Computer Science

Grade: XII

Programming in C

Reference Note

Unit Wise NEB Important Questions for Computer Science XII

Unit-4- Programming in C - 8 Marks

1.) What is C? Write its Features.
2.) What are advantages and disadvantages of C programming Languages?
3.) What is a Data type? Explain the type of data with examples.
4.) What is variable? Explain its types.
5.) What is operator? Write its types and explain any four them.
6.) What is control structure? Write different between break and continue statement with examples.
7.) What is looping? Write different between while and Do while loop with examples.
8.) Define the term array. What is string? Explain any four string handling function with example.
9.) What is functions? Write its features and describe its types.
10.) What is recursion technique? Explain with one example.
11.) What is concept of storage? Differentiate between automatic storage and external storage.
12.) What is a Structure? Explain with one examples.
13.) Differentiate between array and structure.
14.) Differentiate between structure and union.
15.) Differentiate between array and pointer.
16.) Define the term call by value and call by reference with examples.
17.) What is pointer? Explain with examples.
18.) Differentiate between Structure and Pointer.
19.) Differentiate between sequential access and random access techniques of data file.
20.) Differentiate between fprintf and fscanf function.

IMP Selected C Programs

1. Write a program to input any three numbers and find out which one is largest numbers.
2. Write a program to display day using the switch statement depending upon the number entered. i.e. input 1 for Sunday, 7 for Saturday.
3. Write a program to input a number and check whether it is prime or not.
4. Write a program to display the sum of even and odd numbers from 1 to 50.
5. Write a program to accept the age of 10 different employees and count the number of employee.
   a. Whose age is more than or equal to 60
   b. Whose age is less than 35
6. Write a program to enter elements for 3x3 matrixes and Display its sum.
7. Write a program to accept 10 different numbers in array and sort in descending order.
8. Write a program to input a number and find out that number is palindrome or not.
9. Write a program to input the names of N numbers of students and sort them in alphabetical order.
10. Write a program to read a number and make the sum of individual digits and print using recursion technique.
11. Write a program that reads different names and address into the computer and rearrange the names into alphabetical order using the structure variables.
12. Write a program to read N students record store them in data file and display the content in appropriate format by using fprintf and fscanf function.

Unit 4- Programming in C

What is C? Explain its development process.

C is a high level language because no need for any architecture knowledge in normal English form.
C is a compiler because it can translate the whole program at a time so we can call compiler. C is structured programming language. It is called also procedural oriented programming language, function oriented language, module programming language. It is simple, reliable and easy to use.

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

C is a computer language and a programming tool which has grown popular because programmers preferred it. It is a tricky language but a masterful one.
The C programming languages has the following ***features:***

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

Advantage of C language

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

Disadvantage of C Language

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

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

What is header file? List the header files with some functions, where are they used?
A file that is defined to be included at the beginning of a program in C language that contains the definitions of data types and declarations of variables used by the functions in the program is called header file.

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

Some of the frequent used header files are explain below:

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

<u>**Fundamentals of C**</u>

1) What are the character set used in C?
A group of alphabetic, numeric and other characters that have some relationship with C programming language and recognized by compiler is called Character set. A character set can also contain additional characters with other code values.
The keywords, identifiers and other variables are constructed by using character set. The character set consists of following elements.

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

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

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

- ➢ Identifiers consists letters and digits.
- ➢ First character must be an alphabet or underscore.

- Upper case and lowercase are allowed but not same, i.e. Text not same as text.
- Only one special character underscores (_) will used.
  For example, int a_b; Where ***a*** and ***_b*** are valid identifiers.

Keywords:-

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

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

Tokens:

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

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

2) Explain data types used in programming with examples.

Data types:

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

Types of data types in C.
1) Primary data types
2) Secondary data types

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

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

Variable:

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

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

✓ It should not contain any special characters.

Types of variable

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

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

Constant variable:

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

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

Specifier:

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

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

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

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

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

Operator:

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

Types of operator

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

1. Arithmetic Operator

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

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

2. Relational Operator

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

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

3. Logical Operator

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

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

4. Increment and Decrement Operators:

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

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

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

5. Conditional Operator:

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

Expression_1 ? expression_2: expression_3

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

Library function:

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

Control Structure:

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

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

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

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

Types

a) Conditional Statements

i. if statements:

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

Syntax,

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

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

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

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

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

Syntax,

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

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

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

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

iii. <u>if() else if() statements</u>

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

Syntax,

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

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

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

iv. <u>Nested if else statements</u>

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

Syntax,

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

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

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

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

b) Switch case statements

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

Syntax,

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

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

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

1. Looping Statement

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

Types of loop

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

1**. For Loop:-**

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

Syntax,

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

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

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

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

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i, s=0;
    for(i=1;i<=20; i=i+2)
    {
            printf("/n even number are %d",i);
            s=s+i;
    }
    printf("sum of even number is %d", s);
    getch();
}
```

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

```
# include<stdio.h>
# include<conio.h>
```

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

Nested for loop:

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

For example

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

1. Display the following output.

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

2. WAP to display the following output.

```
55555
4444
333
22
1
```

Programs

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

   }
    getch();
}
```

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

         }
        getch();
}
```

<u>While Loop:-</u>

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

Syntax,

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

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

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

<u>Do while loop:-</u>

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

The syntax of the loop is similar to while loop.

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

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

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

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

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

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

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

6	Syntax
	while (expression)
	{
	   //statements
	}
			Syntax
			do
			{
			   //statements
			} while (expression);

<u>The jump Statements</u>

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

1) break statements:

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

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

The continue Statement

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

```
# include<stdio.h>
# include<conio.h>
void main()
{
        int num;
        for(num=1; num!=10; num++);
{
        if(num==7)
        {
                continue;
        }
        printf("%d",num);
        }
        getch();
        }                OUTPUT: 1,2,3,4,5,6,8,9
```

Go to statement:

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

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

Some important C programs

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

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

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

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

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

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

```
#include<stdio.h>
#include<conio.h>
void main()
{
Int i,n;
 int x,y,z;
x=0,y=1,z=0;
printf("Enter the number");
scanf("%d",&n);
for(i=1;i<n;i++)
{
  printf("%d",z);
  z=x+y;
  x=y;
  y=z;
}
getch();
}
```

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

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

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

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i,sum=0;
        for(i=2; i<=100;i+2)
        {
                printf("\n Even number are %d",i);
                sum=sum+i;
        }
        printf("\n Sum of even numbers is %d",sum);
getch();
}
```

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

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

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

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

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

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

```
                                        printf("\n Armstrong number is %d",a);
                                }
                                else
                                {
                                        printf("\n Number is not Armstrong %d",a);
                                }
                        }
                else
                {
                        printf("\n It is not positive number");
                }
                        getch();
                }
```

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

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

10.WAP to display all prime numbers upto 1000.

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

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

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int s,t;
        printf("The square numbers between 100 to 500 are");
        for(s=100;s<=500;s++)
        {
                for (t=1;t<=s;t++)
                {
                        if(s==t*t)
                        {
                                printf("%d  x %d =  %d \n",s,s,t);
                        }
                }
                getch();
        }
```

Arrays and String Function:

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

Advantage of arrays.

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

Disadvantages of arrays.

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

There are two types

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

   Syntax: type array_name[max. size];

   Example int n[10];

   int age[]= {18,12,19,20,16,16,17};

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

   Syntax: type array_name[No.Rows] [No.Cols];

   Example int n[10][5];

```
int matrix[3][3]= {{0,1,2},{3,4,5},{6,7,8}};
```

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

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

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

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

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

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

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

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

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

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        clrscr();
        int I,J, matrix[3][4];
        printf("Enter a matrix of 2x4:\n\n");
        for(I=1; I<=3; I++)
```

```
                {
                        for(J=1; J<=4; J++)
                        {
                                scanf("%d",&matrix[I][J]);
                        }
                }
                Printf("\n The elements of matirx are: \n\n"):
                for(I=1; I<=3; I++)
                {
                        for(J=1; J<=4; J++)
                        {
                                printf("%d\t",matrix[I][J]);
                        }
                        printf("\n");
                }
        }
```

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

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

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

i) ***Whose age is more than or equal to 60***

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

```
        # include<stdio.h>
        # include<conio.h>
         void main( )
        {
                int age[10];
                int count1=0;
                int count2=0;
                int i;
                for (i=1; i<=10; i++)
                {
                        printf("Enter the ages");
                        scanf("%d", &age[i]);
                }
                        for (i=1; i<=10; i++)
                        {
                                if (age[i]>=60)
                                {
                                        count1++;
                                }
                                else
                                {
                                                if(age[i]<=35)
                                                {
                                                count2++;
                                                }
                                 }
                            }
                printf("Employees above or equal to 60 %d",count1);
                printf("Employees under or equal 35 %d",count2);
        }
```

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

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

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

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

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

```
# include<stdio.h>
# include<conio.h>
void main( )
{
        int j, k;
        int num[3][4]={1,2,3,4,8,10,11,5,9,6,2,8}
        clrscr();
        printf("Data stored \t\t sum\n");
        for (j=1; j<=3; j++)
        {
                int sum=0;
```

```
                    for (k=1; k<=4; k++)
                    {
                        printf("%d \t",num[j][k]);
                        sum=sum+num[j]+[k];
                    }
                        printf("=%d',sum);
                        printf('\n");
                    }
            }
```

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

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

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

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

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

<u>String Function:</u>

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

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

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

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

2. strrev( ): It helps to reverse the character of the string.
   Syntax:strrev(string);
3. strupr( ): It converts lowercase letters in string to uppercase.
   Syntax:strupr(string);
4. strlwr( ): It converts uppercase letters in string to lowercase.
   Syntax:strlwr(string);
5. strcpy( ): It is used to copy the content of one string to another.
   Syntax:strcpy(target,source);
6. strcat( ): It is used to concatenate source string to the target string.
   Syntax:strcat(target,source); or strcat(source,target);
7. strcmp( ): It compares two strings on following basis.
   Syntax:strcmp(string1,string2);

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

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

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

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

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

```
# include<stdio.h>
# include<conio.h>
# include<string.h>
void main( )
{
        clrscr( );
        int a;
        char string1[20];
        char string2[20];
        printf("Enter a word");
        scanf("%s",string1);
        strcpy(string2,string1);
        a=strcmp(string2, strrev(string2));
        if (a==0)
        {
                printf("palindrome");
        }
        else
        {
                printf("Not palindrome");
        }
}
```

Function:

Functions are the building block of statements, which takes some data, manipulate them and can return a value. Bug free function can be used repeatedly from other part of the program. There are two types of functions:

1. ***Library Functions (built in or ready made):*** C has the facilities to provide library function for performing some operation. These functions are present in the c library and they are predefined. for eg.
    a. scanf( );
    b. printf( );
    c. getchar( );
    d. putchar( );
2. ***User Defined functions (Defined by user according to need):*** user can creates their own function for performing this type of a function are called user define function.

Syntax of user defined functions

```
# include<stdio.h>
# include<conio.h>
void main( )
{
function ( );
.............
............
}
```

```
function ( )
{
body of the function
}
```

WAP to calculate simple interest using function

```
#include
float interest(float); //function declaration
int main()
{   float si;
    si=interest(); //function call
    printf("Simple interst is %.2f\n",si);

}
float interest() //function definition
{
    float p,t,r,i;
    printf("Enter Principal, Time and Rate");
    scanf("%f%f%f",&p,&t,&r);
    i=(p*t*r)/100;
    return i; //function return value
}
```

WAP to calculate area of rectangle using function.

```
#include
int area (void);
void main()
{
int a;
          a = area();
printf("area is %d",a);

}
int area()
{
int l,b,ar;
          printf("Enter length and breadth");
          scanf("%d%d",&l,&b);
ar = l*b
          return ar;
}
```

Advantage:

1. Big programs can be divided into smaller module using functions.
2. Program development will be faster.
3. Program debugging will be easier and faster.
4. Use of functions reduce program complexity.
5. Program length can be reduced through code reusability.
6. Use of functions enhance program readability.
7. Several developer can work on a single project.
8. Functions are used to create own header file i.e mero.h
9. Functions can be independently tested.

With passing arguments

Program: Write a program to find out the area of circle through given radius as argument using function.

```
# include<stdio.h>
# include<conio.h>
        float area(float);
        void main( )
        {
        clrscr();
        float rad,res;
        printf("enter the radius of a circle");
        scanf("%f",rad)
        res=area(rad)
        printf("your required area is %f ",res);
        }
        float area(float a)
        {
        float mm,pi=3.14;
        mm=pi*a*a;
        return(mm);
        }
```

Program: Write a program in C to create a function to pass two numbers as an argument and to return a sum to the calling function.

```
# include<stdio.h>
# include<conio.h>
        int add(int,int);
        void main( );
        {
                clrscr();
                int a,b,c;
                printf("enter any two numbers");
                scanf("%d%d",&a,&b);
                c=add(a,b);
                printf("The sum of a and b is= %d",c);
                }

                int add(int x, int y)
                {
                int sum;
                sum=x+y;
                return(sum);
                }
```

Without passing arguments

Program: Write a program to find out sum and square of two input number without passing arguments function.

```
# include<stdio.h>
# include<conio.h>
        int sum( );
        int square( );
        void main()
        {
          sum( );
         square( );
        }
        int sum( )
        {
                int a,b, sum;
                printf("enter any two number");
                scanf("%d%d",&a,&b);
                sum=a+b;
                printf("sum is=%d',sum);
                return(sum);
        }
         int square( )
        {
                int a,b, square;
                printf("enter any number");
                scanf("%d%d",&a);
                square=a*a;
                printf("square is=%d',square);
                return(square);
        }
```

Recursive Function:

The function which performs recursion is called recursive function. Recursion is a process by which a function calls itself repeatedly until some specified condition has been satisfied.

Those function which calls itself is known as recursive function and the concept of using recursive functions to repeat the execution of statements as per the requirement is known as recursion. The criteria for recursive functions are:

1. The function should call itself.
2. There should be terminating condition so that function calling will not be for infinite number of time.

Program: ***Write a program to calculate factorials by using recursion process.***

```
#include<stdio.h>
#include<conio.h>
int fact(int);
void main( )
{
        clrscr();
        int n;
        printf("Enter a number");
        scanf("%d",&n);
        printf("The factorial of %d is %d",n,fact(n));
}
int fact(int n)
{
        if(n<=1)
        return (1)
        else
         {
        return(n*fact(n-1));  or              z=(n*fact(n-1));
          }                                   return z;
}
```

Program: ***Write a program read a number and make the sum of individual digits & print using recursion technique.***

```
#include<stdio.h>
#include<conio.h>
 int sum(int)
void main()
{
        clrscr();
        int n;
        printf("Enter the any number");
        scanf("%d",&n);
        printf("The sum of individual digits is %d",sum);
int sum(int n)
{
        int a;
        if(n>0)
        {
        a=n%10;
        n=n/10;
        ruturn(a+sum(n));
        }
}
```

WAP to calculate sum of n-natural number using recursion/recursive function.

```
#include<stdio.h>
int sum (int);
void main()
{
   int n,s;
   printf("Enter any number");
   scanf("%d",&n);
   s = sum(n);
   printf("Sum is %d\n",s);

}
int sum (int n)
{
   if (n<=0)
      return 0;
   else
      return (n+sum(n-1));
}
```

Accessing a Function:

There are two types of accessing a function.

1. ***Call by value:*** Only the values of arguments are same to the function and any change made to the formal arguments do not change the actual arguments.

Program: *Write a C program try to exchange two values by using call by value accessing function.*

```
        # include<stdio.h>
        # include<conio.h>
        int swap (int, int)
        void main( )
        {
                int a,b;
                a=10
                b=20
                printf("Value of a=%d and b=%d",a,b);
                swap (a,b);
                printf("Value of a=%d and b=%d",a,b);
        }
int swap(int x, int y)
{
        int r;
        r=x;
        x=y;
        y=r;
}
```

2. ***Call by reference:*** When we pass address to a function the parameters receiving the address should be pointer. The process of calling a function by using pointer to pass the address of the variable is known as call by reference.

Program: *Write a C program to exchange two values by using call by reference accessing function.*

```
        # include<stdio.h>
        # include<conio.h>
        int swap(int *, int *)
        void main( )
        {
                int a,b;
                a=10
                b=20
                printf("Value of a=%d and b=%d",a,b);
                swap(&a,&b);
                printf("Value of a=%d and b=%d",a,b);
        }

    int swap(int *x, int *y)
    {
        int r;
        r=*x;
        *x=*y;
        *y=r;
    }
```

Structure and Union

Structure:

Structure is a collection of variables under a single name. As an example, if we want to store data with multiple data types like roll number, student name, address, phone number and class of a student then C supports structure which allows us to wrap one or more variables with different data types. Each variable in structure is called a structure member. To define a structure, we use struct keyword.

Syntax:
```
        struct  struct_name
        {
                Structure_members(s);
        };
```

Structure Declaration

```
    struct  student
    {
        int  roll;
        char  name[50];
        int  age;
        char  section;
        float  height;
    };
```

We can also declare a structure

```
        struct  student
        {
                int  roll;
                char  name[50];
                int  age;
                char  section;
                float  height;
        } s1,s2;
```

We can also declare array of variables at a time of declaring a structure as:

```
        struct  student
        {
                int  roll;
                char  name[50];
                int  age;
                char  section;
                float  height;
        } s[100];
```

Union:

Unions like structure contain members whose individual data types may differ from one another. However the member that composes a union all share the same storage area within the computer memory were as each member within a structure is assigned its own unique storage area.

Syntax

```
Union  union_name
{
Union_member (s);
}
```

S.N.	Structure	S.N.	Union
1	Structure is designed by using 'struct' keyword.	1	Union is designed by using 'union' keyword.

2	*Syntax:* struct structure_name
	{
	Data_type member1;
	Data_type member1;
	}
		2	*Syntax:* union union_name
			{
			Data_type member1;
			Data_type member1;
			}

3	All the member of the structure variable can be processed at a given time.	3	Only one member of the union variable can be processed at a time because only one member of the union variable can be active at a time.
4	We use structure variable if memory is large and have to store values all of the variables.	4	We use union variables if memory is less and have to store one variable in one of the declared variables or members.
5	Structures are broadly used in programming.	5	Unions are not used broadly as much as structures.
6	Structure declaration takes large amount of spaces to store data and values.	6	Union declaration shares the same area of memory to save storage space.

Example 1, Write a C program to read different structure variables and display them.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
        struct   student
        {
                int roll;
                char name[30];
                char section;
                float height;
        } s1
        printf("\n Enter roll numer");
        scanf("%d",&s1.roll );
        printf("\n Enter name:");
        gets(s1.name);
        printf("\n Enter section:");
        scanf("%c",&s1.section);
        printf("\n Enter height");
        scanf("%f,&s1.height);
        printf("\n You have entered:\n);
        printf(" Roll Numer %d Name is %s section %c and height is %f",
        s1.roll,s1.name,s1.section,s.height);
}
```

Program: *Write a program to input the employee name and their basic salary of n employees and display the record in proper format.*

```
#include<stdio.h>
#include<conio.h>
void main ()
{
        struct employee
        {
                char name[10];
                int salary;
        };
struct employee e[100],t;
int i,j,n;
printf("\n How many records");
scanf("%d", &n);
for(i=0; i<n; i++)
{
        printf("\n Enter name of employee");
        gets(e[i].name);
```

```
                printf("\n Enter salary of employee:");
                scanf("%d",&e[i].salary);
        }
        printf("\n Employee \t \t salary);
        printf("\n ==============================");
        for(i=0; i<n; i++)
        {
                printf("\n %s \t\t\t %d",e[i].name,e[i].salary);
        }
        printf("\n===============================");
}
```

Program: *Write a program that reads different names and addresses into the computer and rearrange the names into alphabetical order using the structure variables.*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main ()
{
int i,j,n;
        struct student
        {
                char name[50];
                char address[50];
        };
        struct student s[200],t;
        printf("\n how many records do you want store?");
        scanf("%d",&n);
        for (i=0; i<n; i++)
        {
                        printf("\n Enter name of the student:");
                        gets(s[i].name);
                        printf("\n Enter address of the student:");
                        gets(s[i].address);
        {
        for (i=0; i<n; i++)
        {
                for (j=0; j<n-1; j++)
                {
                        If(strcmpi (s[j].name,s[j+1].name)>0)
                        {
                                t=s[j];
                                s[j]=s[j+1];
                                s[j+1]=t;
                        }
                }
        }
        Printf("\n\n The records of the student after sorting\n");
        Printf("\n Name of student           Address        ");
        for (i=0; i<=n; i++)
        {
                printf("\n %s %s", s[i].name,s[i].address);
        }
}
```

Pointer

A pointer is a variable that a points to a references a memory location where data is stored. Each memory cell in the computer has an address which can be used to access its location. A pointer variable points to a memory location rather than a value.

- We can access and change the contents of the memory location.
- A pointer variable contains the memory location of another variable.
- The asterisk tells the compiler that you are creating a pointer variable.

The pointer declaration syntax is as shown below.

Pointer_type *pointer_variable_name;

For e.g. int *p;

Address (&) and indirection (*) operator

The address (&) operator and immediately preceding variable returns the address of the variable associated with it. Hence, the address of (&) operator returns the address of memory location in which the variable is stored.

The indirection (*) operator is used to extract value from the memory location stored in the particular memory location whose address is stored in pointer variable.

The syntax of declaring address operator to pointer variable is as follows.

Pointer_variable = &variable_name;

For Example

```
int   *ptr, num=25;
ptr      = &num;
```

Program: Write a complete program to display address and value of the pointer variable.

```
#include<stdio.h>
void main( )
{
        int *p;
        int age=17;
        p=&age;
        printf("\n Value of age is %d", age);           output= 17
        printf("\n Value of age is %d",*p);         output=17
        printf("\n Value of age is %d",*(&age));    output=17
        printf("\n Address of age is %u",p);           output=2686784
        printf("\n Address of age is %u",&age);     output=2686784
}
```

Pointer to Arithmetic

```
        #include<stdio.h>
        #include<conio.h>
        void main( )
        {
                int *p;
                int age= 17
                p=&age;
                printf(" \nValue of age is %d",age);                     output=17
                printf("\n Increment on age is %d",++age);               output=18
                printf("\n Address of age is %u",p);                     output=2686784
                printf("\n Increment in pointer is %u",++p);             output=
        }
```

Write a C program to increment pointer.

```
        #include<stdio.h>
        #include<conio.h>
        void main( )
        {
                int n[ ]={10,20,30,40};
                int *aptr=n;
                int i;
                printf("\n Pointer notation");
                for (i=1; i<=4; i++);
                {
                        printf("*(aptr+%d)=%d\n",i*(aptr+i));           output  =10
                }                                                               =20
        }                                                                       =30
                                                                                =40
```

Assignment pointer value to function

Program: Write a program to pass pointer variable to function sum them and display after returning it.

```
                #include<stdio.h>
                #include<conio.h>
                Int add(int *p, int *k)
                void main( )
                {
                        int a,b,c=0;
                        printf("\n Enter two numbers");
                        scanf("%d %d",&a,&b);
                        c=add(&a,&b);
                        printf("\n Sum of two numbers %d and %d is %d",a,b,c);
                }
```

```
add (int   *x, int *y)
        {
                int r;
                r=*x+*y;
                return (r);
        }
```

Array of pointers:

Program: Write a C program to assign array, place these array in pointer variable and display array value along with its address.

```
#include<stdio.h>
void main( )
{
        int i, arr[10];
        Int *ptr[10];
        for (i=1; i<=10; i++)
        {
                Arr[i]=i;
        }
        for (i=1; i<=10; i<=10; i++)
        {
                ptr[i]=&arr[i];
        }
        for (i=0; i<=10; i++)
        {
                printf("\n value arr[%d] stored at address of %u",arr[i],ptr[i]);
        }
}
```

Pointer to Pointer

Program: Write a program to demonstrate the use of pointer to pointers:

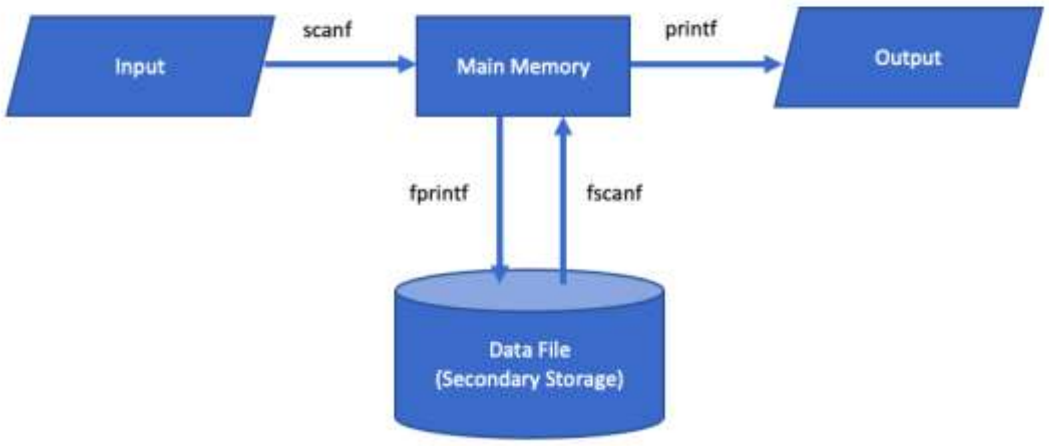
```
#include<stdio.h>
Void main( )
{
        int a=10, *b, **c;
        b=&a;
        c=&b;
        printf("\n Value of a is %d",a);                    output= 10
        printf("\n Value of a is %d",*b);                   output= 10
        printf("\n Value of a is %d",**c);                  output= 10
        printf("\n address of a is %u",b);                  output= 2686788
        printf("\n address of a is %u",&a);                 output= 2686788
        printf("\n address of b is %u",c);                  output= 2686784
        printf("\n address of b is%u", &b);                 output= 2686784
}
```

Advantage of Pointer

- Runtime memory creation.
- Runtime memory deletion.
- Hard Access through pointer.
- Data Structure Based Pointer.
- Applications of Pointer Ms-word, Excel, Access, SQL.

Working with files

Data file much application required that information be written is stored on the memory device in the form of a data file. Thus, data file access and alter that information whenever information in C.



Sequential and Random Access File

Sequential Access File:

The sequential access files allow reading the data from the file in one after another i.e. in sequence. There is no predefined order for accessing data file. All the processes are declare and assigned by the compiler during run time of the program.

Random Access File:

The random access files allow reading data from any location in the file. Sometimes, we need to read data file from reverse, middle and from specific location. To achieve this, C defines a set of functions to manipulate the position of the file. The inbuilt function fseek( ), lseek( ), rewind( ) and ftell( ) are the some of the common examples of random access files.

Opening, Reading, Writing and Appending on/from Data File:

Once the file pointer has been declared, the next step is to open file. There is an inbuilt function to open a file. The function fopen( ) is used to create a steam for use and links of new open file. This function return a file pointer and takes two parameter one for name of file and other for mode for the file. The syntax is as follows;

```
FILE    *f;
f = fopen ("file_name.extension",  "mode_of_open");
```

The modes of the data file are as follows:

S.N.	***Mode***	***Description***
1	"r" / "rt"	It opens a text file to read only mode.
2	"w" / "wt"	It creates a text file to write only mode.
3	"a" / "at"	It appends text to already data containing file.
4	"r+t"	It opens a text file for read and write mode.
5	"w+t"	It creates a text file for read and write mode.
6	"a+t"	It opens or creates a text file and read mode.
7	"rb"	It opens a binary file for read only mode.
8	"wb"	It creates a binary file for write only mode.
9	"ab"	It opens or create a binary file for append mode.
10	"r+b"	It opens a binary file for read and write mode.
11	"w+b"	It creates a binary file for read and write mode.
12	"a+b"	It opens or creates a binary file for append mode.

Functions

1. fputc= Store character into the file.
2. fputs= Store string in to the file.
3. fgetc= fetch character from the file.
4. fgets= fetch string from the file.
5. fwrite= store data (structure) in to the file.
6. fread= fetch data (structure) from the file.
7. fprintf= store data in to file.
8. fscanf= fetch variable from the file.

Opening a data file

Syntax:
FILE *fptr
fptr = fopen (“filename” , ”mode”)
Where, File name can be “library.txt”, “student.dat” ..etc
Mode:
“w” to write/store data in a data file.
“r” to display/read/retrieve/access data from a datafile.
“a” to add/append data in existing datafile.

Store/write data

Syntax:
fprintf(fptr , ”format specifiers” ,variables);
Eg; suppose if we want to store name, disease, age and bed number of a patient then, it is written as

```
fprintf(fptr , "%s %s %d %d", n, d, a, b);
```

Where, variable are initialized as:

```
char n[10], d[10];
int a, b;
```

Program example

1) Create a datafile "patient.txt" and store name, disease, age and bed number of a patient.

```
#include<stdio.h>
void main()
{
   char n[10], d[10];
   int a, b;
   FILE *fptr;
   fptr = fopen("patient.txt","w");
   printf("Enter name, disease, age and bed number");
   scanf("%s %s %d %d", n, d, &a, &b);
   fprintf(fptr,"%s %s %d %d\n", n, d, a, b);
   fclose(fptr);
}
```

[Note: This program will store only single record to store multiple record we have to use loop as following programs.

2) Create a datafile "student.txt" and store name, class and marks obtained in 3 different subject for few students/n-students.

```
#include<stdio.h>
void main()
{
   char n[10];
   int c, e, ne, m, i, num;
   FILE *fptr;
   fptr = fopen("student.txt","w");
   printf("How many record?");
   scanf("%d",&num);
   for(i=1;i<=num;i++)
   {
   printf("Enter name class and 3 marks");
   scanf("%s %d %d %d %d",n, &c, &e, &ne, &m);
   fprintf(fptr,"%s %d %d %d %d \n",n, c, e, ne, m);
   }
   fclose(fptr);
}
```

3) Create a datafile "student.txt" and store name, class and marks obtained in 3 different subject until user press "y" / as per user requirement.

```
#include<stdio.h>
int main()
{
   char n[10],ch[3];
   int c, e, ne, m;
   FILE *fptr;
   fptr = fopen("student.txt","w");
```

```
  do
  {
  printf("Enter name class and 3 marks");
  scanf("%s %d %d %d %d",n, &c, &e, &ne, &m);
  fprintf(fptr,"%s %d %d %d %d\n",n, c, e, ne, m);
  printf("Press Y to continue");
  scanf("%s",ch);
  } while (strcmp(ch,"Y") == 0 || strcmp(ch,"y")==0);
  fclose(fptr);
 }
```

Add/Append data

1) A datafile “student.txt” contain name, class and marks obtained in 3 different subject of few students. Write a C program to add 200 more records.

```
#include<stdio.h>
 void main()
{
   char n[10];
   int c, e, ne, m, i;
   FILE *fptr;
   fptr = fopen("student.txt","a");
   for(i=1;i<=200;i++)
   {
   printf("Enter name class and 3 marks");
   scanf("%s %d %d %d %d", n, &c, &e, &ne, &m);
   fprintf(fptr,"%s %d %d %d %d \n",n, c, e, ne, m);
   }
   fclose(fptr);
 }
```

2) A datafile “student.txt” contain name, class and marks obtained in 3 different subject of few students. Write a C program to add more records until user press “y” / as per user requirement.

```
#include<stdio.h>
  void main()
{
   char n[10], ch[3];
   int c, e, ne, m;
   FILE *fptr;
   fptr = fopen("student.txt”,”a”);
   do
   {
   printf("Enter name class and 3 marks");
   scanf("%s %d %d %d %d", n, &c, &e, &ne, &m);
   fprintf(fptr,"%s %d %d %d %d\n", n, c, e, ne, m);
   printf("Press Y to continue");
   scanf("%s",ch);
   } while (strcmp(ch,"Y") == 0 || strcmp(ch,"y")==0);
   fclose(fptr);
   }
```

Read/Display/retrieve/access data from a datafile

Syntax:

fscanf(fptr , ”format specifiers” ,variables);

Eg; suppose if we want to display/read name, disease, age and bed number of a patient from data file then, it is written as

```
fscanf(fptr , “%s %s %d %d”, n, d, &a, &b);
```

Where, variable are initialized as:

```
char n[10], d[10];
int a,b;
```

EOF: End of file

1) A d#include <stdio.h>

```
void main()
{
   char n[10];
   int c, e, ne, m;
   FILE *fptr;
   fptr = fopen("student.txt","r");
   printf("Name\tPercentage\n");
   while(fscanf(fptr,"%s %d %d %d %d",n,&c,&e,&ne,&m) != EOF)
   {
      printf("%s %d  %d  %d  %d", n, c, e, ne, m);
   }
   fclose(fptr);
}
```

2) A datafile “student.txt” contain name, class and marks obtained in 3 different subject of few students. Write a C program to read and display only records whose name is Ram.

```
#include<stdio.h>
void main()
{
   char n[10];
   int c, e, ne, m;
   FILE *fptr;
   fptr = fopen("student.txt","r");
   while(fscanf(fptr,"%s %d %d %d %d",n,&c,&e,&ne,&m) != EOF)
   {
     strlwr(n);
     if (strcmp(n,”ram”) == 0)
     {
      printf("%s %d  %d  %d  %d", n, c, e, ne, m);
```

```
            }
          }
          fclose(fptr);
        }
```

3) A datafile “student.txt” contain name, class and marks obtained in 3 different subject of few students. Write a C program to read and display only records who pass in all subjects.

```
#include<stdio.h>
void main()
{
   char n[10];
   int c, e, ne, m;
   FILE *fptr;
   fptr = fopen("student.txt","r");
   while(fscanf(fptr,"%s %d %d %d %d",n,&c,&e,&ne,&m) != EOF)
   {
     if (e>=40 && ne>=40 && m>=40)
      {
       printf("%s %d  %d  %d  %d", n, c, e, ne, m);
      }
   }
   fclose(fptr);
 }
```

4) A datafile “student.txt” contain name, class and marks obtained in 3 different subject of few students. Write a C program to read and display only records who fail in any one subject.

```
    #include<stdio.h>
    void main()
    {
       char n[10];
       int c, e, ne, m;
       FILE *fptr;
       fptr = fopen("student.txt","r");
       while(fscanf(fptr, "%s %d %d %d %d", n, &c, &e, &ne, &m) != EOF)
       {
         if (e<40 || ne<40 || m<40)
        {
          printf("%s %d  %d  %d  %d", n, c, e, ne, m);
        }
       }
       fclose(fptr);
    }
```

5) A datafile “student.txt” contain name, class and marks obtained in 3 different subject of few students. Write a C program to read and display only name and percentage of all students

```
#include<stdio.h>
int main()
{
   char n[10];
   int c, e, ne, m;
   float p;
   FILE *fptr;
   fptr = fopen("student.txt","r");
   while(fscanf(fptr, "%s %d %d %d %d", n, &c, &e, &ne, &m) != EOF)
   {
      p = (e+ne+m)/3;
      printf("%s %f", n, p);
   }
   fclose(fptr);
 }
```

6) A datafile “student.txt” contain name, class and marks obtained in 3 different subject of few students. Write a C program to read and display only records of all students who secure distinction.

```
#include<stdio.h>
int main()
{
   char n[10];
   int c, e, ne, m;
   float p;
   FILE *fptr;
   fptr = fopen("student.txt","r");
   while(fscanf(fptr, "%s %d %d %d %d", n, &c, &e, &ne, &m) != EOF)
   {
     p = (e+ne+m)/3;
     if (p>=80)
     {
      printf("%s %d  %d  %d  %d", n, c, e, ne, m);
     }
   }
   fclose(fptr);
 }
```

Program: ***Write a program to read a line and store it in a data file and display the contents.***

```
#include<stdio.h.
#include<conio.h>
void main( )
char I;
FILE   *f,      *q;
```

```
                        f = fopen("store.txt","w");
                        while((i=getchar())!='\n')
                        {
                                fputc(i, f);
                        }
                        fclose( f);
                        q=fopen ("store.txt","r");
                        while((i=fgetc(q))!=EOF)
                        {
                        Printf("%c",i);
                        }
                }
```

Program: ***Write a C program to read N students record store them in data file and display the content in appropriate format by using fprintf and fscanf function.***

```
#include<stdio.h>
void main( )
{
  FILE   *f;
  f= fopen (''ratna.txt","w");
  int   i, rn,  n;
  char name[25],     add[35];
  printf("\n how many records?");
  scanf("%d", &n);
  printf("Enter %d   student   roll number,  name  and  address:\n",n);
  for (i=o; i<n; i++)
  {
          scanf("%d %s  %s",&rn,name,add);
          fprintf( f, "%d\t%s\t%s\n ",rn,name,add);
  }
  fclose(f);
  printf("\n The %d data records",n);
  f= fopen ("ratna.txt","r");
  while (fscanf( f, "%d  %s   %s",&rn,name,add)!=EOF)
  {
          printf("\n %d\t%s\t%s",rn,name,add);
  }
  fclose(f);
}
```

Program: Write a program using C language that reads successive records from the new data file and display each record on the screen in an appropriate format.

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

```
{
        struct student.
        {
                char name[50];
                char add[80];
        };
        struct student    s;
        char next='Y' ;
        FILE   *fp;
        fp = fopen ("C:\\student.txt","w");
        while(next=='Y' || next=='y')
        {
                printf("\n Enter the name of the student");
                gets(s.name);
                printf("\n Enter the address of the student");
                gets(s.add);
                fwrite(&s, sizeof (s), 1, fp);
                printf("\n Do you want to write next record (Y/N)");
                                }
                                fclose(fp);
                                fp=fopen ("C:\\student.txt", "r");
                                printf("\n Name of student Address");
                                while (fread (&s, sizeof(s), 1,f(p)==1)
                                {
                                        printf("\n %s \t\t\t %s",s.name,s.add);
                                }
                                fclose (fp);
```

Program: ***Write a C program to read sentence until enter key pressed. Put every words in data file by using fputs( ) fucnction and display the content of file by using fgets( ) function.***

```
#include<stdio.h>
#include<conio.h>
Void main( )
{
        Char a[20];
        FILE   *f;
        f= fopen ("senten.doc", "w");
        printf("\n Enter sentence");
        gets(a)
        while (strlen (gets(a))>0)
        {
                fputs(a,f);
        }
```

```
                    fclose(f);
                    f = fopen ("senten.doc", "r");
                    if (f== NULL)
                    printf("\n Cannot open the file");
                    else
                    {
                            While (fgets (a,19,f)!=NULL)
                            {
                                    printf("%s",a);
                            }
                    }
                    fclose(f);
            }
```

Program: ***Write a C program to read ages of N students, store them in age.txt file and find out the average age after reading from data file.***

```
        #include<stdio.h>
        #include<conio.h>
        Void main( )
        {
                int ag,n, i, sum=0;
                float avg;
                FILE *f;
                f= fopen("age.txt", "w");
                printf("\n How many student:");
                scanf("%d",&n);
                for(i=0; i<n; i++)
                {
                        printf("\n Enter %d student ages",i+1);
                        scanf("%d",&ag);
                        putw(ag,f);
                }
                fclose (f);
                f= fopen ("age.txt", "r");
                printf("\n Calculating average age from file");
                for (i=0; i<n; i++)
                {
                        sum=sum+getw(f);
                }
                avg = (float) sum / n ;
                printf("\n Average age of %d student is %f", n, avg);
                fclose(f);
        }
```

Program: *Write C program to read already created data file student.txt and display it contents if it opens otherwise print message "The file ……does not exists".*

```
#include<stdio.h>
#include<conio.h.
Void main( )
{
        char i;
        FILE   *f;
        f = fopen ("student.txt", "r");
        If (f== NULL)
        {
                printf("\n The student.txt file does not exists");
        }
        else
        {
                while ( ( i=fgetc(f))!=EOF)
                {
                        printf("%c",i);
                }
        }
        fclose(f);
```

Some IMP Notes

S.N.	Fwrite	S.N.	fprintf
1	Binary format writing	1	No binary format.
2	Size of data.	2	Size of data is not fix
3	Number of records.	3	No number of records.
4	We store the data in block form so not in format form.	4	Format string used to store the data in the format form.
5	Fixed block size.	5	No block size.

S.N.	Structure	S.N.	Pointer
1	Structure is use to store the difference type of data member like as char, int, float etc.	1	Where is pointer is use to store the address of another location (variable).
2	Structure is use to maintain the records.	2	Whereas pointer is use to create link list.
3	Structure can't be use to access the hardware.	3	Pointer can be use to access the hardware.
4	Structure can't be use to create memory run time.	4	But pointer are use to create the memory runtime.
5	Structure can be declared as pointer variable.	5	Pointer can be declared inside the structure.

S.N.	Break Keywords (Statement)	S.N.	Continue Keywords (Statement)
1	The break statement is use to terminate the loop unconditionally.	1	The continue statement is used return the pointer at the beginning of the loop.
2	The break statement can also be use inside the switch statement.	2	But the continue can be use inside the switch statement.
3	The break statement is use to terminate the loop.	3	The continue statement is use to repeat set of statements.

4	Example
	`void main( )`
	`{`
	`int i;`
	`for (i=1; i<=10; i++)`
	`{`
	`If (i==5)`
	`{`
	`break;`
	`}`
	`printf("%d",i);`
	`}`
	`}`
	Output= 1,2,3,4.
		4	Example
			`void main( )`
			`{`
			`int num;`
			`for (num=1; num!=10; num+1)`
			`{`
			`If(num==7)`
			`{`
			`continue;`
			`}`
			`printf("%d",num);`
			`}`
			`}`
			Output= 1,2,3,4,5,6,8,9.

S.N.	Array	S.N.	Pointer
1	Array is create continue memory location with same type.	1	Pointer creates only one block of memory but access all memory of another variable.
2	Array takes huge of memory.	2	Pointer takes less and less memory.
3	Array is to be fixed at the time of declaration.	3	Whereas pointer size can be change at the run time.
4	In array we can free the memory run time.	4	But using pointer the dynamic allocated memory can be make free at run time.
5	If we allocated huge sized of array and we get few information then unnecessary memory will be occupied.	5	But this problem can resolved by using the pointer.
6	Using array technique to accessing the array elements is more different comparison to pointer.	6	It is easier than array.

Web References:

3. https://www.javatpoint.com
4. https://www.w3schools.com
5. https://www.tutorialspoint.com
6. https://www.google.com
7. https://www.wikipedia.org