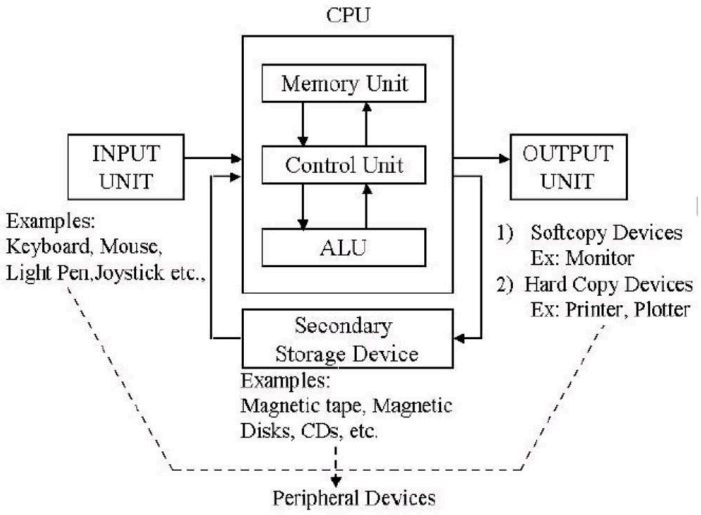
UNIT-1

Introduction of Computer-

Computer System is an electronic data processing device that does the following:

- Accept and store input data.
- Process the data input.
- And output the processed data in the required format.

Block Diagram of a Digital Computer



Input Unit

This unit contains devices with the help of which we enter data into computer.This unit makes link between user and computer. The input devices translate the human being information into the form understandable by computer.

CPU (Central Processing Unit)

CPU is considered as the brain of the computer. CPU perform all types of data processing operations.It stores data , intermediate results and instructions(program).It controls the operation of all parts of computer.

CPU itself has following three components

- ALU(Arithmetic Logic Unit)
- Memory Unit
- Control Unit

Input Unit

This unit assists for reading data from the user's end. Following are few of the important input devices which are used in Computer Systems

- Keyboard
- Mouse
- Joy Stick
- Light pen
- Track Ball
- Scanner
- Graphic Tablet
- Microphone
- Magnetic Ink Card Reader(MICR)
- Optical Character Reader(OCR)
- Bar Code Reader
- Optical Mark Reader

Output Unit

Output unit consists of devices with the help of which we get the information from computer. This unit is a link between computer and users.

Following are few of the important output devices which are used in Computer Systems

- Monitors
- Graphic Plotter
- Printer

Classification of Computer

Computer can be broadly classified by their speed and computing power. Like the following-

Type	Specifications
PC (Personal Computer)	Single user computer system. Moderately powerful microprocessor.
WorkStation	Single user computer system. Similar to Personal Computer but have more powerful microprocessor.
Mini Computer	Multi-user computer system. Capable of supporting hundreds of users simulaneously.
Main Frame	Multi-user computer system. Capable of supporting hundreds of users simulaneously.Software technology is different from minicomputer.
Supercomputer	An extremely fast computer which can perform hundreds of millions of instructions per second.

Computer Generations-

Generation in computer terminology is a change in technology a computer is/was being used. Initially, the generation term was used to distinguish between varying hardware technologies. But nowadays, generation includes both hardware and software, which together make up an entire computer system.

Following are the main five generations of computers -

Generation & Description
First Generation The period of first generation : 1946-1959. Vaccum tube based.
Second Generation The period of second generation : 1959-1965. Transistor based.
Third Generation The period of third generation : 1965-1971. Integrated Circuit based.
Fourth Generation The period of fourth generation : 1971-1980. VLSI microprocessor based.
Fifth Generation The period of fifth generation : 1980-onwards.ULSI microprocessor based

Computer Hardware:

Physical component of a computer are called hardware.

Example of hardware in computer system are the processor, memory devices, printer, keyboard, mouse etc.

Program:

- A program is a sequence of instructions written to solve a particular problem.

Computer Software:

The software is the collection of program, procedure, rules and data.

Example of software: word processor, web browser, compiler, interpreter, Excel, PowerPoint, Photoshop, MySQL etc.

Types of Software:

There are two types of software:

i) System software
ii) Application software

i) System Software:

System software is a software which controls the execution of a hardware.

Ex: Operating system, Compiler, Drivers, Interpreter, Assemblers, Linker, Loader etc.

ii) Application software:

Application software is a general-purpose software which uses system program to execute application program.

Ex: web browser, database system, word processing system etc.

Operating System-

- An operating system is a program that acts as an interface between the software and the computer hardware.
- It is an integration set of specialized programs that are used to manage overall resources and operations of the computer.
- It is specialized software that controls and monitors the execution of all other programs that reside in the computer, including application programs and other system software.

Examples: Windows XP/7/8 ..., Linux, UNIX, Android etc.

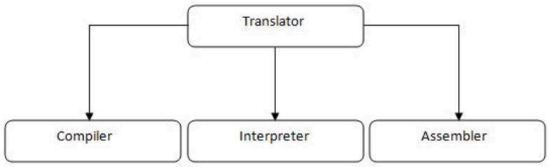
Functions of Operating System-

i) as a government
ii) as a resource Allocator
iii) as a control program

Translator

- Translator is a program which convert source program into object program.
- There are three Translators

1)Compiler
2)Interpreter
3)Assembler



1) Compiler:-

A compiler is a program which translates the source code written in a high level language into object code / target code written in machine language.

1) Interpreter:-

An interpreter translates a high-level language statement in a source program to a machine code and executes it immediately, before translating the next source language statement. When an error is found, the execution of the program is halted and an error massage is displayed on the screen of the computer.

2) Assembler

Assembler is a program which convert assembly language program into machine level language.

Memory

A memory is just like a human brain. It is used to store data and instruction. Computer memory is the storage space in computer where data is to be processed and instructions required for processing are stored.

Memory is primarily of three types

- Cache Memory
- Primary Memory/Main Memory
- Secondary Memory

Cache Memory

Cache memory is a very high speed semiconductor memory which can speed up CPU. It acts as a buffer between the CPU and main memory.
It is used to hold those parts of data and program which are most frequently used by CPU. The parts of data and programs are transferred from disk to cache memory by operating system, from where CPU can access them.

ADVANTAGE

- Cache memory is faster than main memory.
- It consumes less access time as compared to main memory.
- It stores the program that can be executed within a short period of time.
- It stores data for temporary use.

DISADVANTAGE:

- Cache memory has limited capacity.
- It is very expensive.

Primary Memory (Main Memory)

Primary memory holds only those data and instructions on which computer is currently working. It has limited capacity and data get lost when power is switched off.
It is generally made up of semiconductor device. These memories are not as fast as registers. The data and instruction required to be processed earlier reside in main memory. It is divided into two subcategories **RAM** (**Random Access Memory**) and **ROM** (**Read Only Memory**).

Characteristic of Main Memory

- These are semiconductor memories.
- It known as main memory.
- Usually volatile memory.
- Data is lost in case power is switch off.
- It is working memory of the computer.
- Faster than secondary memories.
- A computer cannot run without primary memory.

Secondary Memory

This type of memory is also known as external memory or non-volatile. It is slower than main memory. These are used for storing Data/Information permanently. For example: Hard Disk, CD-ROM,DVD etc.

Characteristic of Secondary Memory

- These are magnetic and optical memories.
- It is known as backup memory.
- It is non-volatile memory.
- Data is permanently stored even if power is switched off.
- It is used for storage of the data in the computer.
- Computer may run without secondary memory.
- Slower than primary memories.

Algorithm

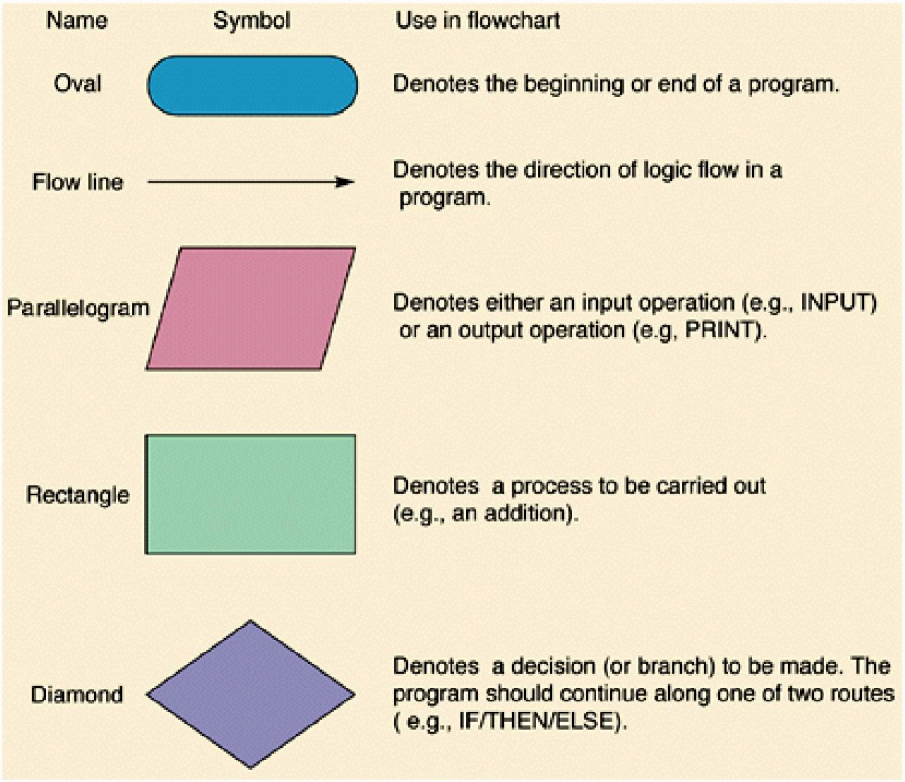
To make a computer do anything, you have to write a computer program. To write a computer program, you have to tell the computer, step by step, exactly what you want it to do. The computer then "executes" the program, following each step mechanically, to accomplish the end goal.
When you are telling the computer what to do, you also get to choose how it's going to do it. That's where computer algorithms come in. The algorithm is the basic technique used to get the job done.
Every algorithm should have the following five characteristics:

1. Input
2. Output
3. Definiteness
4. Effectiveness
5. Termination

Flow Chart

A flow chart is a graphical or symbolic representation of an algorithm. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction. The symbols of flowchart are-

-



Storage Classes

auto - This is the default storage class, which initialize a local variable. It initialize a variable with a garbage value. The syntax will be-

int a; or auto int a;

register – These variables are stored inside the registers(a kind of fast access memory) instead of RAM. So they are faster than auto. They are initialized by garbage value. The Syntax will be-

register int a;

static – These variable share a single copy to each function of a program. So change the value in any function reflects to other function. They are initialized by 0. The Syntax will be –

static int a;

extern – They are used as a global variable. Default initialize value will be 0. the Syntax will be-

extern int a;

UNIT-2

Introduction of C Language-

1. C was developed by Denish Ritche in 1972 at AT & T Bell Lab.
2. It is a high level programming language, some times called middle level programming language.
3. It is a case sensitive language.
4. The extention of C program must be .c or .cpp.
5. C uses compiler for converting high level code into machine level code.

A Simple C Program -

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("Hello Rishab");
        printf("\nWelcome in C Language");
        getch();
}
```

Step By Step Demostration of the program -

1. #include is known as preprocessor directive and it attach the given header file with our program at compile time.
2. main() function indicates that program is begin now.
3. {} (curly braces) indicates a block
4. clrscr() is a library function , defined inside conio.h which clears screen at run time
5. printf() is also a library function defined inside stdio.h which print the givcen message on he screen
6. getch() defined inside conio.h and it pauses your output, until you press any key from the keyboard.
7. \n changes a new line
8. ;(semi colon) indictes the compiler that statement is end now

Que- WAP which genertates the following output-
Welcome Friends
I The World of Programming
Hope you will enjoy

Tokens- The smallest individual units of a program are known as tokens.It consist the following elements-

1. Identifiers
2. Constant
3. Keywords
4. DataType
5. Operators

1. Identifiers- The name of any programming element is known as identifiers. like name of variable, function etc.

2. Constant- They are the fixed values which did not change during execution of a program.

like

        10,20,30
        "ram","shyam"

3. Keywords- They are the reserve words which implements any particular meaning in your program. there are 32 keywords in c language.

like -

        int,for,if,while,do etc

4. Datatype- It specifies the type of a variable.

        Variable- They represent a memory location where we can put any value according to their datatype.
        Syntax
                datatype variable_name;

5. Opeartors- They are the symbols which perform any particular operation with their operands. They could be classified into following categories-

1. Assignment Operator
2. Arithmetical Operator
3. Relational Operator
4. Logical Operator
5. Increment/Decrement Operator

6. Conditional Operator
7. Shorthand Operators
8. Bitwise Operators
9. sizeof() operators

1. Assignment Operator- The equals to sign(=) is known as assignment operator and it assign the value from right to left.

Syntax

```
variable=value;
<--------------------
ex
int a;
a=10;
```

Que- WAP for swapping the value of two variables with each other.
a=10;
b=20

2. Arithmetical Operator- They are used for performing the arithmetic operations. They are-

Operator	Meaning
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo Division

Que- WAP for adding two values.
Que- WAP for calculaing the area of rectangle
Que- WAP for calculating area of a circle.
Que- WAP which contains a number having three digits & write a program for reversing the digits of number.

scanf() function - This function is used for reading any value at run time-

Syntax

scanf("format_specifiers",&variable_list)

Format_Specifiers

%c	- Characters
%d	- Integer
%f	- Float
%lf	- Double
%s	- String
%x	- Hexadecimal
%o	- Octal
%u	- Unsigned Integer

Que- Read two values & Add Them.
Que- Read marks of a student in P,C,M and calculate their total percetnage & division
Que- Read Principal amount,rate & Time & Calculate simple intrest.
si=p*r*t/100

3. Relational Operator- They are used for comparing two or more than two variables. they return 1 if condition is true otherwise return 0. They are-

Operator	Meaning
<	is less than
<=	is less than or equals to
>	is greater than
>=	is greater than or equals to
==	is equals to
!=	is not equals to

4. Logical Opeartors- They are used for working with more than one conditions. They are-

Operator	Meaning
&&	Logical AND
||	Logical OR

! Logical NOT

5. Increment/Decrement Operator- The ++ sign is known as increment operator and -- sign is known as decrement operator. ++ increase 1 into its operand and -- decrease 1.
Here is two term known as-

Pre Incremenet/Decrement =>++/--Operand => First increase or decrease then perform operation
Post Incremenet/Decrement =>Operand++/-- => First perform operation then increase or decrease

6. Conditional Operator- It is also known as ternary operator and it used as a substitute of if else statement.
Syntax

condition?true statement:false statement;

Que- Check given number is even or odd?
Que- Read two values & print the largest.
7. Shorthand Operators- They are tricky method for working with operators, like
+=
-=
*=
/=
num=5
num=num+2; => num+=2
num=num*5;=>num*=5;

8. sizeof() operators- It returns how much memory space (bytes) is occupied by variable in memory.
9. Bitwise Operators- The bitwise operators works on binary bits.

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

Decision Making Statements

They are used for making the decision inside your program. The Decision Making Statements are-
1. if statement
2. switch statement

1. if statement- We can use different variations of if statement, These are-

a. if else statement
b. if else if ladder statement
c. netsed if statement

a. if else statement- if we have a condition & their are two aspect of the condition, first for true and second for false, then we use such type of statement.

Syntax

```
if(condition)
{
        true statement block
}
else
{
        false statement block
}
```

Que- Check given number is even/odd.
Que- Check The Triangle is equilateral or not?
Que- Read age of a person & print whether he/she can vote or not?
Que- Read basic salary & calculate the allowances like the following-

if basic>5000 then

ta=5% of basic
da=10% of Basic
hra=6% of Basic
pf=15% of Basic

gross_sal=basic+ta+da+hra;
net_sal=gross - pf

if basic<=5000 then

ta=2% of basic
da=5% of Basic
hra=3% of Basic
pf=7% of Basic
gross_sal=basic+ta+da+hra;
net_sal=gross - pf

b. if else if ladder statement- if we have more than one conditions then we can use if else if ladder statement.

Syntax

```
if(condition-1)
        statement-1
else if(condition-2)
        statement-2
_________________
_________________
else if(condition-n)
        statement-n
else
        default statement
```

Que- Read 3 values & print the largest one.

Que- WAP for calculating the electric bill. Read the value of current reading & last reading and calculate unit , rate and amount like the following-

```
unit=cr-lr;
rate
        if unit>1000 then rate=2
        if unit is b/w 500 to 1000 then rate=3
        if unit<500 then rate=4
amount=unit*rate
```

Que- Read marks of a students in physics, chemistry & math & calculate and print their total , percentage & division.

c. nested if statement - if, if statement contains one or more than one if statement then this term is known as nested if statement.

Syntax

```
if(condition)
{
        _________________
        _________________
        if(condition)
        {
                ---------------
                ----------------
        }
}
```

Que- Read 3 values & print the largest one.

switch case statement- It is also a decision making statement.It works faster than if statement, but the disadvantages is that it did not support logical and relational operator.

Syntax

```
switch(expression)
{
        case value-1:
                statement-1;
                break;
        case value-2:
```

```
                statement-2;
                break;
        _____________________
        _____________________
        case value-n:
                statement-n;
                break;
        default:
                default statement
    }
```

Que- Read day number & print the day name.

Que- Read two values & then print the following menu-

1. Add
2. Subtract
3. Multiply
4. Divide

then read the choice(1-4) & print the result according to choice.

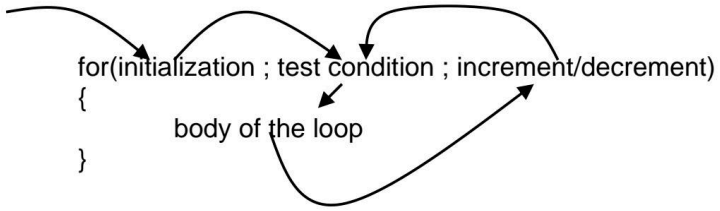
UNIT-3

Loop – Loop statements are used for repeating a part of program during finite or infinite times. It is also known as iterative or repetitive statement. It could be classified into two categories-

a) Entry Controlled Loop
b) Exit Controlled Loop

a) Entry Controlled Loop – This type of loop first checks the condition & if condition is satisfied then execute the body of the loop. "for loop" & "while loop" belongs to this category.
b) Exit Controlled Loop – This type of loop execute at least one time either condition is satisfied or not. "do while loop" belongs to this category.

for loop : It is an entry controlled loop & the general form of this loop will be-



```
for(initialization ; test condition ; increment/decrement)
{
        body of the loop
}
```

Important Questions –

Que- 1: WAP for printing the table of given number.

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

Que-2 : WAP for calculating the factorial of given number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int num,i,fact=1;
        clrscr();
        printf("Enter Number :");
        scanf("%d",&num);
        for(i=num;i>=1;i--)
        {
                fact=fact*i;
        }
        printf("\nFactorial : %d",fact);
        getch();
}
```

Que- 3: WAP for printing the Fibonacci series up to the given limit.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a=0,b=1,c,i,limit;
        clrscr();
        printf("Enter Limit Number :");
        scanf("%d",&limit);
        if(limit>=2)
        {
                printf("%d\n%d\n",a,b);
                for(i=1;i<=limit-2;i++)
                {
                        c=a+b;
                        printf("%d\n",c);
                        a=b;
                        b=c;
                }
        }
        getch();
}
```

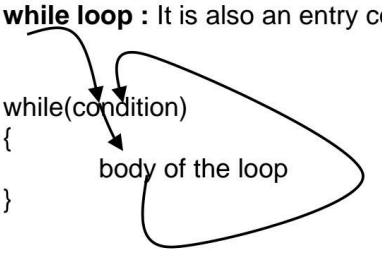
Que- WAP which read base number(x) & power number(n) & calculate x^n.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int x,n,res=1,i;
        clrscr();
        printf("Enter Base Number :");
        scanf("%d",&x);
        printf("Enter Power Number :");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                res=res*x;
        }
        printf("\nResult : %d",res);
        getch();
}
```

Exercise on for loop-

Que-1 : WAP for printing 1 to 10.
Qie-2 : WAP for printing 10 to 1.
Que-3 : WAP for printing number & their square up to the given limit.
Que-4 : WAP for printing the list of even numbers between 1 to 100.
Que-5 : WAP

while loop : It is also an entry controlled loop & the general form of this loop will be-



```
while(condition)
{
        body of the loop
}
```

Important Questions –

Que- WAP for reversing the digits of a given number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int num,r,rev=0;
        clrscr();
        printf("Enter Number :");
        scanf("%d",&num);
        while(num>0)
        {
                r=num%10;
                rev=rev*10+r;
                num=num/10;
        }
        printf("\nReverse : %d",rev);
        getch();
}
```

Que – WAP for printing the sum & average of digits, of a given number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int num,sum=0,avg,r,nod=0;
        clrscr();
        printf("Enter Number :");
        scanf("%d",&num);
        while(num>0)
        {
                r=num%10;
                sum=sum+r;
                nod++;
                num=num/10;
        }
        avg=sum/nod;
        printf("\nSum of Digits : %d",sum);
        printf("\nAverage of Digits : %d",avg);
        getch();
}
```

Que- WAP which read a number & check whether given number is Palindrome or not?

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int num,r,rev=0,n;
        clrscr();
        printf("Enter Number :");
        scanf("%d",&num);
        n=num;
        while(num>0)
        {
                r=num%10;
                rev=rev*10+r;
                num=num/10;
        }
        if(n==rev)
                printf("\nNumber is Palindrome");
        else
                printf("\nNumber is Not Palindrome");
        getch();
}
```

Que- WAP which read a number & check whether given number is Armstrong or not?

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int num,r,s=0,n;
        clrscr();
        printf("Enter Number :");
        scanf("%d",&num);
        n=num;
        while(num>0)
        {
                r=num%10;
                s=s+(r*r*r);
                num=num/10;
        }
        if(n==s)
                printf("\nNumber is Armstrong");
        else
                printf("\nNumber is Not Armstrong");
        getch();
}
```

Exercise on while loop-

Que-1: WAP for printing the largest digit of a given number.
Que-2: WAP for calculating the factorial of a given number using while loop.
Que-3: WAP which count how many even & odd digits in a given number.

do while loop : it is an exit controlled loop, means it will execute at least one time either condition is satisfied or not. The general form will be-

```
do{
        body of the loop
}while(condition);
```

Difference between while loop & do while loop

while Loop	do while loop
1. it is entry controlled loop. 2. it first check the condition & if satisfied then execute body of loop. 3. Most frequently used.	1. it is an exit controlled loop. 2. it execute the body of loop either condition is satisfied or not? 3. Occasionally used.

Example of do while loop

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,ch;
        clrscr();
        do{
                printf("\nEnter First Number :");
                scanf("%d",&a);
                printf("Enter Second Number :");
                scanf("%d",&b);
                printf("Sum : %d",a+b);
                printf("\nPress 1 For Continue The Program :");
                scanf("%d",&ch);
        }while(ch==1);
        printf("\nBye-Bye");
        getch();
}
```

break Statement- This statement is used for terminating the flow of loop as per the given condition.

Example

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

Que- Check given number is prime or not?

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int num,i,isprime=0;
        clrscr();
        printf("Enter Number :");
        scanf("%d",&num);
        for(i=2;i<=num-1;i++)
        {
                if(num%i!=0)
                        isprime=1;
                else
                {
                        isprime=0;
                        break;
                }
        }
        if(isprime==1)
                printf("\nGiven Number is Prime");
        else
                printf("\nGiven Number is not Prime");
        getch();
}
```

continue statement- This statement is used for ignoring the sequence of loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i;
        clrscr();
        for(i=1;i<=10;i++)
        {
                if(i>=5 && i<=8)
                        continue;
                printf("%d\n",i);
        }
        getch();
}
```

Exercise

Que- WAP which read three numbers & print the LCM of these numbers.

Nesting of loop statement- if loop statement contains one or more than one another loop statement into its body then this term is known as nesting of loop statement.

Que-1: WAP for printing the table of 2 to 10.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i,j;
        clrscr();
        for(i=2;i<=10;i++)
        {
                for(j=1;j<=10;j++)
                {
                        printf("%d\t",i*j);
                }
                printf("\n");
        }
        getch();
}
```

Exercise-

Que-1: WAP for printing the list of prime nos between 1 to 100.

Que-2: WAP for printing the list of Palindrome numbers between 1 to 1000.

Que-3: Write programs for printing the following structures-

1 12 123 1234 12345	1 01 010 1010 10101	A AB ABC ABCD ABCDE	1 23 456 78910	1 12 123 1234 12345
12345 1234 123 12 1	***** **** *** ** *	12345 1234 123 12 1	* *** ***** *******	

Functions

Functions are also known as sub programs, they are the named block which perform any particular action when they are called. Functions could be classified into two categories, they are-

a) Library Functions

b) User Defined Functions

a) Library Functions- They are built in functions & provide by 'C' standard library. They are defined inside any particular header file, only we call these functions without knowing the background details. Some most frequently used library functions are – printf(), scanf(), getch(), clrscr(), strlen(), sqrt() etc.

b) User Defined Functions- The functions which are defined by the user as per the requirement are known as user defined functions. The main objectives for creating an UDF are-

- **i.** Dividing a big problem into different small modules.
- **ii.** Implementing the concept of reusability
- **iii.** Increasing the readability of the program
- **iv.** Decreasing the complexity of the program.
- **v.** Scalability

The general form of an UDF will be-

```
return_type function_name(argument_list)
{
        body of the function
}
```

return_type- It specifies which type of value will be return by the function. Here we specify the **data type** of return value, if there is nothing to return then write **void** here.

function_name- Here we specify the name of function, it may be any valid name.

argument_list- Here we specify the name & type of the return value. If there is no argument then you can either leave it blank or write void here.

body of the function- Here we specify the task of the function.

Note – During creation of an UDF we must follow the following three steps-

1) **Function Declaration or Function Prototype –** Here we specify the signature of the function. Like-

```
void demo(void)// function name demo contains no argument & also not return a value
void demo(int a,int b)// function name demo contains two argument but not return a value
int demo(int a)// function name demo contains one argument and return a value
```

It take place either outside the main() or inside the main() function.

2) **Function Calling –** Here we invoke the function for performing the actual task. It take place inside the main() function or any other function.
3) **Function Definition-** Here we specify that what will be actually perform by the function. It take place outside the body of main() function or any other function.

Types of User Defined Function – An UDF could be classified into following categories-

a) Function with no argument & no return value
b) Function with argument but no return value
c) Function with argument & return value

a) Function with no argument & no return value- This type of functions are used for performing the static operations. They not contain any argument as well as not return any value at the calling position. When we call these function the control transfer on there definition & after performing the task they return on calling position.

Example- WAP which generate the following output-

```
**********************************************************
                    Welcome
**********************************************************
**********************************************************
#include<stdio.h>
#include<conio.h>
void line(void);
void main()
{
        clrscr();
        line();
        printf("\n\t\t\tWelcome\n");
        line();
        line();
        getch();
}
void line(void)
{
        int i;
        for(i=1;i<=70;i++)
        {
                printf("*");
        }
        printf("\n");
}
```

Exercise :

Que-1: create a function named info() which print your name & address when it will called.

b) **Function with argument but no return value-** This type of functions are used for performing the dynamic operations according to given value. They contain argument but not return any value at the calling position. There are two concepts about the arguments they are-

i. **Real Argument-** The arguments which are passed at the time of function calling are known as real argument.
ii. **Formal Argument-** The arguments which are passed at the time of function definition are known as formal argument. The type & order of formal argument must be same as the real argument but the name must be differ.

Example- Create a function named as table() which contains a number as argument & print the table of that number.

```
#include<stdio.h>
#include<conio.h>
void table(int);
void main()
{
        clrscr();
```

Exercise :

Que-1: Create a function named reverse() which contains a number as argument & print the digits of number in reverse form.

Que-2: Create a function named as fibo() which contains a limit number as argument & print the Fibonacci series up to the given limit.

```
        int num;
        printf("Enter Number :");
        scanf("%d",&num);
        table(num);
        getch();
}
void table(int n)
{
        int i;
        for(i=1;i<=10;i++)
        {
                printf("%d\n",n*i);
        }
}
```

c) **Functions with argument & return value-** This type of functions are used for performing the dynamic operations & they return a value at the calling position with the help of return keyword.

Example: Create a function named as fact() which contains a number as argument & return the factorial of that number.

```
#include<stdio.h>
#include<conio.h>
int fact(int);
void main()
{
        clrscr();
        int num,res;
        printf("Enter Number :");
        scanf("%d",&num);
        res=fact(num);
        printf("\nFactorial : %d",res);
        getch();
}
int fact(int n)
{
        int i,f=1;
        for(i=n;i>=1;i--)
        {
                f=f*i;
        }
        return f;
}
```

Exercise :

Que-1: Create a function named as power() which contains base number x & power number n & return x to the power n.

Que-2: Create a function named as max() which contains three numbers as argument & return the largest value.

Que-3: Create a function named as prime() which contains a number as argument & return 1 if number is prime otherwise return 0.

Recursion: When a function called itself inside its own body then this term is known as recursion & this type of functions are known as recursive functions. It is used for repeating any particular task.

Example : Create a recursive function named as fact() which contains a number as argument & return the factorial of that number.

```
#include<stdio.h>
#include<conio.h>
int fact(int);
void main()
{
        clrscr();
        int num,res;
        printf("Enter Number :");
        scanf("%d",&num);
        res=fact(num);
        printf("\nFactorial : %d",res);
        getch();
}
```

Exercise :

Que- Create a recursive function named as reverse() which contains a number as argument & print the digits of number in reverse form.

Que- Create a recursive function named as fibo() which print the Fibonacci series up to the given limit.

```
int fact(int n)
{
        if(n==1)
                return 1;
        else
                return n*fact(n-1);
}
```

Unit-4
Array

Array – It is the collection of same type of data type that shares a common name. It is also known as subscripted variable. It could be classified into following categories-

a) Single Dimensional Array
b) Double Dimensional Array

a) Single Dimensional Array – This type of array contains single rows with multiple columns. The general form of this type of array will be-

datatype arrayname[size];

ex.

Int arr[5];

Important Questions –

Que-1: Read 5 values using array & print them.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i,arr[5];
        clrscr();
        printf("Enter Five Values \n");
        for(i=0;i<=4;i++)
        {
                scanf("%d",&arr[i]);
        }
        printf("\nThe Given Values are \n");
        for(i=0;i<=4;i++)
        {
                printf("%d\n",arr[i]);
        }
        getch();
}
```

Que-2: WAP for searching an element from an array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int arr[5];
        int i,found=0,num;
        printf("Enter Five Values \n");
        for(i=0;i<=4;i++)
        {
                scanf("%d",&arr[i]);
        }
        printf("\nEnter Number For Search :");
        scanf("%d",&num);
        for(i=0;i<=4;i++)
        {
                if(arr[i]==num)
                {
                        found=1;
                        break;
                }
        }
        if(found==1)
                printf("\nNumber Exist in Array");
        else
                printf("\nNumber Not Exist in Array");
        getch();
}
```

Que-3: WAP for printing the largest element from an array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int arr[5];
        int i,max;
        clrscr();
        printf("Enter Five Values \n");
        for(i=0;i<=4;i++)
        {
                scanf("%d",&arr[i]);
        }
        max=arr[0];
        for(i=0;i<=4;i++)
        {
                if(max<arr[i])
                        max=arr[i];
```

Que-4: WAP for sorting the elements of an array.(Selection Sort).

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int arr[]={190,20,300,40,50};
        int i,j,temp;
        clrscr();
        printf("Before Sorting Elements are \n");
        for(i=0;i<=4;i++)
        {
                printf("%d\n",arr[i]);
        }
        for(i=0;i<=4;i++)
        {
                for(j=i+1;j<=4;j++)
                {
                        if(arr[i]>arr[j])
```

```
        }
        printf("\nLargest Value is : %d",max);
        getch();
}
```

```
                        {
                                temp=arr[i];
                                arr[i]=arr[j];
                                arr[j]=temp;
                        }
                }
        }
        printf("\nAfter Sorting Elements are \n");
                for(i=0;i<=4;i++)
        {
                printf("%d\n",arr[i]);
        }

        getch();
}
```

Exercise on Single Dimensional Array-

Que-1: WAP for printing the Binary Equivalent of given integer value.
Que-2: WAP for counting how many even & odd elements in an array.
Que-3: WAP for printing the prime elements from an array.

b) Double Dimensional Array – This type of array contains multiple rows with multiple columns. The general form of this type of array will be-

Datatype ArrayName[Rows][Cols];
Ex.
int mat[3][3];

mat[0][0]	mat[0][1]	mat[0][2]
mat[1][0]	mat[1][1]	mat[1][2]
mat[2][0]	mat[2][1]	mat[2][2]

Important Questions -

Que-1: WAP which read 9 values for a matrix of 3*3 & print the elements in matrix form.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int mat[3][3],i,j;
        printf("\nEnter Nine Numbers For Matrix of 3*3\n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        scanf("%d",&mat[i][j]);
                }
        }
        printf("\nMatrix is \n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        printf("%d\t",mat[i][j]);
                }
                printf("\n");
        }
        getch();
}
```

Que-2: WAP for multiplying two matrixes.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int a[3][3],b[3][3],c[3][3],i,j,k;
        printf("\nEnter Nine Numbers For First Matrix \n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        scanf("%d",&a[i][j]);
                }
        }
        printf("\nEnter Nine Numbers For Second \n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        scanf("%d",&b[i][j]);
                }
        }
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        c[i][j]=0;
                        for(k=0;k<=2;k++)
                        {

c[i][j]=c[i][j]+(a[i][k]*b[k][j]);
```

```
                        }
                }
                printf("\n");
        }
        printf("\nFirst Matrix is \n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        printf("%d\t",a[j][i]);
                }
                printf("\n");
        }
        printf("\nSecond Matrix is \n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        printf("%d\t",b[j][i]);
                }
                printf("\n");
        }
        printf("\nMultiply of Both Matrix is \n");
        for(i=0;i<=2;i++)
        {
                for(j=0;j<=2;j++)
                {
                        printf("%d\t",c[j][i]);
                }
                printf("\n");
        }
getch();
}
```

Exercise on Double Dimensional Array-

Que-1: WAP for transposing a matrix.

Que-2: WAP for adding two matrixes.

String Handling

String is an array of characters, for declaring a string variable we will use the following syntax-

```
char variablename[size];
ex.
char n[35];
```

Que- Read A name & print it with hello message.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char n[40];
        clrscr();
        printf("Enter Name :");
        scanf("%s",n);
        printf("Hello %s",n);
        getch();
}
```

Note- the scanf() function read a word, not a sentence.

Que- Read A name & print it with hello message.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char n[40];
        clrscr();
        printf("Enter Name :");
        gets(n);
        printf("Hello %s",n);
        getch();
}
```

Note- the gets() function read a complete sentence.

Note- When we press enter key for terminating the string the compiler automatically placed a Null Character('\0') at the last, which indicates string is end now, like-

U	N	I	T	E	D	'\0'

Important Questions

Que-1: Read a name & print the length of name.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char n[40];
        int i;
        clrscr();
        printf("Enter Name :");
        gets(n);
        for(i=0;n[i]!='\0';i++)
        {}
        printf("\nLength is : %d",i);
        getch();
}
```

Que-2: Read a name & print characters of name in reverse form.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char n[40];
        int i,j;
        clrscr();
        printf("Enter Name :");
        gets(n);
        for(i=0;n[i]!='\0';i++)
        {}
        printf("\nThe Reverse is :");
        for(j=i-1;j>=0;j--)
        {
                printf("%c",n[j]);
        }
        getch();
}
```

Que-3: Check given string is palindrome or not?

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char n[40];
        int i,j,palin=0;
        clrscr();
        printf("Enter Name :");
        gets(n);
        for(i=0;n[i]!='\0';i++)
        {}
        i=i-1;
        for(j=0;n[j]!='\0';j++,i--)
        {
                if(n[j]==n[i])
                        palin=1;
                else
                {
                        palin=0;
                        break;
                }

        }
        if(palin==1)
                printf("String is Palindrome");
        else
                printf("String is not Palindrome");
        getch();
}
```

Important String Handling Functions: String handling functions are defined inside string.h header file, they provide the facility for processing string easily. The important string handling functions are-

1. **strlen() –** this function contains a string as argument & return the length of the string.
2. **strcpy() –** this function contains two strings as argument & copies the value of second argument into first argument(string variable).
3. **strcat()-** this function contains two strings as argument & returns the concatenation of both string.
4. **strcmp()-** this function contains two strings as argument & return 0 if both strings are equal otherwise return a non zero value.

Structure

Structure- Structure is a technique by which we can create a User Defined Data Type. Structure allows you for create a complex data type in which more than one variable with different data type are encapsulated. The general form of a structure will be-

```
struct structurename
{
        member-1;
        member-2;
        _____________________
        _____________________
        member-n;
};
```

Example

```
struct stu
{
        char name[30];
        int age;
        char address[50];
};
```

Creating variable of structure – for creating variable of structure we will follow the following syntax-

Syntax

Struct structurename variable1,variable2,…….

Ex.

Struct stu s;

For accessing the members of structure we will use the member ship operator (.).

Example: Read name, age & address of a student & print them.

```
#include<stdio.h>
#include<conio.h>
struct stu
{
        char name[30];
        int age;
        char addr[50];
};
void main()
{
        clrscr();
        struct stu s;
        printf("Enter Name :");
        gets(s.name);
        fflush(stdin);
        printf("Enter Age :");
        scanf("%d",&s.age);
        fflush(stdin);
        printf("Enter Address :");
        gets(s.addr);
        fflush(stdin);
        printf("\nName : %s",s.name);
        printf("\nAge : %d",s.age);
        printf("\nAddress : %s",s.addr);
        getch();
}
```

Array of Structure: Like any other data type we can also declare the variable of structure as an array like the following-

struct structurename variablename[size];

for example if we declare the variable of stu structure as an array then it will allocate the memory like the following –

struct stu s[5];

name age address	name age address	name age address	name age address	name age address
s[0]	s[1]	s[2]	s[3]	s[4]

Array of Structure Example :

Example : Read maks of 5 student in physics, chemistry & math & calculate and print their total, percentage & division.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct stu
{
        char name[30],div[30];
        int p,c,m,tot,per;
};
void main()
{
        clrscr();
        struct stu s[5];
        int i;
        for(i=0;i<=4;i++)
        {
                printf("\nEnter Record Number : %d\n",i+1);
                printf("Enter Name :");
                gets(s[i].name);
                fflush(stdin);
                printf("Enter Marks of Physics :");
                scanf("%d",&s[i].p);
                fflush(stdin);
                printf("Enter Marks of Chemistry :");
                scanf("%d",&s[i].c);
                fflush(stdin);
                printf("Enter Marks of Math :");
                scanf("%d",&s[i].m);
                fflush(stdin);
                s[i].tot=s[i].p+s[i].c+s[i].m;
                s[i].per=s[i].tot/3;
                if(s[i].per>=60)
                        strcpy(s[i].div,"First");
                else if(s[i].per>=45 && s[i].per<60)
                        strcpy(s[i].div,"Second");
                else if(s[i].per>=33 && s[i].per<45)
                        strcpy(s[i].div,"Third");
                else
                        strcpy(s[i].div,"Failed");
        }
        for(i=0;i<=4;i++)
        {
                printf("\nRecord Number : %d\n",i+1);
                printf("\nName : %s",s[i].name);
                printf("\nTotal : %d",s[i].tot);
                printf("\nPercetage : %d",s[i].per);
                printf("\nDivision : %s",s[i].div);

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

Excersice:

Que-1: Write a program for storing the basic details like Name, Course, Branch of 5 students & print them.

Que-2: Design a structure named as Date which contains three integer members date, month & year.

Write a program which read two dates & check whether both are equals or not?

Union

Union can be defined as a user-defined data type which is a collection of different variables of different data types in the same memory location. The union can also be defined as many members, but only one member can contain a value at a particular point in time.
Union is a user-defined data type, but unlike structures, they share the same memory location.

Syntax:

```
union union_name
{
   union element;
   union element;
   ...
   union element;
} [one or more union variables];
```

Enumerated Data Types

Enumeration (or enum) is a user-defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

- Keyword enum is used for this data type.
- **Syntax :** enum enumeration_name { identifier1 , identifier 2, ……., identifier n}.
  Enumeration name is optional.

Example : enum COLORS { RED , BLUE , BLACK , GREEN , YELLOW ,PURPLE} ;

Unit-5
Pointer

Pointer is a variable which points the address of some other variables. For creating a pointer variable we will use the following syntax-

datatype *variablename;

ex.

int *p;

in pointers we use two types of operators these are –

* - value at the address

& - address at the value

Basic Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int n,*p;
        n=10;
        p=&n;
        printf("\nValue & Address Through Variable \n");
        printf("\nValue : %d",n);
        printf("\nAddress : %u",&n);
        printf("\nValue & Address Through Pointer\n");
        printf("\nValue : %d",*p);
        printf("\nAddress : %u",p);
        getch();
}
```

Pointer with Array (Pointer Arithmetic): Remember that a pointer variable could not hold the address of an array. It only reference the base address of an array like the following –

Ex.

```
int arr[5];
int *p;
p=arr;
```

1024	1026	1028	1030	1032
10	20	30	40	50

*p

here by using ++ we can move on next location & by using -- we can move on previous location. This term is known as pointer arithmetic.

Pointer with Array Example :

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int arr[]={10,20,30,40,50};
        int i,*p;
        p=arr;
        printf("\nValue\tAddress\n");
        for(i=1;i<=5;i++)
        {
                printf("%d\t%u\n",*p,p);
                p++;
        }
        getch();
}
```

Pointer with function- There are two mechanisms for passing a value to a function. These are –

Call By Value – In this case we pass the value as function argument. In this case all the changes which occur inside the function will not reflect outside the function.

Call By Address or Call By Reference - In this case we pass the address as function argument. In this case all the changes which occur inside the function will reflect outside the function.

Call By Value Example

```
#include<stdio.h>
#include<conio.h>
void swap(int,int);
void main()
{
        clrscr();
        int a=10,b=20;
        printf("\nBefore Swap a=%d and b=%d\n",a,b);
        swap(a,b);
        printf("\nAfter Swap a=%d and b=%d\n",a,b);
        getch();
}
void swap(int x,int y)
{  int z;
    z=x;
    x=y;
    y=z;  }
```

Output – the value of a & b will not change here.

Call By Address or Call By Reference Example

```
#include<stdio.h>
#include<conio.h>
void swap(int *,int *);
void main()
{
        clrscr();
        int a=10,b=20;
        printf("\nBefore Swap a=%d and b=%d\n",a,b);
        swap(&a,&b);
        printf("\nAfter Swap a=%d and b=%d\n",a,b);
        getch();
}
void swap(int *x,int *y)
{  int z;
    z=*x;
    *x=*y;
    *y=z;  }
```

Output – the value of a & b will be change here.

Dynamic Memory Allocation

Allocating memory at run time is known as dynamic memory allocation. It is implemented by pointers. C language provides some function which are used for implanting DMA. These are –

malloc() function – This function is used for allocating a block of memory. The general form will be-
Syntax

```
*ptr=(datatype *)malloc(sizeof(datatype))
```

calloc() function – This function is used for allocating multiple block of memory. The general form will be-
Syntax

```
*ptr=(datatype *)calloc(number_of_blocks,sizeof(datatype))
```

free() function – This function is used for releasing the memory which is allocated by calloc() or malloc() function.

File Handling

File handling is a technique by which we can store the processed data in form of file & we can access that data whenever required. For creating working with a file we must create a file pointer like the following –

```
FILE *fp;
```

Some important file handling functions are –

fopen()- this function initialize a file pointer. The syntax will be-

```
fopen(file_name,file_mode);
```

file_mode – it specifies the file opening mode like-

- w – write mode(creating a file)
- r- read mode (reading the content of a file)
- a – append mode (modifying an existing file).

fgetc() – gets a character from a file stream(memory).
fputc() – puts a character on a file stream(memory).
Fclose()- closes a file.

Important Questions –

Que-1: Read data from keyboard & store that data in a file.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        FILE *fp;
        char fn[15],ch;
        printf("Enter File Name For Craete a New File
:");
        scanf("%s",&fn);
        fp=fopen(fn,"w");
        printf("\nInput Contents of File, Press Ctrl+Z or
F6 For Exit\n");
        while((ch=getchar())!=EOF)
        {
                fputc(ch,fp);
        }
        puts("File Created");
        getch();
}
```

Que-2: Write a program for reading the contents of an existing file.

```
#include<stdio.h>
#include<process.h>
#include<conio.h>
void main()
{
        clrscr();
        FILE *fp;
        char fn[15],ch;
        printf("Enter File Name For Read :");
        scanf("%s",&fn);
        fp=fopen(fn,"r");
        if(fp==NULL)
        {
                puts("File Not Exist");
                getch();
                exit(0);
        }
        while((ch=fgetc(fp))!=EOF)
        {
                printf("%c",ch);
        }
        getch();
}
```

Que-3: WAP for creating the duplicate file of any existing file.

```
#include<stdio.h>
#include<process.h>
#include<conio.h>
void main()
{
        clrscr();
        FILE *fp1,*fp2;
        char source[30],target[30],ch;
        printf("Enter Source File Name:");
        scanf("%s",&source);
        fp1=fopen(source,"r");
        if(fp1==NULL)
        {
                puts("File Not Exist");
                getch();
                exit(0);
        }
        printf("Enter Target File Name:");
        scanf("%s",&target);
        fp2=fopen(target,"w");
        while((ch=fgetc(fp1))!=EOF)
        {
                fputc(ch,fp2);
        }
        puts("Copy Created...");
        getch();
}
```

Standard C Preprocessors

The preprocessors directives are processed at the time of program compilation. They are generally starts with # symbol. Some important preprocessors are –

#include – This preprocessor directive is used for attaching a header file with our program at compile time.

#define – This preprocessor is used for defining an identifier or macro.

A Macro Example –

```
#include<stdio.h>
#include<conio.h>
#define max(a,b) (a>b?a:b)
void main()
{
        clrscr();
        int x,y;
        printf("Enter Two Numbers :\n");
        scanf("%d%d",&x,&y);
        printf("\nLargest : %d",max(x,y));
        getch();
}
```

Command Line Arguments

Command line arguments are the arguments which the user gives from the operating system's command line during the time of execution. Earlier, we used main() functions without arguments. These command line arguments are handled by the main() function.

If you want to pass command line arguments then you will have to define the main() function with two arguments. The first argument defines the number of command line arguments and the second argument is the list of command line arguments.

Syntax:

```
int main(int argc, char *argv[])
```