**Module 2 – Introduction to Programming**

**Overview of C Programming**

**THEORY EXERCISE**:

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

* The C programming language is a general-purpose high-level language developed by Dennis Ritchie at Bell Labs in the 1970s. It was intended to be a low-level language that provided direct access to hardware and system resources, making it highly efficient and flexible. Due to its simplicity and elegance, C has become the building block for many other programming languages.
* The C programming language’s history can be traced to Dennis Ritchie, who developed it at Bell Labs in the early 1970s. It evolved from an earlier language called B, which was developed by Ken Thompson. B itself was based on the BCPL language.
* C gained popularity due to its portability and efficiency. In 1978, Brian Kernighan and Dennis Ritchie published “The C Programming Language,” which became the go-to reference for C programmers. Since then, C has been widely used and has influenced the development of other programming languages.
* C programming language offers several advantages over other [programming languages](https://iies.in/blog/best-embedded-systems-programming-languages/), including:
* Efficiency: C allows for direct memory manipulation and low-level access to system resources. This results in highly efficient code execution.
* Portability: C code can be easily ported to different platforms without major modifications, thanks to its wide availability of compilers and libraries.
* Speed: C is known for its fast execution speed, making it suitable for developing performance-critical applications.
* Control: C gives programmers fine-grained control over memory management and system resources.
* Compatibility: C code can be easily integrated with code written in other languages like C++, Java, and [Python](https://iies.in/crash-courses/python-crash-course/).
* **LAB EXERCISE**:

Research and provide three real-world applications where C programming is extensively used, such as in embedded systems, operating systems, or game development.

* Operating systems: All modern operating systems, such as Windows, Linux, and macOS, have a significant portion of their codebase written in C.
* Embedded systems: Devices like washing machines, smart TVs, and medical equipment often run on embedded systems programmed using C.
* Mobile applications: Some parts of mobile applications, including high-performance tasks and system-level operations, may be implemented in C for efficiency.
* Gaming consoles: Popular gaming consoles’ core software and engines are often developed using C.
* Network infrastructure: Routers and networking devices rely on software written in C to handle complex networking protocols.

**2. Setting Up Environment**

**THEORY EXERCISE**:

* 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.

**The installation process for Dev-C++ involves the following steps:**

* Download Dev-C++ IDE from <http://www.bloodshed.net/dev/devcpp.html>
* Click on source forge link under “Downloads -> Dev-C++ 5.0 beta 9.2 (4.9.9.2) (9.0 MB) with Mingw/GCC”
* Save the .exe file in your local machine.
* Double click on the exe file.
* Start the installation by clicking Next button and I Agree button.
* Choose the destination folder as “C:\Dev-Cpp” (It is there by default) and click “Install’ button and finally click finish button. You can modify this destination folder path if you want.
* Once installation is completed, go to your desktop and right click on My Computer -> properties -> advanced system settings -> advanced tab. Then, click on “Environment Variables” button and then “New”. You will get a popup window as shown below.
* Change the System variable as given below.
* Variable name: PATH  
  Variable value: C:\Dev-Cpp\bin;
* Once you done with above settings, then you can start Dev C++ by clicking start –> Dev C++ as shown below.
* Open Dev C++ window and click on file -> new -> project. Then, select Console Application. Choose “C project” and “Make Default Language” check box.
* Click on file -> new -> source file and type a sample program and save it as sample.c
* Click on “Compile & Run” button to compile and execute our program .

**3. Basic Structure of a C Program**

**THEORY EXERCISE**:

* Explain the basic structure of a C program, including headers, main function,

comments, data types, and variables. Provide examples.

Structure of C Program:

* Header: #include<stdio.h>
* Main():Int main()

{

Variable declaration: Int a=10;

Body: Printf(“%d”,a);

Return: Return 0;

}

* Syntax to include a header file in C: #include.
* stdio.h – Defines core input and output functions.
* Syntax to Declare main method: int main(){}
* Variable Declaration: It refers to the variables that are to be used in the function, the variables are to be declared before any operation in the function.

Example: int main()

{

int a;

. .

}

* Body: Body of a function in C program, refers to the operations that are performed in the functions. It can be anything like manipulations, searching, sorting, printing, etc.

Example: int main()

{

int a;

printf("%d", a); . .

}

* Return Statement: The return statement refers to the returning of the values from a function. This return statement and return value depend upon the return type of the function. For example, if the return type is void, then there will be no return statement. In any other case, there will be a return statement and the return value will be of the type of the specified return type.

**4. Operators in C**

**THEORY EXERCISE**:

Write notes explaining each type of operator in C: arithmetic, relational,

logical, assignment, increment/decrement, bitwise, and conditional operators.

An arithmetic operator performs mathematical operations such as addition,subtraction, multiplication, division etc on numerical values (constants and variables).

| Operator | Meaning of Operator |
| --- | --- |
| + | addition or unary plus |
| - | subtraction or unary minus |
| \* | Multiplication |
| / | Division |
| % | remainder after division (modulo division) |

An assignment operator is used for assigning a value to a variable. The most common assignment operator is =

| Operator | Example | Same as |
| --- | --- | --- |
| = | a = b | a = b |
| += | a += b | a = a+b |
| -= | a -= b | a = a-b |
| \*= | a \*= b | a = a\*b |
| /= | a /= b | a = a/b |
| %= | a %= b | a = a%b |

Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in [decision making](https://www.programiz.com/c-programming/c-if-else-statement) and [loops](https://www.programiz.com/c-programming/c-for-loop).

| Operator | Meaning of Operator | Example |
| --- | --- | --- |
| == | Equal to | 5 == 3 is evaluated to 0 |
| > | Greater than | 5 > 3 is evaluated to 1 |
| < | Less than | 5 < 3 is evaluated to 0 |
| != | Not equal to | 5 != 3 is evaluated to 1 |
| >= | Greater than or equal to | 5 >= 3 is evaluated to 1 |
| <= | Less than or equal to | 5 <= 3 is evaluated to 0 |

Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in [decision making in C programming](https://www.programiz.com/c-programming/c-if-else-statement).

| Operator | Meaning | Example |
| --- | --- | --- |
| && | Logical AND. True only if all operands are true | If c = 5 and d = 2 then, expression ((c==5) && (d>5)) equals to 0. |
| || | Logical OR. True only if either one operand is true | If c = 5 and d = 2 then, expression ((c==5) || (d>5)) equals to 1. |
| ! | Logical NOT. True only if the operand is 0 | If c = 5 then, expression !(c==5) equals to 0. |

Bitwise Operators

During computation, mathematical operations like: addition, subtraction, multiplication, division, etc are converted to bit-level which makes processing faster and saves power.

Bitwise operators are used in C programming to perform bit-level operations.

| Operators | Meaning of operators |
| --- | --- |
| & | Bitwise AND |
| | | Bitwise OR |
| ^ | Bitwise exclusive OR |
| ~ | Bitwise complement |
| << | Shift left |
| >> | Shift right |

Increment and Decrement Operators

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1. These two operators are unary operators, meaning they only operate on a single operand.

**5. Control Flow Statements in C**

**THEORY EXERCISE**:

Explain decision-making statements in C (if, else, nested if-else, switch).

Provide examples of each.

**Simple if Statement:** Syntax of the if statement is as given below:

If(condition){

Statement;

}

**If-else Statement :**

In some situations, you may have to execute statements based on true or false under certain conditions, therefore; you use if-else statements. If the condition is true, then if block will be executed otherwise the else block is executed.

Syntax of the if-else statement is as given below:

If(condition)

Statement1;

Else

Statement2;

**Nested if-else Statements**

The nested if-else statements consist of another if or else. Therefore; if the condition of “if” is true (i.e., an outer if) then outer if’s if block is executed which contains another if (that is inner if) and if the condition of if block is true, statements under if block will be executed else the statements of inner if’s “else” block will be executed.

If the outer “if” condition is not true then the outer if’s “else” block is executed which consists of another if. The outer else’s inner if the condition is true then the statement under outer else’s inner if is executed else the outer else’s else block is executed.

Syntax of the nested if-else statement is as given below:

If(condition1)

{

If(condition2)

{

Statement1;

}

Else

{

Statement2;

}

Else

{

If(condition3)

{

Statement3;

}

Else

{

Statement4;

}

}

## switch-case Statement:

The switch-case statement is a [decision-making statement](https://www.tutorialspoint.com/cprogramming/c_decision_making.htm) in C. The if-else statement provides two alternative actions to be performed, whereas the switch-case construct is a multi-way branching statement. A switch statement in C simplifies multi-way choices by evaluating a single variable against multiple values, executing specific code based on the match. It allows a [variable](https://www.tutorialspoint.com/cprogramming/c_variables.htm) to be tested for equality against a list of values.

switch(expression)

{

case x:

// code block

break;

case y:

// code block

break;

default:

// code block

break;

}

**6. Looping in C**

**THEORY EXERCISE**:

Compare and contrast while loops, for loops, and do-while loops. Explain the

scenarios in which each loop is most appropriate.

* While loops, for loops, and do-while loops are all types of loops that are used in programming to repeat a block of code based on a condition. The main differences between them are when the condition is evaluated and how many iterations are required:
* **While loop**

Evaluates the condition first, then executes the statement. This loop is used when the number of iterations is unknown or the condition is long-running.

* **For loop**

Used when the number of iterations is known in advance. The syntax for a for loop includes initialization, expression, and increment.

* **Do-while loop**

Executes the statements first, then evaluates the condition. This loop guarantees at least one execution of the loop body. It's used when the number of iterations is unknown and the loop must be executed at least once.

* Loops are essential for automating repetitive tasks, iterating through data structures, and creating interactive programs

**7. Loop Control Statements**

**THEORY EXERCISE**:

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

* **Break**: Terminates the current iteration of a loop or switch statement, and transfers control to the next statement. For example, to break out of a for loop when i is equal to 4, you can use the following code:

int i;

for (i = 0; i < 10; i++) {

if (i == 4) {

break;

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

}

* **Continue**: Skips the current iteration of a loop, and transfers control to the beginning of the next iteration. For example, to skip the value of 4 in a for loop, you can use the following code:

int i;

for (i = 0; i < 10; i++) {

if (i == 4) {

continue;

}

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

}

* **Goto**: Transfers control to a labeled statement within the same function. The syntax for a goto statement is goto label; or label: goto label;. The label is a user-defined identifier that indicates the target statement.

.

**8. Functions in C**

**THEORY EXERCISE**:

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

* A function is a block of code which only runs when it is called.
* You can pass data, known as parameters, into a function.
* Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.
* The declaration establishes the names and characteristics of a function but does not allocate storage for it, while the definition specifies the body for a function, associates an identifier with the function, and allocates storage for it.
* We use the function name followed by the argument list in parentheses to call a function. For example, we can use the following code to call the sum function that we defined earlier: int a = 5; int b = 10; int c = sum(a, b); In this code, we are calling the sum function with a and b as its parameters.

**9. Arrays in C**

**THEORY EXERCISE**:

Explain the concept of arrays in C. Differentiate between one-dimensional and

multi-dimensional arrays with examples.

* In C, an array is a collection of data items of the same type that are accessed using a common name. The main difference between one-dimensional and multi-dimensional arrays is their structure and how data is accessed and manipulated:
* **One-dimensional arrays**

Store elements in a single line, similar to a list. One-dimensional arrays are the simplest form of arrays.

* **Multi-dimensional arrays**

Store elements in a table format, similar to a matrix. Multi-dimensional arrays can have more than one level or dimension, such as two-dimensional (2D) or three-dimensional (3D) arrays.

**10. Pointers in C**

**THEORY EXERCISE**:

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

are pointers important in C?

* In C, pointers are variables that store the memory address of other variables, allowing for indirect access and manipulation of data in the computer's memory. Pointers are a powerful and important feature of C that can improve performance and reduce code.
* Here's how to declare and initialize pointers in C:
* **Declaration**

To declare a pointer variable, use the syntax type \*var-name. Here, type is the base type of the pointer, and var-name is the name of the pointer variable. The asterisk (\*) is used to designate a variable as a pointer.

* **Initialization**

To initialize a pointer, assign the address of a variable to it. This process is known as pointer initialization. You can use the address operator (&) to find the address of a variable.

**11. Strings in C**

**THEORY EXERCISE**:

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

* The strlen() function in C is used to find the length of a string.
* The strcpy() function in C is used to copy one string to another.
* The strcat() function in C is used to concatenate two strings.
* The [strcmp()](https://www.geeksforgeeks.org/strcmp-in-c/) is a built-in library function in C. This function takes two strings as arguments and compares these two strings lexicographically.
* The strchr() function lexicographically compares the first n characters from the two null-terminated strings and returns an integer based on the outcome.

**12. Structures in C**

**THEORY EXERCISE**:

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

access structure members.

The structure in C is a user-defined data type that can be used to group items of possibly different types into a single type. The **struct keyword**is used to define the structure in the C programming language. The items in the structure are called its **member** and they can be of any valid data type. Additionally, the values of a structure are stored in contiguous memory locations.

We have to declare structure in C before using it in our program. In structure declaration, we specify its member variables along with their datatype. We can use the struct keyword to declare the structure in C using the following syntax:

Syntax

struct structure\_name {

data\_type member\_name1;

data\_type member\_name1;

....

....

};

To use structure in our program, we have to define its instance. We can do that by creating variables of the structure type. We can define structure variables using two methods:

### 1. Structure Variable Declaration with Structure Template

struct structure\_name {

data\_type member\_name1;

data\_type member\_name1;

....

....

}variable1, varaible2, ...;

### 2. Structure Variable Declaration after Structure Template

/ structure declared beforehand

struct structure\_name variable1, variable2, .......;