**Module 2 – Introduction to programming**

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

* C programming, developed in the early 1970s by Dennis Ritchie at Bell Labs, is one of the most influential programming languages in computing history.
* It was created as an evolution of the B programming language.
* The importance of c programming lies in its simplicity, efficiency, and performance.
* As c matured, it became the foundation for many modern languages such as C++, C#, java, and python.
* C language is used today because is fast, efficient, and close to hardware.it provides low-level memory access, making it ideal for system programming.

1. **Describe the steps to install C compiler and setup an integrated Development like DevC++, VS code, or CodeBlocks.**

* Describe in below
* Dev C++
* Step 1: download Dev C++
* Go to google
* Step 2: install Dev C++
* Run the download installer and follow the instructions.
* Step 3: start coding
* Open Dev C++, create a new project or source file(.c), write your code, and hit compile & run

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

* Explain below
* Header files
* Header files contain definitions of functions and macros. They are included at the beginning of the program using the #include directive.
* E.g. #include<stdio.h>
* Comments
* Comments are used to explain code and are ignored by the compiler. There are two types:

1. Single-line comment: //
2. Multi-line comment: /\*…\*/

* E.g. // this is single line comment

/\* this is multiline comment\*/

* The main function
* This is the entry point of every C program. Execution starts from here.
* E.g. int main()

{

// code goes here

Return 0;

}

* Data Types
* C has several basic data types for declaring variables.
* E.g.

|  |  |  |
| --- | --- | --- |
| Data Type | Description | Example |
| Int | Integer values | Int age=25; |
| Float | Floating point | Float pi=3.14; |
| Char | Single characters | Char grade=’A’; |
| double | Double precision float | Double largepi=3.1415926535; |

* Variables
* Variables are contains for storing data. You must declare them before use.
* E.g. int a=30;
* Complete example of a basic C program

#include<stdio.h>

Main ()

{

Int age = 25;

Float height = 5.2;

Char grade = ‘A’;

Printf (“Age: %d\n”, age);

Printf (“height: %f\n”, height);

Printf (“grade: %c\n”, grade);

}

* Output

Age: 25

Height: 5.2

Grade: A

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

* Operators are below:
* Relational operators
* Used to compare two values. They return 1 (true) or 0(false).
* E.g.
* == (equal to)
* != (not equal to)
* > (greater than)
* < (less than)
* >= (greater than or equal to)
* <= (less than or equal to)
* Logical Operators
* Used to perform logical operations, mainly in conditions.
* E.g.
* && (logical AND)
* || (logical OR)
* ! (logical NOT)
* Assignment Operators
* Used to assign values to variables.
* E.g.
* = (assign)
* += (add and assign)
* -= (subtract and assign)
* \*= (multiply and assign)
* /= (divide and assign)
* Increment/Decrement operators
* Used to increase or decrease a variable’s value by 1.
* E.g.
* ++ (increment)
* -- (decrement)
* Bitwise operator
* Operate on bits and perform bit-by-bit operations.
* E.g.
* & (AND)
* `(OR)
* ^ (XOR)
* ~ (NOT)
* << (Left Shift)
* >> (Right Shift)
* Conditional Operator
* Used as a shorthand for if-else.
* E.g.
* Condition?
* A > b? a: b

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

* Below:
* If-else Statement
* Syntax: if (condition)

{

// code if condition is true

}

Else

{

// code if condition is false

}

* E.g.

Int age = 18;

If (age >= 18)

{

Printf (“eligible for vote.);

}

Else

{

Printf (“Not eligible.”);

}

* Nested if-else Statement
* This is used when multiple conditions need to be checked in a sequence.
* Syntax:

If(condition)

{

//code

}

Else if (condition2)

{

// code

}

Else

{

// default code

}

* E.g.

Int marks = 85;

If (marks >= 90)

{

Printf (“grade A”);

}

Else if (marks >= 75)

{

Printf (“grade B”);

}

Else

{

Printf (“grade C”);

}

* Switch Statement
* Used when you want to select one option from many based on the value of a variable.
* Syntax:

Switch (expression)

{

Case value 1:

// code

Case value 2:

// code

Break;

default:

// code

}

* E.g.

int day = 3;

switch (day) {

case 1:

printf ("Monday");

break;

case 2:

printf("Tuesday");

break;

case 3:

printf("Wednesday");

break;

default:

printf("Invalid day");

}

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

* Below:
* While Loop
* Syntax:

While (condition)

{

// code

}

- description: checks the condition before executing the loop body.

* Use Case: when the number of iterations is not known in advance.
* For Loop
* Syntax:

for (initialization; condition; increment)

{

// code

}

* Description: combines initialization, condition-checking, and updating in one line.
* Use case: when the number of iterations is known.
* Do-while Loop
* Syntax:

Do {

// code

} while (condition);

* Description: executes the loop body at least once, then checks the condition.
* Use case: when the loop must run at least once, regardless of the condition.

1. **Explain the use of break, continue, and goto statements in C. provide examples of each.**

* Below:
* Break
* The break statement is used to exit from a loop or a switch statement before It finishes its normal execution.
* E.g.

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

{

If(i==3)

break;

printf (“%d”, i);

}

* Continue
* The continue statement allow an unconditional jump to another part of the program. Although it’s not recommended due to readability issues, it can be used in special cases.
* E.g.

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

{

If(i==3)

Continue;

Printf (“%d”, i);

}

* Goto
* The goto statement allows an unconditional jump to another part of the program.
* E.g.

Int x = 1;

If(x==1)

Goto skip;

Printf(“hey\n”);

Skip:

Printf(“bye\n”);

1. **What are the function in C? explain function declaration, definition, and how to call a function. Provide examples.**

* In c programming, functions are blocks of code that perform a specific task.
* Function Declaration
* A function declaration tells the compiler about the function’s name, return type, and parameters before its actual definition.
* E.g. int add (int a, int b);
* Function definition
* The function definition contains the actual code that runs when the function is called.
* E.g. return\_type function\_name (parameter

-list) {

// function body

}

* Function call
* To use a function, you “call” it by using its name and passing the required arguments.
* E.g. int result = add (3,5);

1. **Explain the concepts of arrays in C. Differentiate between one-dimensional and multi-dimensional arrays with example.**

* An arrays in C is a collection of elements of the same data type stored at contiguous memory locations.
* It allows us to store multiple values in a single variable instead of declaring separate variables for each value.
* Two types of array:
* One-dimensional array
* A one-dimensional array is a list of elements. It is like a row of elements.
* Syntax: data\_type array\_name[size];
* Two-dimensional array
* A multi-dimensional array is an array of arrays. The most common is a 2D array, which can be thought of as a table with rows and columns.
* Syntax: data\_type array\_name[row\_size] [column\_size];
* Difference:

|  |  |  |
| --- | --- | --- |
| feature | One-dimensional Array | Multi-dimensional Array |
| structure | linear | Tabular |
| syntax | Int a [5]; | Int a [2][3]; |
| Accessing elements | A [2] | A [1][2] |
| Use case | Simple list of items | Matrices, tables, grids |

1. **Explain what pointers are in C and how they are declared and initialized. Why are pointers important in C?**

* A pointer is a variable that stores the memory address of another variable. Instead of holding data directly, it holds the address where the data is stored in memory.
* Declaration and initialization of pointers:
* To declare a pointer, use the \* symbol before the pointer name.
* Syntax: data\_type \*pointer\_name
* E.g.

Int a = 10;

Int \*ptr;

ptr = &a;

* Why are pointers important in C?
* Direct memory access
* Call by reference
* Dynamic memory allocation
* Building data structures

1. **Explain sting handling functions like strlen (), strcpy (), strcat (), strcmp (), and strchr (). Provide examples of when these functions are useful.**

* Below:
* Strlen ()
* Returns the length of a string
* Syntax: size\_t strlen (const char \*str);
* E.g.

Char a [] = “hey”;

Int len = strlen(a);

* Strcpy ()
* Copies a string from source to destination
* Syntax: char \*strcpy (char \*dest, const char \*src);
* E.g.

Char src [] = “World”;

Char dest [10];

Strcpy (dest, src);

* Strcat ()
* Concatenates one string to the end of another
* Syntax: char \*strcat (char \*dest, const char \*src);
* E.g.

Char str1[20] = “hello”;

Char str2[20] = “world”;

Strcat (str1, str2);

* Strcmp ()
* Compares two strings
* Syntax: int strcmp (const char \*str1, const char \*str2);
* E.g.

Strcmp (“abc”, “abc”);

Strcmp (“abc”, “abd”);

* Strchr ()
* Finds the first occurrence of a character in a string.
* Syntax: char \*strchr (const char \*str, int c);
* E.g.

Char 8ptr = strchr (“hey”, ‘e’);

1. **Explain the concept of structures in C. describe how to declare, initialize, and access structure members.**

* A structure is user define data that group different types of variables in one name
* Key points:
* Structure group different type of data
* Use dot (.) operator to access members
* Use arrow (->) operator with pointers
* Declare structure

struct Person {

char name [50];

int age;

float height;

};

* Initialize members

Strcpy (p1.name, "Alice");

p1.age = 25;

p1. height = 5.4;

* Access structure members

Printf ("%s", p1.name);

1. **Explain the importance of the handling in C. discuss how to perform file operations like opening, closing, reading, and writing files.**

* Importance: file handling in C is essential for storing data permanently. It allows programs to read from and write to files on disk, enabling data persistence across program executions.
* File operations in C:
* File opening
* Use fopen () to open a file
* Syntax: FILE \*fp;
* fp = fopen ("file.txt", "r");
* File closing
* Use fclose () to close an open file
* Syntax: flcose(fp);
* File reading
* Functions: fgetc (), fgets (), fread ()
* E.g.

FILE \*fp = fopen ("data.txt", "r");

char str [100];

fgets (str, 100, fp);

printf ("%s", str);

fclose(fp);

* File writing
* Functions: fputc (), fputs (), fprintf (), fwrite s ()
* E.g.

FILE \*fp = fopen ("data.txt", "w");

Fprintf (fp, "Hello, world! \n");

fclose(fp);