Practical No.1

Aim: Basic C++ Program Structure

a. Write a program to display "Hello, World!".
b. Use #include <iostream>, main () function, and return 0;
c. Compile and run using command line (g++) or IDE.

Name: Sandip T. Hatnore
Class:FY.BSc.CS(A**)**
Roll No: 37
Subject: Object Oriented Programming with C++
Sign:

A. Write a program to display "Hello, World!".

Source Code:

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

Output:

```
Hello, World!

--------------------------------
```

Source Code:

Program 1: Add Three Numbers

```
#include <iostream>
using namespace std;
int main() {
    int a, b, c, sum;
    cout << "Enter three numbers: ";
    cin >> a >> b >> c;
    sum = a + b + c;
    cout << "The sum is: " << sum << endl;
    return 0;
}
```

Program 2: Calculate Area of a Rectangle

```
#include <iostream>
using namespace std;
int main() {
    float length, width, area;
    cout << "Enter length of the rectangle: ";
    cin >> length;
    cout << "Enter width of the rectangle: ";
```

```
    cin >> width;
    area = length * width;
    cout << "Area of the rectangle is: " << area << endl;
    return 0;
}
```

Output:

Program 1:

```
Enter three numbers: 22 32 43
The sum is: 97

--------------------------------
```

Program 2:

```
Enter length of the rectangle: 14
Enter width of the rectangle: 12
Area of the rectangle is: 168

--------------------------------
```

C.Compile and run using command line (g++) or IDE.

✅ Using Command Line (g++)

Step 1: Write your code

Save your C++ code in a file, for example:
main.cpp

Step 2: Compile

Open your terminal (or Command Prompt / PowerShell on Windows) and run:

bash

CopyEdit

```
g++ main.cpp -o main
```

- main.cpp = your source file
- -o main = output file name (this will be your executable)

Step 3: Run the program

On Linux/macOS:

bash

CopyEdit

```
./main
```

On Windows:

c

CopyEdit

```
main.exe
```

✅ Using an IDE (like Code::Blocks, VS Code, or CLion)

1. Open the IDE
2. Create a new project or file
3. Paste/write your code
4. Click “Build” or “Run” (or use shortcuts like F5, Ctrl+Shift+B, or Shift+F10, depending on the IDE)

Practical No.2

Aim: Using Data Types and Operators

A. Declare and initialize int, float, char, bool.
B. Write a calculator for +, -, *, /, %.
C. Use logical operators (&&, ||, !) in conditional statements.

Name: Sandip T. Hatnore
Class:FY.BSc.CS(A**)**
Roll No: 37
Subject: Object Oriented Programming with C++
Sign:

A.Declare and initialize int, float, char, bool.

Source Code:

```
#include <iostream>
#include <string>   // Needed for string type
using namespace std;
int main() {
    int age = 19;
    float height = 5.6f;
    char grade = 'A';
    bool isPassed = true;
    string name = "Sandip";
    cout << "Name: " << name << endl;
    cout << "Age (int): " << age << endl;
    cout << "Height (float): " << height << endl;
    cout << "Grade (char): " << grade << endl;
    cout << "Passed (bool): " << isPassed << endl;
    return 0;
}
```

Output :

```
Name: Sandip
Age (int): 19
Height (float): 5.6
Grade (char): A
Passed (bool): 1

--------------------------
```

B.Write a calculator for +, -, *, /, %.

Source Code:

```
#include <iostream>
using namespace std;
int main() {
    int a, b;
```

```
    cout << "Enter two numbers: ";
    cin >> a >> b;
    cout << "Addition: " << a + b << endl;
    cout << "Subtraction: " << a - b << endl;
    cout << "Multiplication: " << a * b << endl;
    cout << "Division: " << a / b << endl;
    cout << "Modulus: " << a % b << endl;
    return 0;
}
```

Output :

```
Enter two numbers: 43 23
Addition: 66
Subtraction: 20
Multiplication: 989
Division: 1
Modulus: 20

--------------------------------
```

C. Use logical operators (&&, ||, !) in conditional statements.

Source Code:

```
#include <iostream>
using namespace std;
int main() {
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    (a > 0 && b > 0) && cout << "Both numbers are positive.\n";
    (a < 0 || b < 0) && cout << "At least one number is negative.\n";
    !(a == b) && cout << "The numbers are not equal.\n";
    (a == b) && cout << "The numbers are equal.\n";

    return 0;
}
```

Output :

```
Enter two numbers: 42 65
Both numbers are positive.
The numbers are not equal.

--------------------------------
```

Practical No. 3

Aim: Conditional Statements and Loops

A. Check whether a number is even or odd using if...else.
B. Menu-driven program using switch
C. Use for, while, and do...while to:
   i. Print a triangle pattern
   ii. Find the sum of first N numbers

Name: Sandip T. Hatnore
Class:FY.BSc.CS(A**)**
Roll No: 37
Subject: Object Oriented Programming with C++
Sign:

A.Check whether a number is even or odd using if...else.

Source Code:

```
#include <iostream>
using namespace std;
int main() {
    int number;
    cout << "Enter a number: ";
    cin >> number;
    if (number % 2 == 0)
        cout << number << " is Even." << endl;
    else
        cout << number << " is Odd." << endl;
    return 0;
}
```

Output :

```
Enter a number: 7
7 is Odd.

--------------------
```

```
Enter a number: 6
6 is Even.

--------------------
```

B.Menu-driven program using switch.

Source Code:

```
#include <iostream>
using namespace std;
int main() {
    int choice;
    cout << "Menu:\n";
    cout << "1. Add\n";
    cout << "2. Subtract\n";
    cout << "3. Multiply\n";
    cout << "4. Divide\n";
```

```
    cout << "Enter your choice: ";
    cin >> choice;
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    switch (choice) {
        case 1:
            cout << "Sum = " << a + b << endl;
            break;
        case 2:
            cout << "Difference = " << a - b << endl;
            break;
        case 3:
            cout << "Product = " << a * b << endl;
            break;
        case 4:
            if (b != 0)
                cout << "Quotient = " << a / b << endl;
            else
                cout << "Cannot divide by zero!" << endl;
            break;
        default:
            cout << "Invalid choice." << endl;
    }
    return 0;
}
```

Output :

```
Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter your choice: 2
Enter two numbers: 67
54
Difference = 13

--------------------------------
```

C.Use for, while, and do...while to:

i. Print a triangle pattern

Source Code:

```
#include <iostream>
using namespace std;
int main() {
    int rows;
    cout << "Enter number of rows: ";
    cin >> rows;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            cout << "* ";
        }
        cout << endl;
    }
    return 0;
}
```

Output :

```
Enter number of rows: 5
*
* *
* * *
* * * *
* * * * *

--------------------------------
```

ii. Find the sum of first N numbers

Source Code:

```
#include <iostream>
using namespace std;
int main() {
    int N, i = 1, sum = 0;
```

```
    cout << "Enter a number N: ";
    cin >> N;
    while (i <= N) {
        sum += i;
        i++;
    }
    cout << "Sum of first " << N << " numbers is: " << sum << endl;
    return 0;
}
```

Output :

```
Enter a number N: 4
Sum of first 4 numbers is: 10

--------------------------------
```

Practical No.4

Aim: Arrays and String Manipulation

A. Declare and display elements of a 1D array.
B. Sort and search elements in an array.
C. Use character arrays to reverse a string.

Name: Sandip T. Hatnore
Class: FY.BSc.CS(A**)**
Roll No: 37
Subject: Object Oriented Programming with C++
Sign:

A. Declare and display elements of a 1D array.

source code:

```
#include &lt;iostream&gt;
using namespace std;
int main() {
// Declaration and initialization of an array
int arr[5] = {10, 20, 30, 40, 50};
// Accessing elements of the array
cout &lt;&lt; &quot;Element at index 2: &quot; &lt;&lt; arr[2] &lt;&lt; endl;
// Modifying elements of the array
arr[3] = 60;
cout &lt;&lt; &quot;Modified element at index 3: &quot; &lt;&lt; arr[3] &lt;&lt;
endl;
// Calculating the sum of all elements
int sum = 0;
for (int i = 0; i &lt; 5; i++) {
sum += arr[i];
}
cout &lt;&lt; &quot;Sum of all elements: &quot; &lt;&lt; sum &lt;&lt; endl;
return 0;
}
```

Output-

```
Output:

Element at index 2: 30
Modified element at index 3: 60
Sum of all elements: 170
```

B. Sort and search elements in an array

Source code:

```
// C++ Program to how to sort an array using
// std::sort() function
#include &lt;iostream&gt;
using namespace std;
int main() {
int arr[] = {5, 4, 1, 2, 3};
// Calculate the size of the array
int n = sizeof(arr) / sizeof(arr[0]);
```

```
// Sort the array using std::sort()
sort(arr, arr + n);
for (auto i : arr)
cout &lt;&lt; i &lt;&lt; &quot; &quot;;
return 0;
}
```

output-

```
1 2 3 4 5
```

C. Use character arrays to reverse a string

Source code:

```
#include &lt;iostream&gt;
using namespace std;
int main() {
string s = &quot;Hello World&quot;;
// Using reverse() function to reverse s
reverse(s.begin(), s.end());
cout &lt;&lt; s;
return 0;
}
```

output-

```
dlroW olleH
```

Practical No. 5

Aim: Constructors and Destructors

A. Create a class with a default constructor to initialize values.
B. Use parameterized constructors to accept data.
C. Implement a copy constructor.
D. Add a destructor to show object cleanup.

Name: Sandip T. Hatnore
Class:FY.BSc.CS(A)
Roll No: 37
Subject: Object Oriented Programming with C++
Sign:

A.Create a class with a default constructor to initialize values.

Source Code:

```
#include <iostream>
using namespace std;
class Student {
private:
   string name;
   int rollNo;
   float marks;
public:
   // Default constructor
   Student() {
      name = "Unknown";
      rollNo = 0;
      marks = 0.0;
      cout << "Default constructor called!" << endl;
   }
   // Function to display data
   void display() {
      cout << "Name: " << name << endl;
      cout << "Roll No: " << rollNo << endl;
      cout << "Marks: " << marks << endl;
   }
};
int main() {
   // Creating an object (default constructor is called automatically)
   Student s1;
   // Display initialized values
   s1.display();
   return 0;
}
```

Output:

```
Default constructor called!
Name: Unknown
Roll No: 0
Marks: 0
```

B.Use parameterized constructors to accept data.

Source Code:

```
#include <iostream>
using namespace std;
class Square {
private:
    int side;
public:
    // Parameterized constructor
    Square(int s) {
        side = s;
    }
    void area() {
        cout << "Area of square = " << side * side << endl;
    }
};
int main() {
    Square sq(5); // Passing side length as argument
    sq.area();
    return 0;
}
```

Output:

```
Area of square = 100
```

C.Implement a copy constructor.

Source Code:

```
#include <iostream>
using namespace std;
// Create a demo class
class A {
public:
```

```
    int x;
};
int main() {
        // Creating an a1 object
    A a1;
    a1.x = 10;
    cout << "a1's x = " << a1.x << endl;
    // Creating another object using a1
        // Copy Constructor Calling
    A a2(a1);
    cout << "a2's x = " << a2.x;
    return 0;
}
```

Output:

```
a1's x = 10
a2's x = 10
```

D.Add a destructor to show object cleanup.

Source Code:

```
#include <iostream>
using namespace std;
class MyClass {
private:
    // Pointer to dynamically
    // allocated memory
    int* data;
public:
    MyClass(int value) {
        data = new int;
        *data = value;
        cout << *data << endl;
    }
    // User-defined destructor: Free
    // the dynamically allocated memory
    ~MyClass() {
        // Deallocate the dynamically
        // allocated memory
        delete data;
        cout << "Destructor: Memory deallocated";
```

```
    }
};
int main() {
    MyClass obj1(10);
    return 0;
}
```

Output:

```
10
Destructor: Memory deallocated
```

Practical No. 6

Aim: Function Overloading, Operator Overloading, and Overriding

A. Create overloaded functions (area() for circle, rectangle).
B. Overload + operator to add two complex numbers.
C. Overload == to compare two student objects.
D. Implement base class with virtual function and override it in derived class.

Name: Sandip T. Hatnore
Class:FY.BSc.CS(A)
Roll No: 37
Subject: Object Oriented Programming with C++
Sign:

A.Create overloaded functions (area() for circle, rectangle).

Source Code:

```
#include <iostream>
using namespace std;
class Shape {
public:
   // ?? Member function: calculates area of a circle
   void area(double radius) {
      cout << "Area of circle = " << 3.14 * radius * radius << endl;
   }
   // ?? Member function: calculates area of a rectangle
   void area(double length, double width) {
      cout << "Area of rectangle = " << length * width << endl;
   }
};
int main() {
   Shape s;          // object creation
   s.area(5);        // calls area() for circle
   s.area(4, 6);     // calls area() for rectangle
   return 0;
}
```

Output:

```
Area of circle = 78.5
Area of rectangle = 24
```

B.Overload + operator to add two complex numbers.

Source Code:

```
#include <iostream>
using namespace std;
class Complex {
private:
   float real, imag; // data members
public:
   // Parameterized constructor
   Complex(float r = 0, float i = 0) {
      real = r;
```

```
        imag = i;
    }
    // Operator overloading for +
    Complex operator + (const Complex &obj) {
        Complex temp;
        temp.real = real + obj.real;
        temp.imag = imag + obj.imag;
        return temp;
    }
    // Display function
    void display() {
        cout << real << " + " << imag << "i" << endl;
    }
};
int main() {
    Complex c1(2.5, 3.5), c2(1.5, 4.5);
    Complex c3 = c1 + c2; // calls overloaded + operator
    cout << "First complex number: ";
    c1.display();
    cout << "Second complex number: ";
    c2.display();
    cout << "Sum: ";
    c3.display();
    return 0;
}
```

Output:

```
First complex number: 2.5 + 3.5i
Second complex number: 1.5 + 4.5i
Sum: 4 + 8i
```

C.Overload == to compare two student objects.

Source Code:

```
#include <iostream>
```

```
using namespace std;
class Student {
    int rollNo;
public:
    Student(int r) {
        rollNo = r;
    }
    // Overload == operator
    bool operator==(const Student &obj) {
        return rollNo == obj.rollNo;
    }
};
int main() {
    Student s1(101), s2(101), s3(102);
    if (s1 == s2)
        cout << "s1 and s2 are same\n";
    else
        cout << "s1 and s2 are different\n";
    if (s1 == s3)
        cout << "s1 and s3 are same\n";
    else
        cout << "s1 and s3 are different\n";
    return 0;
}
```

Output:

```
s1 and s2 are same
s1 and s3 are different
```

D.Implement base class with virtual function and override it in derived class.

Source Code:

```
#include <iostream>
using namespace std;
// Base class
class Shape {
public:
```

```
    // Virtual function
    virtual void draw() {
        cout << "Drawing a generic shape" << endl;
    }
};
// Derived class
class Circle : public Shape {
public:
    // Override virtual function
    void draw() override {
        cout << "Drawing a circle" << endl;
    }
};
int main() {
    Shape* shapePtr;   // Base class pointer
    Circle c;
    shapePtr = &c;     // Pointing to derived object
    shapePtr->draw();  // Calls Circle's draw() due to virtual function
    return 0;
}
```

Output:

```
Drawing a circle
```