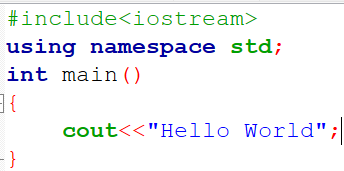
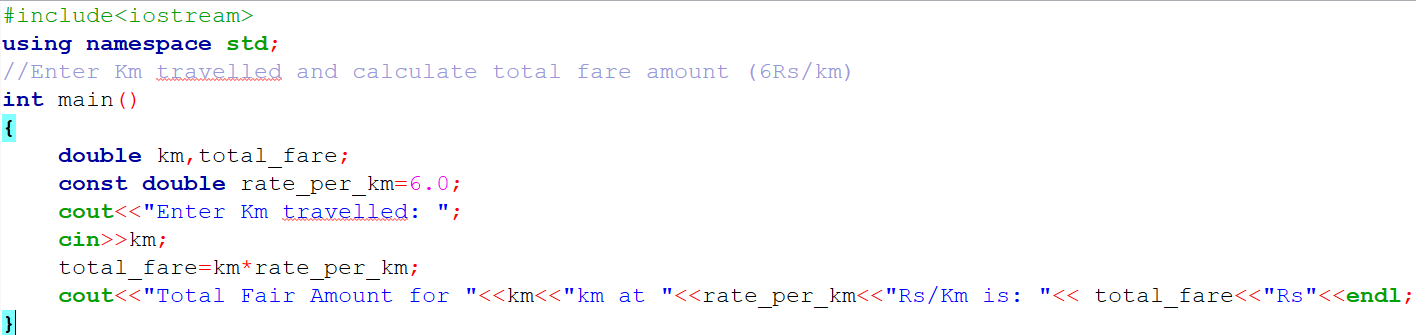
**Module #3 Introduction to OOPS Programming**

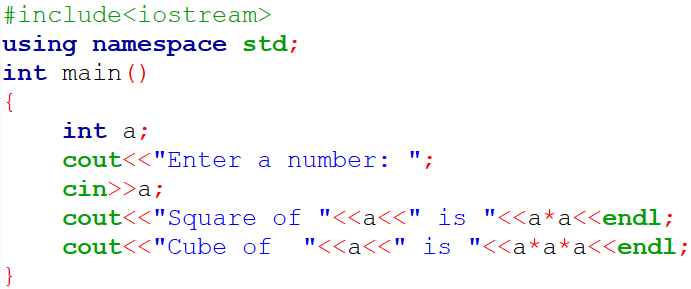
1. **Introduction to C++**

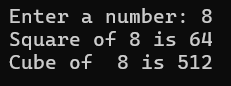
**Lab Exercises:-**

* Write a simple C++ program to display "Hello, World!".



* Basic Input/Output Program.

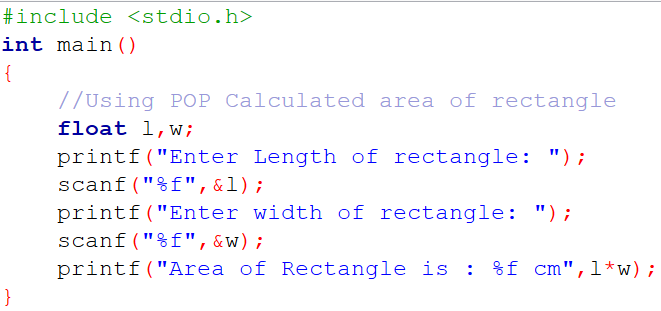
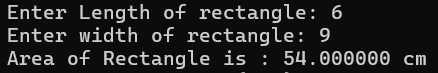


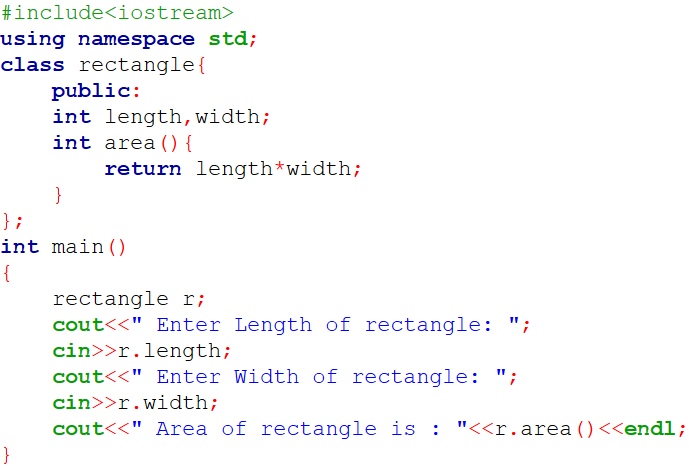
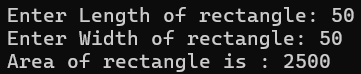


* Write two small programs: one using Procedural Programming (POP) to calculate the

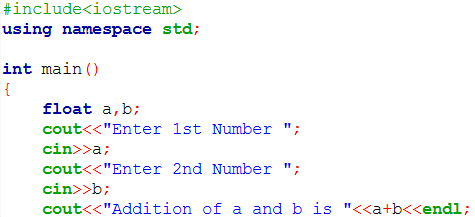
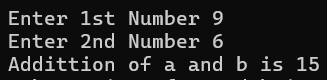
area of a rectangle, and another using Object-Oriented Programming (OOP) with a

class and object for the same task.

* Write a program that asks for two numbers and displays their sum.

**Theory Exercises:-**

* ***What are the key differences between Procedural Programming and Object-Oriented Programming (OOP)?***

1. **Procedural Programming**: Focuses on **functions and procedures** (step-by-step instructions).

**OOP**: Focuses on **objects** (real-world entities with data + behavior).

1. **Procedural:** Program is divided into functions.

**OOP:** Program is divided into classes and objects.

1. **Procedural:** Data is shared and can be accessed by any function.

**OOP:** Data is encapsulated (hidden inside objects); accessed via methods.

1. **Procedural:** Less secure (data can be modified by any function).

**OOP:** More secure (data hiding via encapsulation).

1. **Procedural**: C

**OOP**: C++, Java

* ***List and explain the main advantages of OOP over POP.***

| **Advantage** | **Description** |
| --- | --- |
| 1. Modularity | OOP organizes code into classes and objects, making it more modular and manageable. POP uses long procedures, making large programs hard to manage. |
| 2. Reusability | In OOP, classes can be reused in other programs using inheritance. In POP, code reuse is limited to copying functions. |
| 3. Data Security | OOP uses encapsulation (data hiding), protecting internal data. POP uses global data, making it less secure. |
| 4. Easy Maintenance | OOP code is easier to update and debug due to separation into smaller, manageable classes. |
| 5. Code Readability | OOP provides clear structure with classes, improving readability compared to flat procedural code. |

* ***What are the main input/output operations in C++? Provide examples.***
* Input:- used to take user input – **cin**
* Output:- used to display ouput on screen – **cout**

Eg:-

#include <iostream> output:- Enter your age: 20

using namespace std; You entered: 20

int main() {

int age;

cout << "Enter your age: ";

cin >> age; // Taking input from user

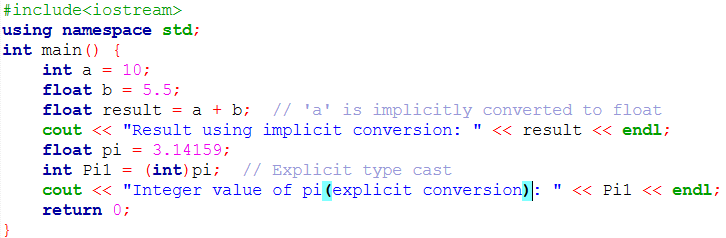
cout << "You entered: " << age << endl;

return 0;

}

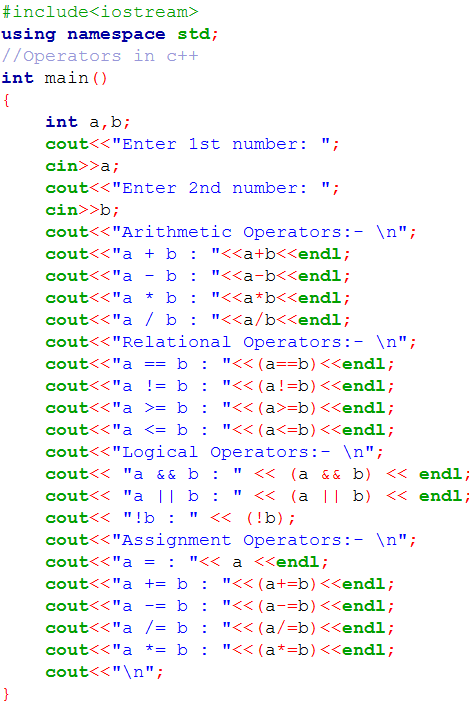
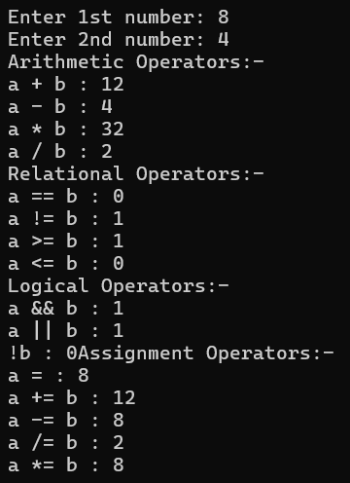
1. **Variables, Data Types, and Operators**

* Write a C++ program that performs both implicit and explicit type conversions and prints the results.





* Write a C++ program that demonstrates arithmetic, relational, logical, and bitwise operators. Perform operations using each type of operator and display the results.



**Theory Exercises:-**

* ***What are the different data types available in C++? Explain with examples.***

1. **Basic datatypes:-** a. **int** - int age = 30;
2. **Float** – float temperature = 25.5f; // 'f' suffix for float literal
3. **Char** – char letter = ‘A’;
4. **Double** – double pi = 3.1415926535;
5. **Derived datatypes:-** a. **Array** - int numbers[5] = {1, 2, 3, 4, 5};

b. **Pointer** - int\* ptr;

int value = 10;

ptr = &value; // ptr now holds the address of 'value'

c. **Functions** - int add(int a, int b) {

return a + b;

}

1. **User – Defined datatypes:-** a. **struct(Structure)** -

struct Student {     
int rollno;         // int variable  
string name;   // string variable

};

int main(){

Student s1; // declare structure variable

s1.rollno = 8;

s1.name = “Aesha”;

}

b**. Class** - class Car { // class declared  
 public:  
 string model;  
 int year;  
 };

Car c1; // car object created

c. **enum (Enumerations)** –

enum Day { Monday, Tuesday, Wednesday };

* ***Explain the difference between implicit and explicit type conversion in C++.***
* **Implicit type conversion :- Implicit conversion** is performed automatically by the compiler without any explicit instruction from the programmer.
* It occurs when converting from a smaller data type to a larger one (e.g., int to double).
* **Explicit type conversion :-** **Explicit conversion**, also known as **type casting** , occurs when the programmer explicitly(manually) converts a value from one data type to another.
* ***What are the different types of operators in C++? Provide examples of each.***

1. Arithmetic Operator:- Used for mathematical calculations.

+, - ,\* , / , %(modulus).

1. Relational Operator:- Used to compare two values.

==, != , > , < , >= , <=.

1. Logical Operator:- Used to combine conditional statements.

&& , || , !

1. Assignment Operator:- Used to assign values to variables.

= , += , -= , \*= , /= , %=

1. Bitwise Operator:- Operator on bits of numbers.

& , | , ~ , ^ , << , >>

1. Increment and Decrement Operator:- Used to increase or decrease the value of a variable by 1.

++ , --

1. Miscellaneous Operator:-

condition ? expr1 : expr2; , sizeof() , &(address) , \*(pointer)

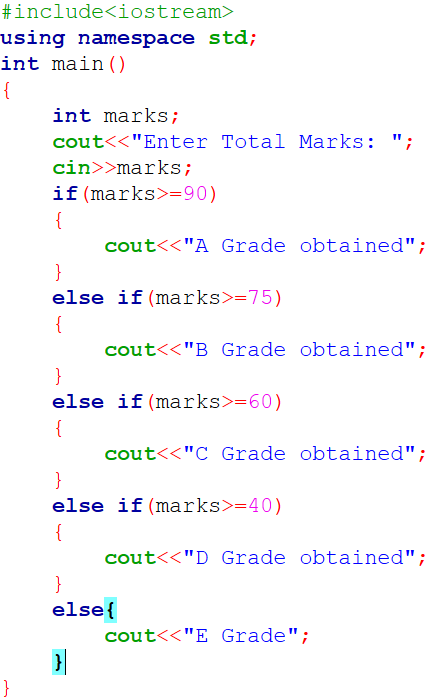
* ***Explain the purpose and use of constants and literals in C++.***
* **Constants** are fixed values that do not change during program execution. Once defined, their value cannot be modified.
* Makes code more **readable** and **maintainable**.
* Used for **fixed values** like PI, conversion rates, limits, etc.
* Eg:- const float PI = 3.14;
* **Literals** are the actual **fixed values** used directly in the code. They are **used** to **assign values** to variables or constants.
* Eg:- int age = 21; // 21 is an integer literal

char grade = 'A'; // 'A' is a character literal

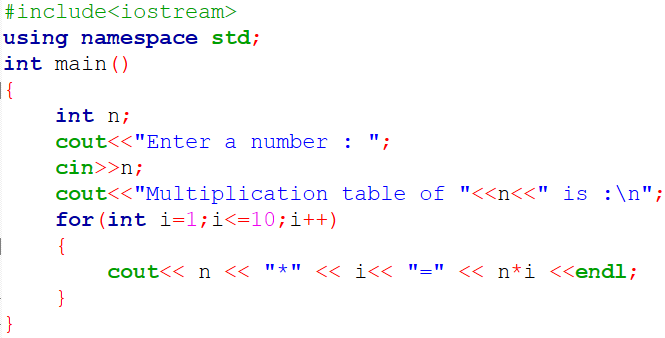
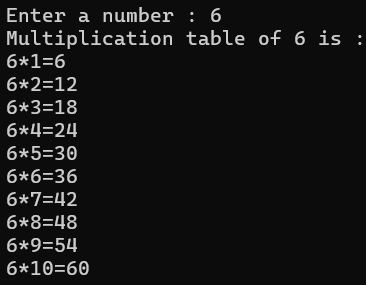
string name = "Aesha"; // "Aesha" is a string literal

1. **Control Flow Statements**

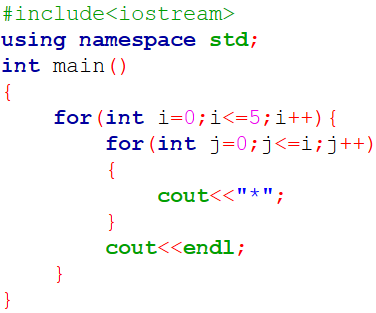
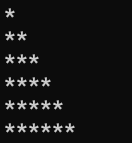
* Write a C++ program that takes a student’s marks as input and calculates the grade based on if-else conditions.



* Write a C++ program to display the multiplication table of a given number using a for loop.

* Write a program that prints a right-angled triangle using stars (\*) with a nested loop.



**Theory Exercises:-**

* ***What are conditional statements in C++? Explain the if-else and switch statements.***

Conditional statements in C++ allows a program to make decisions and execute different blocks of code based on whether a specific condition is true or false.

* **If-Else Statement:** if (condition) {

// Code to be executed if the condition is true

} else {

// Code to be executed if the condition is false

}

The if-else statement is used to execute one block of code if a condition is true and another block if the condition is false.

* **Switch Statement:** switch (choice) {

case constant1:

// Code to be executed if choice matches constant1

break;

case constant2:

// Code to be executed if choice matches constant2

break;

// ... more cases

default:

// Code to be executed if no case matches the choice

}

The switch statement is a multi-way branch statement that is used to select any one option from the given options. In switch case sequence is maintained. Switch case is automatically called and executed based on user selected options.

* ***What is the difference between for, while, and do-while loops in C++?***
* **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 {

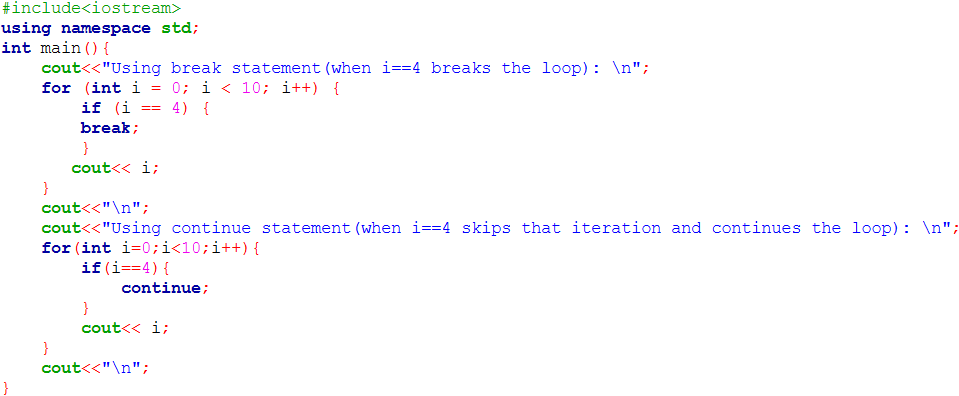
// block of code

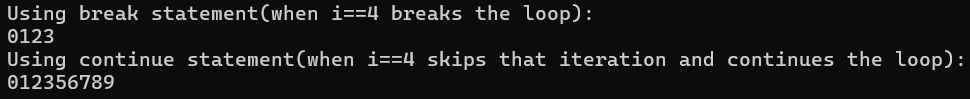
} while (condition);

* **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.
* ***How are break and continue statements used in loops? Provide examples.***

The **break** and **continue** statements are used to **control** the **flow** **of** **execution** within loops.

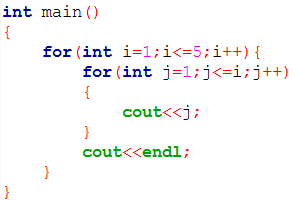
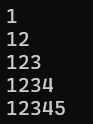
**Break** immediately exits the loop, while **continue** skips the current iteration and proceeds to the next.





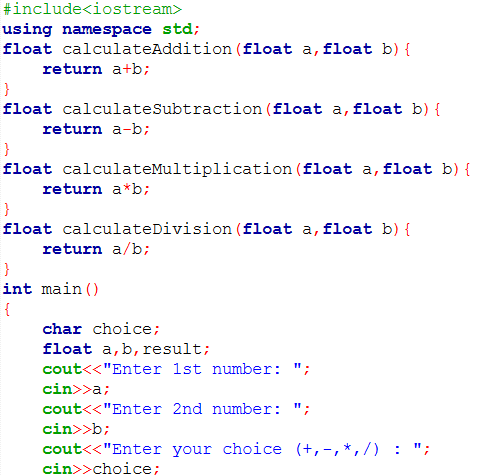
* ***Explain nested control structures with an example.***

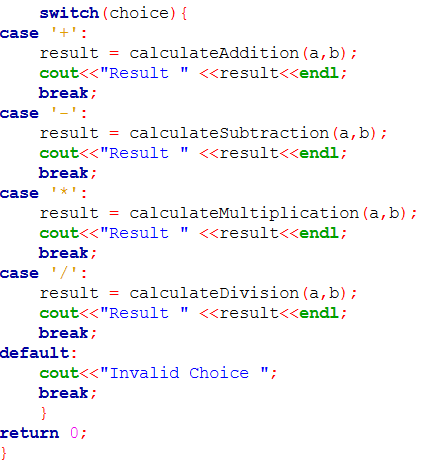
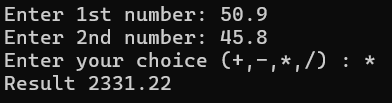
A nested control structure refers to a control flow statement placed inside another control flow statement. For eg:- for loop inside another for loop.

* *

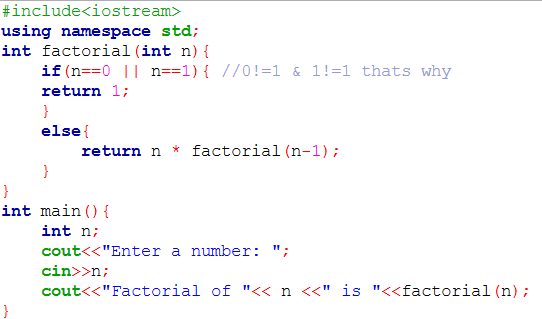
1. **Functions and Scope**

* Write a C++ program that defines functions for basic arithmetic operations (add, subtract, multiply, divide). The main function should call these based on user input.



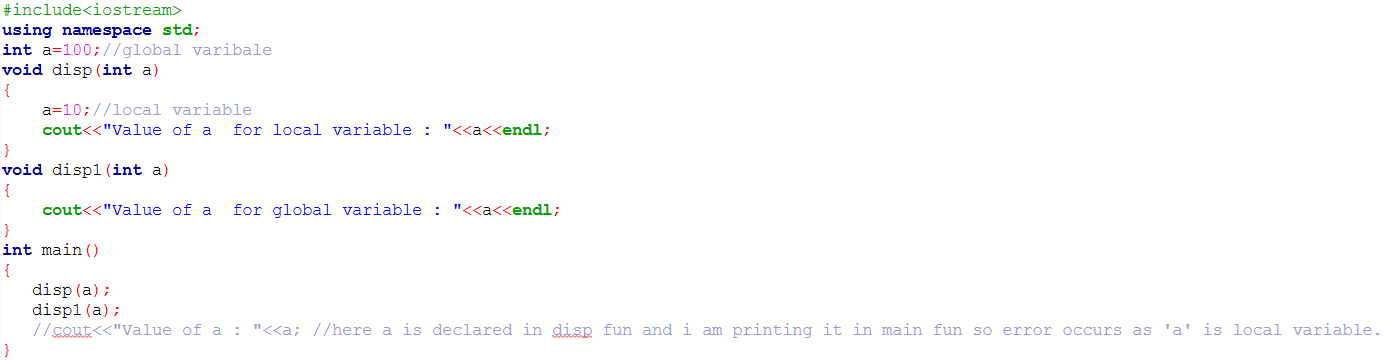
 

* Write a C++ program that calculates the factorial of a number using recursion.

**



* Write a program that demonstrates the difference between local and global variables in C++. Use functions to show scope.





**Theory Exercises:-**

* ***What is a function in C++? Explain the concept of function declaration, definition, and calling.***
* **Function:-** It is a set of statements that take inputs, do some specific computations and produce output.
* **Function Declaration:-**

Eg:- 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.

Eg:- 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())

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

* ***What is the scope of variables in C++? Differentiate between local and global scope****.*

In C++, the scope of a variable defines the region of the program where that variable is visible and can be accessed.

* **Local scope:-** Variables with local scope are declared inside a specific block of code, such as a function, a loop, or any other block enclosed within curly braces {}.
* A local variable is only accessible within the block where it is declared. It cannot be accessed from outside that block.
* **Global scope:-** Variables with global scope are declared outside of all functions or blocks, typically at the top of a program file.
* A global variable is accessible from any part of the program after its declaration. Any function or block within the program can read or modify its value.
* ***Explain recursion in C++ with an example.***

Recursion in C++ is a technique where a function calls itself directly or indirectly to solve a problem.

*Already explained in program above.*

* ***What are function prototypes in C++? Why are they used?***

A function prototype is a declaration of a function before its actual definition.

It tells the compiler: The function name

The return type

The parameters (if any)

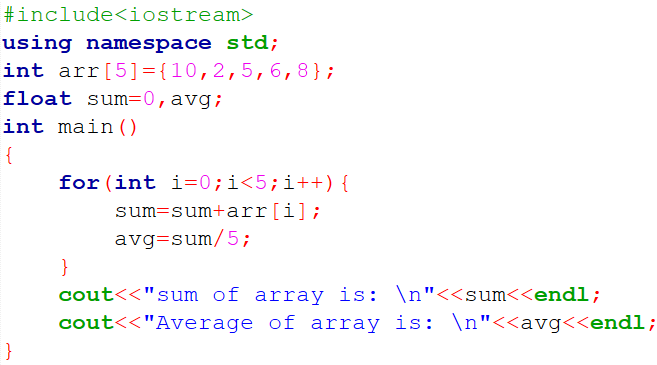
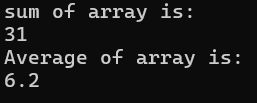
Syntax:- return\_type function\_name(parameter\_list);

Why it is used:-

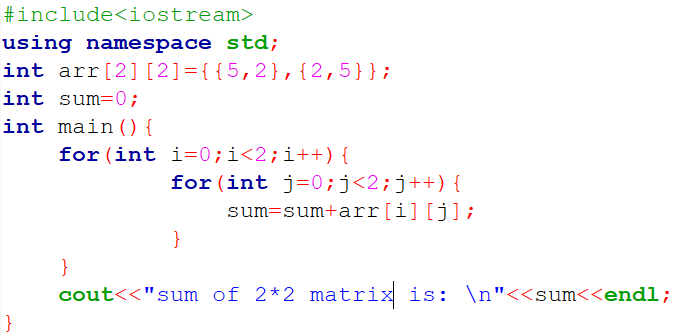
* Tells the compiler a function exists before it is used.
* Allows functions to be defined after the main() function.
* Helps in type checking of parameters during compilation***.***

1. **Arrays and Strings**

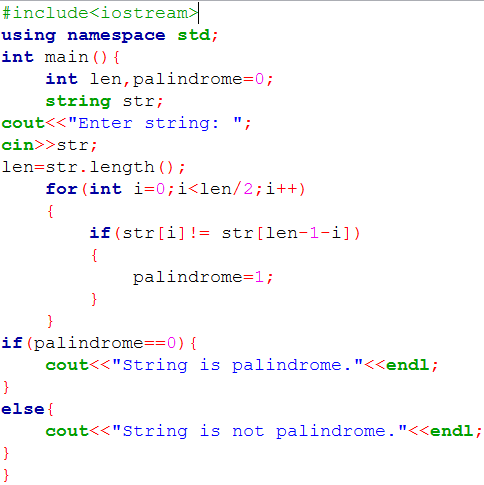
* Write a C++ program that accepts an array of integers, calculates the sum and average, and displays the results.

* Write a C++ program to perform matrix addition on two 2x2 matrices.

* Write a C++ program to check if a given string is a palindrome (reads the same forwards and backwards).

**Theory Exercises:-**

* ***What are arrays in C++? Explain the difference between single-dimensional and multi- dimensional arrays.***
* Array is group of elements of similar datatypes. Array improves readability of code.
* Array index starts from 0.
* **Single – D array :-** A single-dimensional array, also known as a one-dimensional array, represents a linear list of elements. It can be visualized as a single row or column of data.

Eg:- int arr[5] = {1,2,3};

* **Multi – D array :-** A multi-dimensional array represents a collection of arrays, forming a table-like structure (e.g., rows and columns for a 2D array) or even higher-dimensional structures.

Eg:- int arr[3][3] = {{1,2,3},{4,5,6},{7,8,9}};

2D array named matrix with 3 rows and 3 columns.

* ***Explain string handling in C++ with examples.***

In C++, strings are most commonly handled using the **string class** provided by the **Standard Template Library (STL)**. We have to include the string header and use the std namespace

String operations:-

* 1. **Taking Input**

string name;

getline(cin, name); // allows spaces in input

* 1. **Concatenation**

string first = "Hello", second = "World";

string message = first + " " + second;

* 1. **Compare Strings**

if (first == second) {

cout << "Equal";

}

* 1. **Length of String**

int len = message.length(); // or message.size()

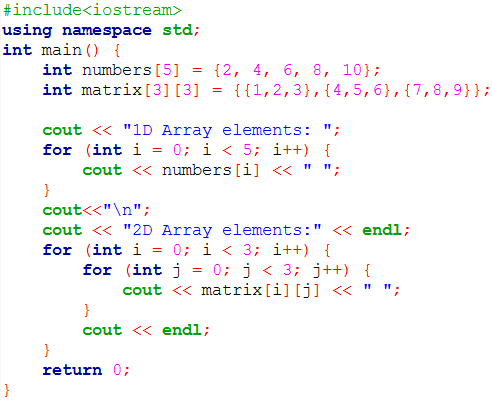
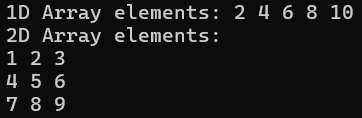
* 1. **Accessing Characters**

char ch = message[0]; // First character

* 1. **Substring**

string sub = message.substr(0, 5); // "Hello"

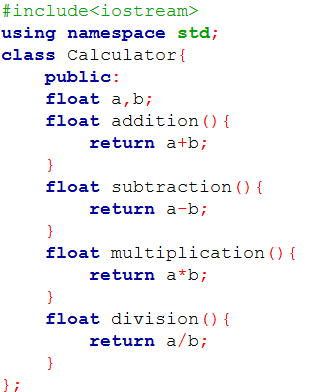
* ***How are arrays initialized in C++? Provide examples of both 1D and 2D arrays.***

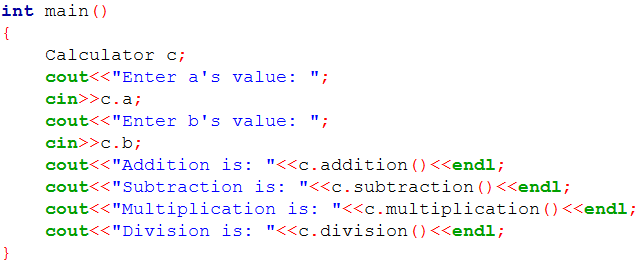
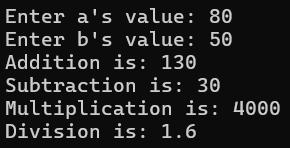
******

1. **Introduction to Object-Oriented Programming**

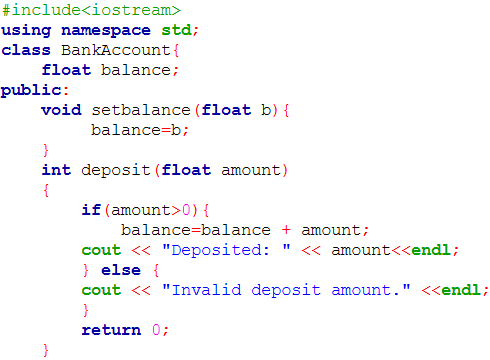
**Lab Exercises:-**

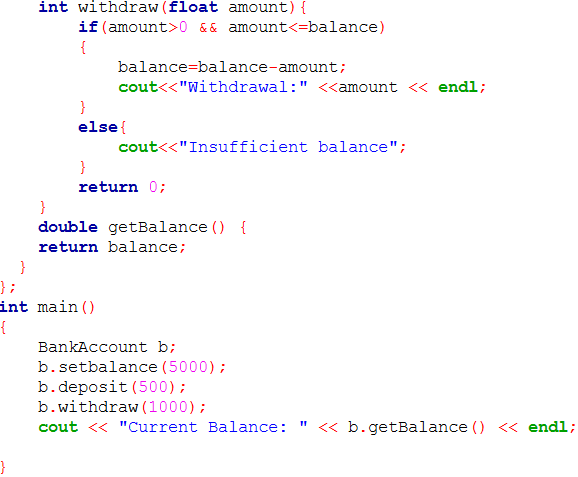
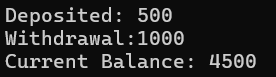
* Write a C++ program that defines a class Calculator with functions for addition, subtraction, multiplication, and division. Create objects to use these functions.



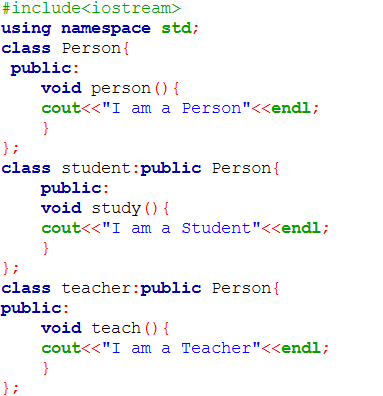
 

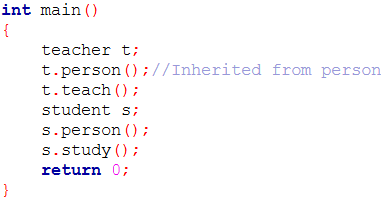
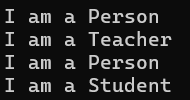
* Create a class BankAccount with data members like balance and member functions like deposit and withdraw. Implement encapsulation by keeping the data members private.



* Write a program that implements inheritance using a base class Person and derived classes Student and Teacher. Demonstrate reusability through inheritance.

****

**Theory Exercises:-**

* ***Explain the key concepts of Object-Oriented Programming (OOP).***

OOP key concepts are:- Encapsulation, Inheritance, Polymorphism, Abstraction.

**Encapsulation**:- Is to make sure that **"sensitive"** data is hidden from users.

* We declare class variables as **private** (cannot be accessed from outside the class).
* If any outsider wants to read or modify the value of a private member, then we can provide public **get** and **set** methods.
* Encapsulation ensures better control of your data, because you (or others) can change one part of the code without affecting other parts.
* Increased security of data

**Inheritance:-** allows **one class** to **reuse attributes** and **methods** from **another class.**

* Two concepts of inheritance **:- base class** (parent) - the class being inherited from , **derived class** (child) - the class that inherits from parent class.
* To inherit from a class, use **:**  symbol.
* Inheritance use when we want to reuse attributes and methods of an existing class in creating a new class.

**Polymorphism:-** Polymorphism means **"many forms".**

1. **Compile – time polymorphism:-** same function name but different parameters known as **method overloading.**
2. **Run – time polymorphism:-** same function name but in different class so inheritance between class is needed and known as **method overridden.**

**Abstraction:-** Abstraction means simplifying complex systems by hiding unnecessary details and exposing only the essential functionalities.

* Focus on **what** an object does, not **how** it does it.
* Improves **security** by exposing only necessary parts of the implementation.
* Abstraction is achieved using **Abstract Class** & **Pure Virtual Function.**
* ***What are classes and objects in C++? Provide an example.***

**Classes:-** A class is a user-defined data type , and it works as a **“blueprint”** for creating objects.

* Class is combination of **variables** and **functions**.
* At last, end the class definition with a semicolon **;** .

Eg:- class MyClass {       // The class  
   public:             // Access specifier  
    int num;        // Attribute (int variable)  
    string name;  // Attribute (string variable)  
};

**Objects:-** An object is created from a class so objects is known as **instance** of class.

* Object is made in main method of program.
* Class cannot access it’s own properties so we need to create object or reference variable of that class.
* While every objects created hold its different memory space .

Eg:- int main(){

MyClass obj; // created object of MyClass

obj.num=20; // Access attributes and set values

obj.name=”Aesha”;

cout<< obj.num <<endl; // printing those values.

cout<< obj.name <<endl;

}

* ***What is inheritance in C++? Explain with an example.***

**Inheritance:-** allows **one class** to **reuse attributes** and **methods** from **another class.**

There are 5 types of inheritance:-

* Single – level Inheritance
* Multi – level Inheritance
* Multiple Inheritance
* Hierarchical Inheritance
* Hybrid Inheritance.

** **

* ***What is encapsulation in C++? How is it achieved in classes?***
* **Encapsulation** in C++ is the process of **wrapping data** (member variables) and the **functions** (member methods) that operate on that data within a single unit, typically a class.
* It can achieved in classes by **hiding data** – member variables are declared as **private** access modifier. This prevents direct access to data from outside the class enhanced security, **control access** – **public** member functions are created to interact with private data. **Get** method used to retrieve value of private variable and **Set** method used to modify them.