Q .1 Write an essay covering the history and evolution of C programming. Explain its importance and why it is still used today.

- C programming language develop in 1972 by Dennis Ritchie at bell laboratories in usa.
- C is pop(Procedure Oriented Programming) language . procedure oriented programming is function, method and block of code.
- C is high-level programming language , human understable and easy to read ,write ,access to the uaser.
- This is a flexible , portability , efficiency operating system , embedded system and application.
- C language basic structure :

```
#include<stdio.h>
main()
{
printf("\n\n\t hello");
}
```

Q.2 Describe the steps to install a C compiler (e.g., GCC) and set up an Integrated Development Environment (IDE) like DevC++, VS Code, or CodeBlocks.

- 1.Google search the c compiler or IDE.

  DEV C++,VS CODE,CODEBLOCK etc…

- 2.choice IDE:

  Dev c++ : download and install the dev c++ valid link in browser.

- 3.configure in IDE :

  Dev c++ : install the dev c++ after the create file.

- 4.write a program and run :

Dev c++ : first program execute in dev c++. .c extension file save and run program.

Q.3 Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples.

- Basic structure in c:

```
#include<stdio.h>
main()
{
    printf("\n\n\t hello");
}
```

Program is set of instruction for the computer perform specific task. #include<stdio.h> is library,

: this is preprocessor , include : keyword in c, <stdio.h> : standard input output header file ,

Main() : function and start the program from here , { } : block of code within write the statement

and condition.

- Comments : comments is part of program but not display output. Comment is use explain code.

comment are two type :

1. Single line comments
2. Multiline comments

1. Single line comments : one line are use single line comment. use symbol(#) hastag.

2. Multiline comment : multi line are use multiline comment. Use symbol (*/____/*).

❖ Data type: data type is type of data and variable store.

Datatype is two type :

1.primitive datatype.

2.non-primitive datatype.

Primitive datatype : primitive datatype is provide language.

Ex : int , float , char, etc…

Non-primitive datatype : non-primitive datatype is provide developer.

Ex : string , array , structure etc…

Integer : declare %d store the positive value small range.

Float : declare %f store the decimal value small range.

Long integer : declare %d store the positive value high range.

double : declare %f store the decimal value high range.

Character : declare %c store the single character value.

String : declare %s store the group of character value.

- Variable : variable is one type of container that hold the value.element(memory) to store the particule value .

  Ex : a=10

  a=123456789

  a=34.56

  a=345678.7656

  a='g'

  a="tops"

example:

```
//value store
include<stdio.h>
Main()
{
Int a=100;
Float b=200;
Printf("\n store the value: %d",a);
Printf("\n store the value: %f",b);
}
```

Q.4 Write notes explaining each type of operator in C:

arithmetic,

relational,

logical,

assignment,

increment/decrement,

bitwise,

conditional operators.

- Operator : Operator is use in two oprants . operator is symbol perform the to get the mathamatics operation.

  Multiple operator in c language

- arithmetic operator: Addition , subtraction , multiplication , division, modulo etc. perform the mathematical operation use arithmetic operator.

  Ex :

addition	+
subtraction	-
multiplication	*
division	/
modulo	%

- relational operator : perform the two value comparison use relational operator.

Ex :

- logical operator : perform the two or more condition check use logical operator.

  Ex : &&(and) , ||(or) , !(not)

Double equal	==
Not equal	!=
Grater than	>
Less than	<
Less than equal	<=
Grater than equal	>=

And	&&
Or	||
Not	!

- assignment operator : perform the assign value use assignment operator.

  Ex: += , -+, *= , /=

a-=b	a=a-b
a*=b	a=a*b
a/=b	a=a/b
a+=b	a=a+b

- increment/decrement :increment operator value+1,decrement operator value-1

  Prefix : ++a , --a postfix : a++ , a--

- bitwise operator : ex : & , | , << , >>

Q.5 Explain decision-making statements in C (if, else, nested if-else, switch). Provide examples of each.

1. If statement : if statement use in one condition that condition only true condition check .

   Syntex : if(condition)

   {

   //statement

   }

Ex :

```
#inlcude<stdio.h>
main(){
Int a=10;
If(a>10){
printf(“hello”);
}
```

2. if_else statement : if_else is use to check the condition is true or false .

Syntex : if(condition)

```
{
    //statement
}
 else
{
    //statement
}
```

Ex:

```
#include<stdio.h>
Main(){
```

```
Int a=18;
If (a>18)
{
    Printf("eligible for voting .");
}
else
{
    Printf("you are not eligible for voting .");
}
```

3. nested if statement : check multiple condition in one time than used to nested if condition within if condition (if into if)

Syntex : if(condition)

```
{
If(condition)
{
        //statement
}
 else
{
        //statement
}
```

```
else
{
    //statement
}
```

Ex :

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

- switch statement :To make menu driven code.Never use relational op. by this. (n==1).switch can be used with only int, char, double data types.

  switch can be used with the keywords :

  switch, case, break, default.

  Syntex :

```
switch(choice)
{
Case 1:  // statement
Break;
Case 2 : //statement
Break;
Default : //statement
```

Q.6 . Compare and contrast while loops, for loops, and do-while loops. Explain the scenarios in which each loop is most appropriate.

→

- **While loop :** while(condition) { execute the code } .while loop condition is true loop is execute and condition is false loop is not execute. while loop is entry control loop, entry control loop is check condition before execute code. Exit the loop condition is false.

e.g :

While loop :

```
 I = 1;                  //(initialization)

While(I <= 5)              //condition
{
 printf (“%d”, i);   //execute code
 I++;                        // increment/decrement
 }
```

- **For loop :** for(initialization , condition , increment/decrement). for loop condition is true loop is execute and condition is false loop is not execute. for loop is entry control loop, entry control loop is check condition before execute code. Exit the loop condition is false.

e.g :

For loop :

```
For
(I=1(initialization; I<=5(condition);i++(increment/decrement))
{
```

```
printf (“%d” , i);          // execute code

 }
```

- **Do while loop :** do { execute the code } while(condition) . do while loop condition first time execute and after condition is true loop is execute and condition is false loop is not execute. do while loop is exit control loop, exit control loop is execute code before check condition. Exit the loop condition is false.

e.g :

Do while loop :

```
I = 1;(initialization)
 Do   {
      printf (“%d” , i);          // execute code
      I++;                              // increment/decrement
      } While (I <= 5);          // (condition)
```

Q.7.Explain the use of break, continue, and goto statements in C. Provide examples of each.

→

Break statement : Break is keyword and break keyword use the break the code, rest of the code will not executed .

Example :

```
#include<stdio.h>
Main()
{
 Int I;
 for(i=0;i<10;i++)
 {
```

```
        If(i==6)
        Break;
        Printf(“%d”,I);
 }
```

Continue statements : The continue statement is used within loops to skip the remaining statements in the current iteration and proceed to the next iteration of the loop.
Example :

```
#include <stdio.h>

main()

{

        for (int i = 1; i <= 10; i++)

        {

                if (i % 2 == 0)

                        {

                         continue; // Skip even numbers

                        }

                printf("%d ", i);

         }

}
```

goto statements : The goto statement allows you to jump to a labeled part of the code. While it can make code harder to read and maintain, it is occasionally used in scenarios such as breaking out of deeply nested loops.

Example :

```
#include<stdio.h>
Main()
{
 Int I;
 I=0;
 abc :
        printf(“%d”,i);
        i++;
        if(i<=10)
        goto abc;
}
```

Q.8.What are functions in C? Explain function declaration, definition, and how to call a function. Provide examples.

→

Function is block of code use the function () round breaket , function is two type :

1) Build in function : build in function is already has a c language provide .it is not created by developer.
2) User define function :user define function is provide by developer.

Mainly four type for user define :

1. Fun with no argument no return value.

2. Fun with argument but no return value.

3. Fun without argument with return value.

4. Fun with arg. with return value.

- Function mainly three type :
  - Function declaration
  - Function calling
  - Function definition

Example :

```
#include<stdio.h>


void hello();  //1. fun. declaration


main()
{
        hello(); //2. fun. calling


}


void hello()  //3. fun. definition
{
        printf("\n\n\t Hello() is called.");
}
```

Q.9.Explain the concept of arrays in C. Differentiate between one-dimensional and multi-dimensional arrays with examples.

→Array : Array is a collection of elements/values with similar data type.

- Types of Arrays :

  1) One Dimentional Array

  e.g int arr[5];

  2) Two Dimentional Array

  e.g int arr[3][3];

  3) Multi Dimentional Array

  e. g int arr[4][3][3];

- Differentiate between one-dimensional and multi-dimensional arrays with example.
  Diffrence :
  →one dimensional array stored single line multiple element, output execute series type.

  →A multi-dimensional array is essentially an array of arrays. The most common type is the two-dimensional array, often used to represent matrices.

- Multi- dimensional arrays

```
#include<stdio.h>
main()
{
        int mat[2][3][3];
```

```
int m, r, c;

for(m=0;m<2;m++)
{
        printf("\n\n\t Matrix [%d] : ", m);
        for(r=0;r<3;r++)
        {
                for(c=0;c<3;c++)
                {
                                printf("\n\n\t Input
                                mat[%d][%d][%d] : ", m, r, c);
                        scanf("%d",&mat[m][r][c]);
                }
        }
}

for(m=0;m<2;m++)
{
        printf("\n\n\t Matrix [%d] : \n\n", m);
        for(r=0;r<3;r++)
        {
                for(c=0;c<3;c++)
                {
                        printf(" %d", mat[m][r][c]);
                }
                printf("\n");
        }
}}
```

Q.10.Explain what pointers are in C. and how they are declared and initialized. Why are pointers important in C?

→ Pointers : a pointer is a variable that stores the memory address of another variable. Instead of holding a direct value like a regular variable, a pointer "points" to the location in memory where the value is stored. This allows for efficient manipulation of data and memory.

Declared : type *pointer_name;

Initialized : Pointers are initialized using the address-of operator &, which gives the memory address of a variable.

example:

int x = 10; // A regular integer variable

int *ptr = &x; // Pointer to an integer, initialized to the address of x

- Why are pointers important in C.

Dynamic Memory Management:

- Pointers allow access to dynamically allocated memory using functions like malloc and free.

Efficiency:

- Pointers enable passing large data structures (e.g., arrays) to functions without copying the entire structure. Instead, only the memory address is passed.

Data Structures:

- Pointers are essential for implementing complex data structures like linked lists, trees, and graphs.

Function Pointers:

- Pointers can store the addresses of functions, enabling dynamic function calls and callback mechanisms.

Array and String Manipulation:

- Arrays and strings in C are closely tied to pointers, as an array name itself is a pointer to its first element.

Low-Level Programming:

- Pointers provide direct access to memory, making them vital for systems programming, hardware interactions, and performance-critical applications.

Example of pointer :

```
int a=10;
int *p=&a; //500000


printf("\n\n\t a = %d", a);
printf("\n\n\t p = %d", *p); //10
printf("\n\n\t p = %p", p); //500000
```

Q.11.Explain string handling functions like strlen(), strcpy(), strcat(), strcmp(), and strchr(). Provide examples of when these functions are useful.

→ String handling functions are a set of standard library functions in C, defined in <string.h>. These functions provide essential tools for

working with strings (character arrays). Here's a detailed explanation of common string-handling functions:

1. strlen()

- **Purpose:** Computes the length of a string (excluding the null terminator \0).
- **Prototype:** size_t strlen(const char *str);
- **Use case:** Determine the size of a string for processing or allocating memory.

  Example :

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

main() {
   char str[] = "Hello, World!";
   printf("Length of the string: %zu\n", strlen(str));
}
```

2. strcpy()

- **Purpose**: Copies the source string (including the null terminator) to the destination string.
- **Prototype**: char *strcpy(char *dest, const char *src);
- **Use case**: Duplicate or overwrite a string.

  Example :

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

 main() {
   char src[] = "Welcome!";
   char dest[20];

   strcpy(dest, src);
   printf("Destination String: %s\n", dest);

}
```

3. strcat()

- **Purpose**: Concatenates (appends) the source string to the end of the destination string.
- **Prototype**: char *strcat(char *dest, const char *src);
- **Use case**: Combine two strings.

Example :

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

 main() {
   char str1[50] = "Hello";
```

```
    char str2[] = ", World!";

    strcat(str1, str2);
    printf("Concatenated String: %s\n", str1);
}
```

4. strcmp()

- **Purpose**: Compares two strings lexicographically.
- **Prototype**: int strcmp(const char *str1, const char *str2);
- **Return Values**:
  - **< 0**: str1 is less than str2.
  - **0**: str1 is equal to str2.
  - **> 0**: str1 is greater than str2.

Use case: Compare strings for sorting or equality checks.

Exmple :

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

main() {
    char str1[] = "apple";
    char str2[] = "banana";

    int result = strcmp(str1, str2);
```

```
if (result < 0)
    printf("%s comes before %s\n", str1, str2);
else if (result > 0)
    printf("%s comes after %s\n", str1, str2);
else
    printf("%s is equal to %s\n", str1, str2);
}
```

5. strchr()

- **Purpose**: Finds the first occurrence of a character in a string.
- **Prototype**: char *strchr(const char *str, int c);
- **Return Value**: A pointer to the first occurrence of the character, or NULL if not found.
- **Use case**: Locate specific characters in a string.

Example :

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

main() {
    char str[] = "Hello, World!";
    char *ptr = strchr(str, 'W');

    if (ptr)
```

```
    printf("Character 'W' found at position: %ld\n", ptr - str);
  else
    printf("Character noundot f.\n");
  }
```

Q.12.Explain the concept of structures in C. Describe how to declare, initialize, and access structure members.

→ a structure is a user-defined data type that groups together variables of different data types under a single name. Structures are used to model more complex entities and are similar to records in other programming languages.

1. Declaring a Structure

To declare a structure, use the struct keyword followed by the structure name and a block of members.

→

```
struct Person {
  char name[50];
  int age;
  float height;
};
```

Here:

- struct Person is the structure type.
- name, age, and height are the members of the structure, which can have different data types.

2.Initializing a Structure

- Structure members can be initialized at the time of declaration:

```
struct Person person1 = {"Alice", 30, 5.5};
```

3.Accessing Structure Members

- Structure members are accessed using the dot operator (.)

```
printf("Name: %s\n", person1.name);
printf("Age: %d\n", person1.age);
printf("Height: %.1f\n", person1.height);
```

Q.13.Explain the importance of file handling in C. Discuss how to perform file operations like opening, closing, reading, and writing files.

→ File handling : file handling in c is the process in which we create , open , read , write , and close operation on a file.

- C language provides different function such as fopen() , fgets() , fputs().
- Fopen() = to open the file in memory by different different modes.
- Fputs() = to write data into file.
- Fgets() = to read the data from the file.
- Fclose()= close a file.

➢ Modes in file handling :

1) ‘r’ : the file is open un read mode.

2)’w’ : creats a text file in write mode.

3)’a’ :open a file append mode.

1) opening a file :

The fopen() function Is used to create a file or open an existing file.

Example :

Fptr=fopen(“demo.txr”,”w”);

2) reading a file:

The “r” mode is used to read a file.

To use to fgets() function to read a data in display.

Example :

Fptr=fopen(“demo.txt”,”r”);

3) writing file :

The “w” mode is used to write a file.

To use to fputs() function to write data in file.

Example :

fptr=fopen("File1.txt","w");

4)close file :

Used to close a file.

Example :

Fclose(fptr);