember large no. of mnemonics.

- No need to knowledge of internal structure of computer architecture for writing source program in HLL.

Disadvantage

- Computer does not understand directly
- More time to require for execution of the program.
- Require more memory space
- Less flexibility
- It needs to be translating program.
- Operating speed is slow.

High Level Language can be further categories as:

i) Procedural oriented language (3GL)
ii) Problem oriented language (4GL)
iii) Natural Language (5GL)

Differentiate between Assembly level language and Machine level language.

S.N.	Assembly Level Language	S.N.	Machine Level Language
1	It can't directly understand by computer.	1	It can directly understand by computer.
2	It need to translator program: Assembler	2	It is not required translator program.
3	It is used to symbolic instructions code.	3	It is used to 0 and 1 code in program.
4	It is also called symbolic/2nd generation language	4	It is also called binary/1st generation language.
5	It takes more time to execution.	5	It takes less time to execution.
6	It is easy to understand & write program.	6	It is difficult to understand & write program.
7	It is easy to debug errors.	7	It is hard to debug errors.

Fourth Generation Language:

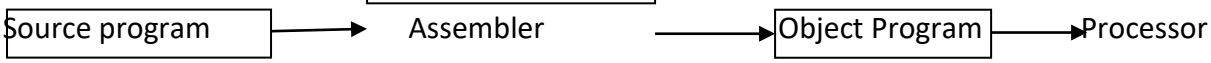
Fourth generation language was developed after HLL, so it is one step ahead from HLL. It is result/problem oriented programming language and it contains database query language. Fourth generation language program is also needed to be translated either by compiler or interpreter into machine understandable code before they are executed as it is not directly understood by the computer. Example of 4GL is SQL (Structure Query Language), Visual Basic, C#, PHP.

Language Processor (Translator)

Language Processor is the programs that translate the program written in other language (source program) to the machine level language program (object program). They are three types.

1. **Assembler:**

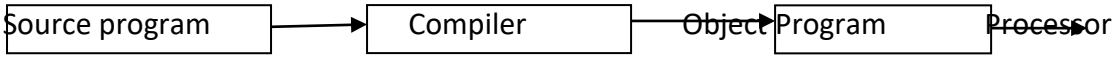
An assembler is a translator or translating program which translates codes of assembly language into equivalent machine level code. The program which is to be translated by assembler is called source program and the translated program by assembler is object program.



Program in ALL Translator Program in MLL

2. Compiler:

A compiler is a translating program that translates the program written in high level language into equivalent machine level language at once. The program written in high level language is called source program and the program translated into machine language is called object program.



Program in HLL/4GL Translator Program in MLL

3. Interpreter:

An interpreter is a translating program which translates the program written using high level language into equivalent machine level language one line at a time. It is similar to compiler but it converts the program's one line at a time into machine level language.

Program in HLL/4GL Translator Program in MLL

Differentiate between Compiler and Interpreter.

S.N.	Compiler	S.N.	Interpreter
1	It translates a complete high level language into machine language at once.	1	It translates high level language program into machine languages instruction line by line.
2	It is slow for debugging.	2	It is faster in debugging.
3	It creates an object program.	3	It does not create an object program.
4	It is not easier to find out and correct mistake in source program.	4	It is easier to find and correct mistake in source program.
5	It compiling process is faster.	5	It compiling process is slower.
6	Examples: C, C++, Visual Basic etc.	6	Examples: BASIC, LISP etc.

List of high level programming language

1) **FORTRAN:** Formula Translator is the first High Level Language which is very powerful language for scientific and engineering computations.
2) **LISP:** List Processing language is functional language for list processing, recursive, and not iterative. It is suitable for solving non numeric operations and used in the field of AI (Artificial Intelligence) and pattern recognition.
3) **CPL:** Combined Programming Language is used polymorphic testing structures and it has list and array. It was a step toward the design of the c language.
4) **COBOL:** Common Business Oriented Language is a classical procedural language aimed at enterprise management.
5) **BASIC:** Beginner's All- purpose Symbolic Instruction Code.
6) **BCPL:** Basic Combined Programming Language is also uses procedures and functions.
7) **PASCAL:** PASCAL is a structured programming language.
8) **PROLOG:** Programming Logic language is used for solving natural logic and in the field of knowledge based system which is a branch of AI.
9) **C:** C is a procedural programming language.
10) **C++:** It is also called Object Oriented version of C, powerful and efficient language for developing software.
11) **JAVA:** Java is also Object Oriented Programming Language for internet web and mobile applications.
12) **.NET:** It is read as 'dot NET' is powerful features of different programming language into one environment. It is most commonly used for solving web based applications and distributed mobile applications.
13) **XML:** eXtensible Markup Language is more advanced markup language than HTML for web content development. It helps to manages sources document in several different formats such as web page, printable documents, PDF files etc.
14) **JavaScript:** JavaScript is a scripting language. It can run inside simple web browser application and calculation. It shares the syntax of C or Java.
15) **ASP:** Active Server Page is server site web scripting language developed by Microsoft Corporation for accessing database from web application.
16) **PERL:** Practical Extraction and Reporting Language is one of the first scripting languages on UNIX system. It is open source and found on UNIX/Linux based server.
17) **PHP:** Hypertext Preprocessor is very popular server site scripting language on UNIX/Linux system. It is very popular for accessing database like My SQL and Oracle. It is open source software.

Testing and Debugging

Bug / Error: Errors are the mistake in the program. The Error can be in syntax or logic of the program.

Testing: Testing is the process of making sure that the program performs the intended task.

There are several stages of testing. They are:

- **Unit testing** involves the individual components' testing.
- **Integration testing** involves the separate components' testing as they are put together.
- **System testing that** involves the whole final form of a program testing.

- **User acceptance testing** involves the user testing the program to see that it is the result what is required.

Debugging: Debugging is the processing of discovery and correction of programming errors (bugs).

- Many syntax and logical errors might have occurred while writing a program.
- Debugging a program can take from a few seconds to months and even years depending on the bug, size of a program and type of system programmer etc.

Syntax and Semantics (Logical) errors

Syntax / Logical: Syntax is the arrangement of words into a sentence. It describes the rules by which words can be combined to make a meaningful sentence. For eg. S.+V.+O. The syntax defines whether the instruction is correct or not. If the syntax of the program is not correct, then the compiler or interpreter detects an error called syntax error and informs the programmer. Then the program execution gets terminated.

Semantic: Semantic is the study of meaning in language. It defines what the sentence means. Although the syntax is correct, it may not be meaningful. For eg.

Aryan eats an apple. = Syntactically and semantically correct.

Aryan eats a house. = Syntactically correct and but semantically not correct.

Aryan eat a house. = Syntactically and semantically incorrect.

S.N.	Syntax	S.N.	Semantics
1	It is common format.	1	It is pseudo code format.
2	It is the rule of the semantics.	2	It is the example of syntax.
3	Computer can understand the syntax.	3	Computer couldn't understand the semantics.

S.N.	Syntax error	S.N.	Logical error
1	It is the error of spelling mistake & grammar mistake.	1	It is the mistake of doing error inputting.
2	Single programmer can easily detect the error.	2	Group of programmer are necessary.
3	It is simple to find out.	3	It is difficult to find out.
4	It takes less time to find out.	4	It is long time to find out.

Runtime Error: The error which appears during the execution or runtime of program is called runtime error. The error which appears during the execution or runtime of a program is called runtime error. Such errors appear when the computer is asked to divide by zero or when the variable is assigned a large value which is beyond its capacity. The run-time error causes the termination of program execution.

Programming Fundamentals

Data types

Introduction to Data Type:-

- Data type specifies the type of data that is to be used in a program.
- A data types defines the set of data in any high level language which specifies the values and stored in memory.

1. **Integer:** All positive natural numbers, negative natural numbers and zero are called integers. Integer variable can take two bytes memory hold data in computer.
2. **Characters and String:** All alphabets including uppercase, lowercase, symbols etc are considered as character. Even space is also considered as a character. It requires one byte memory to hold the data in computer.
3. **Numeric Real:** The data type which contains any numeric representation is called real data type. The value may be signed or unsigned, fractions or exponential as well. Example: 0, 0.5, 4.7234, $3.0e^{-4}$, etc are real numbers.
4. **Floating:-** All fractional numbers are considered as floating point numbers. Variables like: weight, height, currency etc. It requires four bytes memory to hold the data in computer.
5. **Boolean:** A Boolean data type can only take two data values: either true(1) or false(0). For eg. Marital status of a person is either married or unmarried.

Keyword, Variable and Constant

Keyword:

Keyword is a special reserved word in programming language whose meaning has been already defined to the compiler. A keyword performs a specific task in computer program. For eg.int, float, if, while, etc.

Variable:

Variable is a named location in memory (RAM) of computer which can hold some value. The value assigned to the variable may be changed during the execution of the program, hence it is called variable. For eg. L=10 and B=5, here L and B are variable and 10 and 5 are value.

Constant:

It is a value which does not change during the execution of the program. For eg. A=10 and B=5 are constants.

Operation, Operand and Operator

Operation: An Operation in program is defined as an action upon the given data. The examples of operation are as follows:

- Arithmetic Operation
- Relational Operation
- I/O Operation
- Data Moving Operation etc.

Operand: Operand is the different type of data on which different operations take place. The data may be value or variable.

Operator: An Operator is a sign or symbol which performs an operation or evaluation on one or more operands. There are different types of operator they are Arithmetic Operator (+,-,*,/,%), Relational Operator (=,>,<,>=,<=,!=), Logical Operator (AND, OR and NOT) etc.

Program Design Tools

Algorithms:

An algorithm is a step by step by of instructions to solve a particular problem. It is an effective procedure for solving a problem in a finite number of steps. It is written in simple English language.

- It should be finite number of steps.
- It should be in a simple language.
- It should have an input.
- It should give an output after its execution.
- It should be independent of programming language.

***Example:** Write an algorithm to read the principle amount, time and rate and find simple interest.*

1. Start / Begin
2. Read / Input P,T and R.
3. Calculate SI=P*T*R/100.
4. Print / Display SI.
5. End / Close.

Flowchart:

A flowchart is a graphical representation of program logic in terms of symbols. It represents the program using the symbols. It is a program planning tool for organizing the sequence of steps necessary to solve a problem.

Types of Flowchart

1. ***System flowchart:*** The flowchart which shows the overview of the data flow and sequence of operations in a system is called system flowchart.

2. Program Flowchart: The flowchart which shows the detailed diagram to solve a particular problem of the program is called program flowchart.

S.N.	System Flowchart	S.N.	Program Flowchart
1	It is gives the direction of a program.	1	It is oriented to solve the problem through a program.
2	It controls all the mechanism of the modules or the programs.	2	This flowchart is known as program flowchart.
3	It gives internal infrastructure of the program.	3	The size of flowchart depends on nature of the problem.

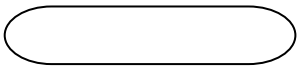
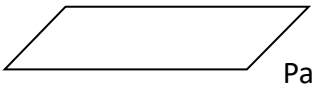
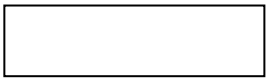
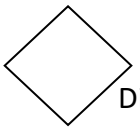
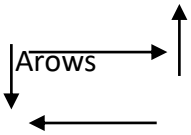
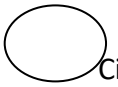
Advantage of flowchart

- Communication
- Proper documentation
- Proper debugging
- Effective analysis
- Efficient coding
- Efficient program maintenance

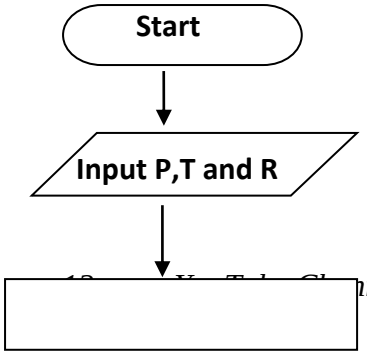
Disadvantage of flowchart

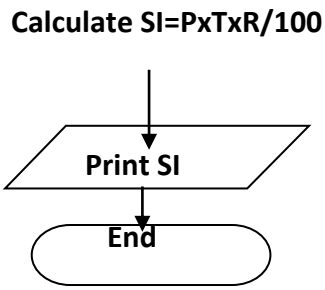
- Complex logic
- Difficult to Alternations and modifications
- Hard to Reproduction
- Time consuming work

Flowcharting symbol and their meaning.

Symbol Name	***Symbol/Shape***	***Meaning/ Function***
Terminal		= Start or end of the program.
Input/output	 Parallelogram	= Input or output operation.
Processing	 Rectangle	= Processing (or computation steps) function of a program.
Decision	 Diamond	= Decision making and branching.
Flow Lines	 Arows	= Flow line-direction indicates.
Connectors	 Circle	= connector or joining of two parts of program.

***Example:** Draw a flowchart to input the principle amount, time and rate and find the simple interest.*





Pseudo code: It is similar to English structure. It is a language which almost similar to computer language. It isn't a program but it is a form or syntax or general format of a program. We can easily understand it.

Pseudo code for calculating volume of cuboids

Input Length, Breadth and Height

Calculate Volume= Length * Breadth * Height

Output Volume

Decision Tree: It is a tree like structure which explains the alternative possible condition of the particular problem.

Decision Table: Decision table is where number of alternative possible condition of a program is to form a program.

Programming Statement:

An instruction or expression (also called code) written in high level language to do a specific task in a program is called programming statement. A programming statement may consist of keyword, constant, variables, operators, control statement, data type, library function, user-defined function etc. Three types of statement are used:

Simple Statement:

A simplest executable entity is called statement. A simple statement is a basic part of program and it is a single line expression which is used to carry out assignment, calculation or to test logical decision. Examples:

L=10 assign integer value 10 to L variable.

LxB Multiplies the value of L and B.

P>=80 Logical statement to check p is greater than or equal to 80 or not?

Compound Statement:

A single instruction composed of two or more individual instructions is called compound statement. Example of compound statement to calculate circumference of a circle. C=2*pi*r;

Control Statement / Program Control Structure:

A statement that affects the flow of execution through a program is called control statement. Control Statement is also called control structures in high level languages. There are three types of control statements:

1. Sequence
2. Selection/ Selection / Branching
3. Iteration / Repetition / Looping

The structures which regulate the order in which program statements are executed are called Control Structures. There are 3 types of control structure. They are:

- **Sequence:**
  It is the set of program instructions which follow one another and are to be executed unconditionally (not dependent on any program conditions). Instructions are put in a predefined sequence (just like a queue in a cinema hall) and the next instruction is executed by CPU only after the execution of the previous instruction (C never comes before B).

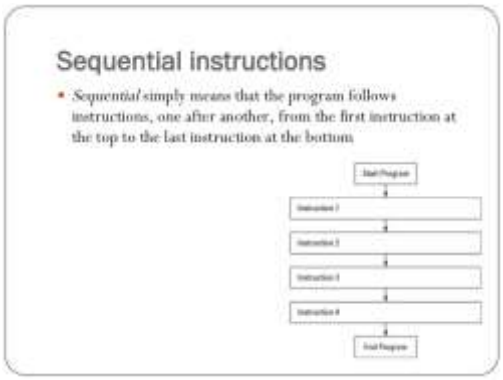


Fig: Sequential Statement

- **Selection:**
  It is the set of instructions which are to be executed conditionally i.e. they are executed based on a condition that can be either true or false. Commonly used logic for selection are if condition, if else condition, if else if condition, nested if else condition and switch case condition.

1. **If condition:**
   If condition is used in case the given problem has only one condition and only one action. Considering either true or false part, if the given condition is true then the statement will be executed. Otherwise, the control exits from the condition.

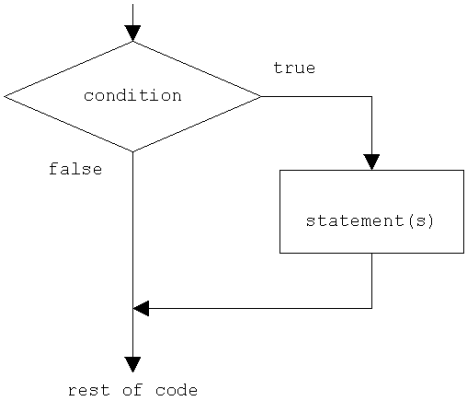
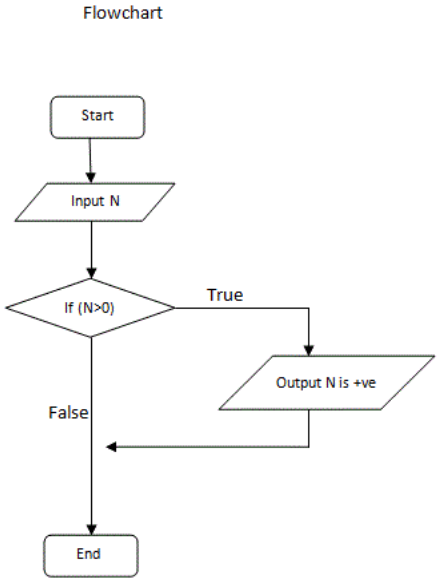


Fig: if condition

Example: Check the number is positive.
Algorithm
Step 1: start
Step 2: input N
Step 3: if (n>0), output is positive
Step 4: end
Figure:



2. **If else condition:**
This condition is used if the problem has one condition but two alternative actions. Here, if the condition is true, statement 1 will be executed; otherwise, statement 2 will be executed.

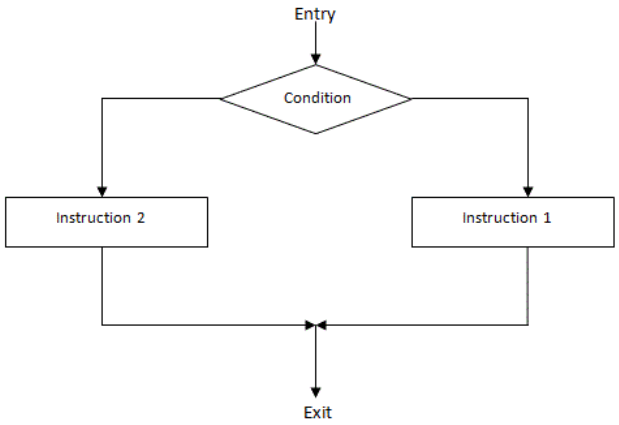
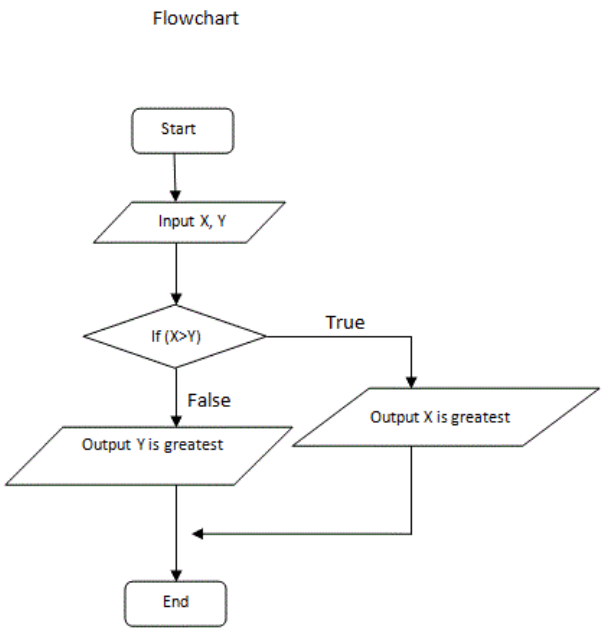


Figure: if else condition

Example: Find the greatest between 2 numbers.
Algorithm
Step 1: start
Step 2: input X, Y
Step 3: if (X>Y)

output X is greatest
else
output Y is greatest
Step 4: end

Figure:



3. **If else if condition:**
   Also known as ladder type if else, we can use this condition if the given problem has more than one interrelated conditions with their respective actions. Here, on a check, if condition 1 is true then, statement 1 is executed. Otherwise, condition 2 is checked and if it is true, statement 2 is executed and so on for next conditions. If all conditions are false, then the last statement will be executed.

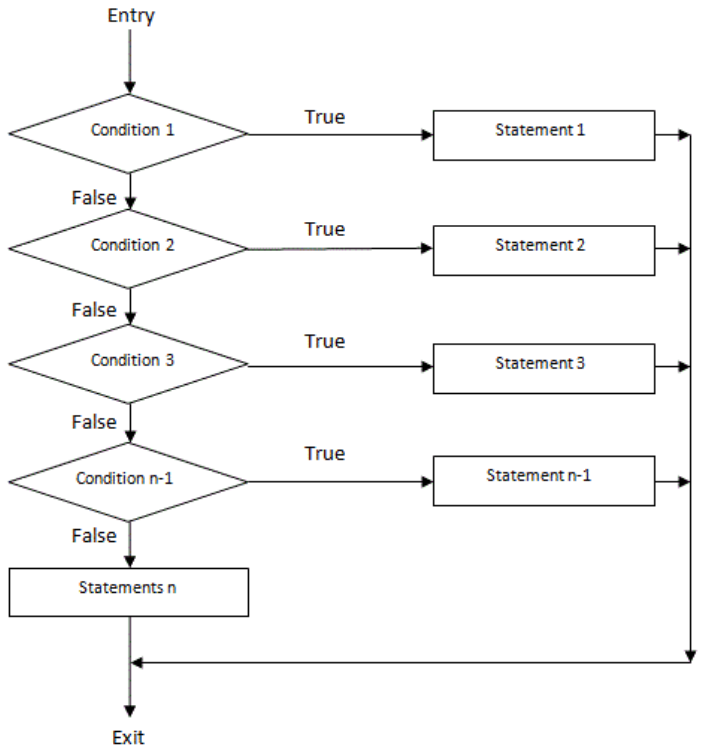
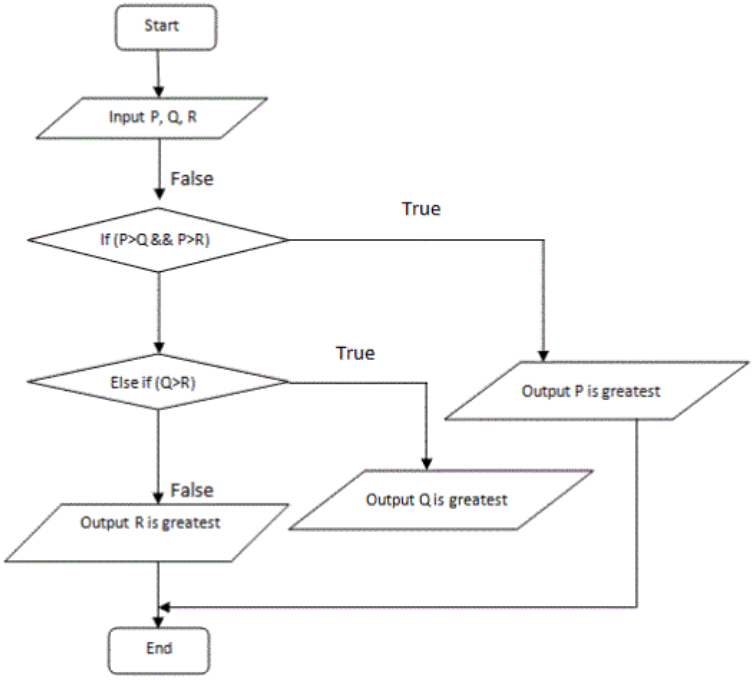


Figure: if else if condition

Example: Find the greatest among 3 numbers.
Algorithm
Step 1: start
Step 2: input P, Q, R
Step 3: if (P>Q && P>R)
output P is greatest
else if (Q>R)
output Q is greatest
else
output R is greatest
Step 4: end

Figure:

Flowchart



4. **Nested if else condition:**
   Nested if else condition is an entire if-else statement which is written within the body of if part or else part of another if else statement. This condition is used when a condition is to be checked that is inside another condition at a time in the same program, to make a decision.

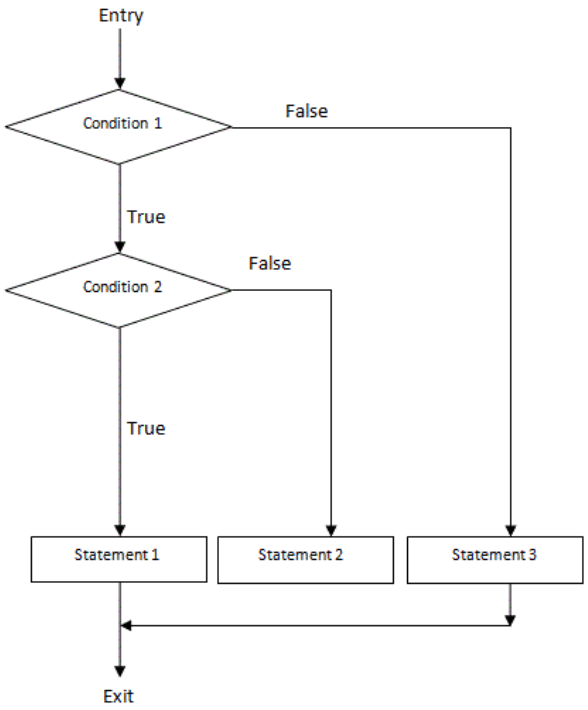
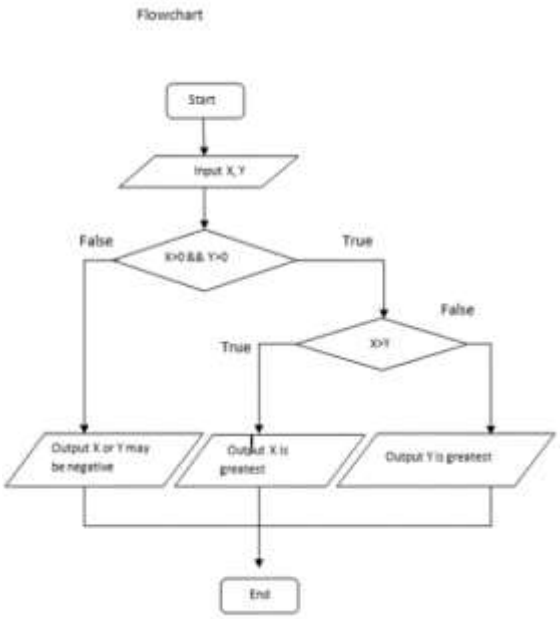


Figure: nested if else condition

Example: Find the greatest 2 positive numbers.
Algorithm
Step 1: start

Step 2: input X, Y
Step 3: if (X>0 && Y>0)
{
if (X>Y)
output X is greatest
else
output Y is greatest
}
output X or Y may be
negative
Step 4: end

Figure:



5. **Switch case condition:**
   If the given problem has one condition and respective more than two actions, then in this type of case scenario, we can use Switch case condition. It is the multiple branching statements which checks the value of the variable to the case value and then, the statements that are associated with it will be executed. If any expression does not match any of the case value, then the default statement will be executed.

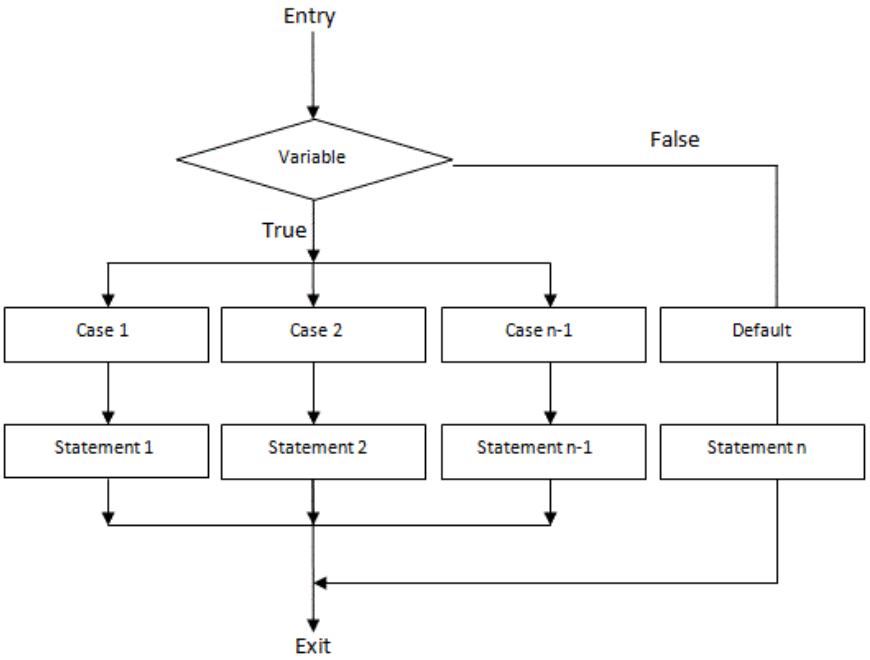


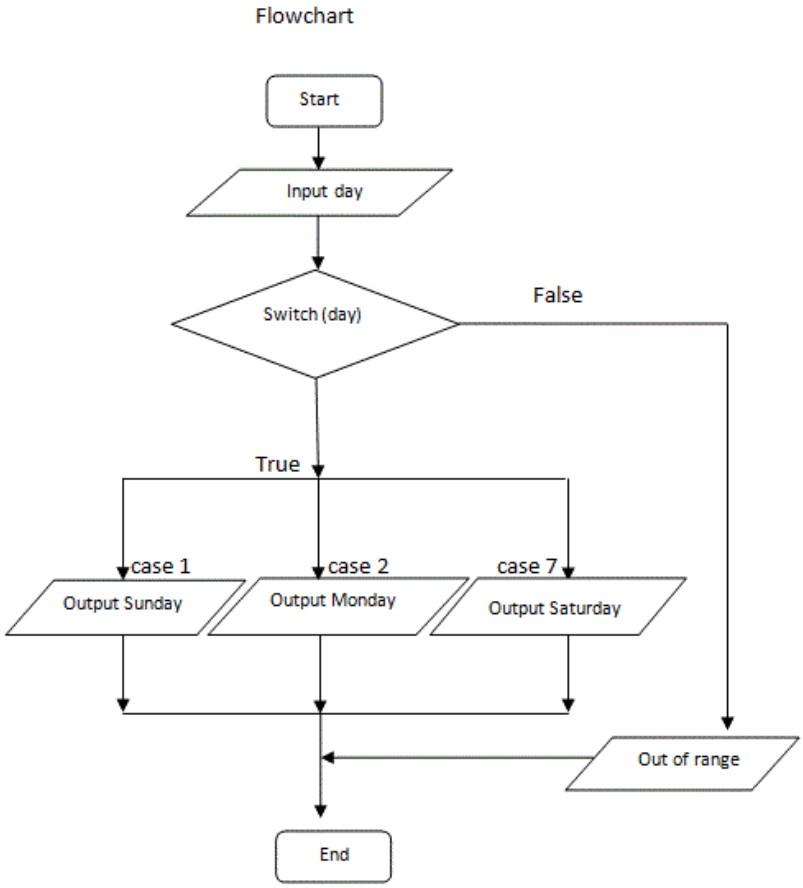
Figure: switch case condition

Example: Write an algorithm and draw a flowchart which takes the integer value 1 to 7 and prints respective day.

Algorithm
Step 1: start
Step 2: switch (day)
{
case 1: output Sunday
break
case 2: output Monday
break
case 3: output Tuesday
break
case 4: output Wednesday
break
case 5: output Thursday
break
case 6: output Friday
break
case 7: output Saturday
break
default: output out of range
}

Step 4: end

Figure:



- **Iteration:** These are the computer instructions which are to be performed repeatedly and conditionally i.e. loop statements are driven by the loop condition. Commonly used logic for iteration are while loop, do while loop and for a loop.

1. **While loop:**
   In this loop, first, the condition is checked by the computer and if the condition turns out to be true, then the statement inside the loop is executed. This process is repeated and the value of increment and decrement operator is always changing. When the condition is false, the loop stops.

   Algorithm Syntax
   Initialization
   while (condition)
   {
   statements
   …….
   …….
   increment/ decrement
   }

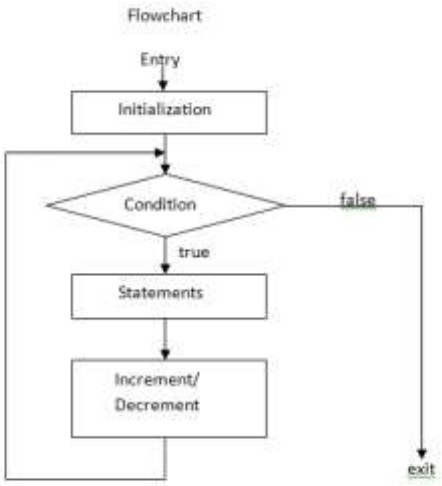
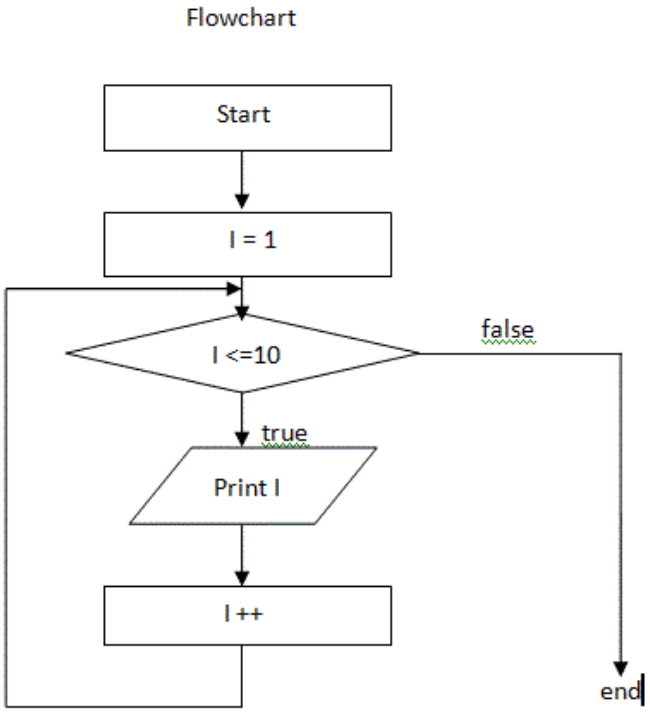


Figure: while loop iteration

Example: Write an algorithm and draw a flowchart to print 1 to 10.

Algorithm
Step 1: start
Step 2: I=1
Step 3: while (I<=10)
{
output I
I++
}
Step 4: stop

Figure:



2. **Do while loop:**
   In this loop, first, the computer checks the initial value; second executes the statements inside the loop and finally, checks the condition. The process is repeated for next pass, if the condition is true. Otherwise, the loop stops. If the condition is initially false, it will execute for at least one time.

   Algorithm Syntax
   Initialization
   do {
   statements
   ……..
   ……..
   increment/ decrement
   } while (condition)

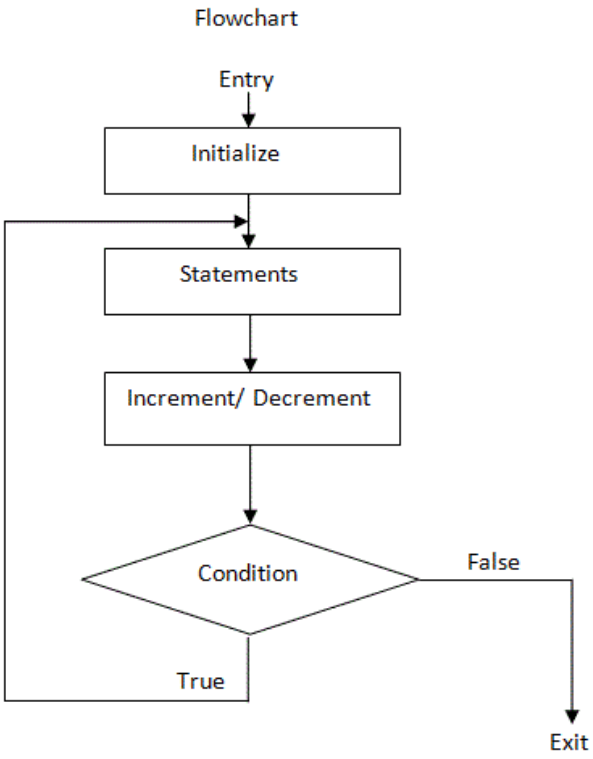
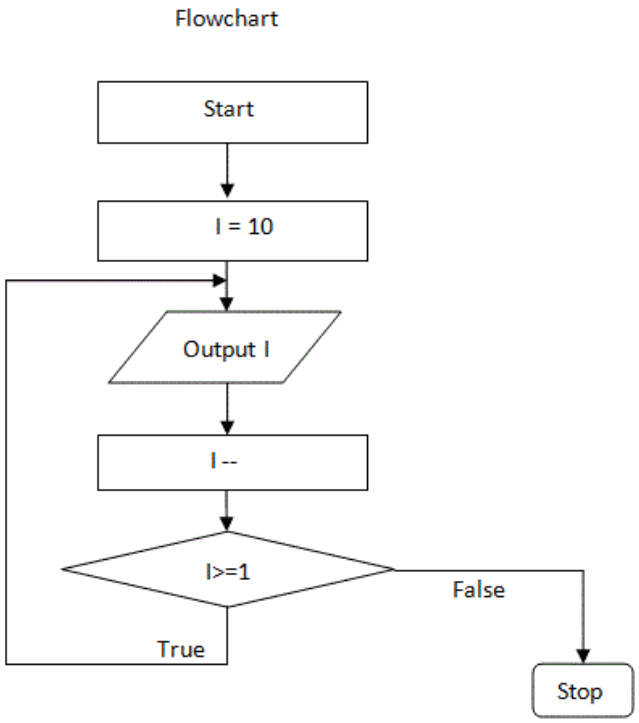


figure: do while loop

Example: Write an algorithm and draw a flowchart to print 100 to 1.
Algorithm
Step 1: start
Step 2: A=100
Step 3: do {
output A
A++
} while (A>1)
Step 4: stop

Figure:



3. **For loop:**
   It is the most commonly used loop. It consists of 3 expressions; initialization, condition and counter, which are defined within a statement.

   Algorithm Syntax
   for (initialization; condition; counter)
   {
   statements
   …….
   …….
   }
   Where, initialization is starting point,
   the condition is stopping point and increment/ decrement is a counter.

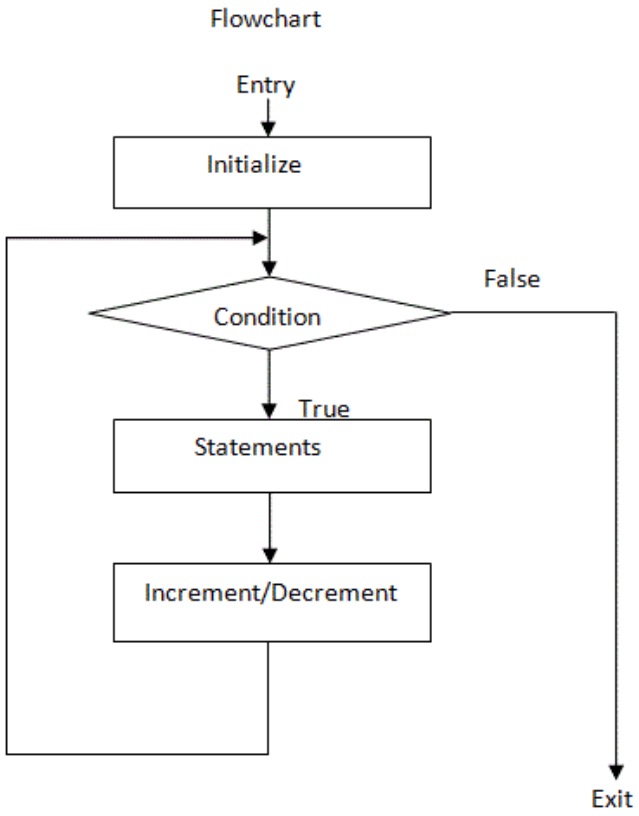
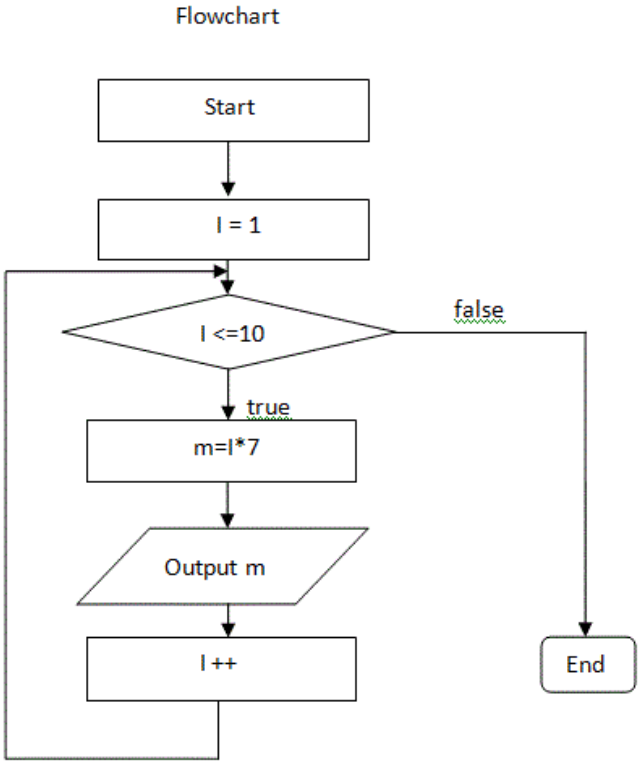


figure: for loop

Write an algorithm and draw a flowchart to print a multiplication table of 7.
Algorithm
Step 1: start
Step 2: for (i=1; i<10; i++)
{
m= i*7
output m
}
Step 4: stop
Write an algorithm and draw a flowchart to print a multiplication table of 7.
Algorithm
Step 1: start
Step 2: for (i=1; i<10; i++)
{
m= i*7
output m
}

Step 4: stop



Data Representation Codes (Internal Storage Encoding of Characters)

INTRODUCTIONTO CODES

A computer can only understand binary numbers which are in the form of two electronic states i.e. high voltage and low voltage. Such notations are further derived into standard codes and such codes can represent the data for users' convenience. Some of the popular codes are:

1. **Absolute Binary (pure binary):** In an absolute binary method, 0 is placed before the binary number to represent positive number and 1 is placed before the binary number to represent a negative number. The most significant bit in binary number denotes the sign bit and the rest bits represent the actual number. The binary number is expressed in 8,16,32,64, etc. bit format.

Fig: pure binary clock with Arduino

2. **BCD (Binary Coded Decimal):** It is a simple system for converting decimal numbers into a binary form where each decimal number is converted separately into binary and placed spaces in between numbers. In BCD, each decimal digit occupies 4 bit. For example, the decimal number 24 can be represented in BCD as $(0010\ 0100)_2$.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Fig: Binary Coded Decimal

3. **ASCII (American Standard Code for Information Interchange):** ASCII is a standard coding system that assigns numeric values to the letter, numbers, punctuation marks and control characters to achieve compatibility with different hardware and peripherals. ASCII was developed in 1968 and was divided into 2 sets: Standard ASCII (7 bits code, 128 characters) and Extended ASCII (8 bits code, 256 characters). Most systems use 8 bit extended ASCII to represent foreign language characters and other graphical symbols.

ANSI Extended ASCII (Windows)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	□	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	□	□
9	□	‘	’	“	”	•	–	—	˜	™	š	›	œ	□	□	Ÿ
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

cplusplus.com

Fig: ANSI Extended ASCII

In ASCII, each character is represented by a unique integer value from 0 to 255. The value 0 to 31 is used for non-printing control characters and the range from 32 to 127 is used to represent the letters of the alphabet and common punctuation symbols. For example: ASCII code for capital letter A is 65, for * is 42, etc. Since, ASCII code uses 8 bits, each character represented in ASCII code occupies 1-byte storage space in a computer.

EBCDIC (Extended Binary Coded Decimal Interchange Code):

It is an 8-bit code system which is commonly used on large IBM mainframe computers, most IBM minicomputers and computers from many other manufacturers. It allows 256 characters to be represented in computers.
In this code, placement of the letters of the alphabet is discontinuous and there is no direct character to character match when converting from EBCDIC to ASCII and vice versa.

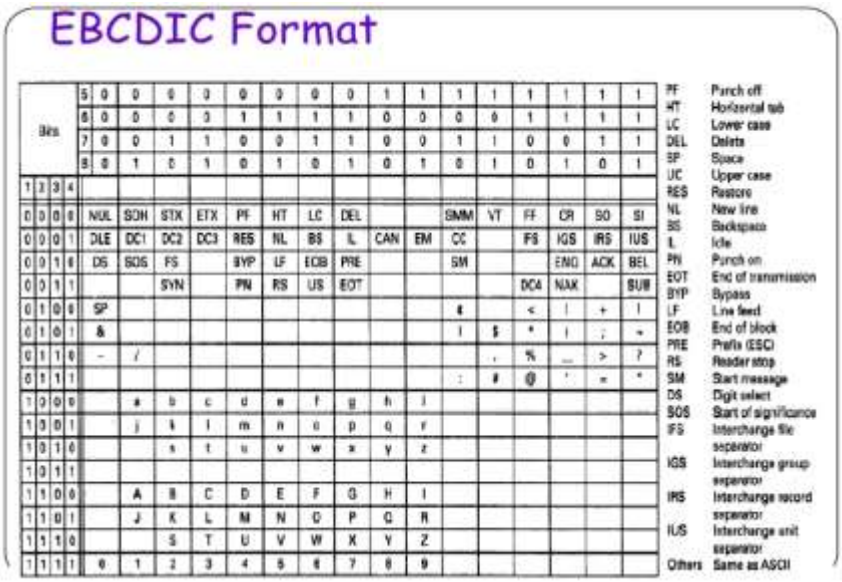


Fig: EBCDIC Format

Unicode: It is a 16-bit character code defined by the Unicode Consortium and International Standard Organization (ISO) that supports up to 65,536 characters. It allows all the characters and symbols in any language in the world to be represented by a single code. For example, the Chinese language has almost 10,000 characters which can be represented by Unicode only. If Unicode is universally adopted, then it will make multilingual software much easier to write and maintain.

Since, Unicode uses 16 bits, each character represented in Unicode occupies 2 bytes storage space in the computer. This coding system has been developed to overcome the drawback of ASCII code that supports only 256 different characters, which is sufficient only for English language but not for all the languages like Chinese, Japanese, etc. which has more than 256 characters. The Unicode Worldwide Character Standard provides up to 4 bytes (32 bits) now.

5.2 C Programming Languages

What is C? Explain its development process.

C is a high level language because no need for any architecture knowledge in normal English form.

C is a compiler because it can translate the whole program at a time so we can call compiler. C is structured programming language. It is called also procedural oriented programming language, function oriented language, module programming language. It is simple, reliable and easy to use.

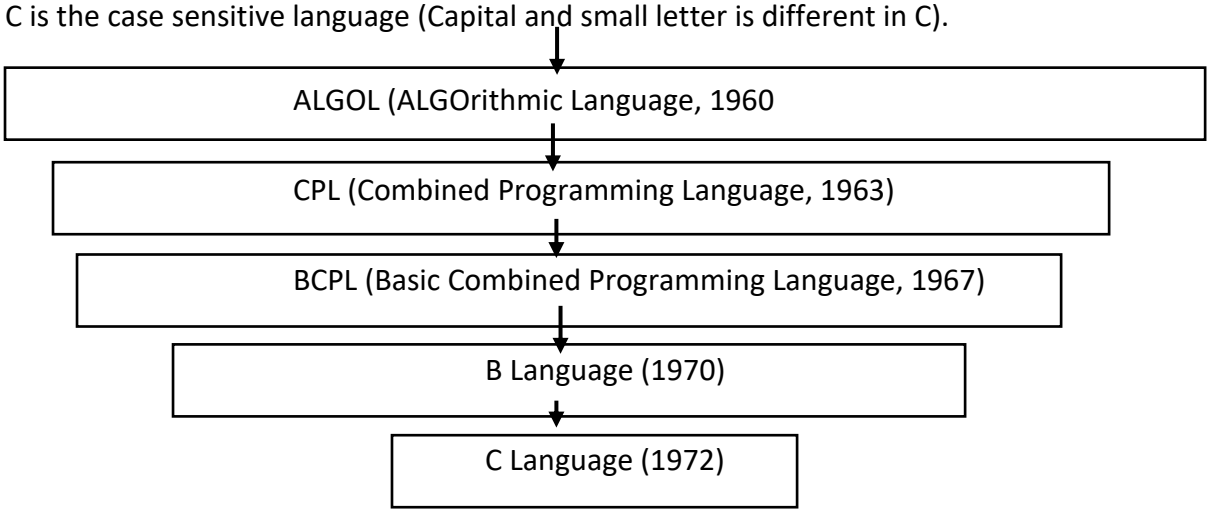
Development Process (History of C):

C was originally developed in the 1970 by Denis Ritchie at bell telephone Laboratory in corporation (now a part of AT and T). It is an outgrowth of two earlier languages called BCPL (Beginners computer programming language) which was also developed language at Bell Laboratory until 1978 from a Brian Kernighan and Ritchie published a definitive description of Language.

1970-1972 = Dennis Ritchie at the Bell Laboratory USA.

1989 = Government of USA clear the C language is the computer language.

C is the case sensitive language (Capital and small letter is different in C).



What are the features of C? What are its advantages and disadvantages?

C is a computer language and a programming tool which has grown popular because programmers preferred it. It is a tricky language but a masterful one.

The C programming languages has the following ***features:***

i) It has small size.
ii) It has extensive use of function call.
iii) It is a strong structural language having powerful data definition methods.
iv) It has low level (Bit Wise) programming available.
v) It can handle low level activities.
vi) Pointer makes it very strong for memory manipulations.
vii) It has level constructors.
viii) It can produce efficient programs.
ix) It can be complied on variety of computers.

Advantage of C language

- It is machine independent programming language.
- It is easy to learn and implement C language.
- It can be implemented from mobile device to mainframe computers.
- It is the mother of all modern programming language.

Disadvantage of C Language

- There is no runtime checking.
- It has poor error detection systems.
- On large programs, it is hard to fix errors.
- It does not support modern programming methodologies oriented programming language.

What is preprocessor? Explain with its types.

The compiler of C has a preprocessor built into it. Lines that begin with # are called pre-processor directives. Each C program must start with proper header files (i.e.<stdio.h>) starting with '#', sign, 'include' and a header file name enclosed within triangle brackets.

- The preprocessor handles directives for source file inclusion (#include), macro definitions (#defiine), conditional inclusion (#if, #ifdef, #elif, #endif) and miscellaneous directive (#pragama, #undef, #error).
- The languages of preprocessor commands are considered as a language within C language. The preprocess offers several feature called preprocessor directives.

What is header file? List the header files with some functions, where are they used?

A file that is defined to be included at the beginning of a program in C language that contains the definitions of data types and declarations of variables used by the functions in the program is called header file.

- Header files commonly contain forward declarations of classes, subroutines, variables, and other identifiers.
- The header file in c is called standard library functions.
- The entire header file has the extension .h
- A header file is used to define constants, variables, macros and functions that may be common to several applications.
- The updating and reading data of any function can be performed by using the header files.

Some of the frequent used header files are explain below:

S.N.	Header File	Description	Main Functions
1	stdio.h	Standard input and output	fpen(), fclose(), rename(), gets(), puts(), getchar(), scanf(), printf() etc.
2	conio.h	Old MS-DOS compiler header file, used for console input & output.	getch(), getche()
3	math.h	Mathematical calculation in C program.	sin(x), cos(x), log(x), pow(x,2), sqrt(x), cbrt(x), ceil(x), floor(x) etc.
4	complex.h	Complex arithmetic	cpow(x,p), csqrt(x), ctan(x), ctanh(x), cabs(x)
5	string.h	String / Words manipulation function	strlen(y), strcpy(z,y), strcmp(z,y), strcat(z,y), strupr(y), strlwr(y) etc.
6	ctype.h	Character manipulation type header file	toupper(y), tolower(y), isupr(y), isspace(), isalnu(y), toascii(y) etc.
7	stdlib.h	General purpose standard library.	rand(), malloc(), calloc(), abort(), exit() abs(), free() etc.

Fundamentals of C

1) What are the character set used in C?

A group of alphabetic, numeric and other characters that have some relationship with C programming language and recognized by compiler is called Character set. A character set can also contain additional characters with other code values.

The keywords, identifiers and other variables are constructed by using character set. The character set consists of following elements.

1) Alphabets:	= Uppercase (i.e. A,B,.....Y,Z) and Lowercase (i.e. a,b,.....y,z)
2) Digits:	= 0,1,2,.....8,9
3) Special Symbols	= + - * / = ( ) , { } < > ' " ! # % & _ ~ \ ; : ?
4) White space characters:=	blank, new line, tab etc.

2. **Define the term identifier, keywords and Tokens.**

Identifiers:-

Identifiers can be defined as the name of the variables, functions, arrays, structures etc created by the programmer. They are the fundamentals requirement of any language. The identifiers are defined according to the following rules:

- Identifiers consists letters and digits.
- First character must be an alphabet or underscore.
- Upper case and lowercase are allowed but not same, i.e. Text not same as text.
- Only one special character underscores (_) will used.

  For example, int a_b; Where ***a*** and ***_b*** are valid identifiers.

Keywords:-

Keywords are the reserved words which have standard, predefined meaning in C language. Keywords cannot be used as names for the variables or other user defined program elements. There are 32 keywords available in C. ***common examples are as follows.***

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

Tokens:

In a C source code, the basic element recognized by the compiler is known as tokens. A token is source-program text that the compiler does not break down into components elements.

- The keywords like int, float, if, for etc.
- Identifiers like main, printf, void etc.
- Constants like a,b,c etc.
- String literals like name, address, phone etc.,and
- Operators like &&, ! etc.
- Punctuation characters such as [,] , { , } , (,) , ; , : are also tokens.

2) Explain data types used in programming with examples.

Data types:

It is the set of keywords to declare variables. A set of data that specifies the possible range of values in a program and stored in memory are called data types. Data types are used to define variables before use it.

Types of data types in C.

1) Primary data types
2) Secondary data types

Primary Data Types: The basic fundamental of data having unit feature on C programming is called Primary Data Type. Example

Data Type	Type	Memory Require	Format Specifies
Char	Character	1 byte	% C
Int	Integer	4/2 byte	%d
Float	Floating point number	4 byte	%f
Long	Floating number	4 byte	%ld
Double	Large floating point number	8 byte	%lf
long double	Very large floating number	12 byte	%lf

Variable:

Variable are simply names that can change the value while executing a program. It allocates memory space inside a memory of computer. A variable can have only one value assigned to it in every time of execution of the program. Its value can change in different executions.

- ***Rules for variable declaration***
  - ✓ They must always begin with a letter, although some systems permit underscore as the first character.
  - ✓ White space is not allowed.
  - ✓ A variable should not be a keyword.
  - ✓ It should not contain any special characters.

Types of variable

1. Numeric Variable: The variable that store numeric data only is called numeric variable. The numeric data may be whole number or fractional number. Examples are integer, floating point and double.

2. String Variable: The variable that stores character data only is called string variable. The string data may be single character or string. Examples are character, array of character (string), table of string.

Types of variable declaration are described below.

- **Static variable declaration:** Any variable which is declared by using keyboard static is called static variable. The value of static variable is remain fixed for the other function but may change within same function boundary.
- **Global variable declaration:** Any variable which is declared before main function is called global variable. This variable can be accessed from any member functions.
- **Local variable declaration:** Any variable which is declared within the function is called local variable. This type of variable can be accessed within the same member function only.

Constant variable:

A constant is fixed entity. It does not change its value during the entire program execution. Constants can be classified as:

1. Integer constants
2. Floating point constants
3. Character constants
4. String constants
5. Symbolic Constants
6. Escape sequence constants

Specifier:

The input and output data are formatted by specific pattern. These Patterns are generated by using specific tokens in C programs. These tokens used to format data are called specifier. Most of the specifier used by printf and scanf functions. Types of mostly used specifier are explained below.

- ❖ **Escape Sequence:** They are a type of specifier. These non printable characters are used to format text on the output screen. These escape sequence character are place after backslash \.

Escape sequence	***Name***	***Meaning***
\'	Single quote	It prints ' in output
\"	Double quote	It prints " in output
\n	New line	It creates new line in output display
\t	Tab	It creates tab or 8 spaces in place of \t

- ❖ **Format Specifier:** The output and input data are display and receive in specific pattern. Format specifier uses the token % and character(s) after it. It is used to format for all types of data i.e. integer, float, character and string.

Format Specifier	***Used by scanf() function***
%d , % i	Signed integer + or – number o to 9
%f	Scans floating point numbers.
%s	String, Collection of character i.e. word
%c	Character, one single key stroke.

Statements: Simple and Compound Statements

Statement: Statement causes the computer to carry out some action. It is terminated by semicolon.

- A smallest executable entity within a program code is called a statement.
- An instruction or one line of code written to do a specific task in a program is called programming statement.
- Statements are the basic building blocks of C programming language. These statements enable the computer to carry out some calculation or perform logical comparison between values and variables.
- A programming statement may consist of keyword, constant, variables, operators, control statement, data type, library function, user-defined function etc.

1. ***Simple statement:***
   A simple statement is a basic part of program and it is a single line expression which is used to carry out assignment, calculation or to test logical decision.
2. ***Compound Statement:***
   A single instruction composed of two or more individual instructions is called a compound statement. This type of statement is used to combine two or more statement in one single line of code.
3. ***Control Statement:***
   A statement that affects the flow of execution through a program is called control statement.

Operator:

An operator is a symbol that operates on a certain data type. The operator generally remains between the two operands. An expression is a combination of variables, constants, and operators written according to the syntax of the language. The data items that operators act upon are called operands.

Types of operator

1. Arithmetic Operator(Binary Operator)
2. Relational Operator (Comparison Operator)
3. Logical Operator (Boolean Operator)
4. Assignment Operator
5. Increment and Decrement Operators (Unary Operator)
6. Conditional Operator (Ternary Operator)
7. Bitwise Operator
8. Comma Operator
9. Size of Operator

1. Arithmetic Operator

The arithmetic operators perform arithmetic operations and can be classified into unary and binary arithmetic operations. The arithmetic operators can operate on any built-in data type. A list of arithmetic operators and their meanings are given below:

Operator	Meaning
+	additional or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	modulo division (returns remainder after division)

2. Relational Operator

The relational operators help to compare two similar quantities and depending on their relation, take some decisions. If the condition is true, it evaluates to an integer 1 and zero if the condition is false. The basic types of relational operator are:

Operator	Meaning
<	less than
>	greater than
<=	less than or equal to

>=	greater than or equal to
==	equal to
!=	not equal to

3. Logical Operator

The logical operators are used to give logical value either true or false. They compare or evaluate logical and relational expressions. There are three logical operators.

Operator	***Meaning***	***Examples***
&&	Logical AND	(a>b) && (a>c)
||	Logical OR	(a>b) || (a>c)
!	Logical NOT	!(a==b)

4. Increment and Decrement Operators:

The increment and decrement operators are very commonly used in C language. The increment operators and decrement operators are extensively used in the loops using structures such as for, while, do, etc. The syntax of the operators is given below.

++<variable name>	Pre increment
--<variable name>	Pre decrement
<variable name>++	Post increment
<variable name)--	Post decrement

The pre increment operator increases the value of the variable by 1 and then the processing does whereas post increment first processes and increase the value of it by 1.

5. Conditional Operator:

A conditional operator is very rarely used. This can be carried out with the conditional operator (? :) An expression that makes use of the conditional operator is called a conditional expression. This can replace the if-else statement. The syntax of conditional operator is:

Expression_1 ? expression_2: expression_3

During evaluating the conditional expression, expression_1 is evaluated at the first step. If expression_1 is true (nonzero), then expression_2 is evaluated and this becomes the value of the conditional expression.

- **Type casting or conversions:**

The process of converting one type of data into another type is called type conversion. Its types are

4. ***Implicit type conversion:*** The conversion of data is performed either during compilation or run time is called implicit type casting. It is the automatic type conversion process performed by compiler itself. The data can be lost during this type of type casting.
5. ***Explicit type conversion:*** Explicit type conversions can either performed by built in functions or by a special syntax generated by coder. These syntax changes one data type to other by using conversion keyword. It is a secure manner of changing variable from one data type to others.

Library function:

The special functions that are well defined in C programming languages are called library functions such as printf(), scanf(),strlen(), sqrt(), tolower(), toupper(), getchar(), putchar() etc.

Control Structure:

Control structures are those programming constructs which control the flow of program statements execution in a program. Types of Control Structure

i) Branching / Decision (Selective Control Structure)
ii) Looping (Repetitive Control Structure)
iii) Jumping (Unconditional Control Structure)

1. **Decision (Selective) Control Structure**

It is mainly used for decision making. It is also called conditional statements. Selection is made on the basis of condition. We have options to go when the given condition is true or false. The flow of program statements execution is totally directed by the result obtained from checking condition.

Types

a) Conditional Statements

i. if statements:

It is used to execute an instruction or block of instructions only if a condition is fulfilled.

Syntax,

```
if(condition)
{
    Statements;
}
```

E.g. WAP to read a number and find even or odd by using if().

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a,r;
    printf("enter the number");
    scanf("%d",&a);
    r=a%2;
    if(r==0)
    {
        printf("number is even %d",a);
    }
    if(r!=0)
    {
        printf("number is odd %d",a);
    }
    getch();
}
```

ii. <u>if else statements</u>

If the condition is true then the if() portion statements are evaluated otherwise else part of the statements are evaluated.

Syntax,

```
if( condition)
{
    Block of statements;
```

```
}
else
{
        Block of statements;
}
```

E.g. WAP input any two numbers and display the largest one.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b;
printf("enter the number");
scanf("%d",&a);
printf("enter the number");
scanf("%d",&b);
if(a>b)
{
        printf("A is greatest number",a);
}
else
{
        printf("B is greatest number",b);
}
getch();
}
```

iii. if() else if() statements

When we have two or more condition to be checked in a series we can use if else if statement. It is also known as multiple conditional statement or multipath conditional statement /if else ladder.

Syntax,

```
if(conditiona 1)
{
        Statements  1;
}
else if(condition  2)
{
        Statement 2;
}
else if(condition n-1)
{
        Statement n-1;
}
else
{
        Statement n;
}
```

```
}
```

e.g. WAP to find the largest number among three input number .

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,c;
printf("\n Enter any three number");
scanf("%d%d%d",&a,&b,&c);
if(a>b&&a>c)
{
        printf("\n Largest number is %d",a);
}
else if(b>a&&b>c)
{
        printf("\n Largest number is %d", b);
}
else
{
        printf("\n Largest number is %d",c);
}
getch();
}
```

iv. Nested if else statements

An entire if else statement written within the body of if part or else part of another if else statement is called nested if else statement. It is used when a condition is to be checked inside another condition at a time in the same program to make decision.

Syntax,

```
if(condition 1)
{
        if(condition 2)
        {
                Statements 1;
        }
        else
        {
                Statements 2;
        }
}
else
{
        Statement 3;
}
```

E.g. WAP that reads marks of five subject and calculate total mark and percentage. Also awards the division on the basis of following criteria.

Percentage	***division***
p>=75	***distinction***
p>=60 and <75	***first***
p>=45 and <60	***second***
p>=35 and <45	***third***
otherwise	failed

```
#include<stdio.h>
#include<conio.h>
Void main()
        {
                int eng,nep,comp,acc,eco,total;
                Float per;
        printf("Enter the five subject mark");
        scanf("%d %d %d %d %d",&eng,&nep,&comp,&acc,&eco);
        total=eng+nep+comp+acc+eco;
        per=total/5;
        if(eng>=35&&nep>=35&&comp>=35&&acc>=35%%eco>=35)
        {
            if(per>=75)
              {
                printf("\n Distinction");
          else if(per>=60)
                {
                printf("\n First");
                }
           else if(per>=45)
                {
                printf("\n Second");
                }
         else
                {
                 printf("\n Third");
                }
        }
else
{
        printf("\n You are failed");
}
getch();
}
```

b) Switch case statements

The switch statement can be used instead of multiple if() else conditional statements. The switch control statement is mainly used to generate menu based programs.

Syntax,

```
Switch(expression 1)
{
        Case condition 1:
                Statements ….
                break;
                .
                .
        Case condition n-1:
                Statements…….
                break;
        default:
                statement n;
}
```

E.g. WAP which reads any two integer values from user and calculates sum, difference and product using switch case statement.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
        int a,b,c,ch;
        printf("enter the two number");
        Scanf("%d%d",&a,&b);
printf("\n 1. Sum");
printf("\n 2. Difference");
printf("\n 3. Product");
        printf("\n Enter your choice");
        Scanf("%d",&ch);
switch(ch)
{
        case 1:
                c=a+b;
                Printf("\n Sum of two number is %d,c);
                break;
        case 2:
                c=a-b;
                printf("\n difference of two number is %d,c);
                break;
        case 3:
                c=a*b;
                printf("\n product of two number is %d,c);
```

```
                    break;
                default:
                    printf("\n Wrong choice");
        }
        getch();
}
```

1. Looping Statement

The looping statement is also called repetitive or iterative control structure. Looping statements are the conditional control flow statements that repeats a certain portion of the program either a specified number of times or until a particular condition is satisfied or true.

Types of loop

i) For Loop ii) While Loop iii) Do while Loop

1**. For Loop:-**

The execution of for loop until the condition is true. The for loop is a entry control loop because it checks the condition at entry point.

Syntax,

```
for( initialization; condition; increment/ decrement)
        {
            // statements
        }
```

1. Write a program to print the natural number from 1 to 10.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i;
    for(i=1;i<=10;i++)
    {
            printf("/n%d",i);
    }
            getch();
}
```

2. Write a program to display even number from 1 to 20 and display their sum also.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i, s=0;
    for(i=2;i<=20; i=i+2)
    {
            printf("/n  %d",i);
            s=s+i;
    }
    printf("sum of even number is %d", s);
    getch();
```

```
}
```

3. Write a program to find out sum of the cubes of first 10 numbers.

```
# include<stdio.h>
# include<conio.h>
void main()
{
    int i,c, sum=0;
    for(i=1;i<=10;i++)
    {
            c=i*i*i;
    sum=sum+c;
    }
printf("/n sum of cube is %d", sum);
    getch();
}
```

Nested for loop:

When for loop is declared inside another for loop is called nested for loop. The life of the inner for loop is depending over the outer for loop. If the outer for loop condition is true then inner for loop is evaluated. And will executes all the statements until condition is true if the inner for loop to be false then the outer for loop condition is reevaluated and so on.

For example

```
for( initialization; condition; increment/ decrement)
    {
        for( initialization; condition; increment/ decrement)
            {
                // statemetns
            }
    }
```

1. Display the following output.

10 20 30 40 50
10 20 30 40 50
10 20 30 40 50
10 20 30 40 50
10 20 30 40 50

2. WAP to display the following output.

55555
4444
333
22
1

Programs

```
# include<stdio.h>
# include<conio.h>
void main()
{
    int R,K;
    for (R=1;R<=5; R++)
    {
            for (K=10;K<=50;K=K+10)
            {
                    printf("\t %d",K);
            }
            printf("\n");

  }
    getch();
}
```

```
# include<stdio.h>
# include<conio.h>
void main()
{
        int R,K;
        for (R=5;R<=1; R++)
        {
                for (K=1;K<=R;K=K++)
                {
                printf("\t %d",R);
                }
                printf("\n");

         }
        getch();
}
```

While Loop:-

The while loop is also a entry control loop. While loop first checks whether the initial condition is true or false and finding it to be true, it will enter the loop and execute the statement.

Syntax,

```
initialization;
while(condition)
{
        // statement
        increment/decrement
}
```

1. Write a program to print even number from 1 to 100.

```
# include<stdio.h>
# include<conio.h>
void main()
{
        int i;
        i=2
        while<i<=100)
        {
                printf("%d\t", i)
                 i=i+2;
         }
        getch();
}
```

Do while loop:-

This loop is an exit control loop. This loop runs at least the once even though the termination condition is set to false. This loop test the condition at exit point hence it is called exit control loop.

The syntax of the loop is similar to while loop.

```
initialization;
        do
        {
            // statement
            increment/decrement
        } while(condition);
```

1. ***Write a program to display odd number from 100 to 1.***

```
# include<stdio.h>
# include<conio.h>
void main()
{
    int i;
    i=99;
    do
    {
        printf("%d\t",i);
        i=i-2;

    }while(i>=0);
    getch();        }
```

2. ***Write a program to read the employee name, address for the N employee and display by using while loops.***

```
# include<stdio.h>
# include<conio.h>
void main()
{
    int num;
    char ename[20],eadd[30];
    printf("\n enter the how many employee number");
    scanf("%d",& num);
    while(num>0)
    {
        printf("\n enter the name, address");
        scanf("%s%s",ename,eadd);
        num=num-1;
    printf("%s%s",ename,eadd);
    }
    getch();
 }
```

S.N.	***While loop***	***S.N.***	***Do while loop***
1	It is an entry controlled loop.	1	It is an exit controlled loop.
2	Testing starting in top	2	Testing started is bottom.
3	It has keyword while	3	It has keyword do and while.
4	If first condition true then the statement is executed otherwise no.	4	But in case at least one time executed statement if the condition is false.
5	Loop is not terminated with semicolon.	5	Loop is terminated with a semicolon.
6	Syntax while (expression) { //statements }		Syntax do { //statements } while (expression);

The jump Statements

i) break statements
ii) continue statements
iii) goto statements

1) break statements:

The break statement is used to terminate loop or to exist form a switch. it can be used within a for, while, do while switch statement.

```
# include<stdio.h>
# include<conio.h>
void main()
{
        int i;
        for(i=1;i<=10;i++)
        {
                if(i==5)
                {
                        break;
                }
        printf("%d",i);
        }
        getch();
}                                   OUTPUT:     1,2,3,4
```

The continue Statement

It skips the remaining statements in the body a while, for or do .....while structure it proceeds with the next iteration of the loop. The continue statement is used to by pass the execution of the further statements.

```
# include<stdio.h>
# include<conio.h>
void main()
{
        int num;
        for(num=1; num!=10; num++);
```

```
{
        if(num==7)
        {
                continue;
        }
        printf("%d",num);
        }
        getch();
        }                    OUTPUT: 1,2,3,4,5,6,8,9
```

Go to statement:

The goto statement is used to send the pointer to the specified label. If the label is not defined then the goto statement will not work.

```
 # include<stdio.h>
# include<conio.h>
void main()
{
   int num;
   Lab:
   printf("Enter a number");
   scanf("%d", &num);
 if(num<100)
   {
   printf("Mark can't be less then 100");
   goto Lab:
   }
else
{
   printf("Valid mark");
}
   getch();
}
```

S.N.	Break Keywords (Statement)	S.N.	Continue Keywords (Statement)
1	The break statement is use to terminate the loop unconditionally.	1	The continue statement is used return the pointer at the beginning of the loop.
2	The break statement can also be use inside the switch statement.	2	But the continue can be use inside the switch statement.
3	The break statement is use to terminate the loop.	3	The continue statement is use to repeat set of statements.
4	Example `void main( )` `{` `int i;` `for (i=1; i<=10; i++)` `{` `If (i==5)` `{` `break;`	4	Example `void main( )` `{` `int num;` `for (num=1; num!=10; num+1)` `{` `If(num==7)` `{` `continue;`

	} printf("%d",i); } } *Output= 1,2,3,4.*		} printf("%d",num); } } *Output= 1,2,3,4,5,6,8,9.*

Some important C programs

1. Reverse order
2. Factorial number
3. Fibonacci series
4. Prime or composite number
5. Even or odd number
6. Palindrome or not.
7. Sum of individual digits
8. Armstrong number or not
9. Multiplication table
10. Find list of prime number
11. Display perfect square number.

1. WAP to input any value and display the that value in reverse order.

```
# include<stdio.h>
# include<conio.h>
void main()
{
        int i,r,n,x;
    printf("enter a number");
    scanf("%d",&n);
        x=0;
        while(n>0)
        {
                r=n%10;
                x=x*10+r;
                n=n/10;
        }
    printf("/n reverse number is%d",x);
        getch();
    }
```

2. WAP to input positive number and find its factorial number.

```
 # include<stdio.h>
  # include<conio.h>
  void main()
  {
        int i,n,f;
        printf("enter a number");
        scanf("%d", &n);
        if (n>0)
        {
                f=1;
                for(i=n;i>=1;i--)
                {
                        f=f*i;
                }
                printf("/n The factorial value is %d",f);
                }
                else
                {
                printf("/n it is not a positive number");
                }
                getch();
        }
```

3. WAP to display the Fibonacci series. 1 1 2 3 5 8 13 ……n.

```
#include<stdio.h>
#include<conio.h>
void main()
```

```
{
Int i,n;
 int x,y,z;
x=0,y=1,z=0;
printf("Enter the number");
scanf("%d",&n);
for(i=1;i<n;i++)
{
 printf("%d",z);
 z=x+y;
 x=y;
 y=z;
}
getch();
}
```

4.WAP read a number and to check the number is prime or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,num;
i=2;
printf("Enter the number");
scanf("%d",&num);
while(i<=num-1)
{
if(num%i==0)
{
        printf("Not a prime number");
        break;
}
i++;
}
printf("number is prime");
getch();
}
```

5.WAP to find out even number from 1 to 100 and find their sum also.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i,sum=0;
        For(i=2; i<=100;i+2)
        {
                printf("\n Even number are %d",i);
```

```
                sum=sum+i;
        }
        printf("\n Sum of even numbers is %d",sum);
getch();
}
```

6.WAP to input a number and find out that number is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,n,r,x;
        x=0;
        printf("\n Enter the any number");
        scanf("%d",&n);
        a=n;
        while(n>0)
        {
                r=n%10;
                x=x*10+r;
                n=n/10;
        }
        if(a==x)
        {
        printf("Number is palindrome");
        }
        else
        {
        printf("Number is not palindrome");
        }
        getch();
}
```

7.WAP to input a positive number and find out the sum of its individual digits.

```
        #include<stdio.h>
        #include<conio.h>
        void main()
        {
                int i,r;n,x;
                printf("\n Enter a number");
                scanf("%d",&n);
                if(n>0)
                {
                        x=0;
                while(n>0)
                {
                        r=n%10;
```

```
                x=x+r;
                n=n/10;
        }
        printf("\n the sum of digits is %d",x);
        }
        else
        {
        printf("\n It is not a positive number");
        }
        getch();
}
```

8.WAP to input a number and check it is Armstrong number or not.

```
#include<stdio.h>
        #include<conio.h>
        void main()
        {
                int i,r,n,x,a;
                printf("Enter a number");
                scanf("%d",&n);
                a=n;
                if(n>0)
                {
                        x=0;
                        while(n>0)
                        {
                                r=n%10;
                                x=x+r*r*r;
                                n=n/10;
                        }
                        if(a==x)
                        {
                                printf("\n Armstrong number is %d",a);
                        }
                        else
                        {
                                printf("\n Number is not Armstrong %d",a);
                        }
                }
        else
        {
                printf("\n It is not positive number");
        }
                getch();
        }
```

9.WAP to display the multiplication table of given number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int,i,n,m;
        printf("Enter the value ");
        scanf("%d",&n);
        for(i=1;i<=10;i++)
        {
        m=n*i;
                printf("\n %d * %d = %d",n,i,m);
        }
        getch();
}
```

10.WAP to display all prime numbers upto 1000.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,r,j,i;
        for(j=1;j<=1000;j++)
        {
                for(i=2;i<j;i++)
                {
                        if(j%i==0)
                        {
                        break;
                        }
                }
                if(i==j)
                {
                        printf("\t%d",j);
                }
                        getch();
        }
```

11.WAP to display all perfect square numbers from 100 to 500.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int s,t;
        printf("The square numbers between 100 to 500 are");
        for(s=100;s<=500;s++)
        {
```

```
        for (t=1;t<=s;t++)
        {
                if(s==t*t)
                {
                        printf("%d  x %d =  %d \n",s,s,t);
                }
        }
        getch();
}
```

Arrays and String Function:

Arrays : An array is a collection of data of the similar type all of which are referred by a single variable name. for example, instead of using 50 individual variables to store name of 50 students, we can use an array to store name of 50 students.

Advantage of arrays.

- It is easier for handling similar types of data in a program.
- It is efficient for solving problems like sorting, searching, indexing etc.
- It is easier to solve matrix related problems.
- Graphics manipulations can be easily be done using array.

Disadvantages of arrays.

- It is not possible to hold dissimilar type of data in an array.
- It is difficult to visualize the multi dimensional array.
- It is static in nature so it is difficult to define the size of array during running time.

There are two types

1. **One/signal dimensional:** The values on an array variable assigned in one row and more than one column are called signal dimensional array.

```
Syntax:     type    array_name[max. size];
Example     int n[10];
            int age[]= {18,12,19,20,16,16,17};
```

2. **Two/Double dimensional:** Two dimensional arrays are capable of storing data in multiple row and columns.

```
Syntax:     type    array_name[No.Rows] [No.Cols];
Example     int n[10][5];
            int matrix[3][3]= {{0,1,2},{3,4,5},{6,7,8}};
```

Program: ***Write a program to read 50 students marks and display them.***

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        int i, M[50];
        printf("Enter the 50 student Marks"):
        for(i=1; i<=50; i++)
        {
                scanf("%d",&M[i]);
        }
        printf("The numbers are:");
        for(i=1; i<=50; i++)
```

```
        {
                printf("%d\t",M[i]);
        }
}
```

Program: Write a program to input 5 numbers with constant values initialization in array to display the sum.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        int  i,sum=0;
        int num[5]={5,10,15,20,25};
        for(I=0;I<5;I++)
        {
                printf("%d\t",num[i]);
                sum=sum+num[I];
        }
        printf("\n Sum of 5 numbers is:%d",sum);
        getch();
}
```

Program: Write a program to input the age of 20 students and count the number of students having age in between 20 to 25.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i, age[10],c=0;
        for(i=0;i<20;i++)
        {
                printf("enter age of students:");
                scanf("%d",&age[I]);
        }
        for(i=0;i<20;i++)
        {
                if(age[i]>=20&& age[i]<=25)
                {
                        c++;
                }
         }
                printf("Total number of students having age between 20 to 25 is %d",c);
                getch();
}
```

Program: Write a program to find the largest number among 'n' numbers.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
        int i, n, num[100],max;
        printf("\n Enter the size of array not more than 100");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter array elements");
                scanf("%d",&num[i])
        }
        max=num[0];
        for(i=1;i<n;i++)
        {
                if(num[i]>max)
                {
                        max=num[i];
                }
        }
        printf("\n Largest number in array is %d",max);
        getch();
}
```

Program: ***Write a program to read a matrix, store it in array and display it.***

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        clrscr();
        int I,J, matrix[3][4];
        printf("Enter a matrix of 2x4:\n\n");
        for(I=1; I<=3; I++)
        {
                for(J=1; J<=4; J++)
                {
                        scanf("%d",&matrix[I][J]);
                }
        }
        Printf("\n The elements of matirx are: \n\n"):
        for(I=1; I<=3; I++)
        {
                for(J=1; J<=4; J++)
                {
                        printf("%d\t",matrix[I][J]);
```

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

Program: ***Write a program to calculate the average age of 10 students.***

```
# include<stdio.h>
# include<conio.h>
 void main( )
{
        int avg, i, sum=0;
        int age[10];
        for (i=1; i<=10; i++)
        {
                print("\n Enter age");
                scanf("%d',&age[i]);
        }
        for (i=1; i<=10; i++)
        {
                sum=sum+age[i];
                avg=sum/10;
        }
        printf("\n Average age=%d", avg);
}
```

Program: ***Write a program to accept the age of 10 different employees and count the no. of employee.***

i) Whose age is more than or equal to 60

ii) ***Whose age is less than 35***

```
# include<stdio.h>
# include<conio.h>
 void main( )
{
        int age[10];
        int count1=0;
        int count2=0;
        int i;
        for (i=1; i<=10; i++)
        {
                printf("Enter the ages");
                scanf("%d", &age[i]);
        }
                for (i=1; i<=10; i++)
                {
                        if (age[i]>=60)
                        {
                                count1++;
```

```
                        }
                        else
                        {
                                if(age[i]<=35)
                                {
                                count2++;
                                }
                         }
                    }
        printf("Employees above or equal to 60 %d",count1);
        printf("Employees under or equal 35 %d",count2);
}
```

Program: ***Write a program to store N numbers in array and print out the sum with the entire array variable.***

```
# include<stdio.h>
# include<conio.h>
 void main( )
{
        int j, sum=0;
        int n,num[10];
        clrscr();
        printf("Enter the how many no. you want to enter"):
        scanf("%d",&n);
        for (j=1; j<=n; j++)
        {
                printf("number stored %d",num[i]);
                sum=sum+num[j];
        }
        printf("\n Sum is %d",sum);
}
```

Program: ***Write a program to accept 10 different numbers in array and sort in descending order.***

```
# include<stdio.h>
# include<conio.h>
void main( )
{
        clrscr( ):
        int array[10];
        int i, j ;
        for (i=1; i<=10; i++)
        {
        printf(" Enter the data serially);
        scanf("%d",&array[i]);
        }
        int temp;
        for( i=1; i<=10; i++)
```

```
        {
            for (j=i+1; j<=10; j++)
            {
                if (array[i] < array[j])
                {
                    Temp=array[i];
                    array[i]=array[j];
                    array[j]= temp;
                }
            }
        }
        printf("Sorting data in descending order\n");
        for( i=1; i<=10; i++)
        {
            printf("%d\n",array[i]);
        }
}
```

Program: ***Write a program to store twelve numbers in double dimensional array and print out the values in table with row wise addition.***

```
# include<stdio.h>
# include<conio.h>
void main( )
{
    int j, k;
    int num[3][4]={1,2,3,4,8,10,11,5,9,6,2,8}
    clrscr();
    printf("Data stored \t\t sum\n");
    for (j=1; j<=3; j++)
    {
        int sum=0;
        for (k=1; k<=4; k++)
        {
            printf("%d \t",num[j][k]);
            sum=sum+num[j]+[k];
        }
            printf("=%d',sum);
            printf('\n");
        }
}
```

Program: WAP to enter elements for 2x2 matrix and display its transpose.

```
#include<stdio.h>
#include<conio.h>
void main()
{
	clrscr();
	int a[2][2];
	int i,j;
	printf("\n Enter number for 2x2 matrices");
	for(i=0;i<2;j++)
	{
	for(j=0;j<2;j++)
		{
		printf("Enter number");
		scanf("%d",&a[i][j]);
		}
	}
	printf("\n Transpose of Matrix is");
		for(i=0;i<2;j++)
			{
			for(j=0;j<2;j++)
				{
					printf("%d",a[i][j]);
				}
			printf("\n");
			}
		getch();
	}
```

Program: WAP to enter elements for 3x3 matrix and display its sum.

```
#include<stdio.h>
#include<conio.h>
void main()
{
	clrscr();
	int m1[3][3], m2[3][3],
	int i,j;
	printf("\n Enter number for 2x2 matrices");
	for(i=0;i<3;j++)
	{
	for(j=0;j<3;j++)
		{
		printf("Enter number for first matrix");
		scanf("%d",&m1[i][j]);
		}
	}
```

```
for(i=0;i<3;j++)
{
for(j=0;j<3;j++)
        {
        printf("Enter number for Second matrix");
        scanf("%d",&m2[i][j]);
        }
}

printf("\n Sum of Two Matrix is");
        for(i=0;i<3;j++)
                {
        for(j=0;j<3;j++)
                {
                        printf("%d",m1[i][j]+ m2[i][j]);
                }
                printf("\n");
        }
getch();
}
```

String Function:

The strings are manipulated by specific string function. These are inbuilt functions and defined within string.h header file.

1. strlen( ): It returns the number of character present in the string.
   Syntax:strlen(string);

Program: ***Write a program to store string in array variable and find the length.***

```
# include<stdio.h>
# include<conio.h>
# include<string.h>
void main( )
{
        int length;
        char n[ ]= "Everest";
        clrscr();
        length= strlen(n);
        printf("\n string=%s",n);
        printf("\n length=%d",length);
}
```

2. strrev( ): It helps to reverse the character of the string.
   Syntax:strrev(string);
3. strupr( ): It converts lowercase letters in string to uppercase.
   Syntax:strupr(string);
4. strlwr( ): It converts uppercase letters in string to lowercase.
   Syntax:strlwr(string);
5. strcpy( ): It is used to copy the content of one string to another.

Syntax:strcpy(target,source);

6. strcat( ): It is used to concatenate source string to the target string.
   Syntax:strcat(target,source); or strcat(source,target);
7. strcmp( ): It compares two strings on following basis.
   Syntax:strcmp(string1,string2);

Program: ***Write a program to show use of trcpy,strrev,strupr and strlwr.***

```
# include<stdio.h>
# include<conio.h>
# include<string.h>
void main( )
{
        char source[25];
        char target[25];
        printf("Enter the any word"):
        gets (source);
        strcpy(target,source);
        printf("Copied word is= %s \n",target);
        printf("Reverse word is=%s \n",strrev(source));
        printf("Word with capital letters= %s \n",strupr(source));
        printf("Word with small letters= %s \n",strlwr(source));
}
```

Program: ***Write a program to read two strings in array and concatenate string.***

```
# include<stdio.h>
# include<conio.h>
# include<string.h>
void main( )
{
        clrscr( );
        char source[10]= "Rajesh";
        char target[10]= "Hamal";
        strcat (source,target);
        printf("\n %s", source);
}
```

Program: ***Write a program to read two strings in array and compare two strings and check that string is palindrome or not.***

```
# include<stdio.h>
# include<conio.h>
# include<string.h>
void main( )
{
        clrscr( );
        int a;
        char string1[20];
        char string2[20];
```

```
        printf("Enter a word");
        scanf("%s",string1);
        strcpy(string2,string1);
        a=strcmp(string2, strrev(string2));
        if (a==0)
        {
                printf("palindrome");
        }
        else
        {
                printf("Not palindrome");
        }
}
```