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 : Swathisri S

Reg. No : 112514020

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.Swathisri S, 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

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

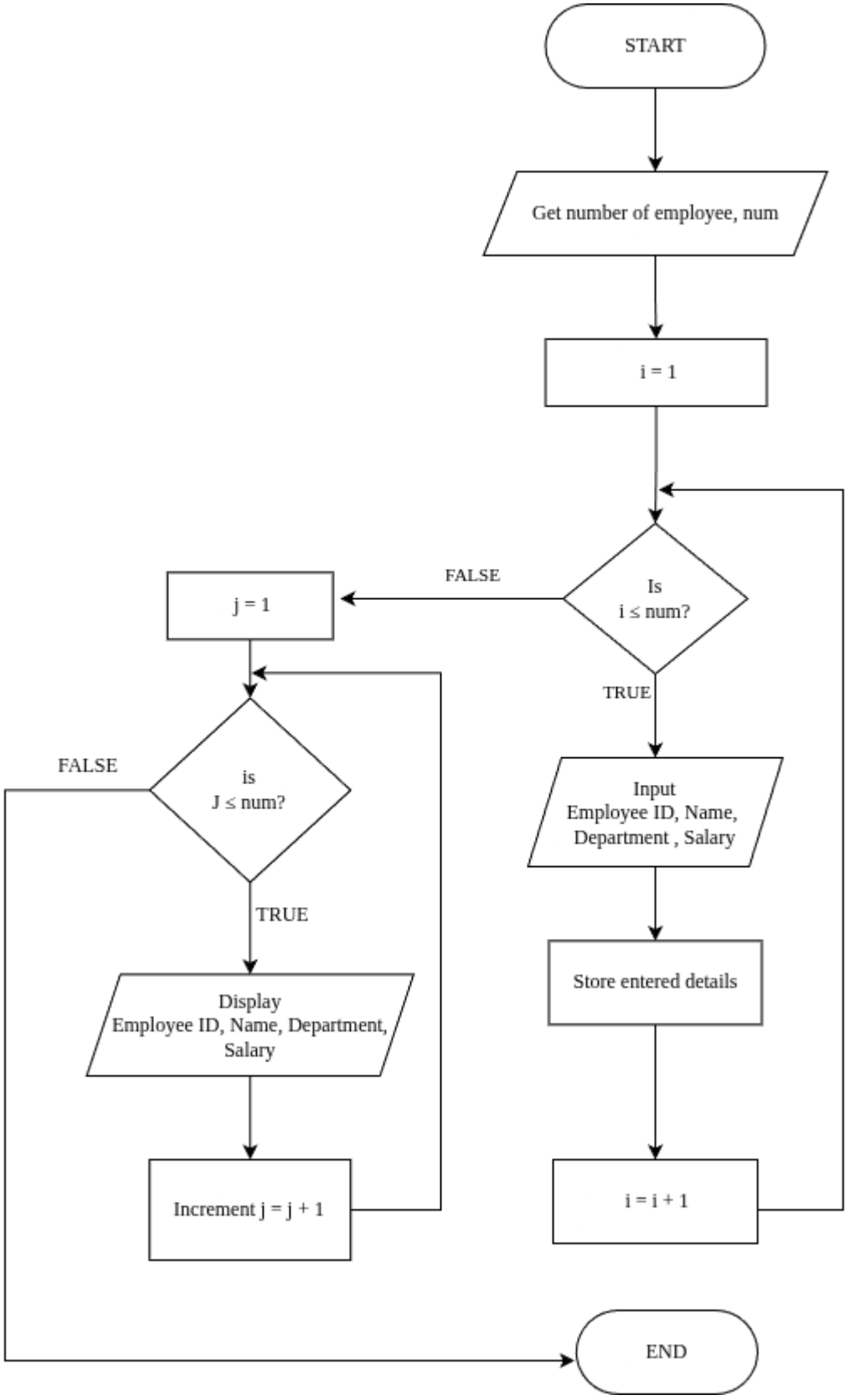
Display salary

Step 11: j = j + 1

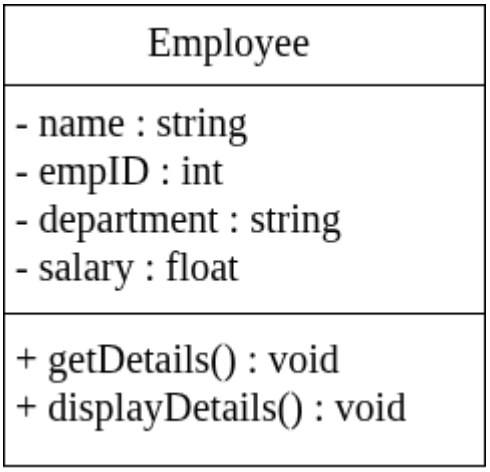
Step 12: if j <= num goto Step 10, else continue

Step 13: Stop

FLOWCHART :


START
Get number of employee, num
i = 1
Is
i ≤ num?
FALSE
TRUE
j = 1
is
J ≤ num?
FALSE
TRUE
Input
Employee ID, Name,
Department , Salary
Store entered details
Display
Employee ID, Name, Department,
Salary
Increment j = j + 1
i = i + 1
END

UML DIAGRAM :



SOURCE CODE :

https://github.com/112514020-create/Record_OOPS/blob/main/EmpClass.cpp

OUTPUT :

```
Enter number of employees: 2

Enter details for employee 1:
Enter Employee ID: 12345
Enter Name: JAY
Enter Department: HR
Enter Salary: 10000

Enter details for employee 2:
Enter Employee ID: 67890
Enter Name: EMMA
Enter Department: IT
Enter Salary: 20000

--- Employee Details ---

Employee ID: 12345
Name: JAY
Department: HR
Salary: Rs.10000
----------------------------

Employee ID: 67890
Name: EMMA
Department: IT
Salary: Rs.20000
----------------------------
```

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

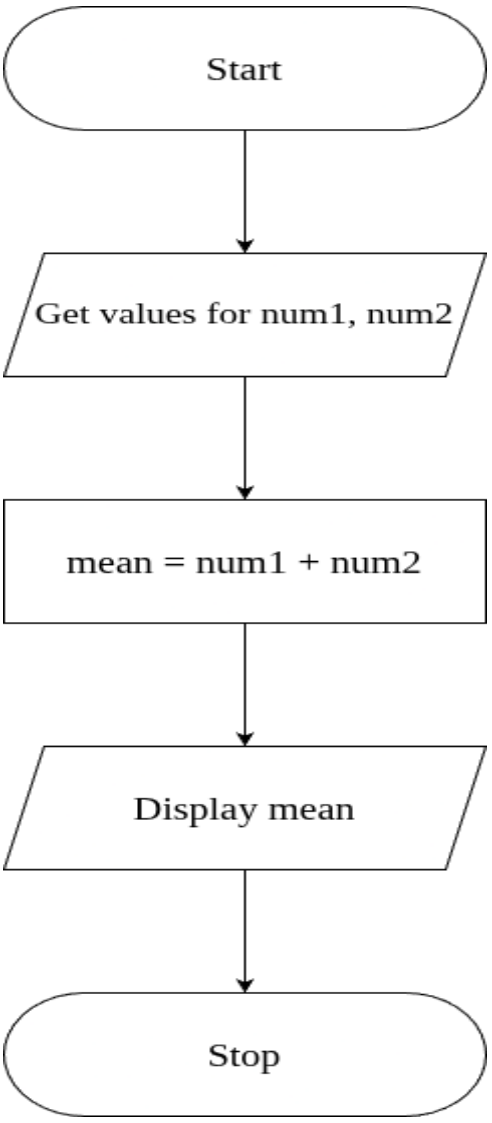
Step 2: Read values num1, num2

Step 3: mean = num1 + num2

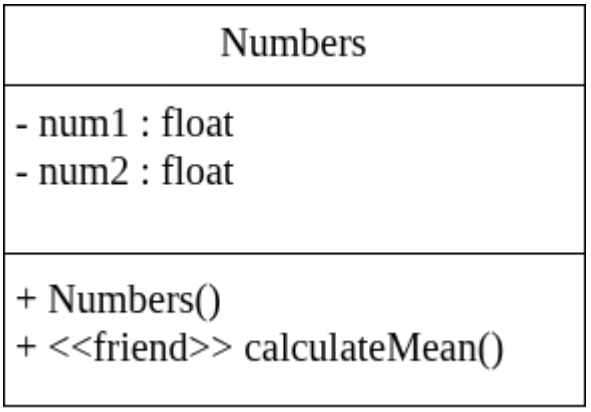
Step 4: Display mean

Step 5: Stop

FLOWCHART :



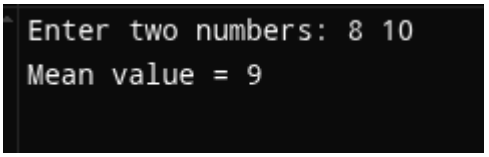
UML DIAGRAM :



SOURCE CODE :

https://github.com/112514020-create/Record_OOPS/blob/main/FriendFnMean.cpp

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.

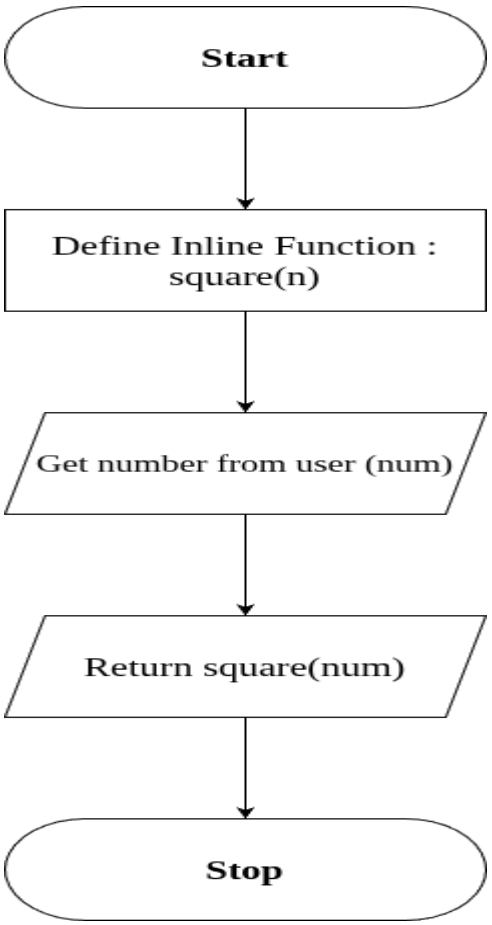
Step 4: Read the value of num from the user.

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

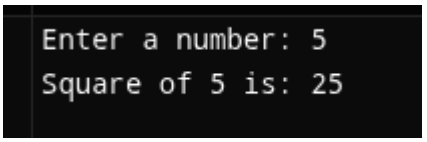
FLOWCHART :



SOURCE CODE :

https://github.com/112514020-create/Record_OOPS/blob/698f27cf11dfb3321eafcbc42157f6d0e0a162c6/InlineFn.cpp

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).

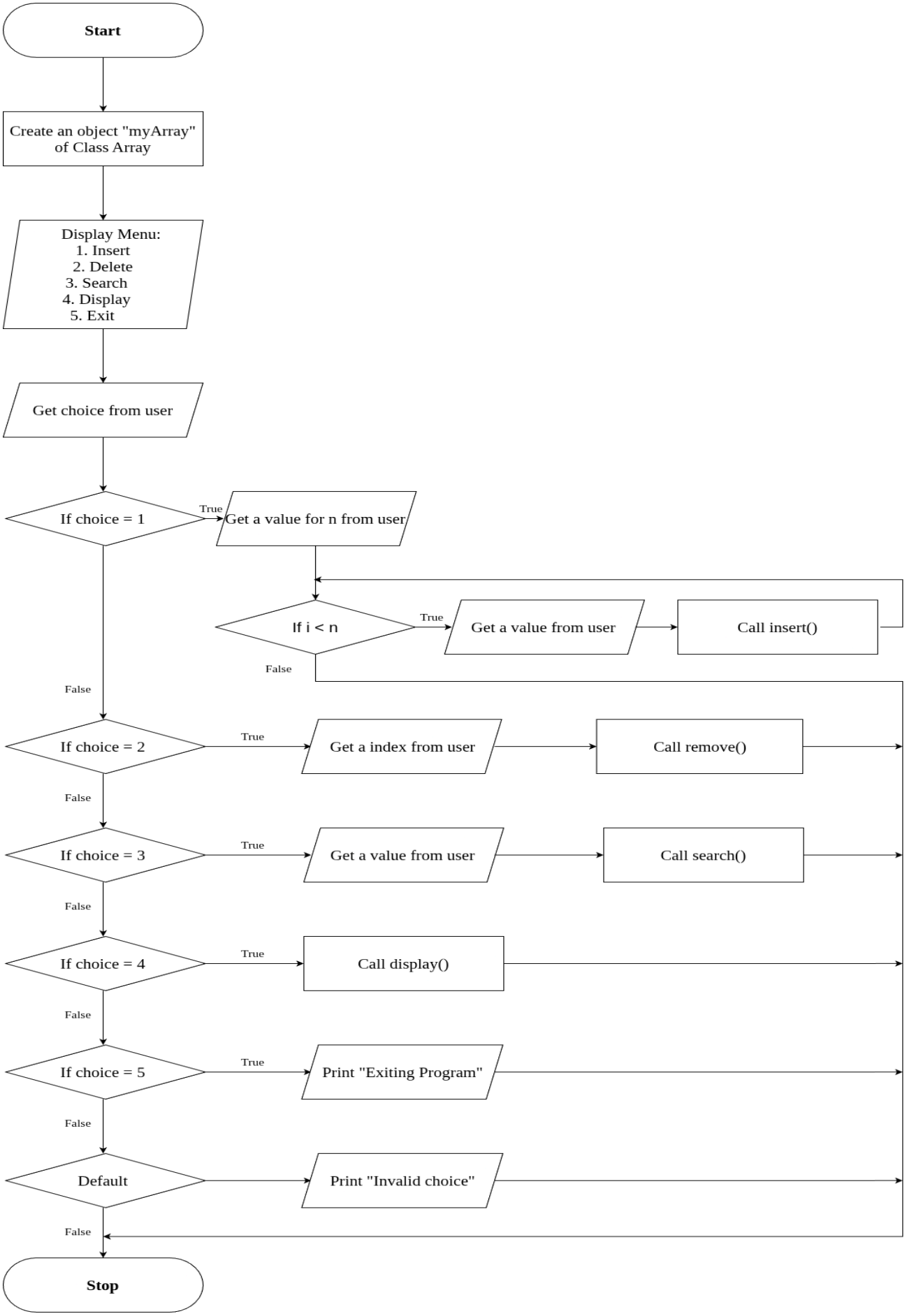
Go back to Step 5 and repeat until i < size becomes False.

Else; Exit the loop.

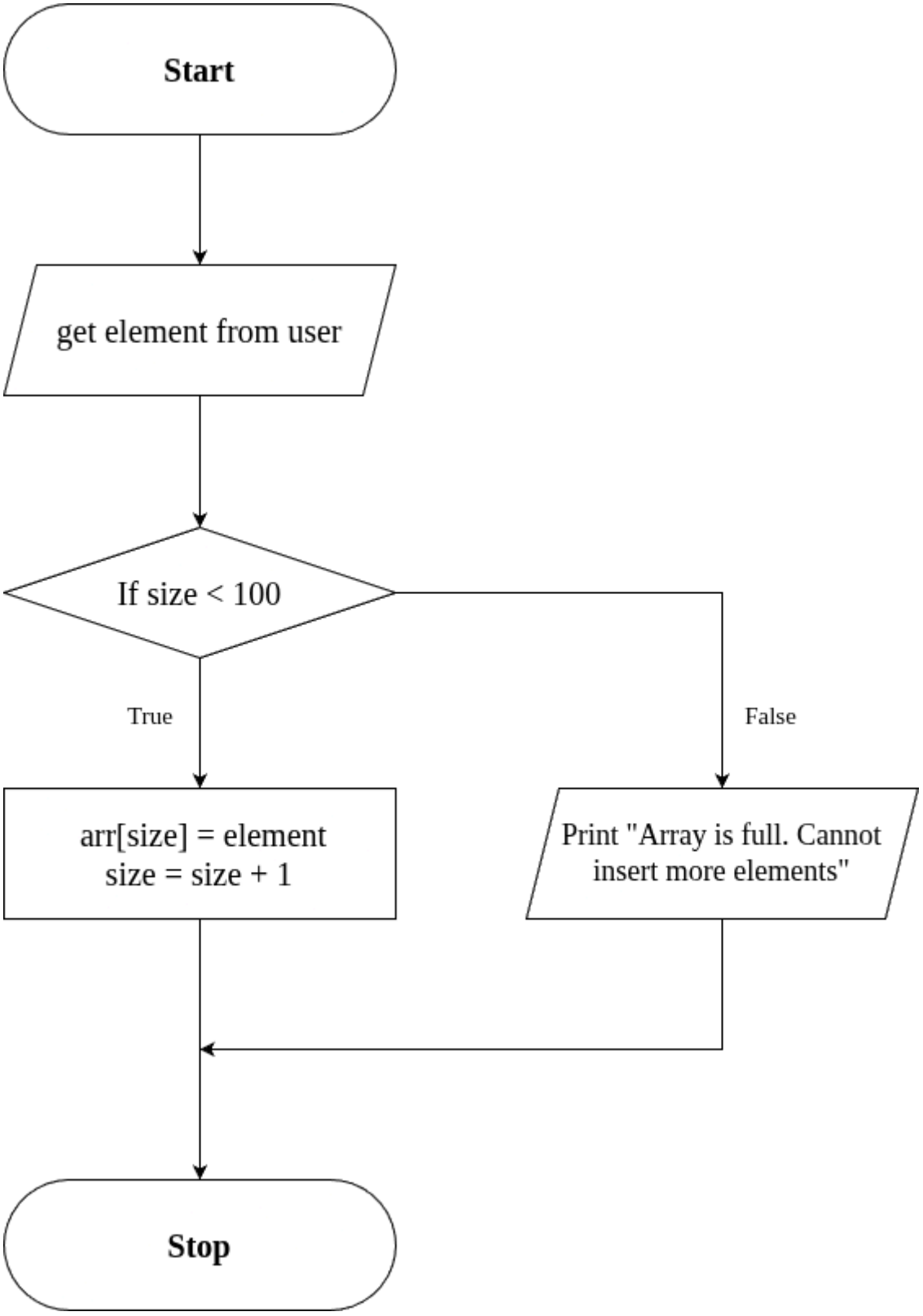
Step 6: After printing all elements, end the loop.

Step 7: Stop the program.

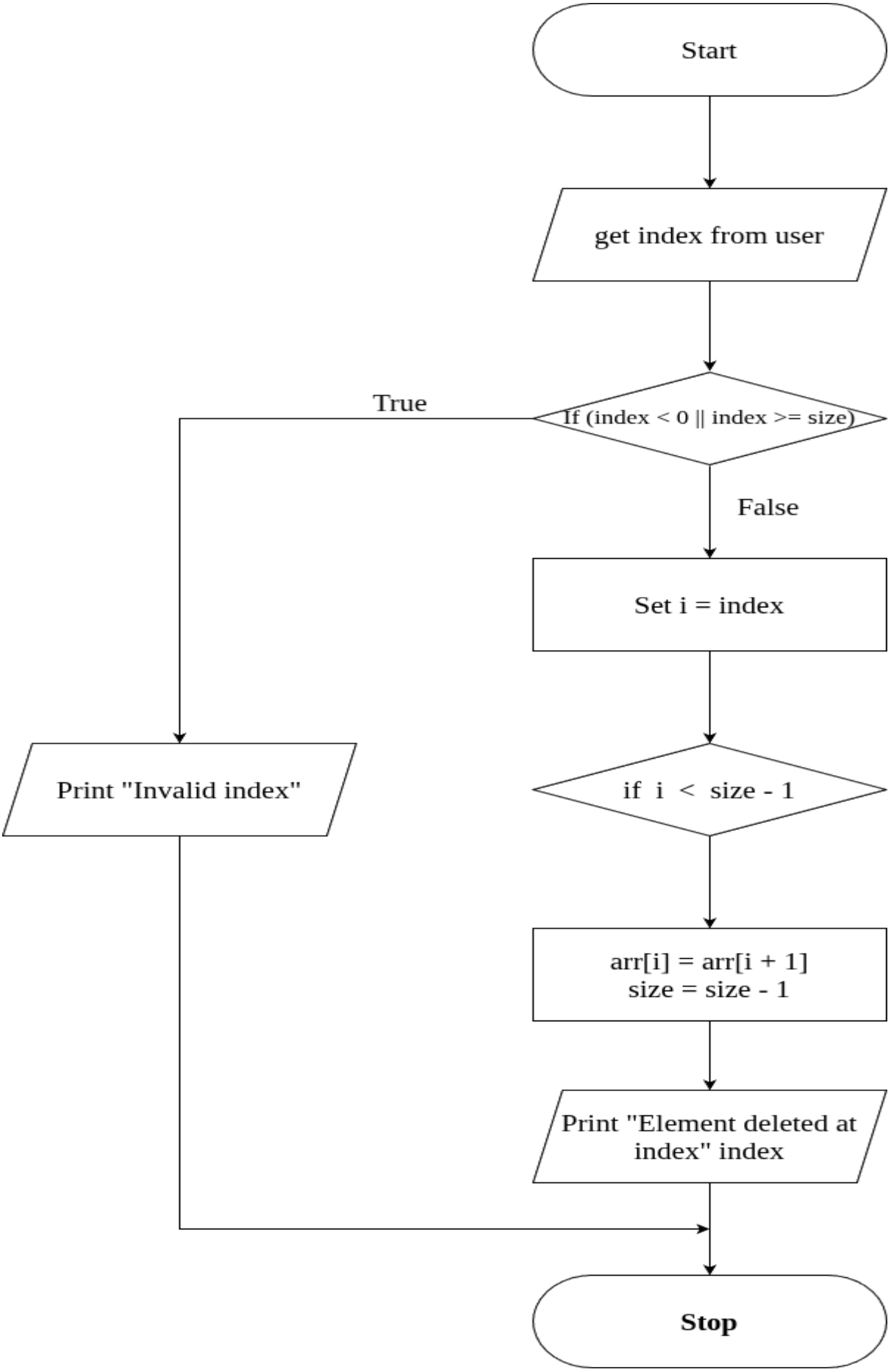
FLOWCHART :


Start
Create an object "myArray" of Class Array
Display Menu:
1. Insert
2. Delete
3. Search
4. Display
5. Exit
Get choice from user
If choice = 1
True
Get a value for n from user
If i < n
True
Get a value from user
Call insert()
False
False
If choice = 2
True
Get a index from user
Call remove()
False
If choice = 3
True
Get a value from user
Call search()
False
If choice = 4
True
Call display()
False
If choice = 5
True
Print "Exiting Program"
False
Default
Print "Invalid choice"
False
Stop

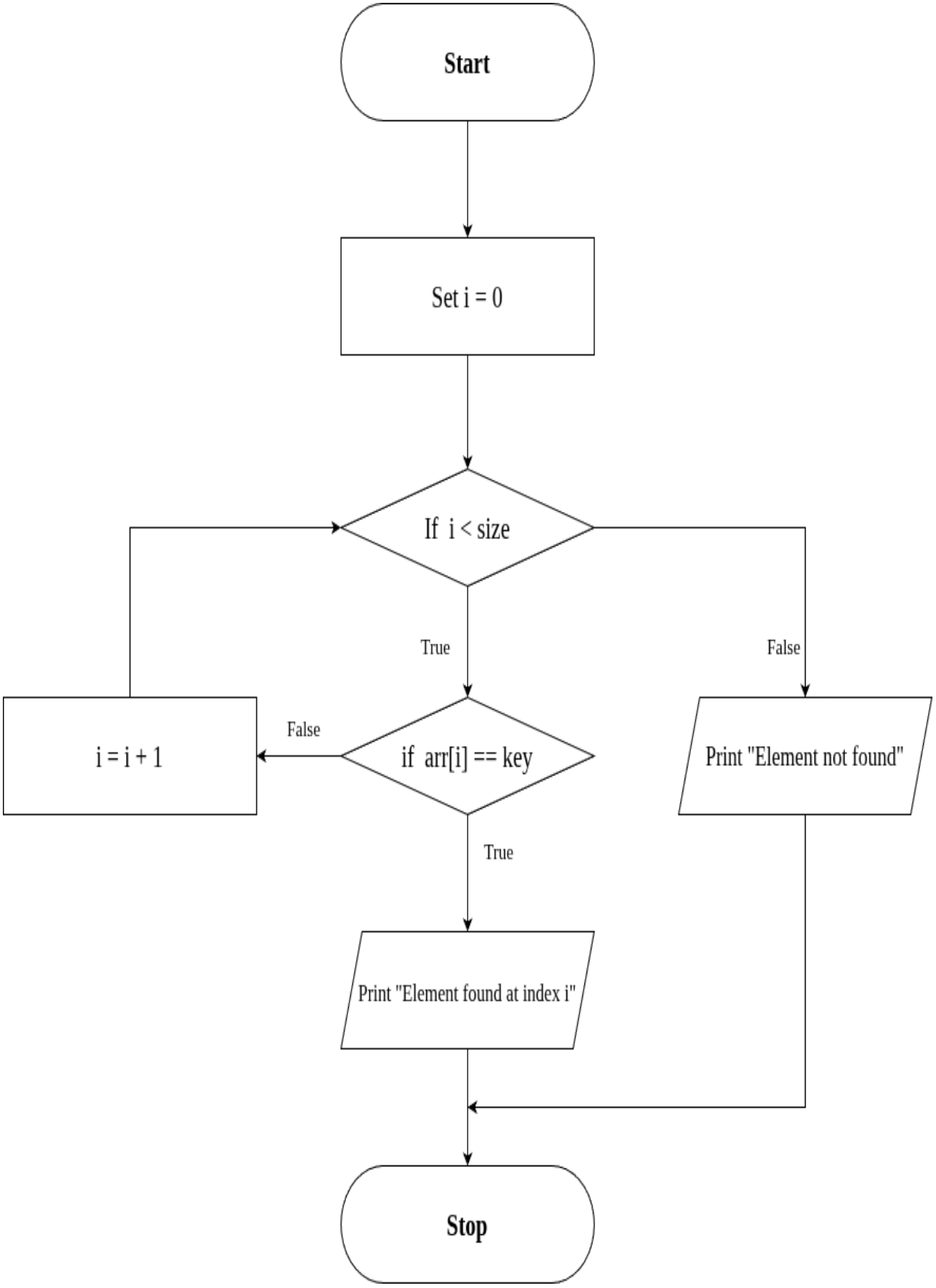
FLOWCHART FOR INSERT FUNCTION :



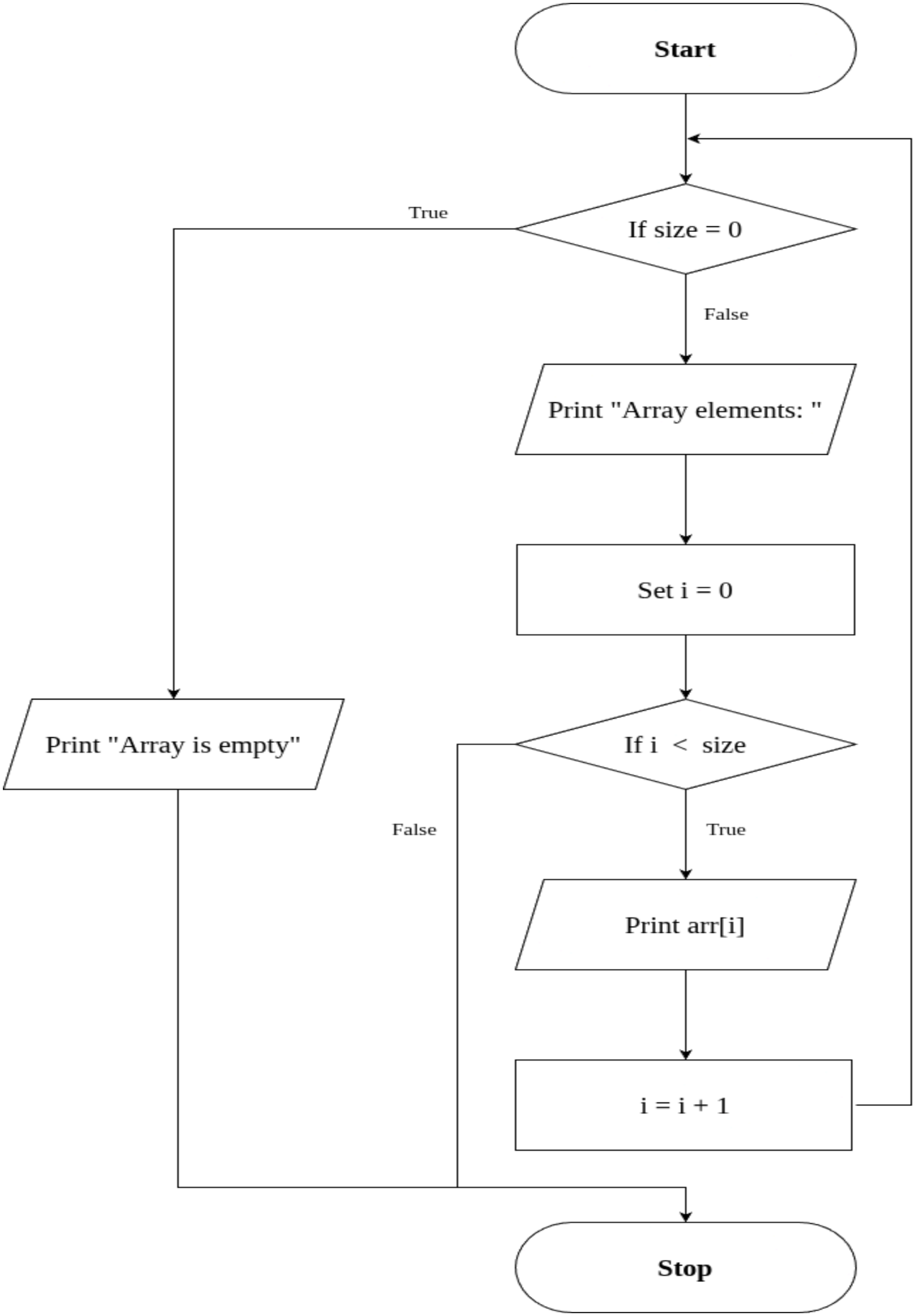
FLOWCHART FOR REMOVE FUNCTION :



FLOWCHART FOR SEARCH FUNCTION :



FLOWCHART FOR DISPLAY FUNCTION :



SOURCE CODE :

https://github.com/112514020-create/Record_OOPS/blob/698f27cf11dfb3321eafcbc42157f6d0e0a162c6/ArrayImpl.cpp

OUTPUT :

```
--- Array Operations ---
1. Insert
2. Delete
3. Search
4. Display
5. Exit
Enter choice: 1
How many values do you want to insert? 5
Enter 5 values: 1 2 3 4 5
```

```
--- Array Operations ---
1. Insert
2. Delete
3. Search
4. Display
5. Exit
Enter choice: 1
How many values do you want to insert? 1
Enter 1 values: 100
Array is full. Cannot insert more elements.
```

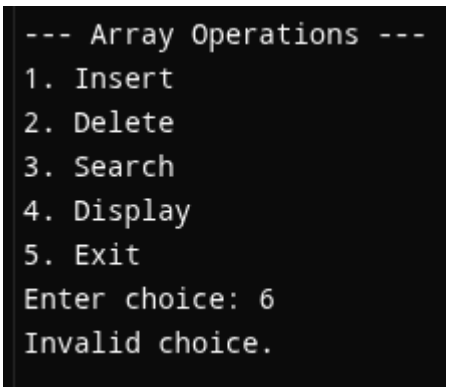
```
--- Array Operations ---
1. Insert
2. Delete
3. Search
4. Display
5. Exit
Enter choice: 2
Enter index to delete: 3
Element deleted at index 3.
```

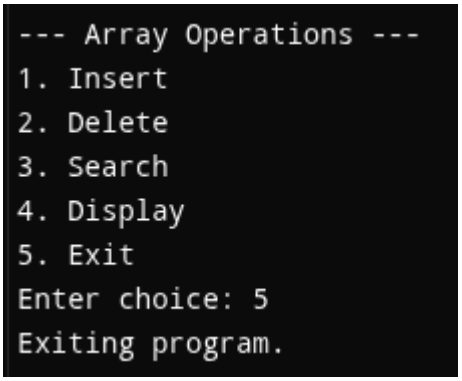
```
--- Array Operations ---
1. Insert
2. Delete
3. Search
4. Display
5. Exit
Enter choice: 2
Enter index to delete: 5
Invalid index.
```

```
--- Array Operations ---
1. Insert
2. Delete
3. Search
4. Display
5. Exit
Enter choice: 3
Enter value to search: 5
Element 5 found at index 3.
```

```
--- Array Operations ---
1. Insert
2. Delete
3. Search
4. Display
5. Exit
Enter choice: 3
Enter value to search: 4
Element 4 not found.
```

```
--- Array Operations ---
1. Insert
2. Delete
3. Search
4. Display
5. Exit
Enter choice: 4
Array elements: 1 2 3 5
```





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:

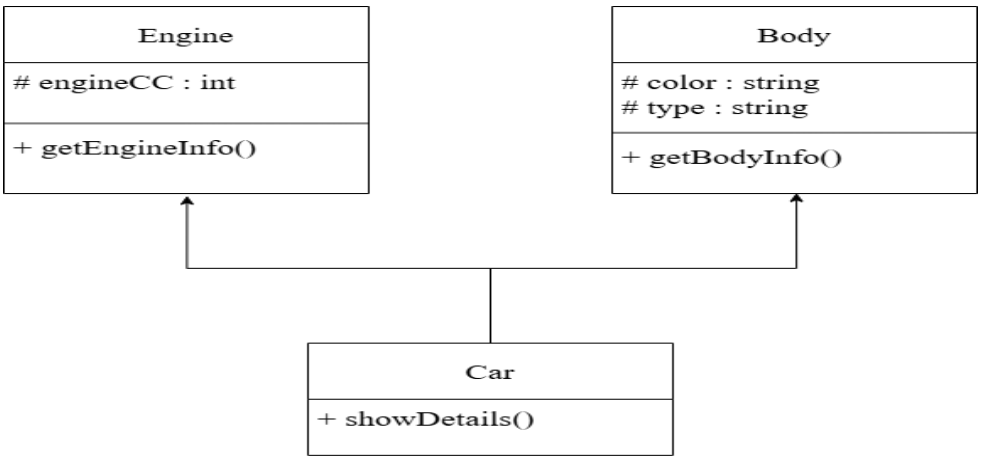
• Displays engineCC, color, and type.

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 :

https://github.com/112514020-create/Record_OOPS/blob/698f27cf11dfb3321eafcbc42157f6d0e0a162c6/MultipleInheritance.cpp

OUTPUT :

```
Enter Engine Capacity (in CC): 2000
Enter Vehicle Color: Blue
Enter Body Type (SUV/Sedan/Hatchback/etc): SUV

--- Vehicle Information ---
Engine Capacity: 2000 CC
Body Color: Blue
Body Type: SUV
```

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

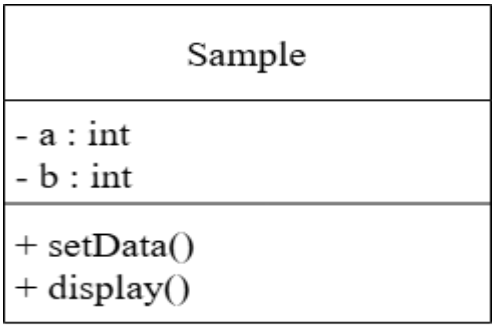
Step 5: In main(), create an object obj of class Sample

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 :

https://github.com/112514020-create/Record_OOPS/blob/698f27cf11dfb3321eafcbc42157f6d0e0a162c6/ThisPointer.cpp

OUTPUT :

```
Value of a: 10
Value of b: 20
```

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.

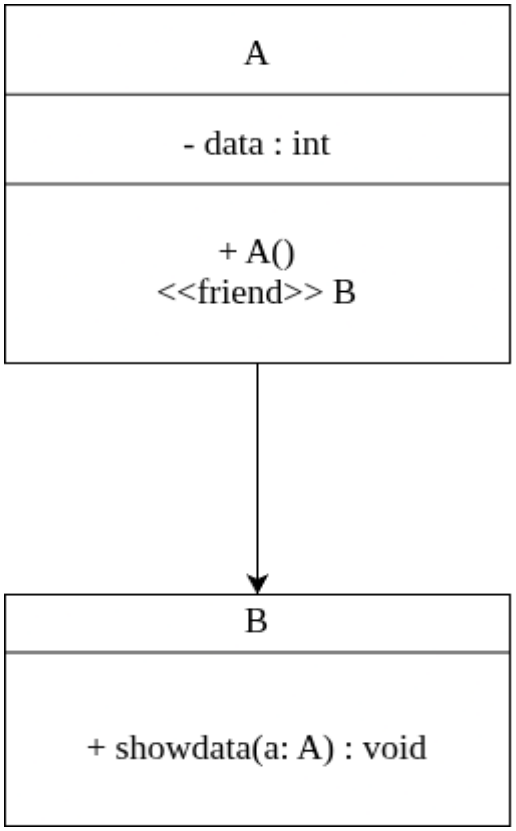
   [Accesses and displays the private variables (name) and (balance) of BankAccount using the reference to the object.]

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 :

https://github.com/112514020-create/Record_OOPS/blob/698f27cf11dfb3321eafcbc42157f6d0e0a162c6/FriendClass.cpp

OUTPUT :

```
Auditor inspecting account...
Account Holder: Alice
Account Balance: 5400
```

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.

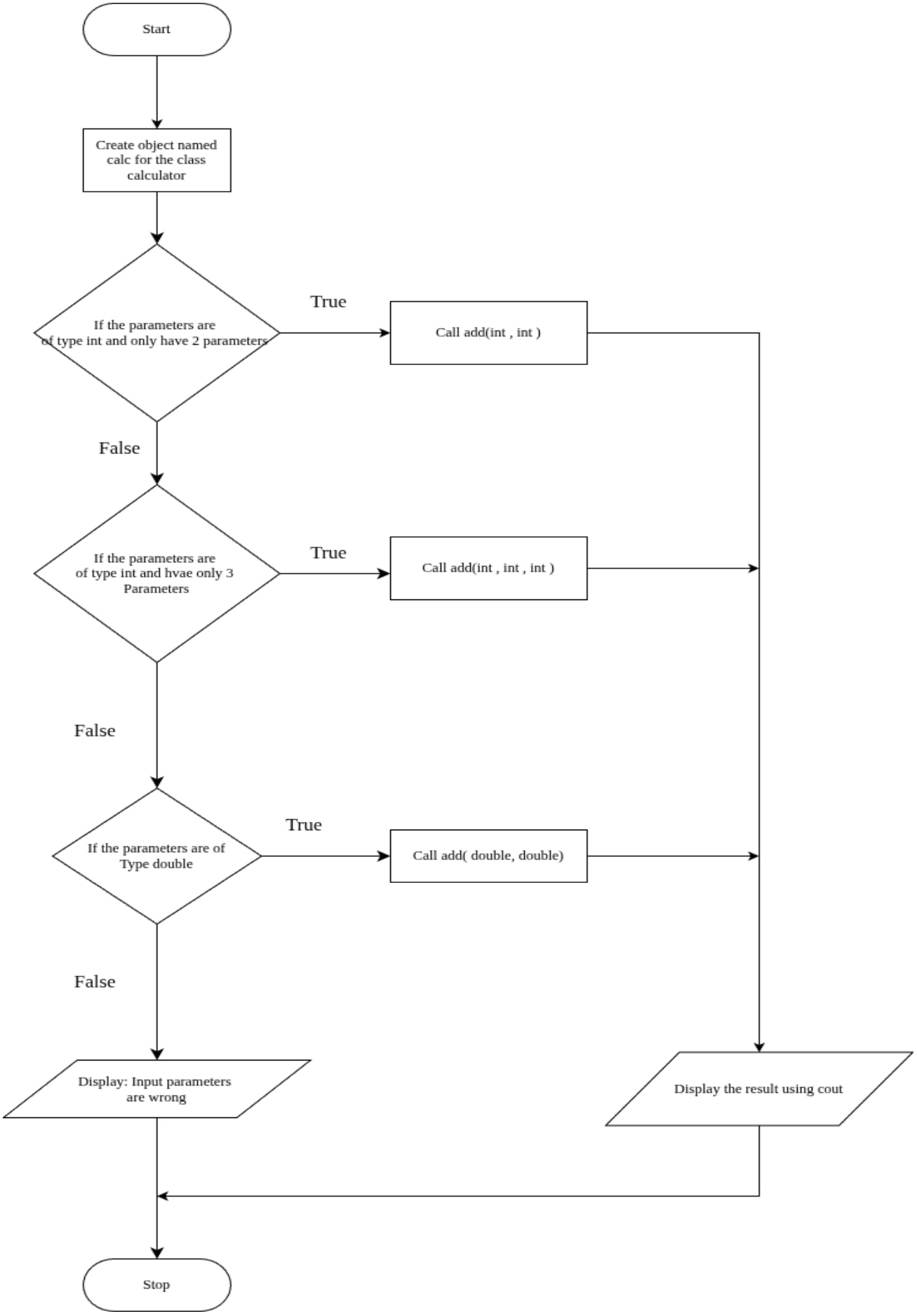
Step 4: In the main() function:

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.

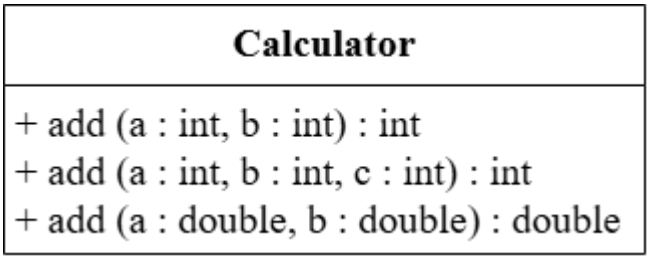
Step 5: Display each result using cout.

Step 6: End.

FLOWCHART :


Start
Create object named calc for the class calculator
If the parameters are of type int and only have 2 parameters
True
Call add(int , int)
False
If the parameters are of type int and hvae only 3 Parameters
True
Call add(int , int , int)
False
If the parameters are of Type double
True
Call add(double, double)
False
Display: Input parameters are wrong
Display the result using cout
Stop

UML DIAGRAM :



SOURCE CODE :

https://github.com/112514020-create/Record_OOPS/blob/698f27cf11dfb3321eafcbc42157f6d0e0a162c6/FriendClass.cpp

OUTPUT :

```
Sum of 2 and 3: 5
Sum of 1, 2 and 3: 6
Sum of 2.5 and 3.7: 6.2
```

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:

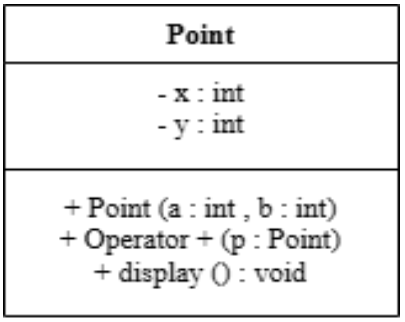
a) Create two objects of Class "Point": p1(2,3) and p2(4,5).

b) Use the overloaded + operator to add the two objects: p3=p1+p2.

c) Display the result using p3.display().

Step 8: Stop the program.

UML DIAGRAM :



SOURCE CODE :

https://github.com/112514020-create/Record_OOPS/blob/698f27cf11dfb3321eafcbc42157f6d0e0a162c6/OperOverloading.cpp

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

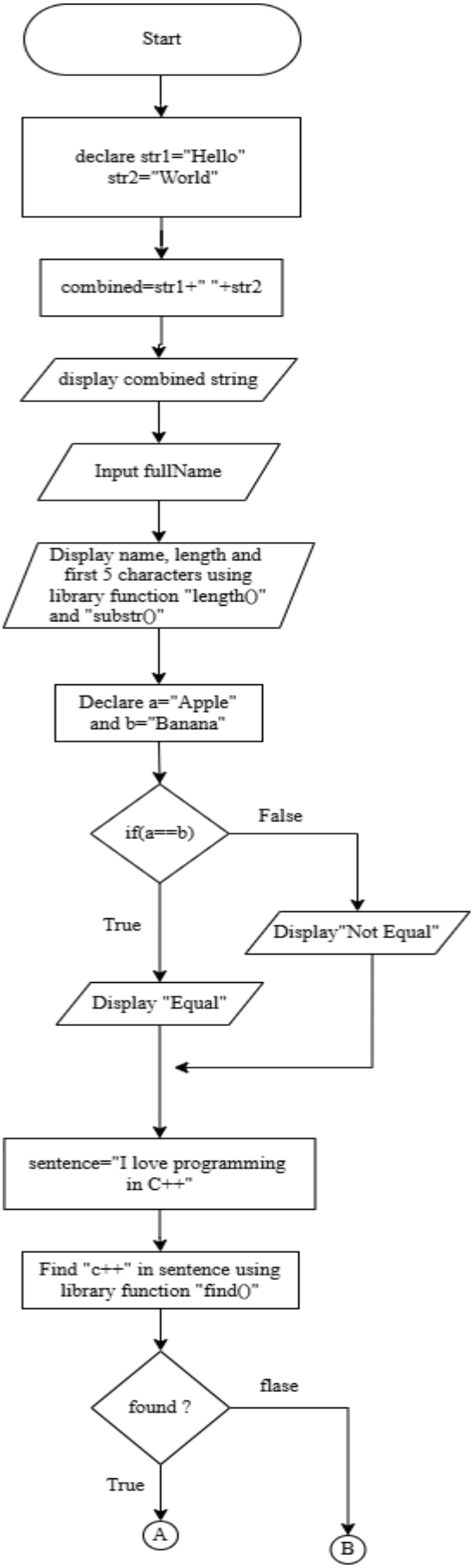
Step 13: Replace "C++" with "Python" in a sentence (using "replace()" library function)

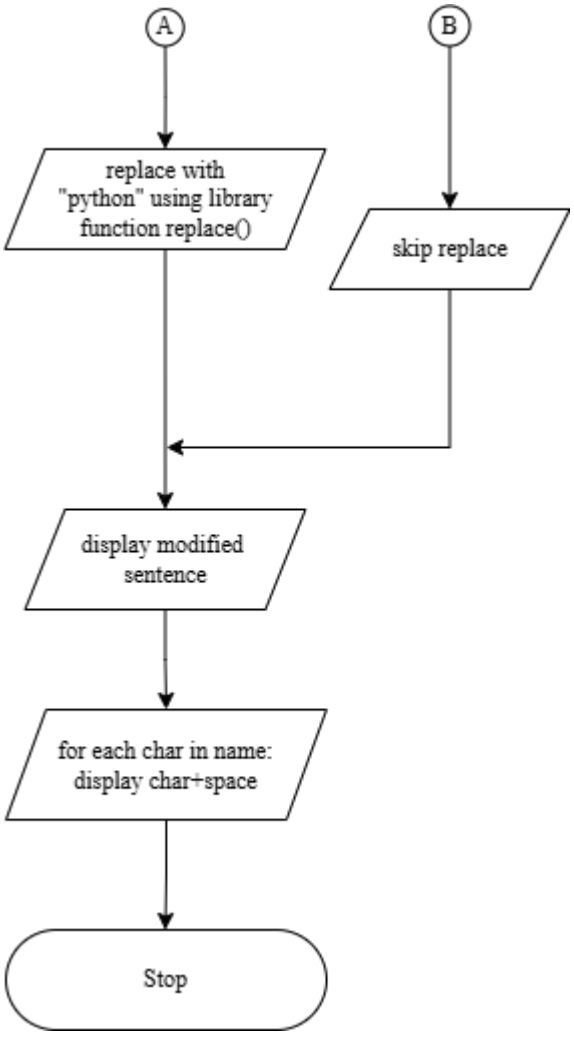
Step 14: Display modified sentence

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

Step 16: End

FLOWCHART :


Start
declare str1="Hello"
str2="World"
combined=str1+" "+str2
display combined string
Input fullName
Display name, length and
first 5 characters using
library function "length()"
and "substr()"
Declare a="Apple"
and b="Banana"
if(a==b)
False
True
Display"Not Equal"
Display "Equal"
sentence="I love programming
in C++"
Find "c++" in sentence using
library function "find()"
found ?
flase
True
A
B



SOURCE CODE :

https://github.com/112514020-create/Record_OOPS/blob/698f27cf11dfb3321eafcbc42157f6d0e0a162c6/StringManipulation.cpp

OUTPUT :

```
Combined string: Hello World
Enter your full name: John Caleb
Your name is: John Caleb
Length of your name: 10
First 5 characters: John
Strings are not equal
Modified sentence: I love programming in Python.
Characters in your name: J o h n   C a l e b
```

RESULT :

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