Module 2

VIKRAM KUMAR

**Que1 - Write an essay covering the history and evolution of C programming. Explain its importance and why it is still used today**

Ans **The history of c language**

* Dennis Ritchie is the developer of c language in 1972
* At & T, s bell lab, USA
* C language named from BCPL language
* BCPL language improves and make B language

**Importance of c programming**

* C set of built-in function
* Its access to low- level system resource
* C has become popular for embedded system
* C is highly portable

**Why it is still used**

* It is still used to its efficiency, portability and ability to connect directly with hardware.
* It is essential in system level programming and performance of critical applications.

**Que2- Describe the steps to install a C compiler (e.g., GCC) and set up an Integrated Development Environment (IDE) like DevC++, VS Code, or Code Blocks.**

Ans-

**1st install GCC compiler**

* Download MinGW
* Install and select components choose “gcc”
* Add bin folder path and set the environment variables

**2nd install the dev-C++**

1. Download dev c++
2. Run and install and complete the setup
3. Open dev c++, and set the path of GCC compilers

**3rd install c/c++ extensions**

1. Download vs code
   * + - Go to vs code website and download the installer for window
2. Install vs code
   * + - Run the downloaded installer and follow the installation instructions
3. Install c/c++ extensions
   * + - Open vs code and go to the extensions view by clicking the square icon in the activity bar on the side of the window
       - Search for c/c++ and install the extension by Microsoft.

**Que3- Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples**

Ans- #include<stdio.h> **(header file)**

Int main ()

**(//) = single line comment**

**(/\* content \*/) = multi-line comments**

{

Int age =10;

Float height =10.1;

Char grades =’a’;

Double = **store big value;**

**(Int, float, char- define the type of data a variable can hold.)**

Variable – **containers for storing data.**

Printf(“age: %d \n”,age);

Printf(“height: %.f \n”,height); **/**

Printf(“grade: %c \n”,grade);

Return 0;

}

**Que4- Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators.**

Ans- arithmetic operators are use for a mathematical operation on operands

There are 5 types of if arithmetic operators

* + - * **Arithmetic operators**

1. **+** addition operator (adds two numbers of values ) **a + b**
2. **–** subtraction operator (subtracts right operand from left operand) **a – b**
3. **\*** Multiply operator ( multiply two numbers) **a\*b**
4. **/** Divide operator (divide two numbers ) **a/b**
5. **%** modules operators( return the remainder ) **a % b**
   * + - **Relation operators**
6. **<** less than
7. **>** greater than
8. **<=** less than or equal to
9. **>+** greater than or equal to
10. **==** equal to
11. **!=**  not equal to
    * + - **Logical operator**
12. **&&** logical **AND**
13. **||** logical **OR**
14. **!** logical **NOT**
    * + - **Assignment operators**
15. **=** simple assignment
16. **+=** plus, and assign
17. **-=** minus and assign
18. **\*=** Multiply and assign
19. **/=** divide and assign
20. **%=** modulus and assign
21. **&=** AND and assign
    * + - **increment/decrement**
22. **a++** post-increment
23. **++a** pre-increment

**Que5- Explain decision-making statements in C (if, else, nested if-else, switch). Provide examples of each.**

Ans-

* **if statement**
  + - * Executes a block of code if the condition is true

Ex:- **if (a>0){**

**printf(“positive number );**

**}**

* **If-else statements**
  + - * Executes one block if the condition is true otherwise executes another block

Ex:- **if (a > 0) {**

**Printf("Positive");**

**} else**

**{**

**Printf("non-positive");**

**}**

* **Nested if-else**
  + - * If or else contains another if-else.

Ex**:- if (a > 0) { if (a % 2 == 0) {**

**printf("Positive even");**

**} else {**

**printf("Positive odd");**

**}**

**} else {**

**printf("non-positive");**

**}**

* **Switch statements**
  + - * In the switch case where user choice ( choice only one case ) individual case from the more than one cases

Ex:- **switch (choice) {**

**case 1:**

**Printf("Option 1");**

**break;**

**case 2:**

**Printf("Option 2");**

**break;**

**default: Printf("Invalid choice");**

**}**

**Que6- Compare and contrast while loops, for loops, and do-while loops. Explain the scenarios in which each loop is most appropriate**

Ans-

* + - * **For loop**
      * **While loop**
      * **Do-while loop**
  + **For loop – (i=1; i<=10; i++) =**

First initializes, then condition check, the executes the body, and last the update is done

* + **While loop – while( i<10){**

**Printf(“hello world”);**

**i++;**

**}**

**Return 0;**

**}**

First Initializes, then condition checks, and then executes the body, and updating can be inside the body

* + **Do-while loop { do**

**{**

**Printf("This loop will run forever.\n");**

**} while (1);**

**return 0;**

**}**

do-while first executes the body and then the condition check is done.

..

**Que7- What are functions in C? Explain function declaration, definition, and how to call a function. Provide examples.**

Ans-

* **What is functions**
  + - * Function is a block of code which has some name for identification
      * Function needs to be defined only once and call it any numbers of time
      * Each function in a program must have a unique name
      * One function name in program must be main()
      * Main() function is the entry point of a c-program
* **there are three keys’ components of functions**

1. Function Declaration

* A function declaration tell the complier about a function’s name, return type and parameters (argument).

1. Function Definition

* The function definition provide the actual implementation of the function.
* This includes the return type, the function name, the parameters, and the body of the function.

1. Function Calling

* A Function call instruct to the compiler to execute the function.

* **There are 4 types of function**

1. With Return Type With Argument
2. With Return Type Without Argument
3. Without Return Type With Argument
4. Without Return Type Without Argument

Q8. Explain the use of break, continue, and goto statements in C. Provide examples of each.

Ans- 1. break Statement

The break statement is used to exit a loop or a switch statement immediately, regardless of the loop's condition.

It is commonly used to stop the execution of a for, while, or do-while loop.

When a break is encountered, control jumps to the statement immediately after the loop or switch block.

Syntax

break;

✅ Example with break in a loop

#include <stdio.h>

int main() {

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

if (i == 5) {

printf("Breaking at i = %d\n", i);

break; // Exit the loop when i equals 5

}

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

}

printf("Loop ended.\n");

return 0;

}

. continue Statement

The continue statement skips the current iteration of a loop and moves control to the next iteration.

It is used to bypass specific conditions without exiting the loop entirely.

Syntax:

continue;

✅ Example with continue in a loop:

#include <stdio.h>

int main() {

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

if (i % 2 == 0) {

continue;

}

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

}

return 0;

}

## . goto Statement

## The goto statement transfers control to a labeled statement elsewhere in the program.

## It can be used to implement jump-based control flow.

## goto is generally discouraged because it makes code harder to read and debug (can lead to spaghetti code).

## Syntax:

## goto label

## label:

## ✅ Example with goto:

## #include <stdio.h>

## int main() {

## int num = 1;

## if (num == 1) {

## goto skip;

## }

## printf("This statement is skipped.\n");

## skip:

## printf("This statement is executed because of goto.\n");

## return 0;

## }

**Que9- Explain the concept of arrays in C. Differentiate between one-dimensional and multi-dimensional arrays with examples.**

Ans- array is a collection of elements of the same type, which store in contiguous memory location

Arrays are useful for storing large amounts of data

* + - * **One dimensional array :-**

one-dimensional array is simple list of elements, of same type

syntax :- int Num [5] = {1,2,3,4,5};

* + - * **Multi-dimensional array :-**

A multi-dimensional array is a array of array. It is representing table of matrix

Syntax :- int Num [2][3] ={{1,2,3},{1,2,3}};

**Que10- Explain string handling functions like Strlen(), Strcpy(), strcat(), strcmp(), and Strchr(). Provide examples of when these functions are useful.**

Ans-

1. **Strlen** (string length) = is used to find the length of the string
2. **Strcpy** (string copy ) = is use for copying the string, duplicate or assign
3. **strcat** (string concatenate) = is concatenate the two string (combine two strings into one)
4. **strcmp**(string compare) = is used to compare two strings (sorting equality checks).
5. **Strchr** (string character) = find the character in string, search or parse a string
6. **Strlwr**(string lower) = is used to upper case character into lower case character
7. **Strupr**(string upper) = is used to lower case character into upper case characters
8. **Strrev**(string reverse) = is used to reverse the strings (last index of string to 0th index of string )print all characters in reverse

Q11. Explain what pointers are in C and how they are declared and initialized. Why

are pointers important in C?

ans-Pointers in C are variables that store the **memory address** of another variable. Instead of storing a value directly, a pointer stores the location where the value is kept in memory.

1. **Data Structures** – Pointers are essential for implementing complex data structures like **linked lists**, **trees**, and **graphs**.
2. **Pointer Arithmetic** – Pointers enable arithmetic operations (e.g., increment, decrement) to navigate through arrays and memory block.
3. **Direct Memory Access** – Pointers allow direct manipulation of memory, which gives you more control over how your program runs.
4. **Dynamic Memory Allocation** – Pointers are used with functions like malloc() and free() to allocate and deallocate memory at runtime.
5. **Function Arguments** – Pointers allow you to pass large data structures (like arrays) to functions more efficiently by passing the memory address instead of copying data.

Q12. Explain the concept of structures in C. Describe how to declare, initialize, and access

Structure member?

### Ans-**Structures in C**

A **structure** in C is a user-defined data type that groups related variables of different data types into a single entity. Structures are useful when you need to represent complex data models, such as records in a database or entities like students, employees, etc.

Declaring a Structure

You define a structure using the struct keyword.

Syntax:

struct StructureName {

dataType member1;

dataType member2;

};

Initializing a Structure

You can initialize structure members in multiple ways:

Direct Initialization at Declaration

struct Student s1 = {"Alice", 20, 3.8};

### **Accessing Structure Members**

To access members of a structure:

* Use the **dot operator (.)** for normal structure variables.
* Use the **arrow operator (->)** for pointers to structures.

**Example Code:**

#include <stdio.h>

#include <string.h>

struct Student {

char name[50];

int age;

float gpa;

};

int main() {

struct Student s1 = {"Alice", 20, 3.8};

struct Student s2;

strcpy(s2.name, "Bob");

s2.age = 22;

s2.gpa = 3.5;

printf("Student 1: %s, Age: %d, GPA: %.2f\n", s1.name, s1.age, s1.gpa);

printf("Student 2: %s, Age: %d, GPA: %.2f\n", s2.name, s2.age, s2.gpa);

struct Student \*ptr = &s1;

printf("Access via pointer: %s, Age: %d, GPA: %.2f\n",

ptr->name, ptr->age, ptr->gpa);

return 0;

}

Q13.Explain the importance of file handling in C. Discuss how to perform file

operations like opening, closing, reading, and writing files.

Ans. Importance of File Handling in C

File handling in C allows programs to store, access, and manipulate data stored on the disk in the form of files. Without file handling, data would be lost once the program terminates because data stored in variables and arrays is held in temporary memory (RAM).

1. Opening a File

To open a file, use the fopen()

FILE \*fp;

fp = fopen("filename.txt", "mode");

FILE \* – A pointer to a FILE type that stores file information.

"filename.txt" – The name of the file.

2. Closing a File

After working with a file, close it using fclose():

fclose(fp);

Closing a file:

Ensures all data is properly saved.

Frees up system resources.

3. Writing to a File

You can write to a file using:

fprintf() – Writes formatted text to a file.

fputs() – Writes a string to a file.

fputc() – Writes a single character to a file.

4. Reading from a File

You can read from a file using:

fscanf() – Reads formatted data.

fgets() – Reads a string.

fgetc() – Reads a single character