

UNIT-1

Introduction to computers: Computer systems, computer Languages, computer number systems.

Introduction to C programming: Background and characteristics of C, Flow Charts, algorithms and pseudo code. Structure of a C Program, Input/output Statements in C, writing C programs, compiling and executing C programs.

S.NO	TOPIC	PAGE NUMBER
1.	Computer systems	
2.	computer number systems	
3.	Computer Languages	
4.	Background and characteristics of C	
5.	Algorithms & Flowchart	
6.	pseudo code	
7.	Structure of a C Program	
8.	Input/output Statements in C	
9.	Compiling and executing C programs.	

1. Computer systems:

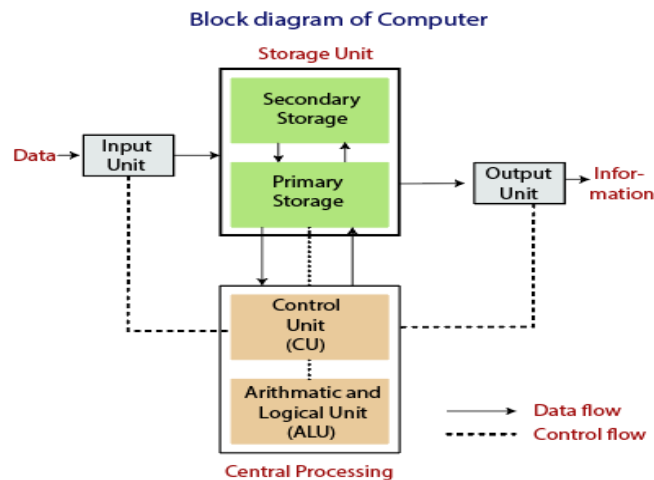
What is a Computer?

The computer is an electronic device which operates under the control of instructions stored in its memory. A computer can take data from the user through input devices (**Input**), process the user given data (**Processing**), produces the result to the user through output devices (**Output**) and stores data (**Information**) for future use.

A Computer can be defined as follows...

The Computer is an electronic device which operates under the control of instructions stored in its memory and it takes the data from the user, a process that data gives the result and stores the result for future use.

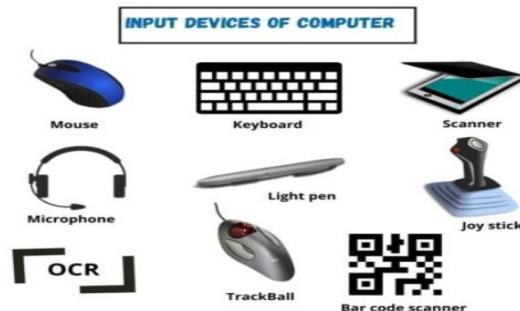
Block Diagram of a Computer:



Mainly the computer system consists of the following parts:

- Input devices
- Central Processing Unit (CPU)
- Storage unit/Memory unit
- Output device

Input devices: The input devices are used to give the data to the computer by user. Some of the input devices are:



Output devices: The output devices are used to display the result on computer to the user.



Central Processing Unit(CPU): The CPU is like the heart/brain of the computer. CPU is responsible for processing all the instructions which are given by the user to the computer system. The CPU is divided into two parts:

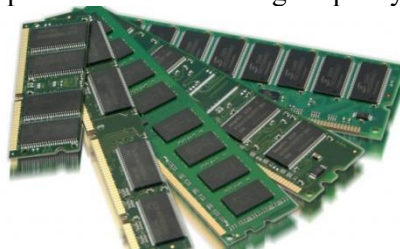
- **Control Unit(CU):** The control unit (CU) controls all the activities or operations which are performed inside the computer system. It receives instructions or information directly from the main memory of the computer.
- **Arithmetic Logic Unit (ALU):** This unit performs Arithmetic and Logical operations.

Memory /Storage Unit:

- Memory unit is used to store the large amount data with the help of “Primary Storage and Secondary Storage”.
 - **Primary Storage:** Primary memory also known as “main memory” which is located in the mother board of system. This is further divided into two types:
 - 1) RAM
 - 2) ROM

RAM (Read Only Memory):

- RAM stands for “Random Access Memory” which stores information temporarily i.e., it stores data which is currently used by the CPU.
- It is “Volatile Memory” because RAM needs power to work, when the computer power is turned off then all the data in it will be erased automatically.
- It allows both “read – write” operations and the storage capacity of RAM is 64MB to 16GB.



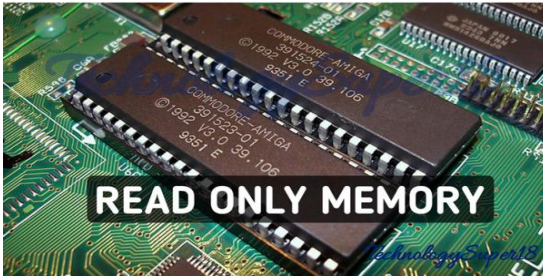
Types of RAM

- RAM can be classified into two types
 - **SRAM:** Static Random Access Memory
 - **DRAM:** Dynamic Random Access Memory.

Static RAM	Dynamic RAM
➤ SRAM uses transistor to store a single bit of data	➤ DRAM uses a separate capacitor to store each bit of data
➤ SRAM does not need periodic refreshment to maintain data	➤ DRAM needs periodic refreshment to maintain the charge in the capacitors for data
➤ SRAM’s structure is complex than DRAM	➤ DRAM’s structure is simplex than SRAM
➤ SRAM are expensive as compared to DRAM	➤ DRAM’s are less expensive as compared to SRAM
➤ SRAM are faster than DRAM	➤ DRAM’s are slower than SRAM
➤ SRAM are used in Cache memory	➤ DRAM are used in Main memory

ROM (Read Only Memory):

- ROM stands for “Read Only Memory” which only read the data and stores this data permanently.
- It is “Non Volatile” memory because the data cannot be erased even the computer power is turned off.
- ROM memory cannot be directly accessed by the CPU. The data is first transferred to RAM and then CPU is accessed from the RAM.



Types of ROM:

- ROM can be classified into three types:
 - **PROM: “Programmable Read Only Memory”.** It is modified only once by a user.
Example: CD - R
 - **EPROM: “Erasable and Programmable Read Only Memory”.** The content of this ROM can be erased by Ultra Violet rays and ROM can be reprogrammed.
Example: CD(R/W)
 - **EEPROM: “Electrically Erasable and Programmable Read Only Memory”.** It can be erased electrically and reprogrammed about 10 thousand times.
Example: pen-drive, memory card.

Differences of ROM:

Types of ROM		
<ul style="list-style-type: none">• PROM:<ul style="list-style-type: none">– Programmable ROM– User can store programs only once.– User can make micro code program can be made that are needed mostly.– The process of making program in PROM is called ‘Burning’.– Example: CD-R	<ul style="list-style-type: none">• EPROM:<ul style="list-style-type: none">– Erasable PROM– Information can be removed by ultra violet rays.– Information can be re-write after removing previous information.– It is cheaper than PROM because it is re-useable.– Example: CD(RW)	<ul style="list-style-type: none">• EEPROM:<ul style="list-style-type: none">◦ Electrically Erasable PROM◦ Information can be removed by electric signals.◦ It is the simplest way to store info in ROM.◦ Now it is used to store BIOS in Memory.◦ Example :Pen Drive

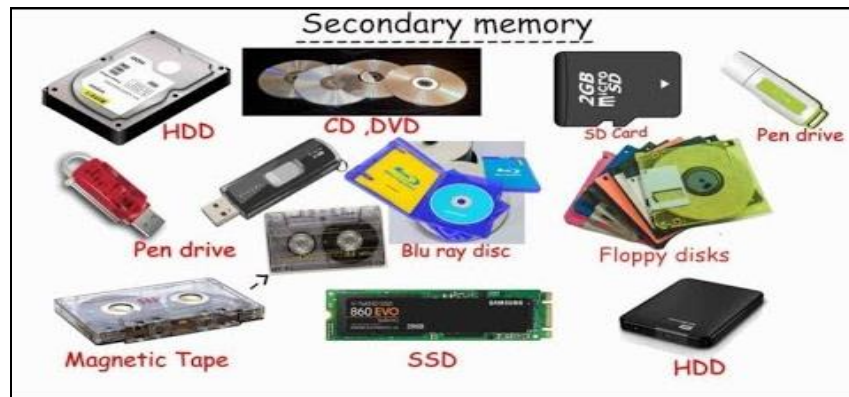
Cache Memory: is a semi-conductor (buffer)memory which lies between main memory and processor. It is mainly used for increasing the speed of RAM.

Difference b/w RAM & ROM:

BASIS FOR COMPARISON	RAM	ROM
Basic	It is a read-write memory.	It is read only memory.
Use	Used to store the data that has to be currently processed by CPU temporarily.	It stores the instructions required during bootstrap of the computer.
Volatility	It is a volatile memory.	It is a nonvolatile memory.
Stands for	Random Access Memory.	Read Only Memory.
Modification	Data in RAM can be modified.	Data in ROM can not be modified.
Capacity	RAM sizes from 64 MB to 4GB.	ROM is comparatively smaller than RAM.
Cost	RAM is a costlier memory.	ROM is comparatively cheaper than RAM.
Type	Types of RAM are static RAM and dynamic RAM.	Types of ROM are PROM, EPROM, EEPROM.

Secondary storage:

- **Secondary Storage:** The place where we store our personal data in computer system is known as “**Secondary Memory**”. We can easily retrieve when the data is needed. It is non volatile in nature so that we cannot lose the data when power supply is off.



There are two methods for accessing the data from it :-

- **Sequential** – This is the method in which we search the data sequentially or line by line until you find the desired data.
E.g., Magnetic tape, etc.
- **Direct** – This is the method in which computer can go directly to the information that the user wants.
E.g. Magnetic disk: hard disk, CD/DVD etc.

History of a computer:

1948 – 1954: First Generation – Vacuum Tubes

These early computers used vacuum tubes as circuitry and magnetic drums for memory. As a result they were enormous, literally taking up entire rooms and costing a fortune to run. These were inefficient materials which generated a lot of heat, sucked huge electricity and subsequently generated a lot of heat which caused ongoing breakdowns.

These first generation computers relied on 'machine language' (which is the most basic programming language that can be understood by computers). These computers were limited to solving one problem at a time. Input was based on punched cards and paper tape. Output came out on print-outs. The two notable machines of this era were the UNIVAC and ENIAC machines – the UNIVAC is the first every commercial computer which was purchased in 1951 by a business – the US Census Bureau.

There are 5 types of Generations.

1st Generation Computers

Year = 1948
Tech = Vacuum Tube Tech. (18000)
Input = Punched Cards
Output = Paper Tapes/Printing Papers
Memory = Magnetic Drums
No Keyboard, Mouse, Screen, CPU

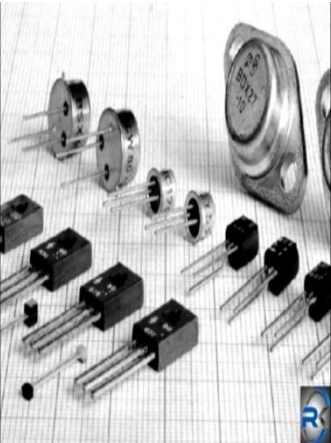



1954 – 1965: Second Generation – Transistors

The replacement of vacuum tubes by transistors saw the advent of the second generation of computing. Although first invented in 1947, transistors weren't used significantly in computers until the end of the 1950s. They were a big improvement over the vacuum tube, despite still subjecting computers to damaging levels of heat. However they were hugely superior to the vacuum tubes, making computers smaller, faster, cheaper and less heavy on electricity use. They still relied on punched card for input/printouts.

2nd Generation Computers

Year = 1954
Tech = Transistors
Input = Punched Cards
Output = Paper Tapes/Printing Papers
Memory = Magnetic Drums
No Keyboard, Mouse, Screen, CPU



1965 – 1975: Third Generation – Integrated Circuits

By this phase, transistors were now being miniaturised and put on silicon chips (called semiconductors). This led to a massive increase in speed and efficiency of these machines. These were the first computers where users interacted using keyboards and monitors which interfaced with an operating system, a significant leap up from the punch cards and printouts. This enabled these machines to run several applications at once using a central program which functioned to monitor memory.

As a result of these advances which again made machines cheaper and smaller, a new mass market of users emerged during the '60s.

3rd Generation Computers

Year = 1965
Tech = Integrated Circuits (Ics)

Keyboard
Mouse
Screen
CPU

are Introduced
in 3rd Generation.




1977 – 1989: Fourth Generation – Microprocessors

This revolution can be summed in one word: Intel. The chip-maker developed the Intel 4004 chip in 1971, which positioned all computer components (CPU, memory, input/output controls) onto a single chip. What filled a room in the 1940s now fit in the palm of the hand. The Intel chip housed thousands of integrated circuits.

The increased power of these small computers meant they could be linked, creating networks. This ultimately led to the development, birth and rapid evolution of the Internet. Other major advances during this period have been the Graphical user interface (GUI), the mouse and more recently the astounding advances in lap-top capability and hand-held devices.

4th Generation Computers

Year = 1975
Tech = Microprocessor

Ex :-
Desktops
Laptops
Palmtops






1989- : Fifth Generation – Artificial Intelligence:

Computer devices with artificial intelligence are still in development, but some of these technologies are beginning to emerge and be used such as voice recognition. AI is a reality made possible by using parallel processing and superconductors. The essence of fifth generation will be using these technologies to ultimately create machines which can process and respond to natural language, and have capability to learn and organise themselves.

5th Generation Computers

Year = Since 1980's
Tech = Artificial Intelligence

Ex :- Robots



Characteristics of computer:

Characteristic	Description
Speed	<ul style="list-style-type: none">Computers can perform millions of operations in single second.
Accuracy	<ul style="list-style-type: none">It never mistakes, It gives accurate result.
Automatic	<ul style="list-style-type: none">They can perform operations without user intervention
Diligence	<ul style="list-style-type: none">Computers can never get tired like humans, they can work for hours without errors.
Versatility	<ul style="list-style-type: none">Means flexible, used if different areas, ex: business, railway, weather.
Memory	<ul style="list-style-type: none">Computers stores a large amount of data and programs in the secondary storage space Ex: Hard disk, CD, DVD, pen drives etc.
Economical	<ul style="list-style-type: none">Using computers we can reduce manpower.

Classification of computers:

TYPES	DESCRIPTION
Super Computers	It is the fastest, most powerful , expansive, and are employed for specialized applications that require immense amounts of mathematical calculations. It is mainly focusing on computational speed. Example: PARAM, Roadrunner. Applications: weather forecasting, nuclear energy research etc..
Mainframe Computers	A very large and expensive computer capable of supporting hundreds, or even thousands, of users simultaneously. It is mainly used to maintain Data Base servers. Example: IBM z Series
Mini computers	It is a multi-user computer system, capable of supporting hundreds of users simultaneously. Mostly used for network purposes.
Micro computers	It is a single user computer system having a moderately powerful microprocessor. It is termed as a computer that is equipped microprocessor as its CPU. Examples: desktop computers, laptops, palmtop computers.

Applications of computers:

Applications	Description
Communication	Internet connects computers all over the world, using electronic mail we can communicate in seconds with a person who is very long distance
Government	Used in keeping records on legal action, revenue records etc..
Business	Used in retail shops to place order, calculate cost, printing receipt & to keep inventory..
Geology	Civil Engineers used to evaluate the earthquake affect on the structure of buildings base on age, soil type, construction material.
Weather	To forecast data about air pressure, temperature, humidity and other values
Online Banking	Cashless society can be made by computers using credit card and debit card.

2. Computer number systems:

Number systems are the technique to represent numbers in the computer system, every value that you are saving or getting into/from computer memory has a defined number system.

Computer supports following number systems:

- Binary number system
- Octal number system
- Decimal number system
- Hexadecimal (hex) number system

Binary Number System:

A Binary number system has only two digits that are 0 and 1. Every number (value) represents with 0 and 1 in this number system. The base of binary number system is 2, because it has only two digits.

Decimal Number System:

Decimal number system has only ten (10) digits from 0 to 9. Every number (value) represents with 0,1,2,3,4,5,6,7,8 and 9 in this number system. The base of decimal number system is 10, because it has only 10 digits.

Octal Number System:

Octal number system has only eight (8) digits from 0 to 7. Every number (value) represents with 0,1,2,3,4,5,6 and 7 in this number system. The base of octal number system is 8, because it has only 8 digits.

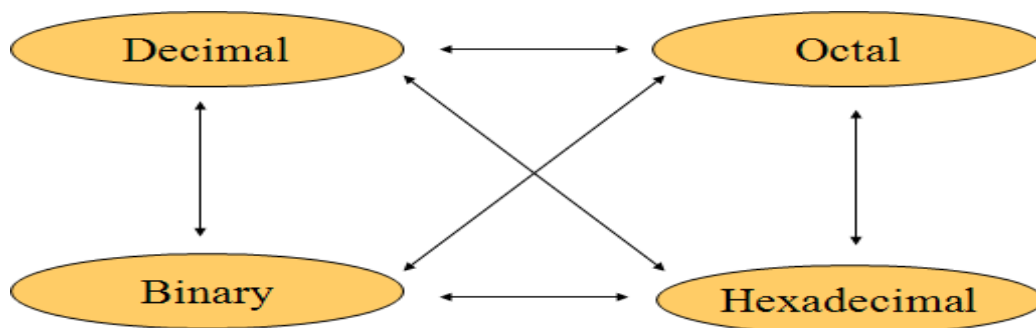
Hexadecimal Number System:

A Hexadecimal number system has sixteen (16) alphanumeric values from 0 to 9 and A to

F. Every number represents with 0,1,2,3,4,5,6, 7,8,9,A,B,C,D,E and F in this number system. The base of hexadecimal number system is 16. Here A is 10, B is 11, C is 12, D is 13, E is 14 and F is 15.

Number system	Base	Used digits	Example	C Language assignment
Binary	2	0,1	$(11110000)_2$	<code>int val=0b11110000;</code>
Octal	8	0,1,2,3,4,5,6,7	$(360)_8$	<code>int val=0360;</code>
Decimal	10	0,1,2,3,4,5,6,7,8,9	$(240)_{10}$	<code>int val=240;</code>
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F	$(F0)_{16}$	<code>int val=0xF0;</code>

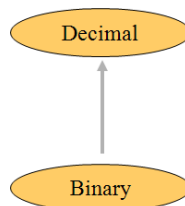
Number System Conversions:



Decimal Conversions:

1. Binary to Decimal
2. Octal to Decimal
3. Hexadecimal to Decimal

Binary to Decimal Conversion:



Technique

- Multiply each bit by 2^n , where n is the “weight” of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results

Example1:

Bit "0"

$$\begin{array}{r} 101011_2 \Rightarrow \\ 1 \times 2^0 = 1 \\ 1 \times 2^1 = 2 \\ 0 \times 2^2 = 0 \\ 1 \times 2^3 = 8 \\ 0 \times 2^4 = 0 \\ 1 \times 2^5 = 32 \\ \hline 43_{10} \end{array}$$

Example2: $(110.101)_2$

Step 1: Conversion of 110 to decimal

$$\Rightarrow 110_2 = (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$

$$\Rightarrow 110_2 = 4 + 2 + 0$$

$$\Rightarrow 110_2 = 6$$

So equivalent decimal of binary integral is 6.

Step 2: Conversion of .101 to decimal

$$\Rightarrow 0.101_2 = (1 \times 1/2) + (0 \times 1/2^2) + (1 \times 1/2^3)$$

$$\Rightarrow 0.101_2 = 1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125$$

$$\Rightarrow 0.101_2 = 0.625$$

So equivalent decimal of binary fractional is 0.625

Step 3: Add result of step 1 and 2.

$$\Rightarrow 6 + 0.625 = 6.625$$

Octal To Decimal Conversion:



Technique

- Multiply each bit by 8^n , where n is the "weight" of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results

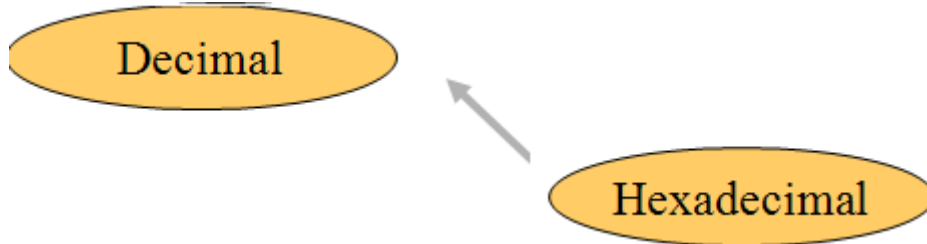
Example1:

$$\begin{array}{r} 724_8 \Rightarrow \\ 4 \times 8^0 = 4 \\ 2 \times 8^1 = 16 \\ 7 \times 8^2 = 448 \\ \hline 468_{10} \end{array}$$

Example2: (67.12172)₈

$$\begin{aligned}
&= 6 \times 8^1 + 7 \times 8^0 + 1 \times 8^{-1} + 2 \times 8^{-2} + 1 \times 8^{-3} + 7 \times 8^{-4} + 2 \times 8^{-5} \\
&= 48 + 7 + 0.125 + 0.03125 + 0.001953125 + 0.001708984375 + 0.00006103515624 \\
&= 58.1599...
\end{aligned}$$

Hexadecimal to decimal:



Technique

- Multiply each bit by 16^n , where n is the “weight” of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results

Example1:

$$\begin{array}{r}
ABC_{16} \Rightarrow \quad C \times 16^0 = 12 \times 1 = 12 \\
\quad \quad \quad B \times 16^1 = 11 \times 16 = 176 \\
\quad \quad \quad A \times 16^2 = 10 \times 256 = 2560 \\
\hline
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad 2748_{10}
\end{array}$$

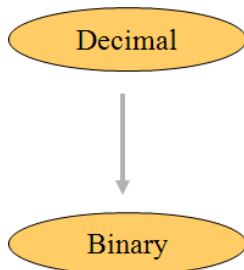
Example2: (5A.28F56)₁₆

$$\begin{aligned}
&= 5 \times 16^1 + A \times 16^0 + 2 \times 16^{-1} + 8 \times 16^{-2} + F \times 16^{-3} + 5 \times 16^{-4} + C \times 16^{-5} \\
&= 5 \times 16^1 + 10 \times 16^0 + 2 \times 16^{-1} + 8 \times 16^{-2} + 15 \times 16^{-3} + 5 \times 16^{-4} + 12 \times 16^{-5} \\
&= 10 + 0.125 + 0.03125 + 0.003662109375 + 0.0000762939453 + 0.0000114440918 \\
&= 10.1599...
\end{aligned}$$

Binary Conversions:

1. Decimal To Binary
2. Octal to Binary
3. Hexadecimal to Binary

Decimal to Binary Conversions:



Technique

- Divide by two, keep track of the remainder
- First remainder is bit 0 (LSB, least-significant bit)

- Second remainder is bit 1
- Etc.

$$125_{10} = ?_2$$

2	125	
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
	0	1

$$125_{10} = 1111101_2$$

Example2: (4.47)₁₀

Step 1: Conversion of 4 to binary

1. 4/2 : Remainder = 0 : Quotient = 2
2. 2/2 : Remainder = 0 : Quotient = 1
3. 1/2 : Remainder = 1 : Quotient = 0

So equivalent binary of integral part of decimal is 100.

Step 2: Conversion of .47 to binary

1. 0.47 * 2 = 0.94, Integral part: 0
2. 0.94 * 2 = 1.88, Integral part: 1
3. 0.88 * 2 = 1.76, Integral part: 1

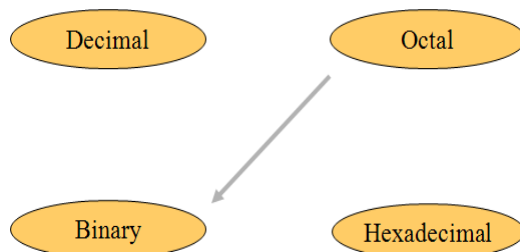
So equivalent binary of fractional part of decimal is .011

Step 3: Combined the result of step 1 and 2.

Final answer can be written as:

$$100 + .011 = 100.011$$

Octal to Binary conversions:



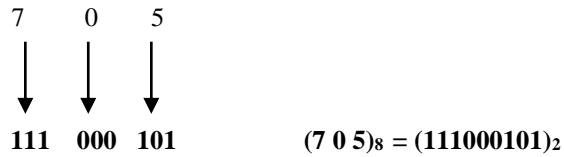
OCTAL	BINARY
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Technique

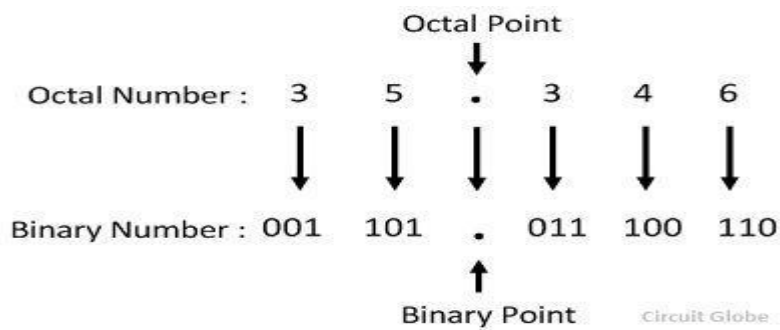
- Convert each octal digit to a 3-bit equivalent binary representation

Example:

$$(705)_8 = ?_2$$

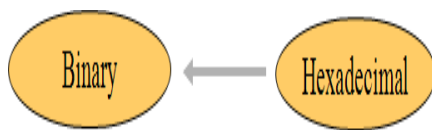


Example2: (35.346)₈



Thus the required binary number is 011 101. 011 100 110

Hexadecimal to Binary:



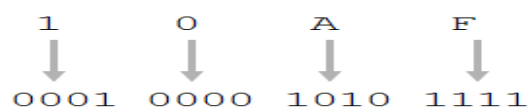
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Technique

- Convert each hexadecimal digit to a 4-bit equivalent binary representation

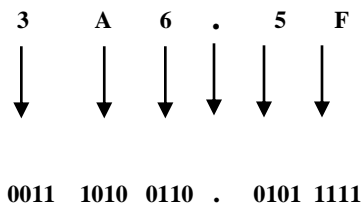
Example:

$$10AF_{16} = ?_2$$



$$10AF_{16} = 0001000010101111_2$$

Example2: (3A6.5F)₁₆



$$(3A6.5F)_{16} = (00111010\ 0110 . 0101\ 1111)_2$$

Octal conversions:

1. Decimal to Octal
2. Hexadecimal to octal
3. Binary to Octal

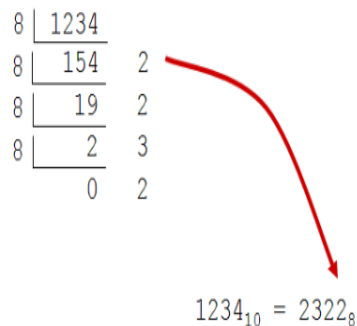
Decimal to Octal Conversions:



Technique 1

- Divide by 8 Keep track of the remainder

$$1234_{10} = ?_8$$



Example2: (1234.16)₁₀

$$(1234)_{10} = (2322)_8$$

$$(0.16)_{10} =$$

Step 1

We multiply 0.16 by 8 and take the integer part

$$0.16 \times 8 = 1.28$$

Integer part = 1

Fractional part = 0.28

As, fractional part is not equal to 0 so we copy it to next step.

Step 2

We multiply 0.28 by 8 and take the integer part

$$0.28 \times 8 = 2.24$$

Integer part = 2

Fractional part = 0.24

As, fractional part is not equal to 0 so we copy it to next step.

Step 3

We multiply 0.24 by 8 and take the integer part

$$0.24 \times 8 = 1.92$$

Integer part = 1

Fractional part = 0.92

As, fractional part is not equal to 0 so we copy it to next step.

Step 4

We multiply 0.92 by 8 and take the integer part

$$0.92 \times 8 = 7.36$$

Integer part = 7

Fractional part = 0.36

As, fractional part is not equal to 0 so we copy it to next step.

Step 5

We multiply 0.36 by 8 and take the integer part

$$0.36 \times 8 = 2.88$$

Integer part = 2

Fractional part = 0.88

As, fractional part is not equal to 0 so we copy it to next step.

Step 6

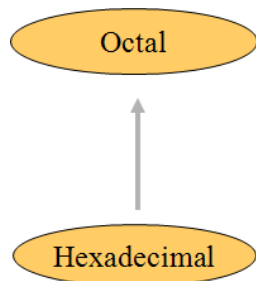
We multiply 0.88 by 8 and take the integer part

0.88 x 8

in this case, we have 5 digits as answer and the fractional part is still not 0, we stop here.

$$0.16_{(10)} = 0.12172$$

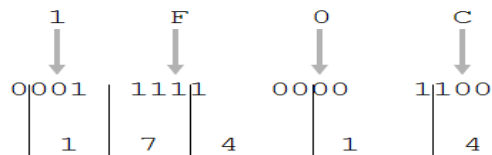
Hexadecimal to Octal conversions:



Technique

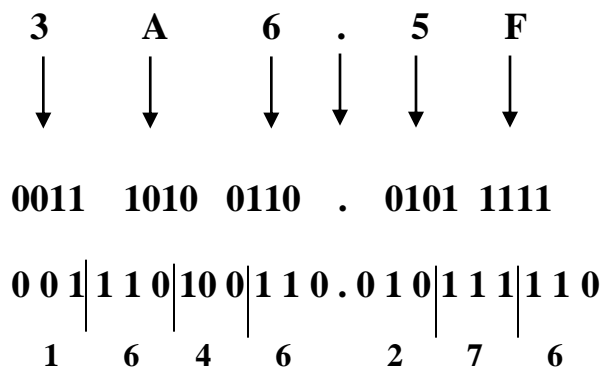
- Step:1 convert hexadecimal value into binary
- Step2: take 3 digits from right to left

$$1FOC_{16} = ?_8$$



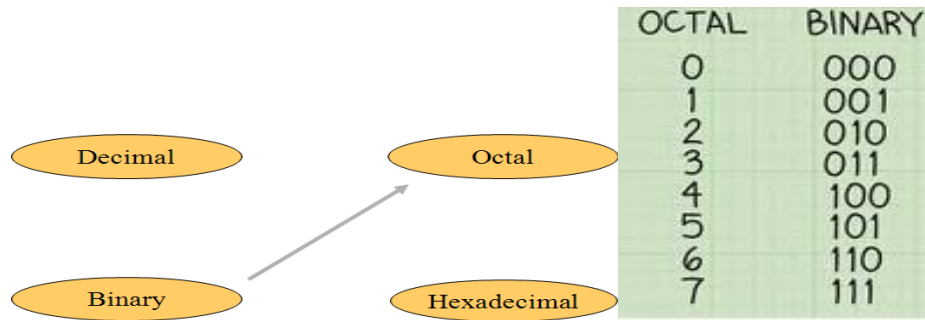
$$1FOC_{16} = 17414_8$$

Example2: (3A6.5F)₁₆



$$(3A6.5F)_{16} = (1646.276)_8$$

Binary to Octal Conversion:



Technique

- Group bits in threes, starting on right
- Convert to octal digits

$$1011010111_2 = ?_8$$

$$\begin{array}{cccc}
 1 & 011 & 010 & 111 \\
 \downarrow & \downarrow & \downarrow & \downarrow \\
 1 & 3 & 2 & 7
 \end{array}$$

$$1011010111_2 = 1327_8$$

Example2: $(11010011.011001)_2$

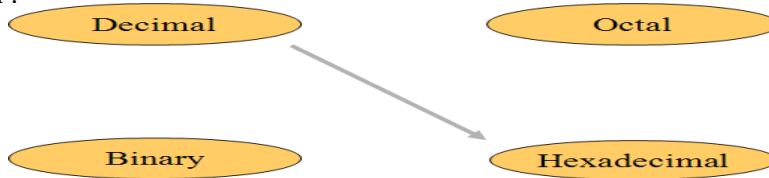
$$\begin{array}{cccccccccccc}
 1 & 1 & | & 0 & 1 & 0 & | & 0 & 1 & 1 & . & 0 & 1 & 1 & | & 0 & 0 & 1 & . \\
 & & & & & & & & & & & & & & & & & & & \\
 3 & & & & 2 & & & & 3 & & . & 3 & & & & & & & 1
 \end{array}$$

$$(11010011.011001)_2 = (323.31)_8$$

Hexadecimal Conversions:

1. Decimal to Hexadecimal
2. Binary to Hexadecimal
3. Octal to Hexadecimal

Decimal to Hexadecimal :



Technique

- Divide by 16
- Keep track of the remainder

Example:

$$1234_{10} = ?_{16}$$

16	1234	
16	77	2
16	4	13 = D
	0	4

$1234_{10} = 4D2_{16}$

Example2: $(1234.00390625)_{10}$

$$(1234)_{10} = 4D2$$

$(0.00390625)_{10} =$

Hexadecimal of 0.00390625

Step 1

 We multiply 0.00390625 by 16 and take the integer part
 $0.00390625 \times 16 = 0.0625$
 Integer part = 0
 Fractional part = 0.0625

As, fractional part is not equal to 0 so we copy it to next step.

Step 2

 We multiply 0.0625 by 16 and take the integer part

$$0.0625 \times 16 = 1.000$$

Integer part = 1

Fractional part = 0

Now the fractional part is 0 so, we stop here.

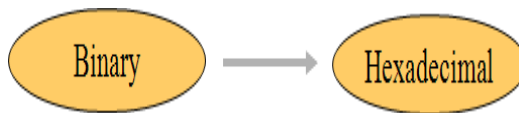
$$(0.00390625)_{10} = (0.01)_{16}$$

$$(1234.00390625)_{10} = (4D2.01)_{16}$$

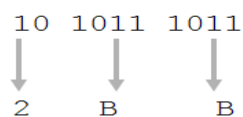
Binary to Hexadecimal:

Technique

- Group bits in fours, starting on right
- Convert to hexadecimal digits

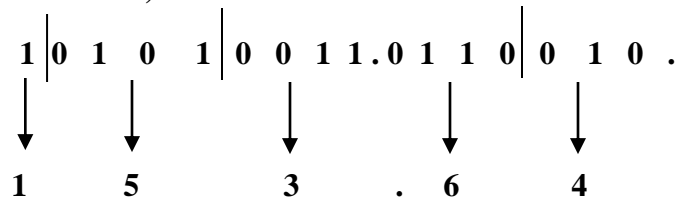


$$1010111011_2 = ?_{16}$$



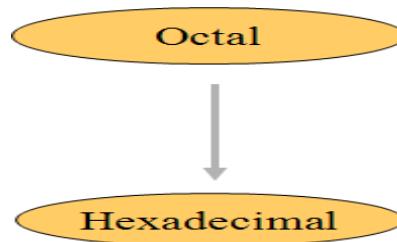
$$1010111011_2 = 2BB_{16}$$

Example2: (101010011.011001)₂



(101010011.011001)₂ = (153.64)₈

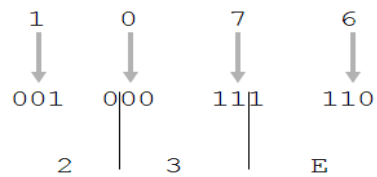
Octal to Hexadecimal:



Technique

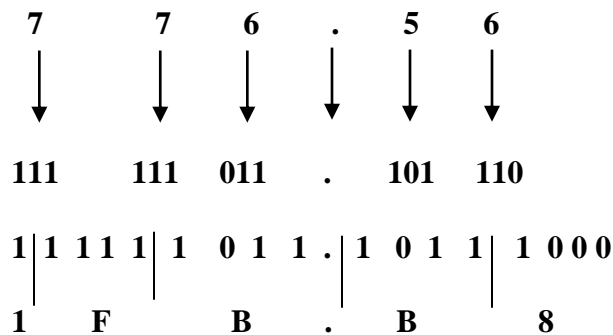
- Step 1: convert octal value into binary
- Step 2: take 4 digits from right to left

$1076_8 = ?_{16}$



$1076_8 = 23E_{16}$

Example2: (776.56)₈



(776.56)₈ = (1FB.B8)₁₆

3. Computer languages:

What is Computer Language?

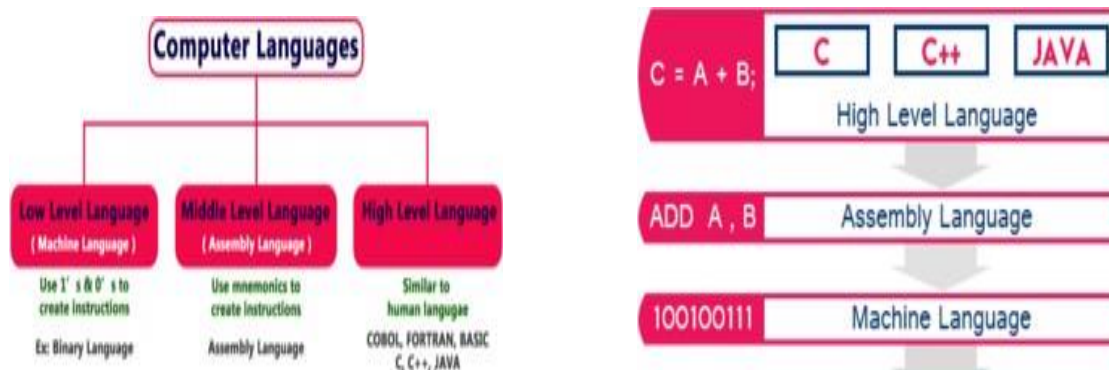
Generally, we use languages like English, Hindi, etc., to make communication between two persons. That means when we want to make communication between two persons we need a language through which persons can express their feelings. Similarly, when we want to make communication between user and computer or between two or more computers we need a language through which user can give information to the computer and vice versa. When a user wants to give any instruction to the computer the user needs a specific language and that language is known as a computer language.

The user interacts with the computer using programs and that programs are created using computer programming languages like C, C++, Java, etc.,

Definition:

Computer languages are the languages through which the user can communicate with the computer by writing program instructions.

Classification of Computer Languages:



Low-Level Language (Machine Language):

Low-Level language is the only language which can be understood by the computer. Binary Language is an example of a low-level language. Low-level language is also known as Machine Language. The binary language contains only two symbols 1 & 0. All the instructions of binary language are written in the form of binary numbers 1's & 0's. A computer can directly understand the binary language. Machine language is also known as the Machine Code. As the CPU directly understands the binary language instructions, it does not require any translator. CPU directly starts executing the binary language instructions and takes very less time

to execute the instructions as it does not require any translation. Low-level language is considered as the First Generation Language (1GL).

Advantages:

- A computer can easily understand the low-level language.
- Low-level language instructions are executed directly without any translation.
- Low-level language instructions require very less time for their execution.

Disadvantages:

- Low-level language instructions are very difficult to use and understand.
- Low-level language instructions are machine-dependent, that means a program written for a particular machine does not execute on another machine.
- In low-level language, there is more chance for errors and it is very difficult to find errors, debug and modify.

Middle-Level Language (Assembly Language):

Middle-level language is a computer language in which the instructions are created using symbols such as letters, digits and special characters. Assembly language is an example of middle-level language. In assembly language, we use predefined words called mnemonics. Binary code instructions in low-level language are replaced with mnemonics and operands in middle-level language. But the computer cannot understand mnemonics, so we use a translator called Assembler to translate mnemonics into binary language. Assembler is a translator which takes assembly code as input and produces machine code as output. That means, the computer cannot understand middle-level language, so it needs to be translated into a low-level language to make it understandable by the computer. Assembler is used to translate middle-level language into low-level language.

Advantages:

- Writing instructions in a middle-level language is easier than writing instructions in a low-level language.
- Middle-level language is more readable compared to low-level language.
- Easy to understand, find errors and modify.

Disadvantages:

- Middle-level language is specific to a particular machine architecture, that means it is machine-dependent.
- Middle-level language needs to be translated into low-level language.
- Middle-level language executes slower compared to low-level language.

High-Level Language:

A high-level language is a computer language which can be understood by the users. The high-level language is very similar to human languages and has a set of grammar rules that are used to make instructions more easily. Every high-level language has a set of predefined words known as Keywords and a set of rules known as Syntax to create instructions. The high-level language is easier to understand for the users but the computer can not understand it. High-level language needs to be converted into the low-level language to make it understandable by the computer. We use Compiler or interpreter to convert high-level language to low-level language.

Languages like COBOL, FORTRAN, BASIC, C, C++, JAVA, etc., are examples of high-level languages. All these programming languages use human-understandable language like English to write program instructions. These instructions are converted to low-level language by the compiler so that it can be understood by the computer.

Advantages:

- Writing instructions in a high-level language is easier.
- A high-level language is more readable and understandable.
- The programs created using high-level language runs on different machines with little change or no change.
- Easy to understand, create programs, find errors and modify.

Disadvantages:

- High-level language needs to be translated into low-level language.
- High-level language executes slower compared to middle and low-level languages.
- Understanding Computer Languages
- The following figure provides a few key points related to computer languages.

4. Background and Characteristics of C

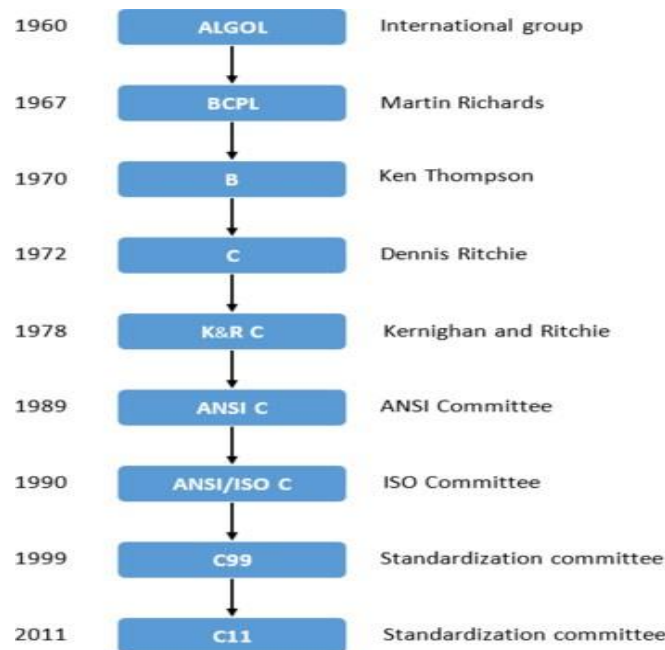
C is a structured programming language. It is also known as function orientated programming language. C programming language was developed in the year of 1972 by Dennis Ritchie at Bell Laboratories in the USA (AT & T). In the year of 1968, research was started by Dennis Ritchie on programming languages like BCPL,CPL. The main aim of his research was to develop a new language to create an OS called UNIX. After four years of research, a new programming language was created with solutions for drawbacks in languages like BCPL & CPL. In the year of 1972, the new language was introduced with the name “Traditional C”.



The name 'c' was selected from the sequence of previous language 'B' (BCPL) because most of the features of 'c' were derived from BCPL (B language).

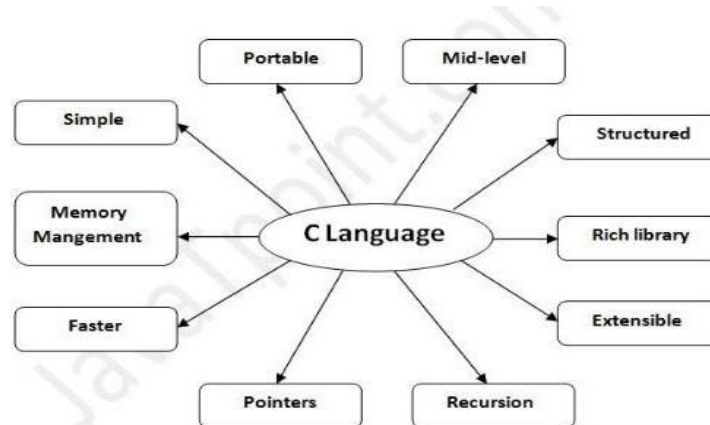
The first outcome of the c language was the UNIX operating system. The initial UNIX OS was completely developed using 'c' programming language.

The founder of the 'C' language, Dennis Ritchie is known as “Father of C” and also “Father of UNIX”.



Timeline History of C language

Characteristics of C:

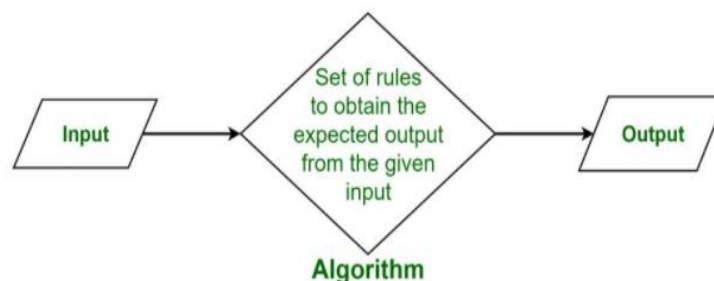


1. **Simple:** C is a simple language in the sense that it provides a **structured approach** (to break the problem into parts), **the rich set of library functions, data types**, etc.
2. **Portable:** Unlike assembly language, c programs **can be executed on different machines** with some machine specific changes. Therefore, C is a machine independent language.
3. **Mid-level programming language:** Although, C is intended to do low-level programming. It is used to develop system applications such as kernel, driver, etc. It also supports the features of a high-level language. That is why it is known as mid-level language.
4. **Structure programming language:** C is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify. Functions also provide code reusability.
5. **Rich libraries:** C provides a lot of inbuilt functions that make the development fast.
6. **Memory management:** It supports the feature of **dynamic memory allocation**. In C language, we can free the allocated memory at any time by calling the **free()** function.
7. **Speed:** The compilation and execution time of C language is fast since there are lesser inbuilt functions and hence the lesser overhead.
8. **Pointer:** C provides the feature of pointers. We can directly interact with the memory by using the pointers. We **can use pointers for memory, structures, functions, array**, etc.
9. **Recursion:** In C, we **can call the function within the function**. It provides code reusability for every function. Recursion enables us to use the approach of backtracking.
10. **Extensible:** C language is extensible because it **can easily adopt new features**.

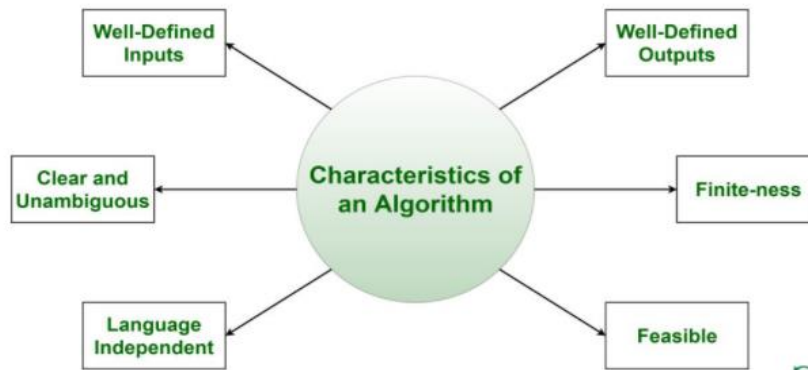
5. Algorithm and Flowchart

What is an Algorithm?

An algorithm is a step-by-step procedure to solve a particular problem. They form the foundation of writing a program.



Characteristics of algorithm:









- **Clear and Unambiguous:** Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-Defined Inputs:** If an algorithm says to take inputs, it should be well-defined inputs.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well.
- **Finite-ness:** It should not end up in an infinite loops or similar.
- **Feasible:** The algorithm must be simple, generic and practical, such that it can be executed upon will the available resources.
- **Language Independent:** It must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

Flowchart:

Flowchart is a diagrammatic representation of sequence of logical steps of a program. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

Flowchart Symbols:

Symbol	Name	Description
	Rectangular or action	A process or action such as calculation, input/output, assignment etc.
	Oval	Represents a complete algorithm Begin/Start, End/Stop.
	Diamond or decision	Which indicates that a decision to be made such as choice between YES or NO.
	Flowlines	Indicates the order in which the actions are to be performed.
	Small circle or connector symbol	When describing a portion of a complete algorithm continued from or will continue on.
	Input or output	The data or input/output.

Example:

Write algorithm and flowchart to calculate average of 3 numbers

Algorithm:

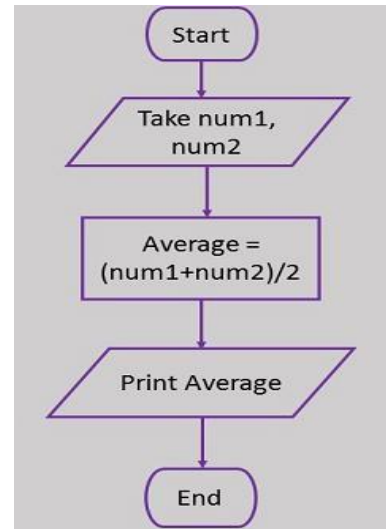
Step 1: start

Step 2: Read num1, num2, num3, average.

Step 3: $\text{average} = (\text{num1} + \text{num2} + \text{num3}) / 3$

Step 4: Print average

Step 5: Stop



Flowchart

Pseudo code:

Pseudo code is a term which is often used in programming and algorithm based fields. It is a methodology that allows the programmer to represent the implementation of an algorithm.

Example:

Adding two numbers

Pseudocode:

BEGIN

INPUT the first number

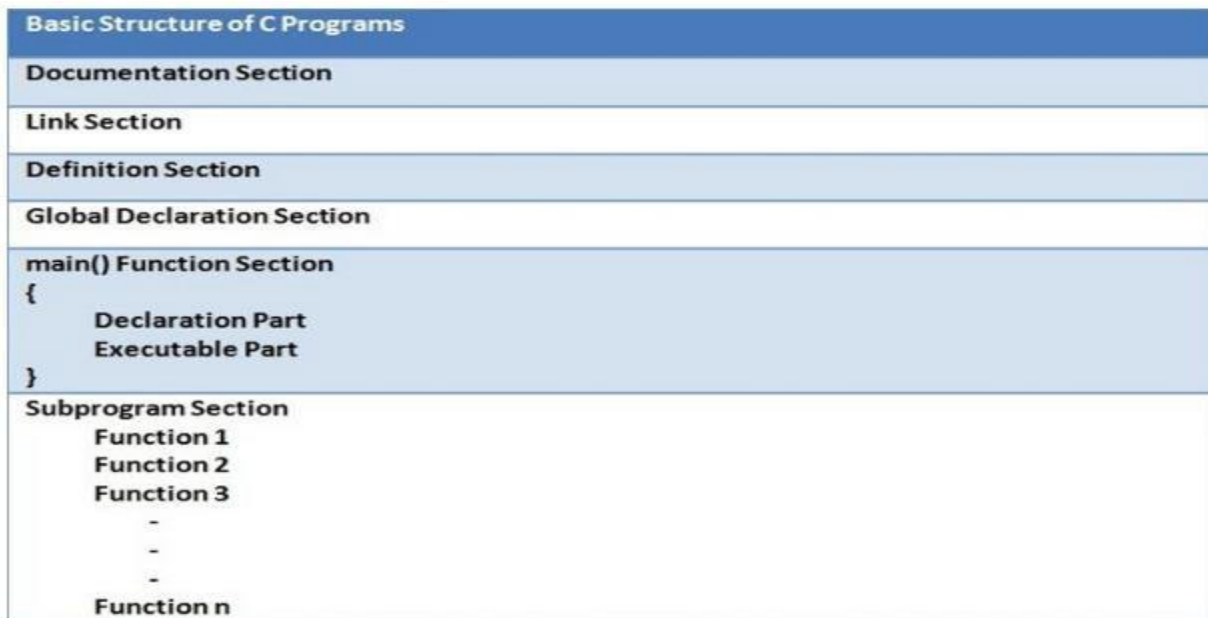
INPUT the second number

Add the two numbers

OUTPUT the sum

END

6. Structure of C:



Documentation Section

This section consists of comment lines which include the name of programmer, the author and other details like time and date of writing the program. Documentation section helps anyone to get an overview of the program.

Link Section

The link section consists of the header files of the functions that are used in the program. It provides instructions to the compiler to link functions from the system library.

Definition Section

All the symbolic constants are written in definition section. Macros are known as symbolic constants.

Global Declaration Section

The global variables that can be used anywhere in the program are declared in global declaration section. This section also declares the user defined functions.

main() Function Section

It is necessary have one main() function section in every C program. This section contains two parts, declaration and executable part. The declaration part declares all the variables that are used in executable part. These two parts must be written in between the opening and closing braces. Each statement in the declaration and executable part must end with a semicolon (;). The execution of program starts at opening braces and ends at closing braces.

Subprogram Section/user defined section

The subprogram section contains all the user defined functions that are used to perform a specific task. These user defined functions are called in the main() function.

7. Input/output Statements in C:

C Input Functions:

C programming language provides built-in functions to perform input operations. The input operations are used to read user values (input) from the keyboard. The c programming language provides the following built-in input functions.

1. scanf()
2. getch()
3. getchar()
4. gets()

Syntax:

```
scanf("format strings",&variableNames);
```

Example Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i;
    printf("\nEnter any integer value: ");
    scanf("%d",&i);
    printf("\nYou have entered %d number",i);
}
```

Output Function:

Programming language provides built-in functions to perform output operation. The output operations are used to display data on user screen (output screen) or printer or any file. The c programming language provides the following built-in output functions...

1. printf()
2. putchar()
3. puts()

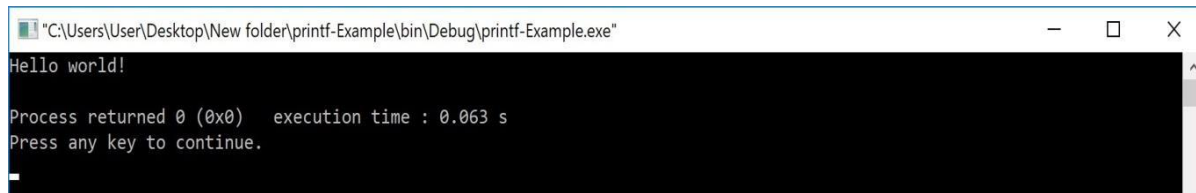
Syntax:

```
printf("message to be display!!!");
```

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    printf("Hello! Welcome to B.Tech smart class!!!");
}
```

Output:



```
"C:\Users\User\Desktop\New folder\printf-Example\bin\Debug\printf-Example.exe"
Hello world!
Process returned 0 (0x0) execution time : 0.063 s
Press any key to continue.
```

In the above example program, we used the printf() function to print a string on to the output screen.

The printf() function is also used to display data values. When we want to display data values we use format string of the data value to be displayed.

Syntax:

```
printf("format string",variableName);
```

Example Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i = 10;
    float x = 5.5;
    printf("%d %f",i, x);
}
```

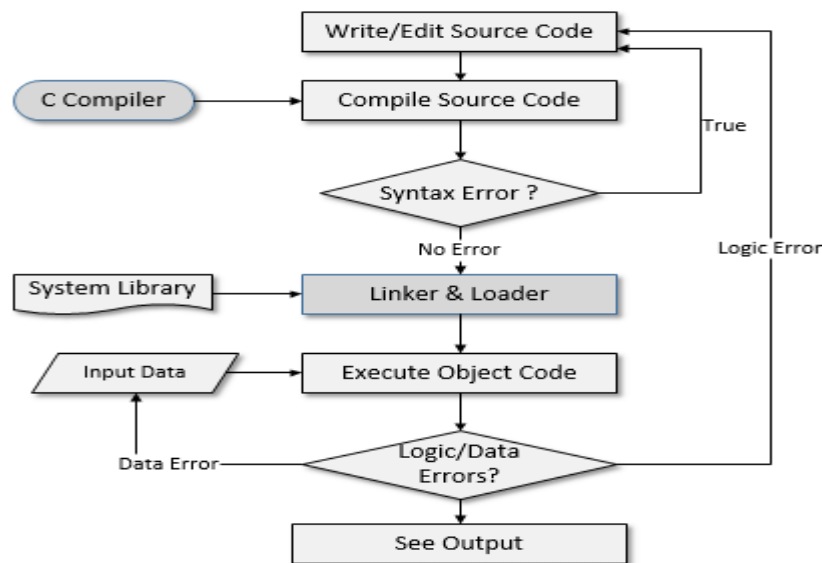
Output:

```
"C:\Users\User\Desktop\New folder\printf-Example\bin\Debug\printf-Example.exe"
```

```
Integer value = 10, float value = 5.500000  
Process returned 0 (0x0)   execution time : 0.047 s  
Press any key to continue.
```

8. Compiling and executing C programs:

- The compilation is a process of converting the source code into machine code. It is done with the help of the compiler.
- The compiler checks the source code for the syntactical or structural errors, and if the source code is error-free, then it generates the object code.



- **Pre-processor:** The source code is the code which is written in a text editor and the source code file is given an extension ".c". This source code is first passed to the pre-processor, and then the pre-processor expands this code.
- **Compiler:** The compiler converts this code into assembly code. Or we can say that the C compiler converts the pre-processed code into assembly code
- **Assembler:** Assembler is used to convert assembly code to machine code. The name of the object file generated by the assembler is the same as the source file. The extension of the object file in DOS is '.obj,' and in UNIX, the extension is '.o'. If the name of the source file is '**hello.c**', then the name of the object file would be 'hello.obj'.
- **Linker:** The main working of the linker is to link the object code of our program with the object code of the library files and other files. The output of the linker is the executable file. The extension of the executable file is '.exe'.

Difference b/w Compiler vs Interpreter:

COMPARISON	COMPILER	INTERPRETER
Input	It takes an entire program at a time.	It takes a single line of code or instruction at a time.
Output	It generates intermediate object code.	It does not produce any intermediate object code.
Working mechanism	The compilation is done before execution.	Compilation and execution take place simultaneously.
Speed	Comparatively faster	Slower
Memory	Memory requirement is more due to the creation of object code.	It requires less memory as it does not create intermediate object code.
Errors	Display all errors after compilation, all at the same time.	Displays error of each line one by one.
Error detection	Difficult	Easier comparatively
Pertaining Programming languages	C, C++, C#, Scala, typescript uses compiler.	PHP, Perl, Python, Ruby uses an interpreter.

Assembler:

An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.

Assemblers are similar to compilers in that they produce executable code. However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code. Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code. Compilers, on the other hand, must convert generic high-level source code into machine code for a specific processor.