Module-2

Overview of C Programming:-

Ans :- (1)

History and Evolution of C Programming:-

→C language is developed by Dennis Ritchie in 1972.

→C is Machine Independent and structured both type of language.

→C is Very Fast, general purpose, High Level, Case-Sensitive and Popular Programming language. The main reason of popular is Fundamental language.

→C language is used to developed game, operating system, application and database.

Importance and why it is still used in Today:-

→C language is very quickly because C is compiled language.

→C language is Procedural language that means step by step instruction are follow.

→C is fast and efficient as compared other Programming language.

→Many Applications that required real-time processing or interact hardware still depends on C language.

→The continued growth of IOT devices and embedded systems that C language remains the language of choice for hardware.

Ans :- (2)

Step to install DevC++:-

1. Open your Web Browser go to DevC++ official download page and click on **Download** button.
2. Then, the download is complete; Double-click the file to start the installation process.
3. Select your language for the installation and click**Ok**.
4. Then Click on **Next** button on the welcome page.
5. Select folder where you want to install DevC++ and Click Next button.
6. Begin the installation process It may take a few minutes. DevC++ is now installed and ready to use on your Windows system.

Ans :- (3)

Basic Structure of C:-

#includes<stdio.h>:- #includes is used to include header files. <stdio.h> is used to standard input output.

main ():- main () is entry point of the program.

{

} is the block of code.

Structure:-s

```
#include<stdio.h>

Main ()

{
```

printf (“Hello World”);

}

➔ Printf is function it is used to display information.

Comments:-

→Comments are not executes the program it is help to the explanation code.
→Comment is write // in C language.

For Example…

```
#include<stdio.h>
Main ()
{
        Printf (“Welcome to Tops”); //use printf function
}
```

Data-Type:-

Define a specify the type of data. There are many data type is available in c language.

1. Int: - Integer data type is used to display whole number. For Example: - 123, 45678,12,3 …

2. Float: - Float is used to display floating pint value. For Example:- 124.5, 13.02,3008.02…..

3. Char: - Character is used to display single character. For Example: - a, b, c….

Variables:-

Variable is used to store value.

For Example:-

```
#include<stdio.h>

Main ()

{

Int a=2; // a is variable

Printf (“%d”, a);

}
```

OUTPUT:-

2

Ans :- (4)

Operator:-

Arithmetic Operators

Operator	Description	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
%	Remainder	x % y
/	Division	x / y

Relational Operator

Operator	Description	Example
<	Less than	a< b
>	Grater then	a>b
==	Equal to	a == b
!=	Not equal to	a != b
<=	Less-than equal to	a <= b
>=	Greater than equal to	a>= b

Increment and Decrement Operators

Operator	Description	Example
++	Increment	d++ , ++d
--	Decrement	e--, --e

Logical Operators

Operator	Description	Example
&	And	o & p
||	Or	o || p
!	Not	! p

Assignment Operators

Operator	Description	Example
=	Assignment	a = b
+=	Add assignment	a += b
-=	Subtract assignment	a -= b
*=	Multiply assignment	a *= b
/=	Divide assignment	a /= b
%=	Modulus assignment	a %= b

Ans :- (5)

Control Flow Statement:-

1. if Statement

The if statement is used to check a condition. If the condition is true, the block of code is executed.

For Example..

```
#include <stdio.h>
Main ()
 {
   Int a = 5;
   If (a > 0)
{
Printf ("a is positive.\n");
   }
```

OUTPUT:-

A is positive.

2.if-else Statement

The if-else statement is used to if is not true. It has two blocks: one for when the condition is true and one for when the condition is false.

For Example…

```
#include <stdio.h>
Main ()
{
   Int a = -3;
   If (a > 0)
{
Printf ("a is positive.\n");
   }
else
{
printf("a is not positive.\n");
   }
}
```

OUTPUT:-

a is not positive.

4.Nested if-else Statements

A nested if-else statement is an if statement inside another if or else block. This is useful to check multiple conditions within the same block.

For Example..

```
#include <stdio.h>
int main()
{
    int a = 10, b = 5;
    if (a > 0)
    {
        if (b > 0)
        {
            printf("Both a and b are positive.\n");
        }
        else
        {
            printf("a is positive but b is not positive.\n");
        }
    }
    else
    {
        printf("a is not positive.\n");
    }
}
```

OUTPUT:-

Both a and b are positive

5.switch Statement

The switch statement is used to select one of many blocks to be executed based on a specific value.

For Example..

```
#include <stdio.h>

main() {
   int day = 3;


   switch (day) {
      case 1:
         printf("Monday\n");
         break;
      case 2:
         printf("Tuesday\n");
         break;
      case 3:
         printf("Wednesday\n");
         break;
```

```
        case 4:
            printf("Thursday\n");
            break;
        case 5:
            printf("Friday\n");
            break;
        case 6:
            printf("Saturday\n");
            break;
        case 7:
            printf("Sunday\n");
            break;
        default:
printf("Invalid day\n");
}
}
```

OUTPUT:-

Wednesday

Ans :- (6)

Loops:-

Loops is a way to repeat a set of instructions multiple times without having to write the same code over and over.

Types of Loop:-

(1) Entry Control :-

1.While Loop:-

While loop is used to repeat a block of code as long as a condition is true.

Syntax:-

```
while (condition)
{
        // block of code be execute
}
```

Example :-

```
#include<stdio.h>
main()
{
        int a=1;
        while (a<=10)
        {
```

```
            printf(" %d ",a);
            a=a+1;
        }
}
```

OUTPUT:-

1 2 3 4 5 6 7 8 9 10

(2)For loop:-

For loop is used to repeat a block of code a specific number of times.

Syntax:-

```
        for (initialization; condition; update)
{
}
```

Example:-

```
        #include <stdio.h>
int main() {
  int i;
  for (i = 1; i < 11; i++) {
    printf("%d\n", i);
  }
}
```

OUTPUT:-

12 4 5 6 7 8 9 10

2.Exit Control loop:-

(1)Do While Loop:-While loop is used to execute block of code after check condition is true.

Syntax:-

```
do {
  // block of code executed
}
while (condition);
```

Example:-

```
#include <stdio.h>

int main() {
  int i = 0;


  do {
    printf("%d\n", i);
    i++;
  }
  while (i < 5);
}
```

OUTPUT :-

0 1 2 3 4

Ans :- (7)

Loop Control Statements:-

Break:-

Break statement is used to jump to the loops.

Example:-

```
#include<stdio.h>
main()
{
Int a;
While(a=1 ; a<10; a++)
    {
        If (if a==6)
        {
        Break;
}
    Printf(“ %d”,a);
}
}
```

OUTPUT:- 1 2 3 4 5

Continue:-

Continue statement is used to skip the current iteration of a loop and move directly to the next iteration.

Example:-

```
#include <stdio.h>

main()

{

  int i = 0;


  while (i < 10)

   {

    if (i == 4)

            {

      i++;

      continue;

    }

    printf("%d\n", i);

    i++;

  }

}
```

OUTPUT:-

0 1 2 3 5 6 7 8 9

GOTO:-

The go to statement is used to transfer control to another part of the program.

EXAMPLE:-

```
#include <stdio.h>

main ()

{

  START:

    printf("Hello World \n");

    printf("How are you? \n");

    goto START;

}
```

OUTPUT:-

Hello World

How Are You ?

Ans :- (8)

Function:-

Function is block of code which can only run when it is called.

EXAMPLE:-

```
#include <stdio.h>

int sum(int a, int b)

{

  return a + b;

}

main()

{

  int add = sum(10, 30);

  printf("Sum is: %d", add);

  return 0;

}
```

OUTPUT:-

```
Sum is: 40
```

Ans :- (9)

Array :-Array is a data structure that stores multiple use of the same data type in a single variable.

Difference between 1D and 2D:-

1D	2D
A 1d store single list of various elements having similar data-types.	A 2d store an array of various array, or list of various list.
It represents multiple data item in the form of a list.	It represents multiple data item in the form of table that contains row and column.
It has only one dimension	It has a total of two dimensions.
One can easy receive it in a pointer.	The parameter that receives it must define arrays.

Example of 1D:-

```
#include <stdio.h>

main()

{

   int arr[5] = {1, 2, 3, 4, 5};

   printf("Elements of the 1D array are:\n");

   for (int i = 0; i < 5; i++)

       {
```

```
        printf("%d ", arr[i]);
    }
}
```

OUTPUT:-

1 2 3 4 5

Example of 2D :-

```
#include <stdio.h>
main ()
{
    int arr[3][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
    };
    for (int i = 0; i < 3; i++)
{
        for (int j = 0; j < 4; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
```

```
    }
}
```

OUTPUT:-

```
1 2 3 4
5 6 7 8
9 10 11 12
```

Ans :- (10)

Pointer:-

A pointer is a variable that stores the memory address of another variable as its value.

→Pointer is Points to a data-type of the same type, and create with * operator.

Pointer Declaration:-

Data_type * Pointer_name;

Pointer Initialization:-

pointer_name = &variable;

Why Pointers are Important:-

1.Memory Management:-

→ Pointers allow direct manipulation of memory addresses. This means you can access and modify data stored at specific memory locations.

2..Efficient Data Handling:-

→When passing large structures or arrays to functions, pointers allow you to pass the memory address of the data instead of copying the entire data.

3. Optimization:-

→ By using pointers, you can optimize performance by reducing memory consumption and execution time.

4. Flexibility and Control:-

In contrast to languages that manage memory automatically (e.g., Java or Python), C gives you explicit control over memory allocation and deal location.

Example:-

```
#include <stdio.h>

main()

{

int myAge = 43;

  int* ptr = &myAge;
```

```
  printf("%p\n", ptr);
  printf("%d\n", *ptr);
}
```

OUTPUT:-

000000000062FE14

43

Ans :- (11)

String :-

→String is a group of character. String handling functions in C are used to manipulate and manage strings,

→Which are arrays of characters The standard C library <string.h> provides a set of functions that allow for string operations such as copying, concatenation, comparison, and searching.

String Function:-

1. strlen():-

This Function is used to calculates the length of a given string. it doesn't count the null value character.

Syntax:-

Int strlen(const char*str);

Example:-

```
#include <stdio.h>
main()
{
   char str[] = "Tops Career Center";
   size_t length = strlen(str);
   printf("String: %s\n", str);
   printf("Length: %zu\n", length);
}
```

OUTPUT:-

```
String: Tops Career Center
Length: 18
```

2. strcpy():-

it is a standard library function in C and is used to copy one string to another.
In C, it is present in **<string.h>** header file.

Syntax:-

```
Char * strcpy(char*dest,const char*src);
```

Example:-

```
#include <string.h>
main()
```

```
{
char source[] = "Welcome To Tops";
 char dest[20];

   strcpy(dest, source);
   printf("Source: %s\n", source);
   printf("Destination: %s\n", dest);
}
```

OUTPUT:-

```
Source: Welcome To Tops
Destination: Welcome To Tops
```

3.strcat():-

It is used for string concatenation. It will append a copy of the source string to the end of the destination string.

Syntax:-

```
Char*strcat(char * dest ,const char * src);
```

Example:-

```
#include <stdio.h>

main()

{

   char dest[50] = "This is an";

   char src[50] = " example";
```

```
    printf("dest Before: %s\n", dest);
    strcat(dest, src);
    printf("dest After: %s", dest);
}
```

OUTPUT:-

dest Before: This is an

dest After: This is an example

3. strcmp():-

The strcmp() is a built-in library function in C.

This function takes two strings as arguments and compares these two strings lexicographically.

Syntax:-

Int strcmp(const char * str1, const char * str2);

Example:-

```
#include <string.h>
main()
{
    char str1[] = "Welcome";
    char str2[] = "To";
    char str3[] = "Tops";
```

```
int result1 = strcmp(str1, str2);
int result2 = strcmp(str2, str3);
int result3 = strcmp(str1, str1);
printf("Comparison of str1 and str2: %d\n", result1);
printf("Comparison of str2 and str3: %d\n", result2);
printf("Comparison of str1 and str1: %d\n", result3)}
```

OUTPUT:-

```
Comparison of str1 and str2: 1
Comparison of str2 and str3: -1
Comparison of str1 and str1: 0
```

4. strchr():-

The strchr() function in C is used to locate the first occurrence of a character in a string.

Syntax:-

```
char *strchr(const char *str, int c);
```

Example:-

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

```
{
    const char *str = "Hello, world!";
    char ch = 'o';
    char *result = strchr(str, ch)
    if (result != NULL) {
        printf("Character '%c' found at position: %ld\n",
ch, result - str);
    } else {
        printf("Character '%c' not found in the string.\n",
ch);
    }
}
```

OUPUT:-

Character 'o' found at position: 4

Ans :- (12)

Concept of Structures in C:-

→a **structure** is a user-defined data type that allows grouping of different types of data under one name.

→It is used when we want to store multiple pieces of information (of different types) related to a single entity.

Declaring a Structure:-

Declare a structure, we use the struct keyword followed by the structure's name, and then the list of its members enclosed in curly braces {}.

Syntax:-

```
struct structure_name {
    data_type member1;
    data_type member2;
};
```

Initializing a Structure:-

Once a structure is declared, you can initialize its members in two ways.

1.At the time of declaration:-

You can initialize a structure at the time of declaration using an initializer list.

Syntax:-

```
struct structure_name variable_name = {value1, value2, ...};
```

2.After declaration:-

You can also initialize members individually after declaring a structure variable.

Example:-

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

```
struct Student {
    char name[50];
    int age;
    float marks;
};
main() {
    struct Student student1 = {"Mohit Patel", 20, 85.5};
    printf("Student Name: %s\n", student1.name);
    printf("Student Age: %d\n", student1.age);
    printf("Student Marks: %.2f\n", student1.marks);
}
```

OUTPUT:-

Student Name: Mohit Patel

Student Age: 20

Student Marks: 85.5

Ans :- (13)

importance of file handling:-

→ File handling in C allows programs to read from and write to files, which is essential for persistent data storage.

→ Without file handling, programs would only be able to operate on data in memory during their execution, which would be lost when the program ends.

1. Opening a File:-

Before performing any operations on a file, you need to **open** it. The fopen() function is used to open a file in a specific mode

Syntax:-

FILE *fopen(const char *filename, const char *mode);

Example:-

```
FILE *file = fopen("example.txt", "r");

if (file == NULL) {

    printf("Error opening file\n");

}
```

2. Writing from a File:-

To write to a file, the file must be opened in a mode that allows writing (like "w", "a", "w+", or "a+").

Syntax:-

```
fprintf(file, "Hello, %s!\n", "World");
```

Example:-

```
FILE *fptr;

fptr = fopen("filename.txt", "w");

fprintf(fptr, "Some text");

fclose(fptr);
```

3. Reading from a File:-

Once a file is open, data can be read using functions like fscanf, fgets, or fgetc. The choice depends on the format and type of data being read.

Example:-

```
FILE *fptr;

fptr = fopen("filename.txt", "r");

char myString[100];

fgets(myString, 100, fptr);

printf("%s", myString);
fclose(fptr);
```