**SRI CHANDRASEKHARENDRA SARASWATHI VISWA MAHAVIDYALAYA**

**(UNIVERSITY ESTABLISHED under section 3 of UGC Act 1956)**

**ENATHUR,** **KANCHIPURAM – 631 561**



**OBJECT ORIENTED PROGRAMMING LAB**

**LABORATORY RECORD**

**Name :** Balakrishna R

**Reg. No :** 112514025

**Class :** I Year B.Sc. (Cyber Security)

**Subject :** UCYF231P60 - Object Oriented Programming Lab

**SRI CHANDRASEKHARENDRA SARASWATHI**

**VISWA MAHAVIDYALAYA**

**(University Established under section 3 of UGC Act 1956)**

****

**BONAFIDE CERTIFICATE**

**This is to certify that this is the bonafide record of work done by**

**Mr./~~Ms.~~ Balakrishna R, with Reg.No 112514025 of I Year B.Sc. (Cyber Security) in the Object Oriented Programming Lab during the year 2025.**

**Staff-in-charge** **Head of the Department**

**Submitted for the Practical Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Internal Examiner** **External Examiner.**

**INDEX**

| **SNo.** | **Date** | **Title** | **Page** | **Sign** |
| --- | --- | --- | --- | --- |
| 1 | 25-07-2025 | Classes & Objects |  |  |
| 2 | 30-07-2025 | Friend Function |  |  |
| 3 | 06-08-2025 | Inline Function |  |  |
| 4 | 08-08-2025 | Array Implementation |  |  |
| 5 | 20-08-2025 | Multiple Inheritance |  |  |
| 6 | 29-08-2025 | This Pointer |  |  |
| 7 | 10-09-2025 | Friend Class |  |  |
| 8 | 17-09-2025 | Function Overloading |  |  |
| 9 | 03-10-2025 | Operator Overloading |  |  |
| 10 | 10-10-2025 | String Manipulation |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

| **1** | **CLASSES AND OBJECTS** | 25-07-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to display Employee details Using Classes and Objects.

**ALGORITHM :**

**Step 1:** Start

**Step 2:** Declare variables:

num (integer) → to store number of employees

For each employee, store:

empID (integer)

name (string)

department (string)

salary (float)

**Step 3:** Display message — “Enter number of employees:”

**Step 4:** Read num

**Step 5:** i = 1

**Step 6:** Display message — “Enter details for employee i:”

Input empID

Input name

Input department

Input salary

**Step 7:** i = i+1

**Step 8:** if i <= num goto Step 6, else continue

**Step 9:** j = 1

**Step 10:** Display message — “--- Employee Details ---”

Display empID

Display name

Display department

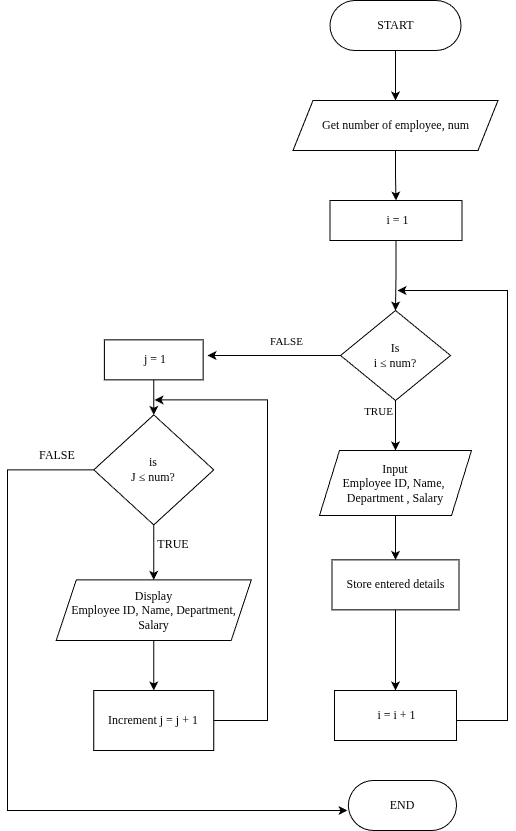
Display salary

**Step 11:** j = j + 1

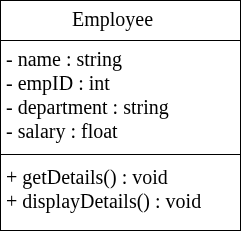
**Step 12:** if j <= num goto Step 10, else continue

**Step 13:** Stop

**FLOWCHART :**



**UML DIAGRAM :**



**SOURCE CODE :**

#include <iostream>

using namespace std;

// Employee class definition

class Employee {

private:

string name;

int empID;

string department;

float salary;

public:

// Method to input employee details

void getDetails() {

cout << "Enter Employee ID: ";

cin >> empID;

cin.ignore(); // To ignore the newline character after empID

cout << "Enter Name: ";

getline(cin, name);

cout << "Enter Department: ";

getline(cin, department);

cout << "Enter Salary: ";

cin >> salary;

}

// Method to display employee details

void displayDetails() {

cout << "\nEmployee ID: " << empID << endl;

cout << "Name: " << name << endl;

cout << "Department: " << department << endl;

cout << "Salary: Rs." << salary << endl;

cout << "-----------------------------" << endl;

}

};

// Main function

int main() {

int num;

cout << "Enter number of employees: ";

cin >> num;

// Creating array of Employee objects

Employee employees[num];

// Getting details for each employee

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

cout << "\nEnter details for employee " << i + 1 << ":\n";

employees[i].getDetails();

}

// Displaying employee details

cout << "\n--- Employee Details ---\n";

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

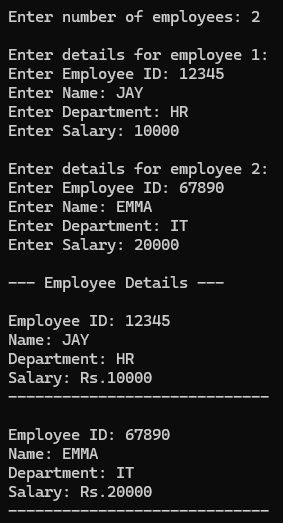
employees[i].displayDetails();

}

return 0;

}

**OUTPUT :**



**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| **2** | **FRIEND FUNCTION** | 30-07-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to find the Mean Value Using Friend Function.

**ALGORITHM :**

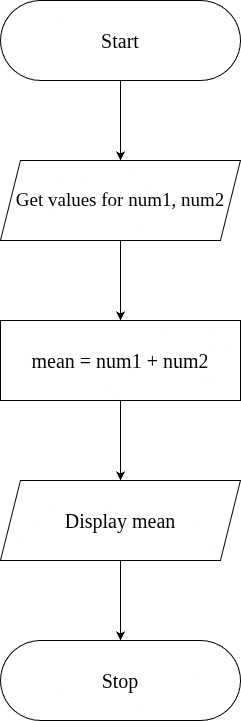
**Step 1:** Start the program

**Step 2:** Read values num1, num2

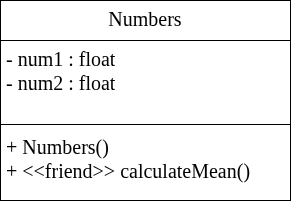
**Step 3:** mean = num1 + num2

**Step 4:** Display mean

**Step 5:** Stop

**FLOWCHART :**

**UML DIAGRAM :**

****

**SOURCE CODE :**

#include <iostream>

using namespace std;

class Numbers {

private:

float num1, num2;

public:

// Constructor to initialize values

Numbers(float a, float b) {

num1 = a;

num2 = b;

}

// Declare friend function

friend float calculateMean(Numbers n);

};

// Definition of friend function

float calculateMean(Numbers n) {

return (n.num1 + n.num2) / 2;

}

int main() {

float a, b;

cout << "Enter two numbers: ";

cin >> a >> b;

Numbers obj(a, b);

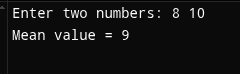
float mean = calculateMean(obj);

cout << "Mean value = " << mean << endl;

return 0;

}

**OUTPUT :**

****

**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| **3** | **INLINE FUNCTION** | 06-08-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to Implement Inline Function.

**ALGORITHM :**

**Step 1:** Start

**Step 2:** Define an inline function square(int n) that returns n \* n.

**Step 3:** In main(), declare an integer variable num.

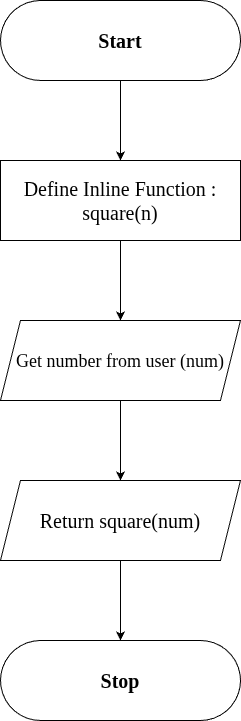
**Step 4:** Read the value of num from the user.

**Step 5:** Call the inline function square(num) and store/print the result.

**Step 6:** Display the square of the given number.

**Step 7:** Stop

**FLOWCHART :**

****

**SOURCE CODE :**

#include <iostream>

using namespace std;

// Inline function definition

inline int square(int n) {

return n \* n;

}

int main() {

int num;

cout << "Enter a number: ";

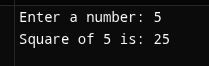
cin >> num;

cout << "Square of " << num << " is: " << square(num) << endl;

return 0;

}

**OUTPUT :**

****

**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| 4 | **ARRAY IMPLEMENTATION** | 08-08-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to Implement Arrays.

**ALGORITHM :**

**Step 1 :** Start the Program

**Step 2 :** Create an Object "myArray" of Class "Array"

**Step 3 :** Display menu as below and get "choice"

1. Insert

2. Remove

3. Search

4. Display

5. Exit

**Step 4 :** If (choice >= 6) go to step -, Else Continue

**Step 5 :** If (choice = 1)

Get value n from user

set i = 0

Check for condition, i < n

On true; get Value for Insert (value)

call myArray.insert(value)

Increment i, i = i + 1

Else; break loop

elseif (choice = 2)

get index to Remove (index)

call myArray.remove(index)

elseif (choice = 3)

get value to search (value)

call myArray.search (value)

elseif (choice = 4)

call myArray.display()

elseif (choice = 5)

go to step 7

**Step 6:** Display "Invalid choice"

**Step 7:** Stop

**ALGORITHM FOR INSERT FUNCTION :**

**Step 1:** Start the program.

**Step 2:** Prompt the user to enter an element to insert into the array.

→ Input: element

**Step 3:** Check the current size of the array.

→ Condition: if size < 100

**Step 4:**

If the condition is TRUE (size < 100):

a) Insert the element into the array at position arr[size].

b) Increment the size of the array by 1.

→ size = size + 1

If the condition is FALSE (size >= 100):

a) Display the message:

→ “Array is full. Cannot insert more elements.”

**Step 5:** Stop the program.

**ALGORITHM FOR REMOVE FUNCTION :**

**Step 1:** Start

**Step 2:** Get index from the user.

**Step 3:** Check the condition, (index < 0 || index >= size)

On true; Go to Step 4

   Else; Go to Step 5

**Step 4:** Print "Invalid index", then Go to Step 10

**Step 5:** Set i = index

**Step 6:** Check the condition, i < size - 1

On true; Go to Step 7

   Else; Go to Step 8

**Step 7:** Shift the elements to the left by one position:

    arr[i] = arr[i + 1]

    Repeat this step until all elements after the deleted index are shifted.

**Step 8:** Decrease the array size by one:

     size = size - 1

**Step 9:** Print "Element deleted at index" followed by the value of index.

**Step 10:** Stop

**ALGORITHM FOR SEARCH FUNCTION :**

**Step 1:** Start

**Step 2:** Initialize i = 0

**Step 3:** Check the condition, i < size

On true; go to Step 4

   Else; go to Step 8

**Step 4:** Check if the current array element matches the search key, arr[i] == key

On true; go to Step 7

   Else go to Step 5

**Step 5:** Increment i by 1, i = i + 1

**Step 6:** Go back to Step 3 (Repeat the checking process for the next element)

**Step 7:** Print "Element found at index" followed by the value of i

   Go to Step 9

**Step 8:** Print "Element not found

**Step 9:** Stop

**ALGORITHM FOR DISPLAY FUNCTION :**

**Step 1:** Start the program.

**Step 2:** Check if the array size is equal to 0, condition (size == 0)

On true; Print "Array is empty".

Go to Step 8 (Stop).

Else; Continue to the next step.

**Step 3:** Print "Array elements:" to indicate that array elements will be displayed.

**Step 4:** Initialize a variable i = 0 (loop counter).

**Step 5:** Check the condition if (i < size).

On true; Print the element arr[i].

Increment i by 1 (i = i + 1).

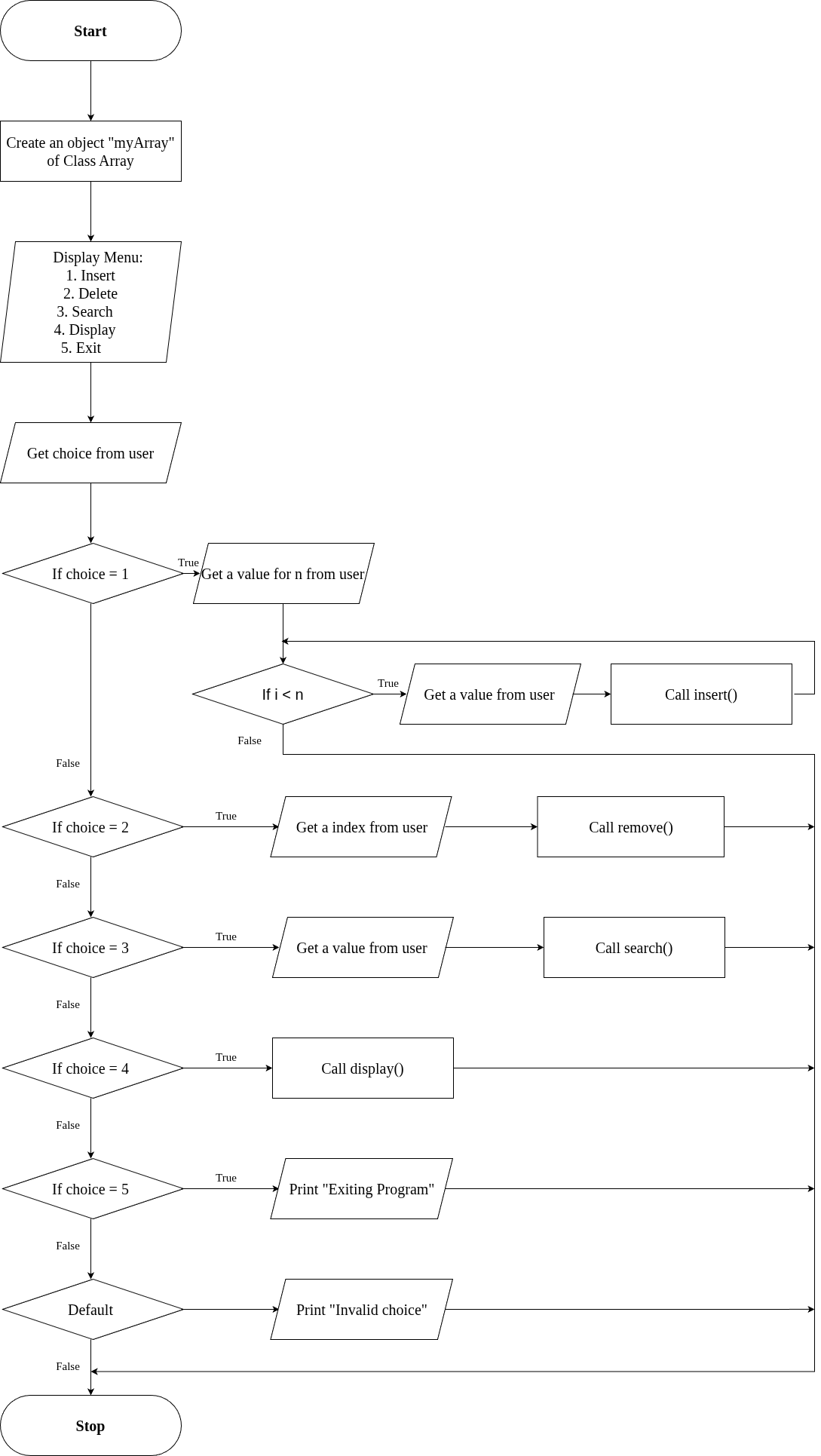
Go back to Step 5 and repeat until i < size becomes False.

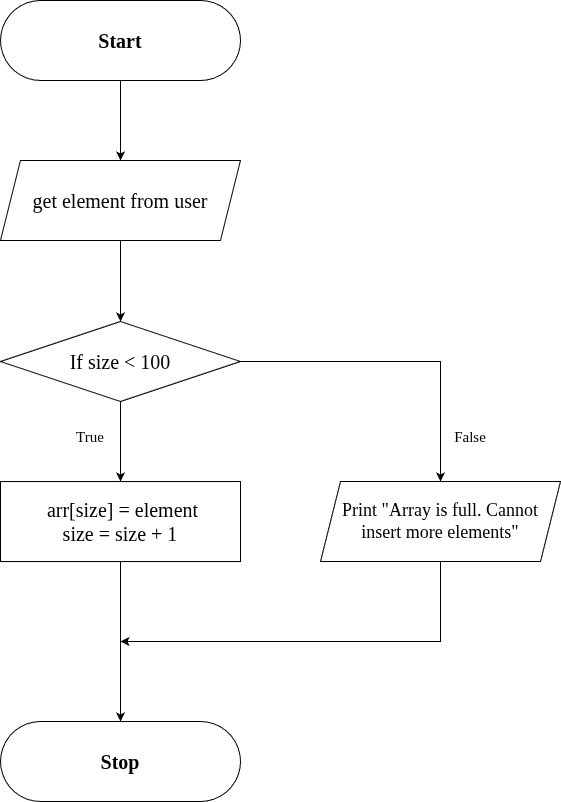
Else; Exit the loop.

**Step 6:** After printing all elements, end the loop.

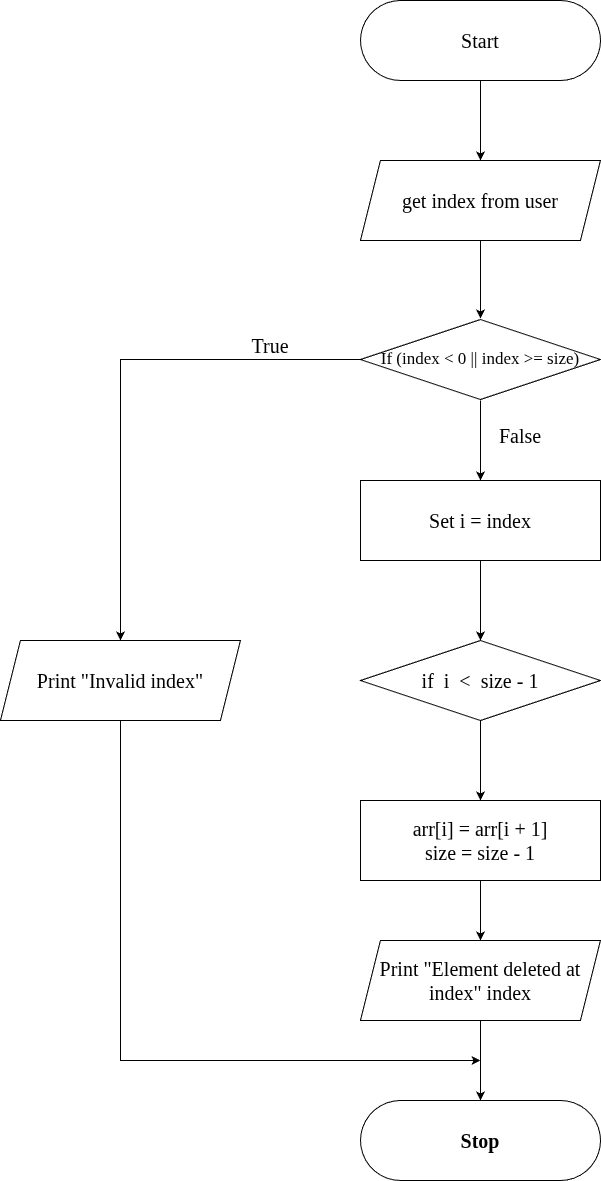
**Step 7:** Stop the program.

**FLOWCHART :**

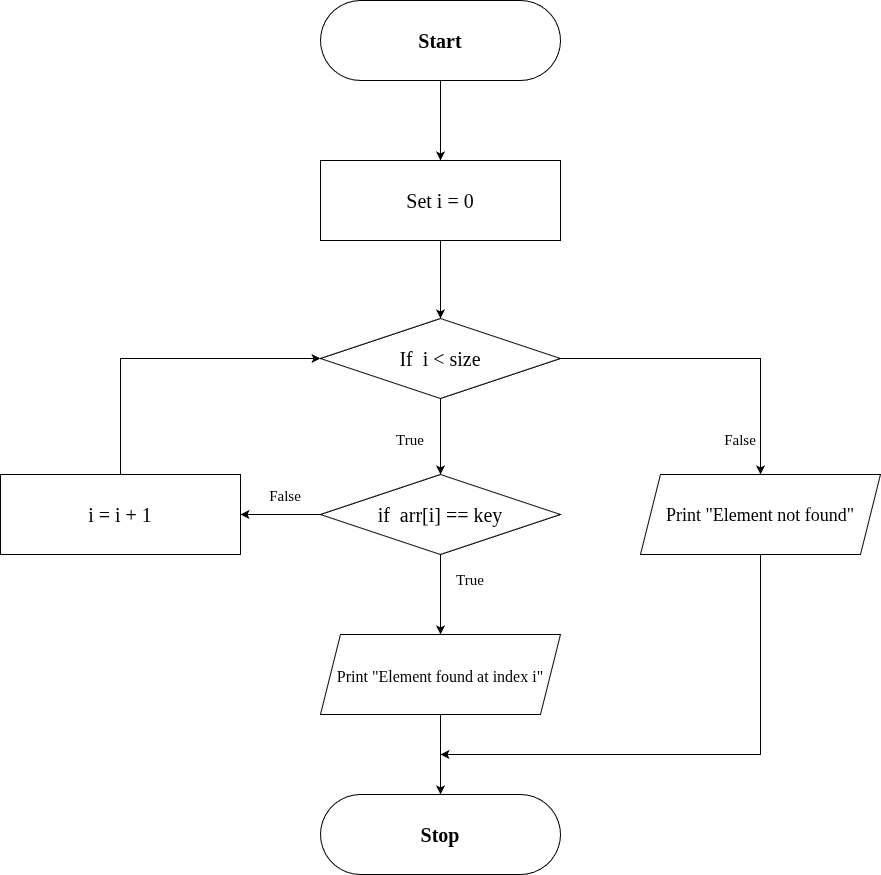
**FLOWCHART FOR INSERT FUNCTION :**

****

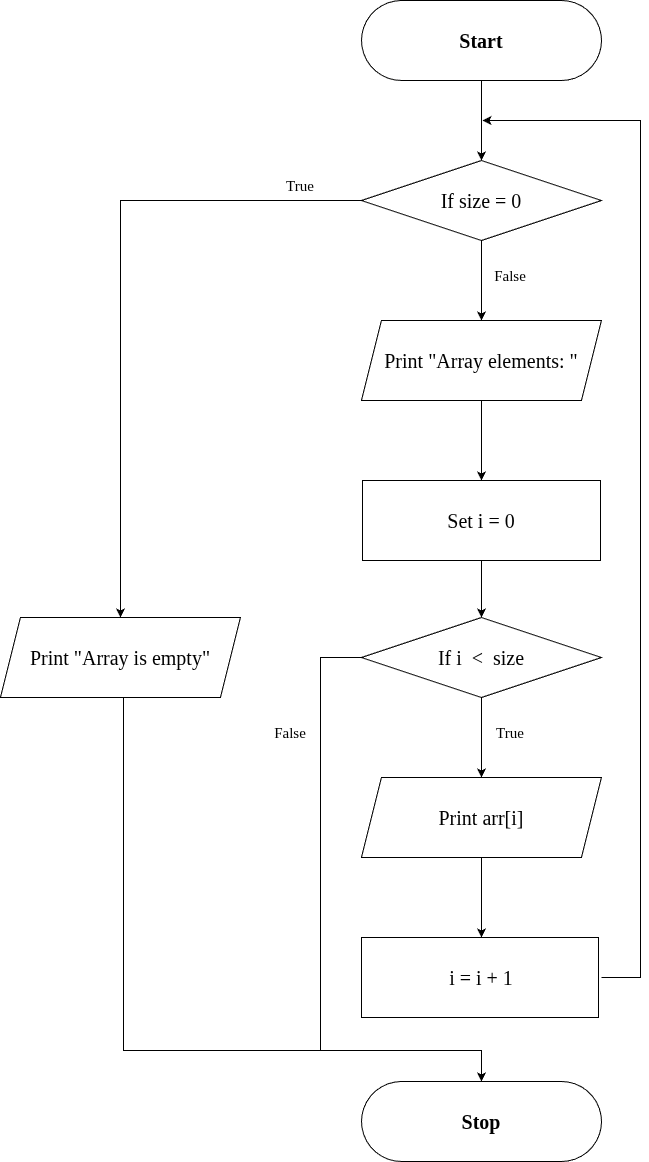
**FLOWCHART FOR REMOVE FUNCTION :**

****

**FLOWCHART FOR SEARCH FUNCTION :**

****

**FLOWCHART FOR DISPLAY FUNCTION :**

****

**SOURCE CODE :**

#include <iostream>

using namespace std;

class Array {

private:

int arr[100]; // Static array with max size

int size; // Current number of elements

public:

Array() {

size = 0;

}

// Insert element at end

void insert(int element) {

if (size < 100) {

arr[size] = element;

size++;

} else {

cout << "Array is full. Cannot insert more elements.\n";

}

}

// Delete element by index

void remove(int index) {

if (index < 0 || index >= size) {

cout << "Invalid index.\n";

return;

}

for (int i = index; i < size - 1; i++) {

arr[i] = arr[i + 1];

}

size--;

cout << "Element deleted at index " << index << ".\n";

}

// Search for an element

void search(int key) {

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

if (arr[i] == key) {

cout << "Element " << key << " found at index " << i << ".\n";

return;

}

}

cout << "Element " << key << " not found.\n";

}

// Display all elements

void display() {

if (size == 0) {

cout << "Array is empty.\n";

return;

}

cout << "Array elements: ";

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

cout << arr[i] << " ";

}

cout << endl;

}

};

// Main function

int main() {

Array myArray;

int choice, value, index;

do {

cout << "\n--- Array Operations ---\n";

cout << "1. Insert\n2. Delete\n3. Search\n4. Display\n5. Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice) {

case 1:

int n;

cout << "How many values do you want to insert? ";

cin >> n;

cout << "Enter " << n << " values: ";

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

cin >> value;

myArray.insert(value);

}

break;

case 2:

cout << "Enter index to delete: ";

cin >> index;

myArray.remove(index);

break;

case 3:

cout << "Enter value to search: ";

cin >> value;

myArray.search(value);

break;

case 4:

myArray.display();

break;

case 5:

cout << "Exiting program.\n";

break;

default:

cout << "Invalid choice.\n";

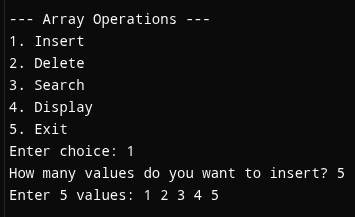
}

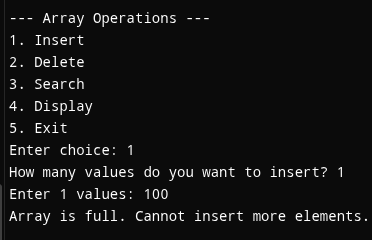
} while (choice != 5);

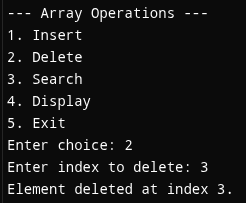
return 0;

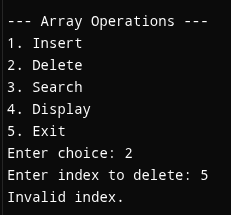
}

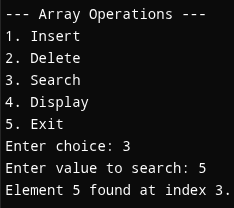
**OUTPUT :**

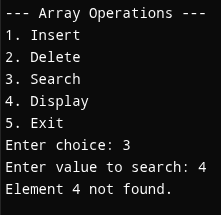
****

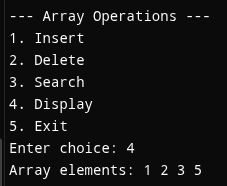
****

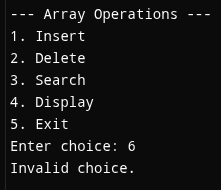
****

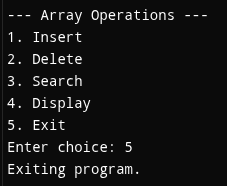
****

****

****

****

****

****

**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| **5** | **MULTIPLE INHERITANCE** | 20-08-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to implement Multiple Inheritance.

**ALGORITHM :**

**Step 1:** Start the program.

**Step 2:** Define the base class "Engine"

  → Declare a protected data member engineCC (integer).

  → Define a public member function getEngineInfo() that:

   • Accepts user input for engineCC.

**Step 3:** Define the base class "Body"

  → Declare protected data members color and type (string).

  → Define a public member function getBodyInfo() that:

   • Takes input for type (SUV/Sedan/Hatchback/etc)

**Step 4:** Define the derived class "Car" that inherits publicly from both "Engine" and "Body".

  → Define a public member function showDetails() that:

   • Displays engineCC, color, and type.

**Step 5:** In the main() function:

  1. Create an object obj of class Car.

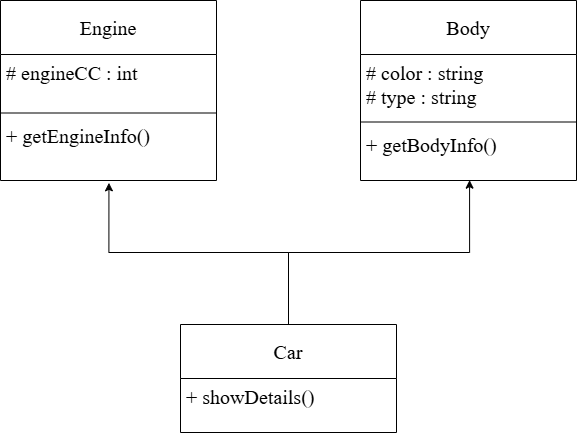
  2. Call obj.getEngineInfo() to input engine details.

  3. Call obj.getBodyInfo() to input body details.

  4. Call obj.showDetails() to display all vehicle details.

**Step 6:** End the program.

**UML DIAGRAM :**

****

**SOURCE CODE :**

#include <iostream>

using namespace std;

// Base Class 1

class Engine {

protected:

int engineCC;

public:

void getEngineInfo() {

cout << "Enter Engine Capacity (in CC): ";

cin >> engineCC;

}

};

// Base Class 2

class Body {

protected:

string color, type;

public:

void getBodyInfo() {

cout << "Enter Vehicle Color: ";

cin >> color;

cout << "Enter Body Type (SUV/Sedan/Hatchback/etc): ";

cin >> type;

}

};

// Derived Class (Multiple Inheritance)

class Car : public Engine, public Body {

public:

void showDetails() {

cout << "\n--- Vehicle Information ---\n";

cout << "Engine Capacity: " << engineCC << " CC" << endl;

cout << "Body Color: " << color << endl;

cout << "Body Type: " << type << endl;

}

};

int main() {

Car obj;

obj.getEngineInfo();

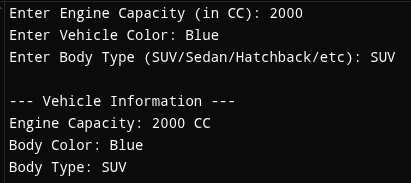
obj.getBodyInfo();

obj.showDetails();

return 0;

}

**OUTPUT :**

****

**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| 6 | **THIS POINTER** | 29-08-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program To Implement This Pointer.

**ALGORITHM :**

**Step 1:** Start the program

**Step 2:** Create a class Sample with data members a and b

**Step 3:** Define setData(int a, int b) member function → Use this->a = a and this->b = b to assign values

**Step 4:** Define display() function to print the values of a and b

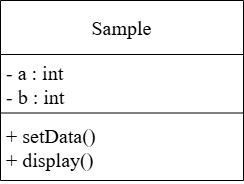
**Step 5:** In main(), create an object obj of class Sample

**Step 6:** Call obj.setData(10, 20)

**Step 7:** Call obj.display() to show stored values

**Step 8:** Stop the program

**UML DIAGRAM :**



**SOURCE CODE :**

#include <iostream>

using namespace std;

class Sample {

int a, b;

public:

void setData(int a, int b) {

// Using this pointer to differentiate between class members and parameters

this->a = a;

this->b = b;

}

void display() {

cout << "Value of a: " << this->a << endl;

cout << "Value of b: " << this->b << endl;

}

};

int main() {

Sample obj;

obj.setData(10, 20); // Call function using object

obj.display(); // Display values

return 0;

}

**OUTPUT :**

****

**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| **7** | **FRIEND CLASS** | **DATE:** 10-09-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to Implement Friend class.

**ALGORITHM :**

**Step 1:** Start.

**Step 2:** Define class Auditor as a forward declaration so it can be referenced in class BankAccount.

**Step 3:** Define class BankAccount:

1. Declare a private string variable (name) and double variable (balance).

2. Define a constructor to initialize (name) and (balance).

3. Implement deposit() and widthrawl() public functions.

4. Declare Auditor as a friend class using the friend class keyword.

**Step 4:** Defines class Auditor:

1. Implement a public member function showAccount that takes an object of class BankAccount as a parameter.

[Accesses and displays the private variables (name) and (balance) of BankAccount using the reference to the object.]

**Step 5:** Defines main function.

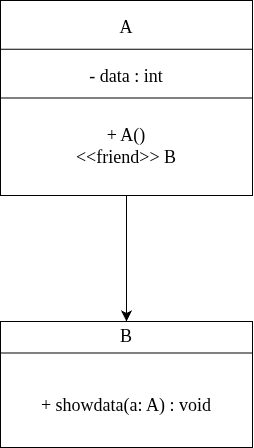
1. Creates object (official) of class Auditor and object (acc) of class BankAccount.

2. Call acc.deposit() and acc.widthrawl() to modify the balance.

3. Call official.showAccount(acc) to display the account details( name and balance ) using the auditor object.

**Step 6:** End.

**UML DIAGRAM :**

****

**SOURCE CODE :**

#include <iostream>

using namespace std;

// Forward declaration

class Auditor;

class BankAccount {

private:

string name;

double balance;

// Declare Auditor as a friend class

friend class Auditor;

public:

BankAccount(string n, double b) : name(n), balance(b) {}

void deposit(double amt) { balance += amt; }

void withdraw(double amt) { balance -= amt; }

};

class Auditor {

public:

void showAccount(const BankAccount &acc) {

cout << "Account Holder: " << acc.name << endl;

cout << "Account Balance: " << acc.balance << endl;

}

};

int main() {

BankAccount acc("Alice", 5000.0);

Auditor official;

acc.deposit(1200.0);

acc.withdraw(800.0);

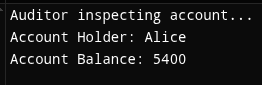
cout << "Auditor inspecting account..." << endl;

official.showAccount(acc);

return 0;

}

**OUTPUT :**



**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| 8 | **FUNCTION OVERLOADING** | 17-09-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to Implement Function overloading.

**ALGORITHM :**

**Step 1:** Start.

**Step 2:** Define a class named Calculator.

**Step 3:** Inside the class, define three overloaded functions named add:

1. Function-1: int add(int a, int b)

-> Computes and returns the sum of two integers.

2. Function-2: int add(int a, int b, int c)

-> Computes and returns the sum of three integers.

3. Function-3: double add(double a, double b)

-> Computes and returns the sum of two double values.

**Step 4:** In the main() function:

1. Create an object (calc) of class Calculator.

2. Call and display the result of the overloaded functions:

if parameters are of type int and only 2 parameters:

-> Call calc.add(int , int)

else if parameterrs are of type int and only 3 parameters:

-> Call calc.add(int , int , int)

else if parameters are of type double:

-> Call calc.add(double , double)

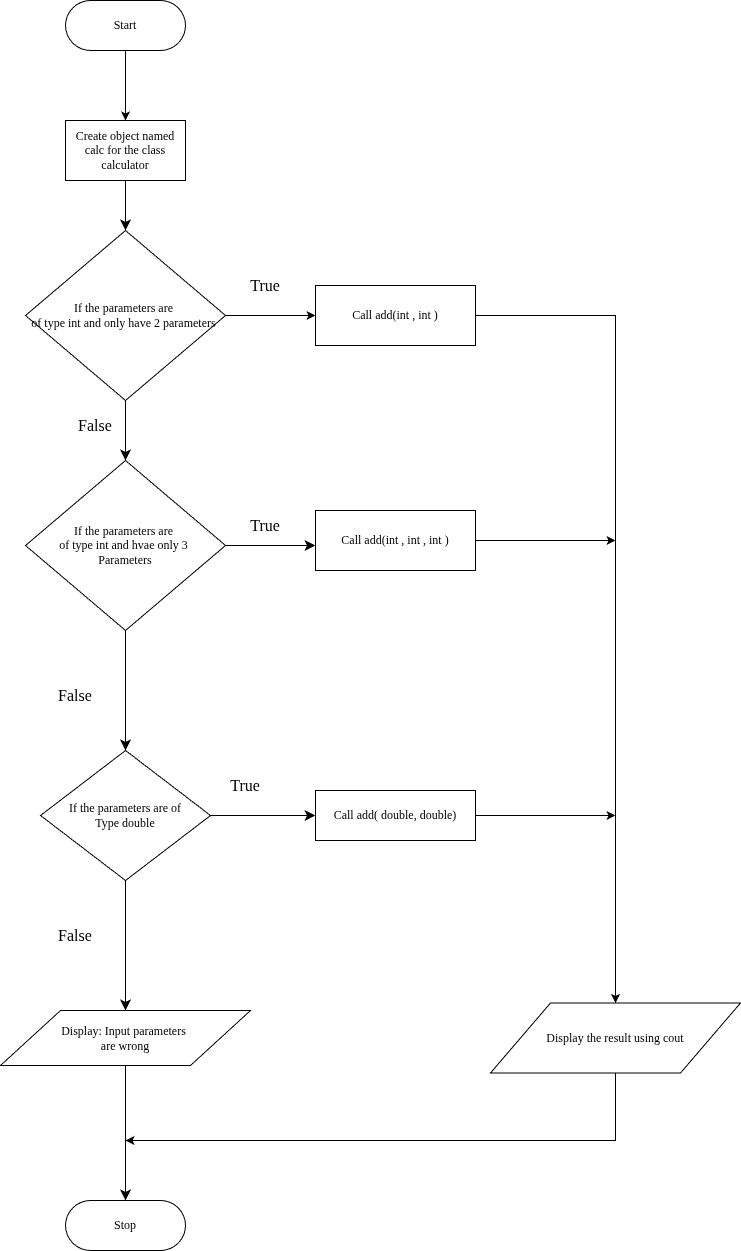
else:

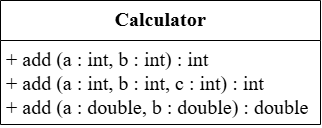
Display: Input parameters are wrong.

**Step 5:** Display each result using cout.

**Step 6:** End.

**FLOWCHART :**

**UML DIAGRAM :**

****

**SOURCE CODE :**

#include <iostream>

using namespace std;

class Calculator {

public:

// Function to add two integers

int add(int a, int b) {

return a + b;

}

// Overloaded function to add three integers

int add(int a, int b, int c) {

return a + b + c;

}

// Overloaded function to add two doubles

double add(double a, double b) {

return a + b;

}

};

int main() {

Calculator calc;

cout << "Sum of 2 and 3: " << calc.add(2, 3) << endl;

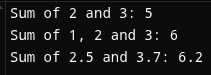
cout << "Sum of 1, 2 and 3: " << calc.add(1, 2, 3) << endl;

cout << "Sum of 2.5 and 3.7: " << calc.add(2.5, 3.7) << endl;

return 0;

}

**OUTPUT :**

****

**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| 9 | **OPERATOR OVERLOADING** | 03-10-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to Implement Operator Overloading.

**ALGORITHM :**

**Step 1:** Start the program.

**Step 2:** Define a class named “Point”.

**Step 3:** Inside the classes, declare two private data members : ‘x’ and ‘y’.

**Step 4:** Create a constructor Point(int a=0,int b =0)to initialize ‘x’ and ‘y’.

**Step 5:** Define an overloaded + operator function :

* Take another object of type point as a parameter.
* Add the ‘x’ and ‘y’ values of both objects.
* Return a new Point object containing the sum.

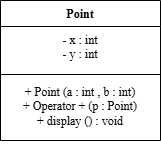
**Step 6:** Define and display() function to print the ‘x’ and ‘y’ values.

**Step 7:** In main() function:

1. Create two objects of Class “Point”: p1(2,3) and p2(4,5).
2. Use the overloaded + operator to add the two objects: p3=p1+p2.
3. Display the result using p3.display().

**Step 8:** Stop the program.

**UML DIAGRAM :**

****

**SOURCE CODE :**

#include <iostream>

using namespace std;

class Point {

private:

int x, y;

public:

Point(int a = 0, int b = 0) : x(a), y(b) {}

// Overload the + operator

Point operator+(const Point& p) {

return Point(x + p.x, y + p.y);

}

void display() {

cout << "(" << x << ", " << y << ")" << endl;

}

};

int main() {

Point p1(2, 3), p2(4, 5);

Point p3 = p1 + p2; // uses overloaded +

p3.display(); // Output: (6, 8)

return 0;

}

**OUTPUT :**



**RESULT :**

Thus the program is compiled and executed successfully with verified output.

| 10 | **STRING MANIPULATION** | 10-10-2025 |
| --- | --- | --- |

**AIM :**

Write a C++ Program to Implement String Concepts.

**ALGORITHM :**

**Step 1:** Start

**Step 2:** Declare string variables (str1, str2, fullName, etc.)

**Step 3:** Initialize str1 = "Hello", str2 = "World"

**Step 4:** Concatenate str1 and str2 (using '+' operator)→ store in (combined)

**Step 5:** Display the combined string

**Step 6:** Prompt user to enter full name

**Step 7:** Read full name using getline()

**Step 8:** Display the full name

**Step 9:** Find and display the length of the full name (using "length()" library function)

**Step 10:** Extract and display the first 5 characters (using "substr()" library function)

**Step 11:** Compare two strings ("Apple" and "Banana") (using "==" operator)

**Step 12:** Display comparison result

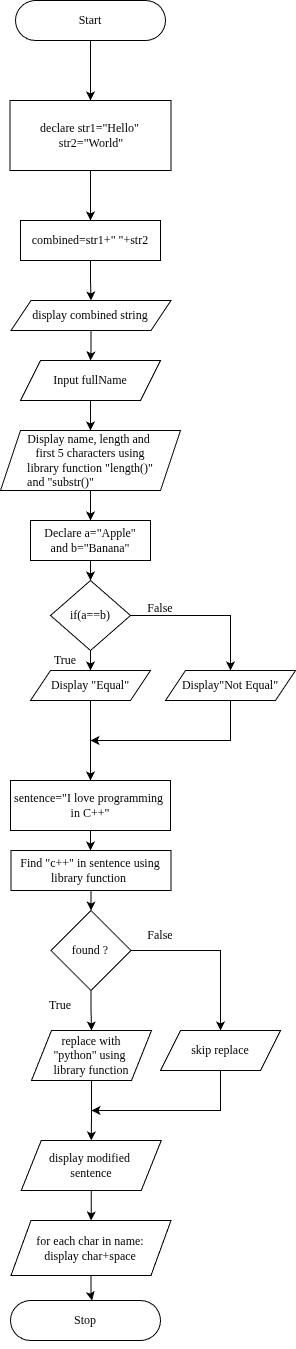
**Step 13:** Replace "C++" with "Python" in a sentence (using "replace()" library function)

**Step 14:** Display modified sentence

**Step 15:** Loop through and display each character of full name

**Step 16:** End

**FLOWCHART :**

****

**SOURCE CODE :**

#include <iostream>

#include <string>

using namespace std;

int main() {

string str1 = "Hello";

string str2("World");

string combined = str1 + " " + str2; // concatenate the str1 and str2 using "+" operator

cout << "Combined string: " << combined << endl;

string fullName;

cout << "Enter your full name: ";

getline(cin, fullName); // safer than cin >> fullName

cout << "Your name is: " << fullName << endl;

cout << "Length of your name: " << fullName.length() << endl; // display length of the string

string sub = fullName.substr(0, 5); //"fullname.substr(0,5)"used to display first 5 character from main string

cout << "First 5 characters: " << sub << endl;

string a = "Apple";

string b = "Banana";

if (a == b) { // compare the string of a and b using "==" operator

cout << "Strings are equal" << endl;

} else {

cout << "Strings are not equal" << endl;

}

string sentence = "I love programming in C++.";

size\_t pos = sentence.find("C++");

if (pos != string::npos) {

sentence.replace(pos, 3, "Python");

}

cout << "Modified sentence: " << sentence << endl;

cout << "Characters in your name: ";

for (char ch : fullName) {

cout << ch << ' ';

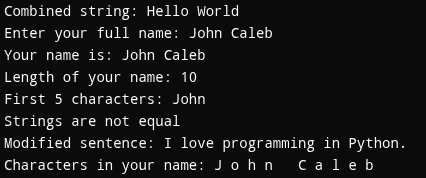
}

cout << endl;

return 0;

}

**OUTPUT :**

****

**RESULT :**

Thus the program is compiled and executed successfully with verified output.