# **Module 2 – Introduction to Programming with C**

## **1. Overview of C Programming**

### Theory Exercise: History and Evolution of C Programming

The C programming language was developed by Dennis Ritchie in the early 1970s at Bell Labs. It was designed as a powerful, efficient language that could run on different hardware platforms and provide low-level access to memory. C evolved from earlier languages like B and BCPL.  
Its close connection with the development of the UNIX Operating System helped it gain widespread adoption. C stands out for its portability, allowing the same program to run on many machine architectures with minimal changes, making it essential in system programming.  
Even today, C remains important for embedded systems, operating systems, and performance-critical applications because of its speed, efficiency, and control over hardware.

### Lab Exercise: Real-World Applications of C Programming

* **Embedded Systems:** C is widely used for programming microcontrollers and embedded devices, providing direct hardware control.
* **Operating Systems:** Important parts of operating systems like Linux and Windows are implemented in C, benefiting from its efficiency.
* **Game Development:** Game engines and performance-critical game components often use C or C++ to achieve fast execution and hardware interaction.

## **2. Setting Up Environment**

### Theory Exercise: Installing a C Compiler and IDE

To start programming with C, you first need to install a compiler and an IDE:  
**1. Install GCC Compiler:**  
- On Windows: Get [MinGW](https://sourceforge.net/projects/mingw/) and follow setup instructions.  
- On macOS: Use Terminal to run xcode-select --install to install Command Line Tools containing GCC.  
- On Linux: Use your package manager, e.g., sudo apt install build-essential on Ubuntu.  
**2. Install IDE:** Choose IDEs like DevC++, Visual Studio Code, or CodeBlocks and configure to use the GCC compiler installed.  
This setup enables easy editing, compiling, and running of your C programs.

### Lab Exercise: Your First C Program

#include <stdio.h>

int main() {

printf("Hello, World!\n");

return 0;

}

Compile and run this program in your configured IDE to print **Hello, World!** on the screen.

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

### Theory Exercise: Basic Structure Explained

A typical C program includes:  
- **Header files:** Like #include <stdio.h>, they bring in standard functions.  
- **Main function:** Code entry point: int main()  
- **Comments:** Explain code, use // or /\* \*/.  
- **Data types and variables:** Hold data like integers, characters, floats.

### Lab Exercise: Program with Variables, Constants, and Comments

#include <stdio.h>

int main() {

// Variable declarations

int age = 21;

char initial = 'J';

float height = 5.7f;

// Display values

printf("Age: %d\n", age);

printf("Initial: %c\n", initial);

printf("Height: %.2f feet\n", height);

return 0;

}

## **4. Operators in C**

### Theory Exercise: Types of Operators

* **Arithmetic:** +, -, \*, /, %
* **Relational:** ==, !=, >, <, >=, <=
* **Logical:** &&, ||, !
* **Assignment:** =, +=, -=, \*=, /=
* **Increment/Decrement:** ++, --
* **Bitwise:** &, |, ^, ~, <<, >>
* **Conditional (ternary):** ? :

### Lab Exercise: Perform Operations on Two Integers

#include <stdio.h>

int main() {

int a, b;

printf("Enter two integers: ");

scanf("%d %d", &a, &b);

printf("Addition: %d\n", a + b);

printf("Subtraction: %d\n", a - b);

printf("Multiplication: %d\n", a \* b);

if (b != 0)

printf("Division: %d\n", a / b);

else

printf("Division: Cannot divide by zero\n");

printf("Is a equal to b? %s\n", (a == b) ? "Yes" : "No");

printf("a > b and a != b: %s\n", (a > b && a != b) ? "True" : "False");

return 0;

}

## **5. Control Flow Statements in C**

### Theory Exercise: Decision-Making Statements

* **if:** Executes if condition is true.
* **else:** Executes if condition is false.
* **Nested if-else:** Multiple conditions tested in sequence.
* **switch:** Selects block based on expression value.

### Lab Exercise: Even/Odd Check and Month Name

#include <stdio.h>

int main() {

int num, month;

printf("Enter a number: ");

scanf("%d", &num);

if (num % 2 == 0)

printf("%d is Even\n", num);

else

printf("%d is Odd\n", num);

printf("Enter month number (1-12): ");

scanf("%d", &month);

switch (month) {

case 1: printf("January\n"); break;

case 2: printf("February\n"); break;

case 3: printf("March\n"); break;

case 4: printf("April\n"); break;

case 5: printf("May\n"); break;

case 6: printf("June\n"); break;

case 7: printf("July\n"); break;

case 8: printf("August\n"); break;

case 9: printf("September\n"); break;

case 10: printf("October\n"); break;

case 11: printf("November\n"); break;

case 12: printf("December\n"); break;

default: printf("Invalid month\n");

}

return 0;

}

## **6. Looping in C**

### Theory Exercise: Loop Types Comparison

- **while loop:** Used when the number of iterations is unknown and depends on a condition.  
- **for loop:** Best for known number of iterations.  
- **do-while loop:** Executes at least once, then repeats while condition is true.

### Lab Exercise: Print Numbers 1 to 10 Using All Loops

#include <stdio.h>

int main() {

int i = 1;

printf("While Loop:\n");

while (i <= 10) {

printf("%d ", i);

i++;

}

printf("\n");

printf("For Loop:\n");

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

printf("%d ", i);

}

printf("\n");

printf("Do-While Loop:\n");

i = 1;

do {

printf("%d ", i);

i++;

} while (i <= 10);

printf("\n");

return 0;

}

## **7. Loop Control Statements**

### Theory Exercise: break, continue, goto

* **break:** Exits the nearest enclosing loop or switch immediately.
* **continue:** Skips the rest of the current loop iteration and continues with the next.
* **goto:** Jumps to a labeled statement. Should be used sparingly.

### Lab Exercise: Use break and continue

#include <stdio.h>

int main() {

printf("Using break to stop at 5:\n");

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

if (i > 5)

break;

printf("%d ", i);

}

printf("\n");

printf("Using continue to skip 3:\n");

for (int i = 1; i <= 5; i++) {

if (i == 3)

continue;

printf("%d ", i);

}

printf("\n");

return 0;

}

## **8. Functions in C**

### Theory Exercise: Functions

Functions encapsulate reusable code. You declare a function with its return type and parameters, define the function body, and call it by name with arguments.

### Lab Exercise: Factorial Using Function

#include <stdio.h>

int factorial(int n) {

if (n == 0) return 1;

return n \* factorial(n - 1);

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

printf("Factorial of %d is %d\n", number, factorial(number));

return 0;

}

## **9. Arrays in C**

### Theory Exercise: Arrays

Arrays hold collections of elements of the same type.  
One-dimensional arrays are lists. Multi-dimensional arrays are arrays of arrays, like matrices.

### Lab Exercise: One-Dimensional and Two-Dimensional Arrays

#include <stdio.h>

int main() {

int arr[5];

printf("Enter 5 integers:\n");

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

scanf("%d", &arr[i]);

}

printf("You entered: ");

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

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

}

printf("\n");

int matrix[3][3], sum = 0;

printf("Enter 9 integers for 3x3 matrix:\n");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

scanf("%d", &matrix[i][j]);

sum += matrix[i][j];

}

}

printf("Sum of all elements: %d\n", sum);

return 0;

}

## **10. Pointers in C**

### Theory Exercise: What Are Pointers?

Pointers hold memory addresses of variables. They let you directly access and manipulate memory, essential for efficiency and advanced programming techniques.

### Lab Exercise: Pointer Usage

#include <stdio.h>

int main() {

int num = 10;

int \*ptr = #

printf("Value of num: %d\n", num);

printf("Value via pointer: %d\n", \*ptr);

\*ptr = 20;

printf("New value of num: %d\n", num);

return 0;

}

## **11. Strings in C**

### Theory Exercise: String Handling Functions

* strlen() – Returns string length.
* strcpy() – Copies strings.
* strcat() – Concatenates strings.
* strcmp() – Compares strings.
* strchr() – Finds character in string.

### Lab Exercise: Concatenate Two Strings

#include <stdio.h>

#include <string.h>

int main() {

char str1[100], str2[100];

printf("Enter first string: ");

fgets(str1, sizeof(str1), stdin);

printf("Enter second string: ");

fgets(str2, sizeof(str2), stdin);

str1[strcspn(str1, "\n")] = 0; // Remove newline

str2[strcspn(str2, "\n")] = 0;

strcat(str1, str2);

printf("Concatenated String: %s\n", str1);

printf("Length: %lu\n", strlen(str1));

return 0;

}

## **12. Structures in C**

### Theory Exercise: Structures Concept

Structures combine different data types into one. You declare them with struct, initialize with values, and access members via dot notation.

### Lab Exercise: Store 3 Students' Details

#include <stdio.h>

struct Student {

char name[50];

int roll\_no;

float marks;

};

int main() {

struct Student students[3];

for (int i = 0; i < 3; i++) {

printf("Enter name for student %d: ", i+1);

scanf("%s", students[i].name);

printf("Enter roll number: ");

scanf("%d", &students[i].roll\_no);

printf("Enter marks: ");

scanf("%f", &students[i].marks);

}

printf("\nStudents' Details:\n");

for (int i = 0; i < 3; i++) {

printf("Name: %s, Roll: %d, Marks: %.2f\n", students[i].name, students[i].roll\_no, students[i].marks);

}

return 0;

}

## **13. File Handling in C**

### Theory Exercise: Importance and Operations

File handling allows storing data permanently. Key operations include opening, closing, reading, and writing files using standard library functions like fopen(), fclose(), fprintf(), and fgets().

### Lab Exercise: Write and Read a File

#include <stdio.h>

int main() {

FILE \*file;

char buffer[100];

file = fopen("myfile.txt", "w");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

fprintf(file, "This is a test string.\n");

fclose(file);

file = fopen("myfile.txt", "r");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

fgets(buffer, sizeof(buffer), file);

printf("File content: %s", buffer);

fclose(file);

return 0;

}

## Extra Lab Exercises to Improve Programming Logic

### 1. Simple Calculator

#include <stdio.h>

int main() {

float num1, num2;

char op;

printf("Enter two numbers separated by space: ");

scanf("%f %f", &num1, &num2);

printf("Enter operator (+ - \* / %): ");

scanf(" %c", &op);

switch(op) {

case '+': printf("Result: %.2f\n", num1 + num2); break;

case '-': printf("Result: %.2f\n", num1 - num2); break;

case '\*': printf("Result: %.2f\n", num1 \* num2); break;

case '/':

if(num2 != 0)

printf("Result: %.2f\n", num1 / num2);

else

printf("Error: Division by zero\n");

break;

case '%':

if((int)num2 != 0)

printf("Result: %d\n", (int)num1 % (int)num2);

else

printf("Error: Division by zero\n");

break;

default: printf("Invalid operator\n");

}

return 0;

}

### 2. Grade Calculator

#include <stdio.h>

int main() {

int marks;

printf("Enter marks: ");

scanf("%d", &marks);

if (marks > 90) printf("Grade A\n");

else if (marks > 75) printf("Grade B\n");

else if (marks > 50) printf("Grade C\n");

else printf("Grade D\n");

return 0;

}

### 3. Prime Number Check

#include <stdio.h>

#include <stdbool.h>

bool isPrime(int n) {

if (n <= 1) return false;

for(int i = 2; i\*i <= n; i++) {

if (n % i == 0) return false;

}

return true;

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

if (isPrime(number))

printf("%d is a prime number.\n", number);

else

printf("%d is not a prime number.\n", number);

return 0;

}

### 4. Maximum and Minimum in Array

#include <stdio.h>

int main() {

int arr[10], max, min;

printf("Enter 10 integers:\n");

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

scanf("%d", &arr[i]);

}

max = min = arr[0];

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

if(arr[i] > max) max = arr[i];

if(arr[i] < min) min = arr[i];

}

printf("Maximum: %d\n", max);

printf("Minimum: %d\n", min);

return 0;

}

### 5. Fibonacci Sequence (Recursive)

#include <stdio.h>

int fibonacci(int n) {

if (n == 0) return 0;

if (n == 1) return 1;

return fibonacci(n-1) + fibonacci(n-2);

}

int main() {

int terms;

printf("Enter number of terms: ");

scanf("%d", &terms);

printf("Fibonacci sequence:\n");

for (int i = 0; i < terms; i++) {

printf("%d ", fibonacci(i));

}

printf("\n");

return 0;

}

### 6. String Reversal Without Built-in Functions

#include <stdio.h>

int main() {

char str[100];

int i, len = 0;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

// Calculate length excluding newline

while(str[len] != '\0' && str[len] != '\n') len++;

printf("Reversed string: ");

for(i = len - 1; i >= 0; i--) {

putchar(str[i]);

}

printf("\n");

return 0;

}

### Lab Challenge: Number Guessing Game

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main() {

int target, guess, attempts = 5;

srand(time(NULL));

target = rand() % 100 + 1; // Random 1-100

printf("Guess the number (1-100). You have %d attempts.\n", attempts);

while(attempts > 0) {

printf("Enter your guess: ");

scanf("%d", &guess);

if (guess == target) {

printf("Correct! You win!\n");

break;

} else if (guess < target) {

printf("Too low. Try again.\n");

} else {

printf("Too high. Try again.\n");

}

attempts--;

}

if (attempts == 0) {

printf("Game over. The number was %d.\n", target);

}

return 0;

}

**5. Fibonacci Sequence (Recursive)**

#include <stdio.h>

int fibonacci(int n) {

if (n == 0) return 0;

if (n == 1) return 1;

return fibonacci(n-1) + fibonacci(n-2);

}

int main() {

int terms;

printf("Enter number of terms: ");

scanf("%d", &terms);

printf("Fibonacci sequence:\n");

for (int i = 0; i < terms; i++) {

printf("%d ", fibonacci(i));

}

printf("\n");

return 0;

}

**6. String Reversal Without Built-in Functions**

#include <stdio.h>

int main() {

char str[100];

int i, len = 0;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

// Calculate length excluding newline

while(str[len] != '\0' && str[len] != '\n') len++;

printf("Reversed string: ");

for(i = len - 1; i >= 0; i--) {

putchar(str[i]);

}

printf("\n");

return 0;

}

**Lab Challenge: Number Guessing Game**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main() {

int target, guess, attempts = 5;

srand(time(NULL));

target = rand() % 100 + 1; // Random 1-100

printf("Guess the number (1-100). You have %d attempts.\n", attempts);

while(attempts > 0) {

printf("Enter your guess: ");

scanf("%d", &guess);

if (guess == target) {

printf("Correct! You win!\n");

break;

} else if (guess < target) {

printf("Too low. Try again.\n");

} else {

printf("Too high. Try again.\n");

}

attempts--;

}

if (attempts == 0) {

printf("Game over. The number was %d.\n", target);

}

return 0;

}