Module – 2 Introduction to Programming

**Overview of C Programming**

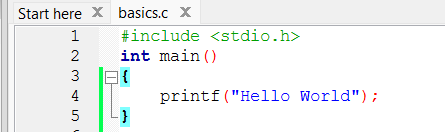
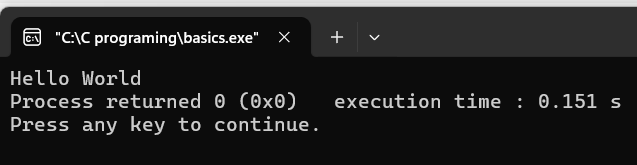
1. **History and evolution of C programming. Explain its importance and why it is still used today.**

* C programming, developed by Dennis Ritchie in the early 1970s, is a foundational language in computer science, known for its efficiency, portability, and influence on subsequent programming languages.
* C programming is still widely used today due to its efficiency, flexibility, and close-to-hardware capabilities.

**Lab Ex:-** Three real-world applications where C programming is extensively used-Operating systems like Linux, macOS, and Android rely on C for core functionalities.

**Setting Up Environment**

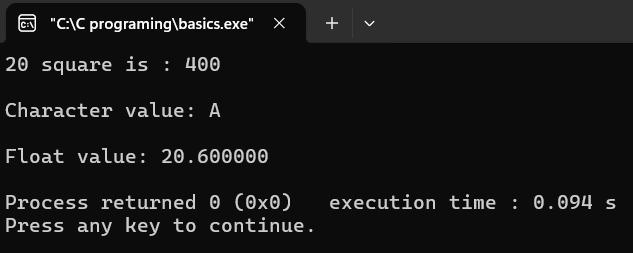
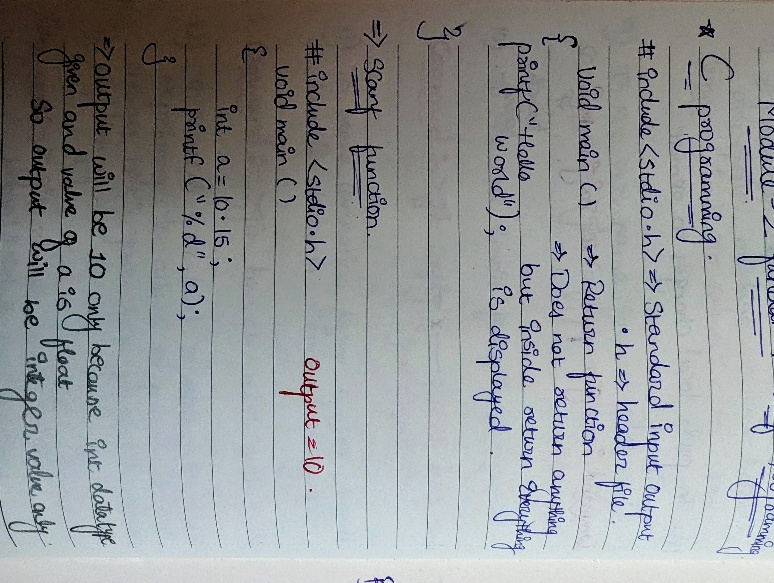
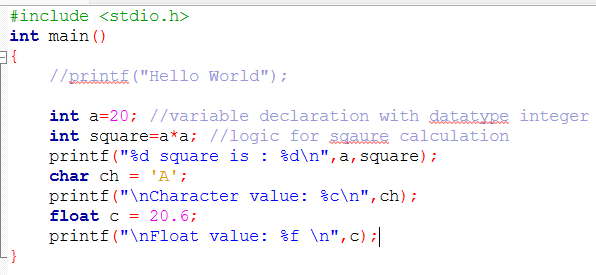
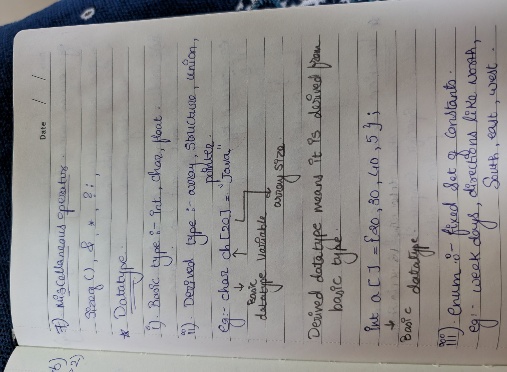
1. **Lab Ex:-** Write your first program to print "Hello, World!" and run it.



**Basic Structure of a C Program**

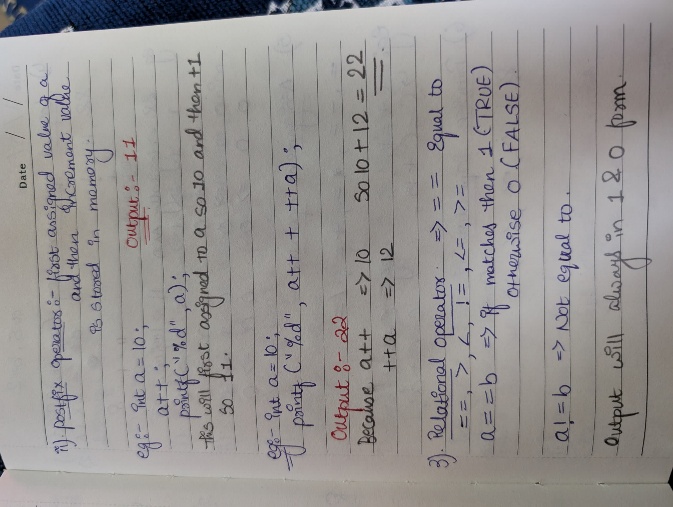
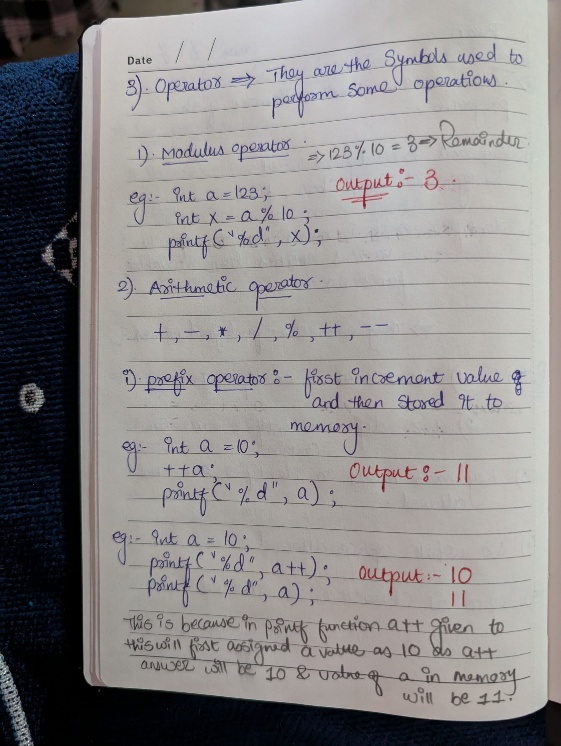
1. **Explain the basic structure of a C program, including headers, main function, comments, datatypes, and variables. Provide examples.**

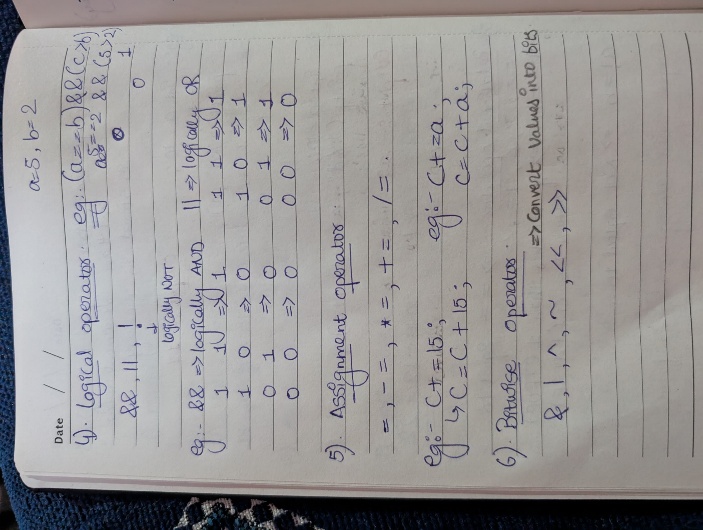
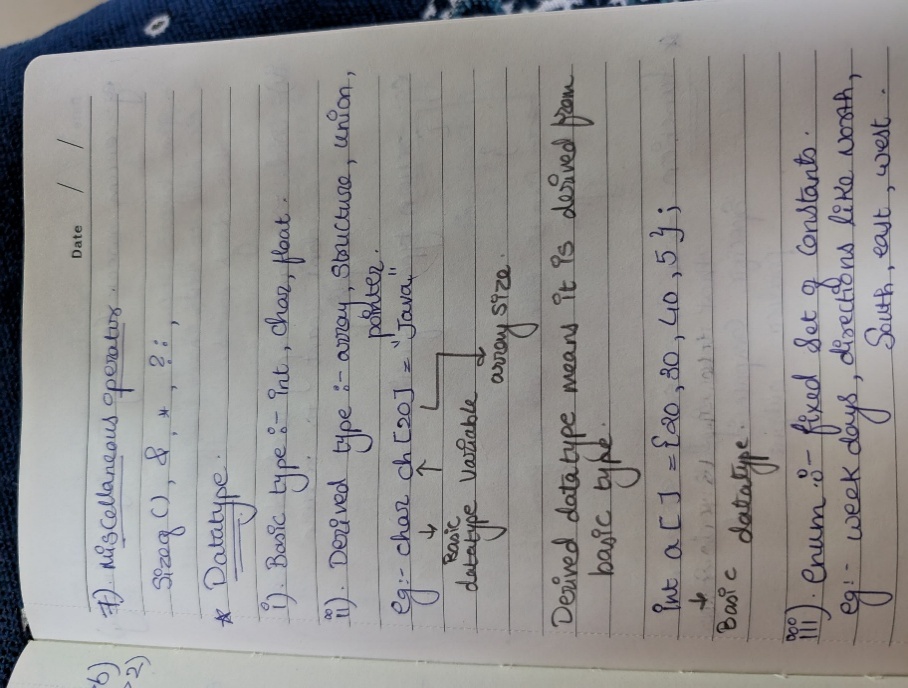
**Lab Ex:-** Write a C program that includes variables, constants, and comments. Declare and use different data types (int, char, float) and display their values.



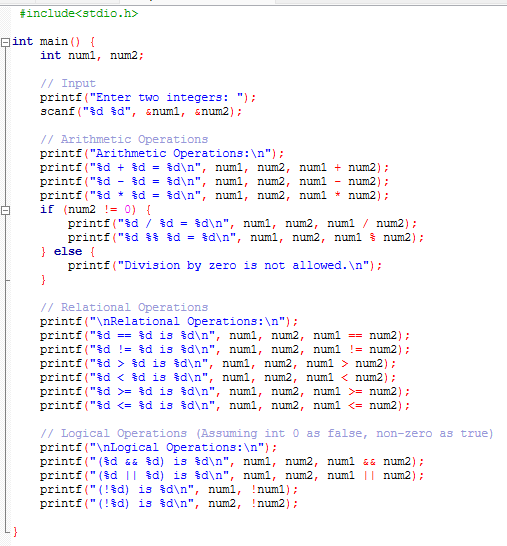
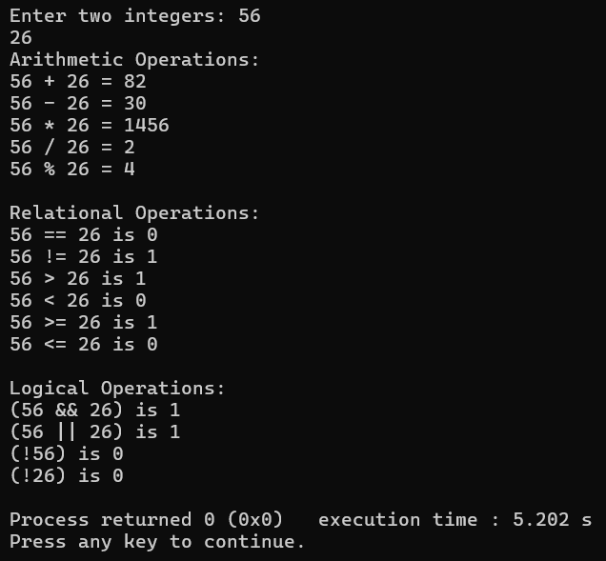
**Operators in C**

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



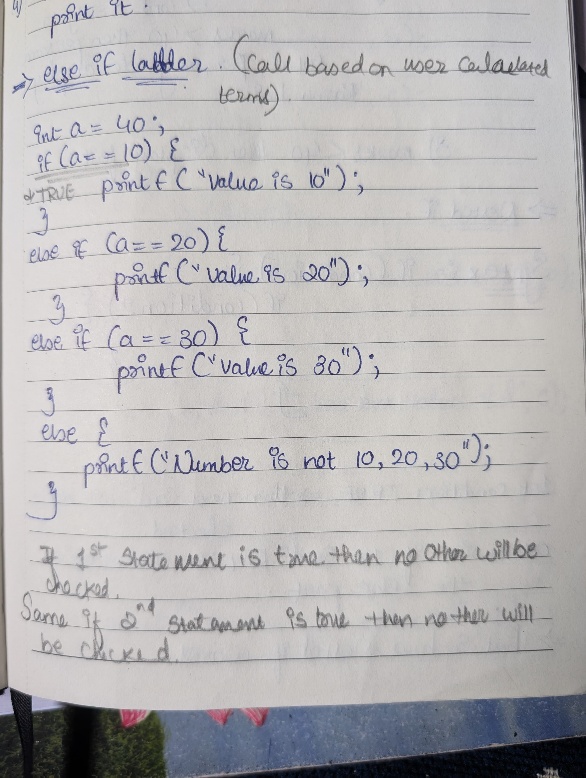
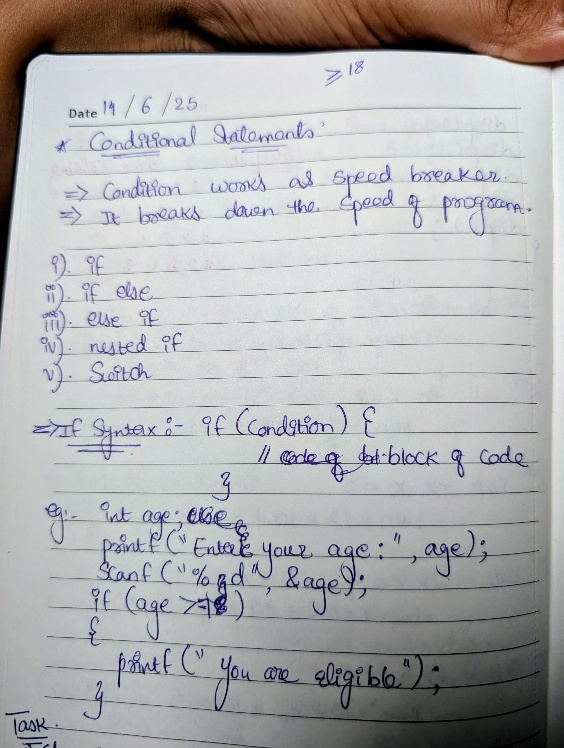


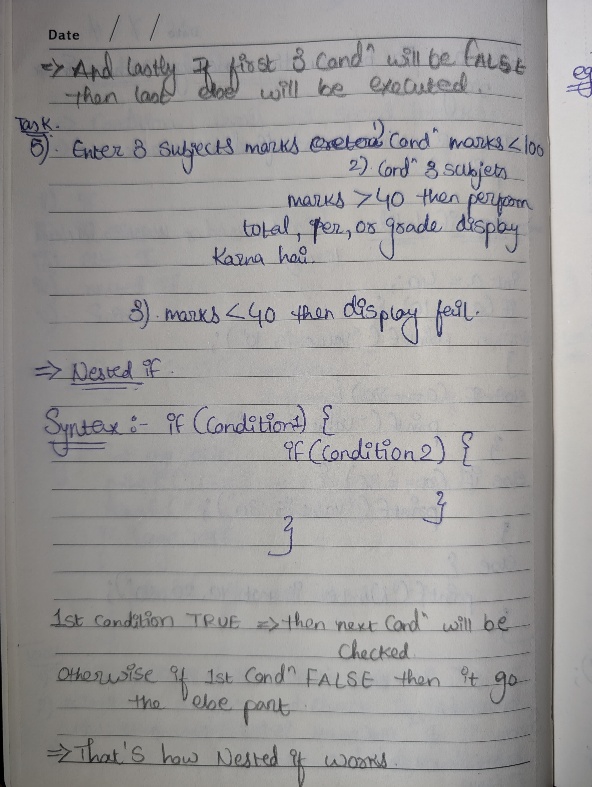
**Lab Ex:-** Write a C program that accepts two integers from the user and performs arithmetic, relational, and logical operations on them. Display the results.

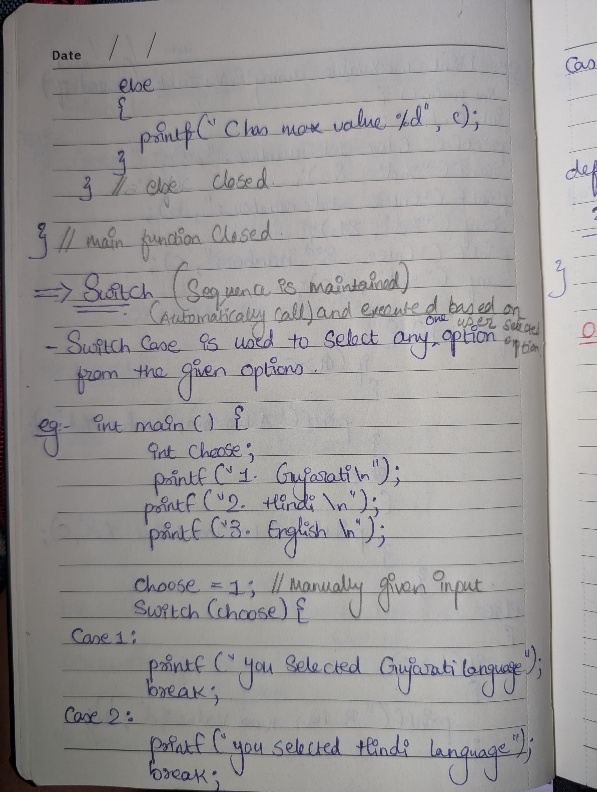
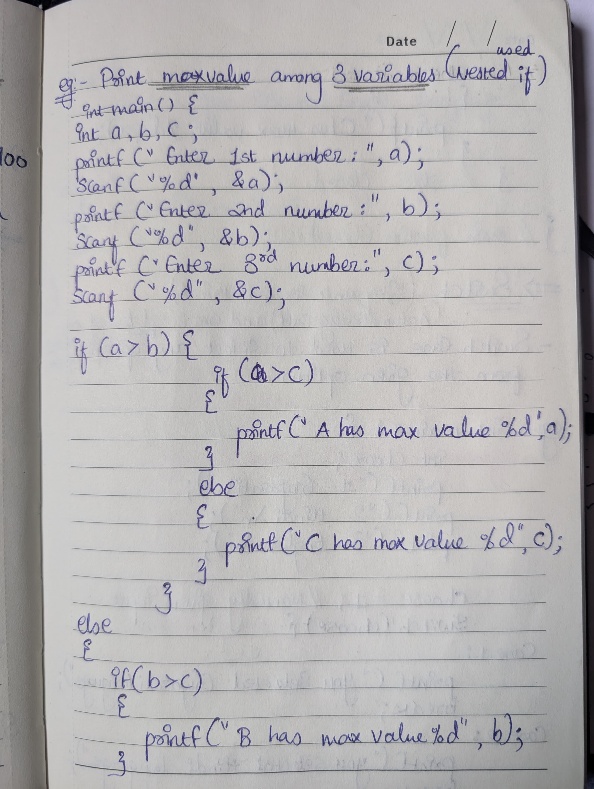


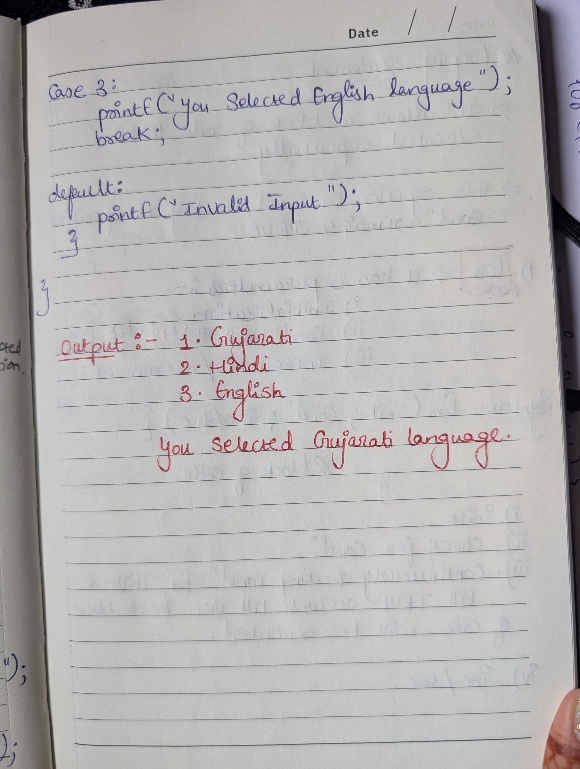
**Control Flow Statements in C**

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

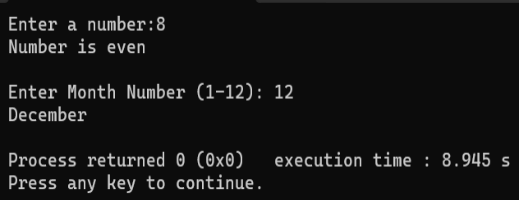
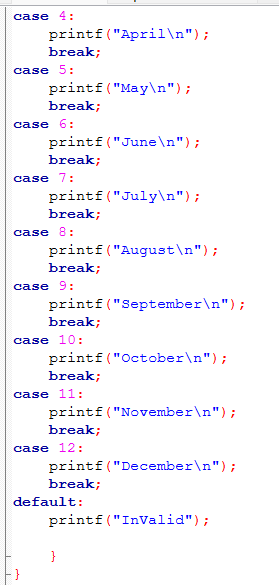
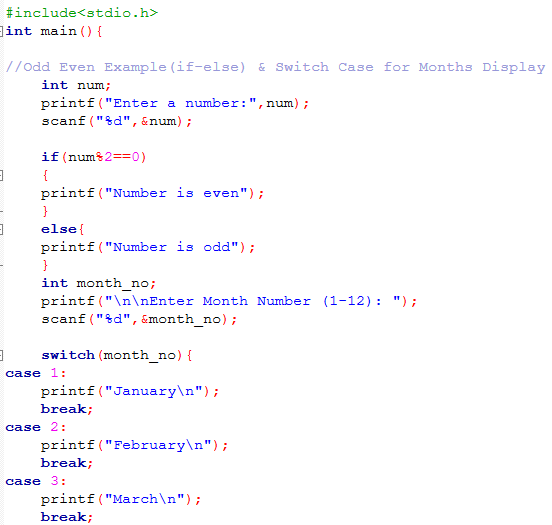








**Lab Ex:-** Write a C program to check if a number is even or odd using an if-else statement. Extend the program using a switch statement to display the month name based on the user’s input (1 for January, 2 for February, etc.).

**Looping in C**

1. **Compare and contrast while loops, for loops, and do-while loops.**

* **For Loop:** Structure.

for (initialization; condition; increment/decrement) {

// block of code

}

* **Execution:** The initialization statement executes once at the beginning. The condition is checked before each iteration, and the loop continues as long as it's true. The increment/decrement statement executes after each iteration. For loops are typically used when the number of iterations is known or can be easily determined.
* **While Loop:** Structure.

while (condition) {

// block of code

}

* **Execution:** The condition is checked before each iteration. The loop body executes only if the condition is true. If the condition is initially false, the loop body will not execute even once.
* **Do-While Loop:** Structure.

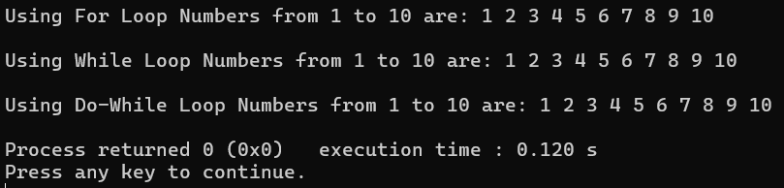
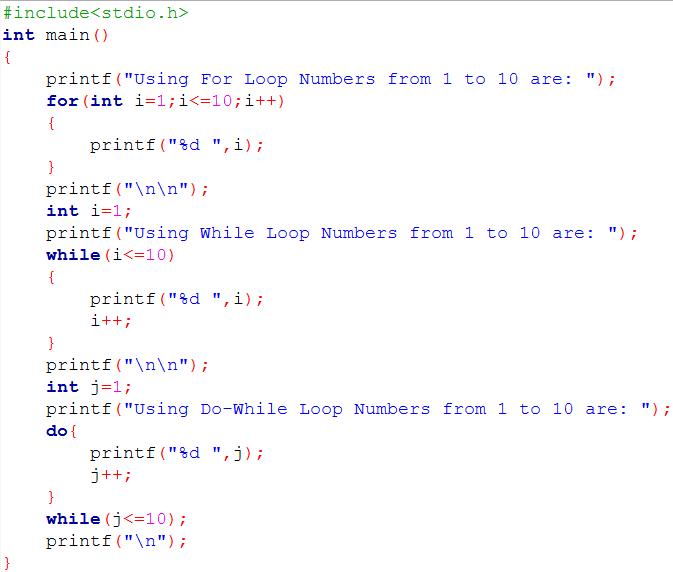
do {

// block of code

} while (condition);

* **Execution:** The loop body executes at least once before the condition is checked. After the first execution, the condition is checked, and the loop continues to execute as long as the condition remains true.

**Lab Ex**:- Write a C program to print numbers from 1 to 10 using all three types of loops (while, for, do-while).

**Functions in C**

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

**Function:-** It is a set of statements that take inputs, do some specific computations and produce output.

**Function Declaration:-**

int add(int a, int b); // Declares a function named 'add' that takes two integers and returns an integer.

**Function Definition:-**

A function definition provides the actual implementation (body) of the function.

int add(int a, int b) {

return a + b; // Returns the sum of a and b

}

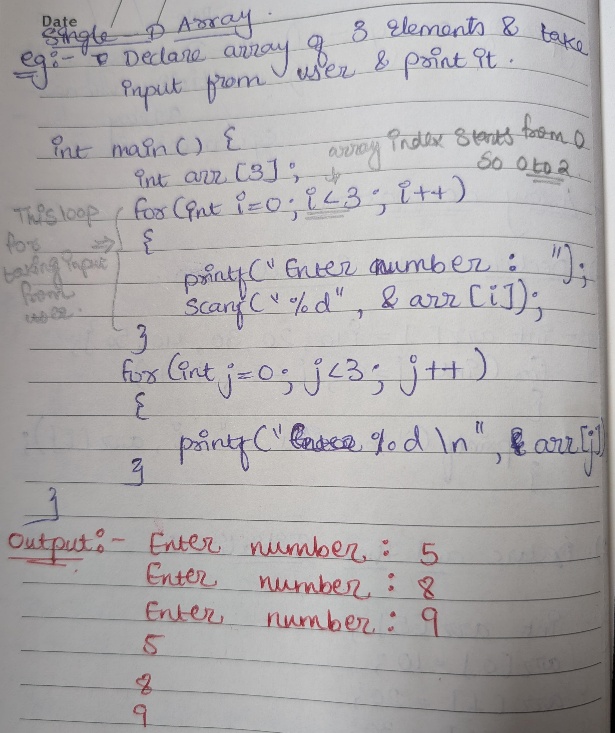
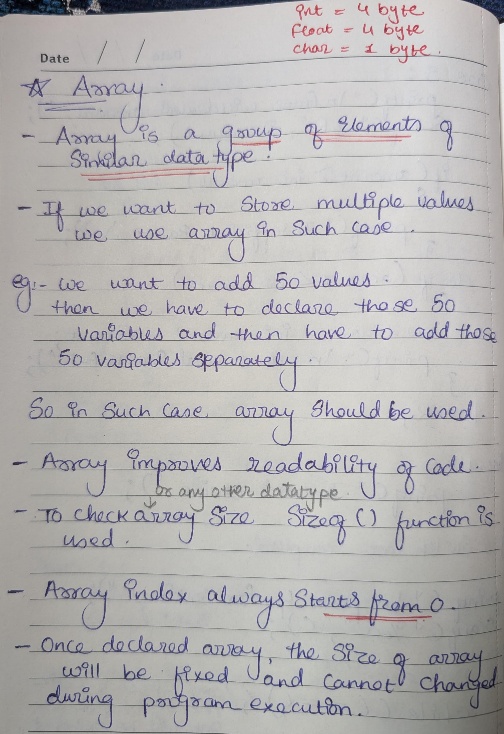
**Function Call:-**

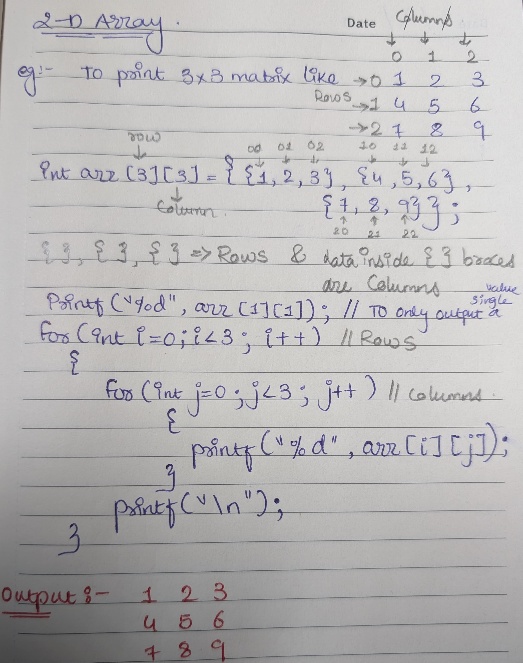
To execute a function, it must be "called" from another part of the program (e.g., from main())

add(num1, num2);//calls the add function, passing num1 and num2 as arguments.

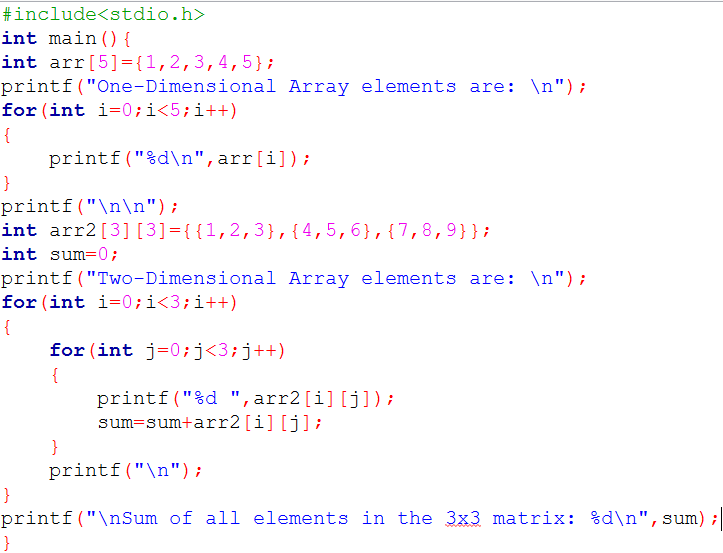
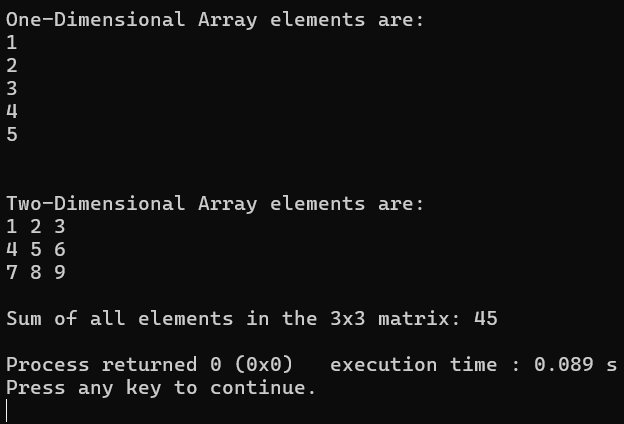
**Arrays in C**

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



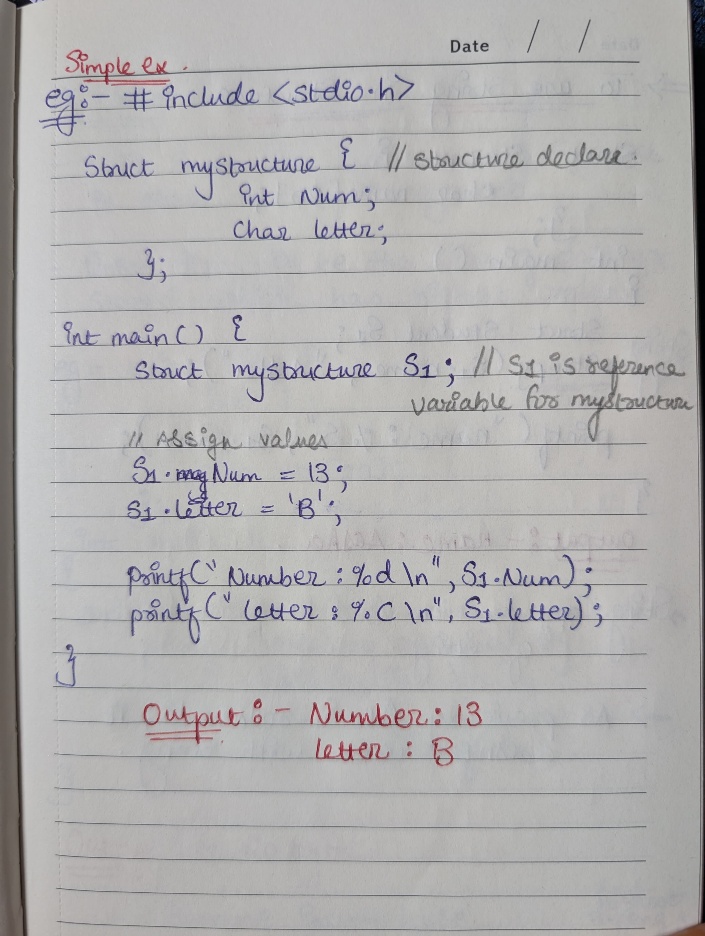
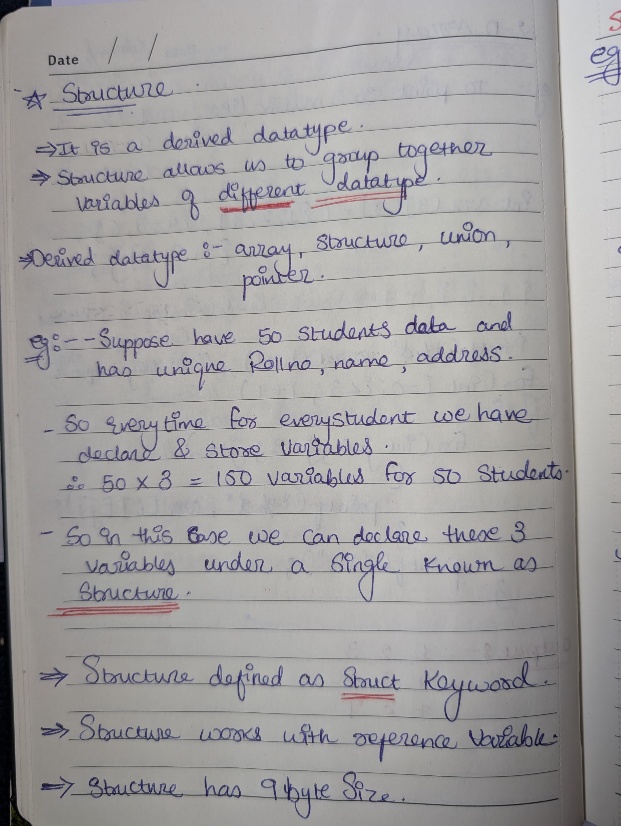


**Lab Ex**:- Write a C program that stores 5 integers in a one-dimensional array and prints them. Extend this to handle a two-dimensional array (3x3 matrix) and calculate the sum of all elements.

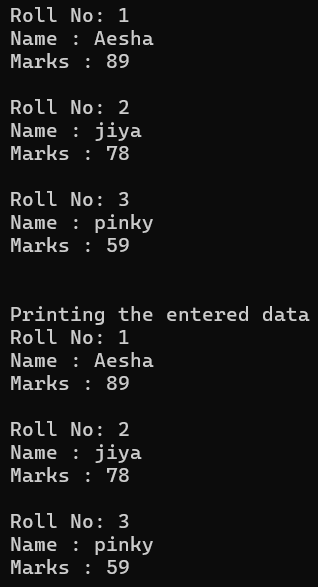
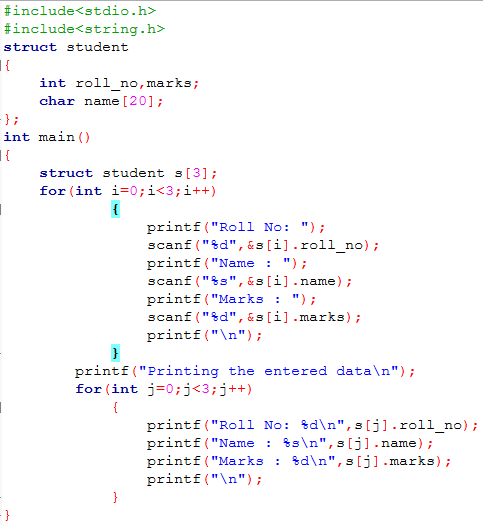
 

**Structures in C**

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

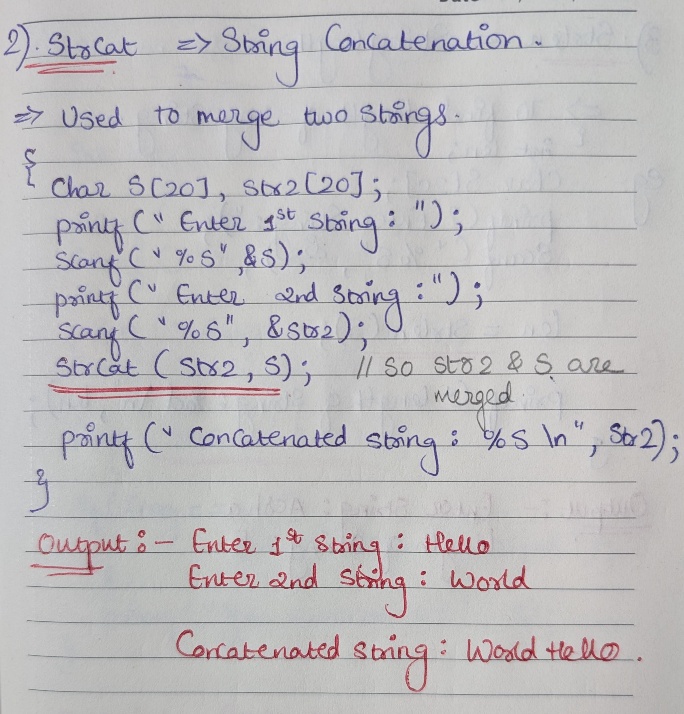
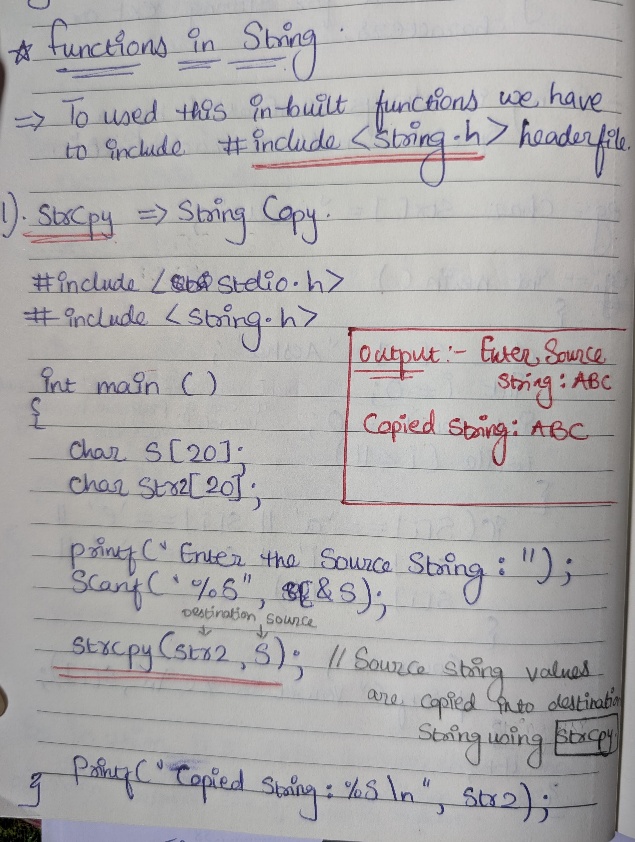
****

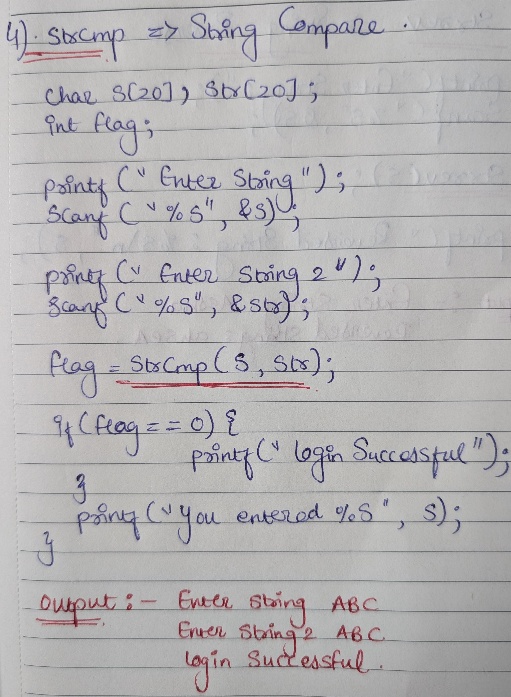
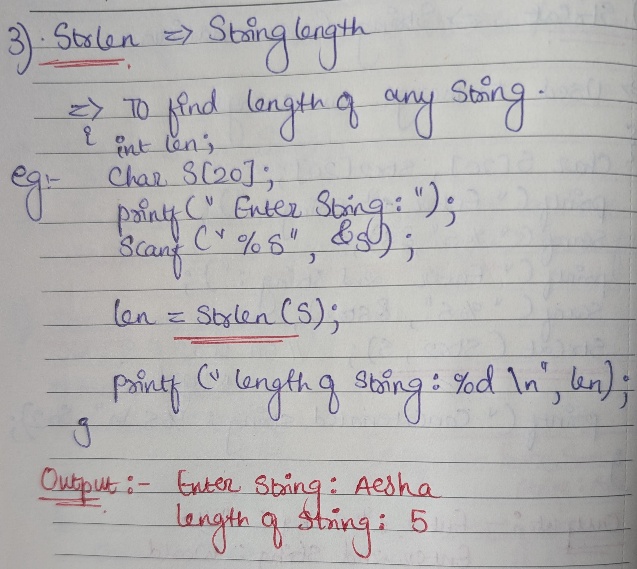
**Lab Ex:-** Write a C program that defines a structure to store a student's details (name, roll number, and marks). Use an array of structures to store details of 3 students and print them.



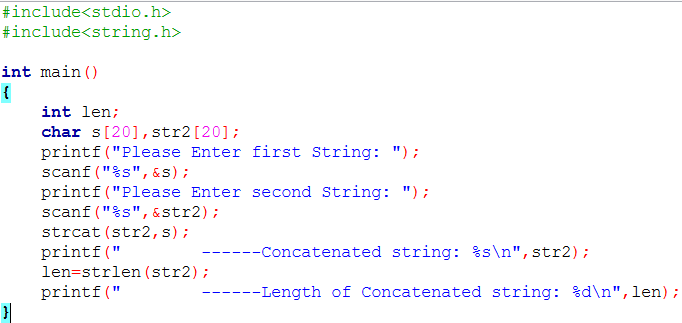
**Strings in C**

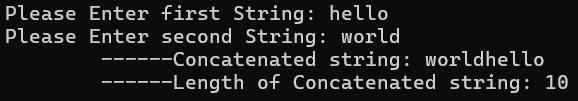
1. **Explain string handling functions like strlen(), strcpy(), strcat(), strcmp(), and strchr(). Provide examples of when these functions are useful.**

****

****

**Lab Ex:-** Write a C program that takes two strings from the user and concatenates them using strcat(). Display the concatenated string and its length using strlen().





**Pointers in C**

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

* Pointer is a variable used to store the memory address of another variable.
* **Declaration:-** data\_type \*pointer\_name; //\* is pointer variable.

Eg:- int \*ptr; // Declares a pointer named ptr that can hold the address of an integer variable.

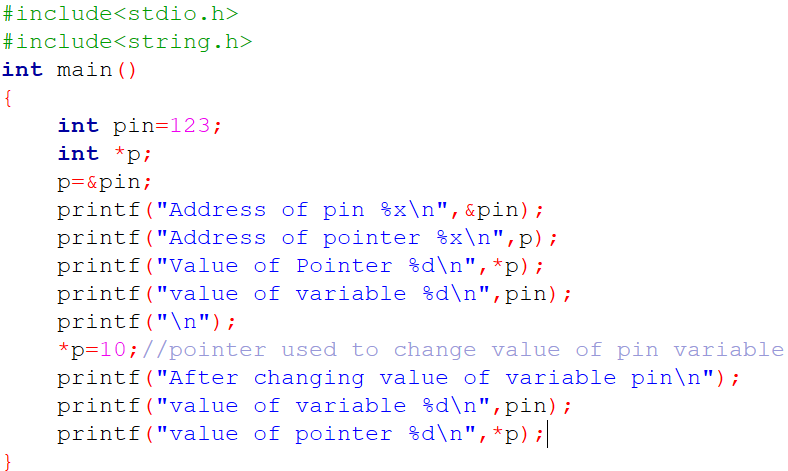
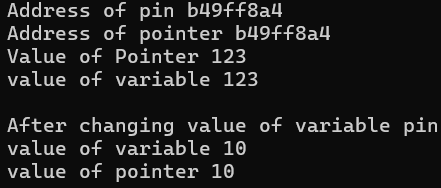
* **Initialization:-** int pin = 123;

int \*ptr;

ptr = &pin; // Initializes ptr to the address of pin variable.

* Instead of holding the actual value, pointer points to address where that value is stored. This allows for efficient memory management and powerful operations like accessing and modifying data directly. So that’s why it is important in c.

**Lab Ex:-** Write a C program to demonstrate pointer usage. Use a pointer to modify the value of a variable and print the result.

**File Handling in C**

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

* File handling refers to how we store the available data or info in a file with the help of a program.
* Key operations include opening, closing, reading, and writing files.
* Without file handling, data is lost when a program terminates. File handling allows data to be saved to a file and reused later.
  1. Opening a File:-
* The **fopen()** function is used to open a file. It takes the filename and mode as arguments.
* Modes:
* "r": Opens for reading.
* "w": Opens for writing (creates a new file).
* "a": Opens for appending (adds to the end of an existing file).
* Eg:- FILE \*fp; // file pointer declare

fp = fopen(“file1.txt”,”r”);

* 1. Closing a File:-
* The **fclose()** function is used to close an opened file.
* Eg:- fclose(fp); //fp is a file pointer.
  1. Reading from a File:-
* **fscanf()** Reads formatted data from a file.
* Eg:- fscanf(fp, "%d %s", &num, str);
* **fgetc()** Reads a single character from a file.
* Eg:- char ch = fgetc(fp);

**Lab Ex:-** Write a C program to create a file, write a string into it, close the file, then open the file again to read and display its contents.

