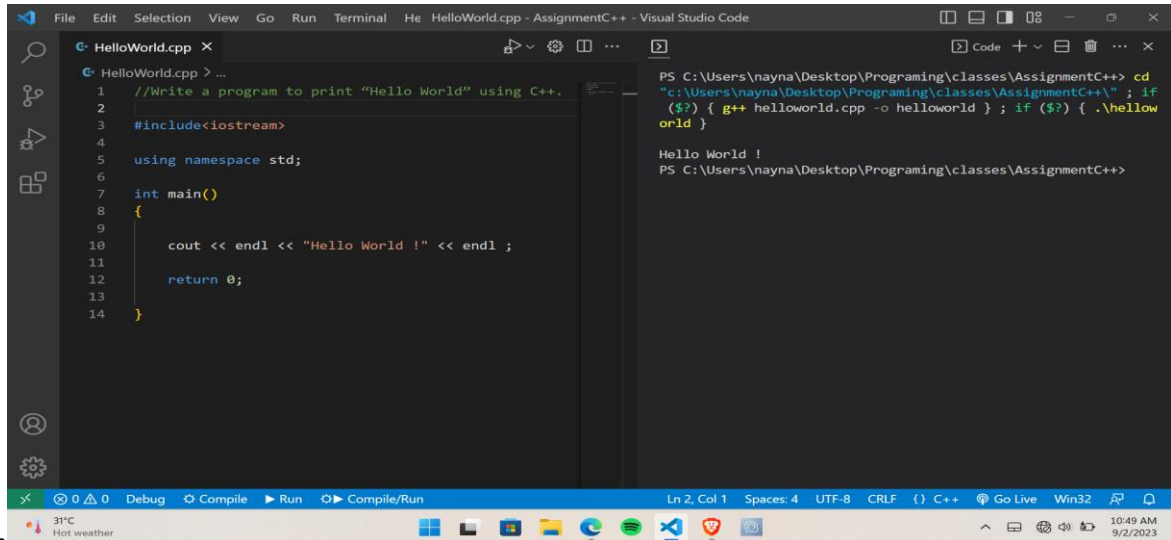


Assignment C++

Module 4: OOP Concepts

Module 4.1: C++ Basic

1> Write a program to print “Hello World” using C++.



The screenshot shows the Visual Studio Code editor with a file named 'HelloWorld.cpp'. The code is as follows:

```
1 //Write a program to print "Hello World" using C++.
2
3 #include<iostream>
4
5 using namespace std;
6
7 int main()
8 {
9
10     cout << endl << "Hello World !" << endl ;
11
12     return 0;
13 }
14
```

On the right, the 'Code' window shows the command prompt output:

```
PS C:\Users\nayna\Desktop\Programing\classes\AssignmentC++> cd "c:\Users\nayna\Desktop\Programing\classes\AssignmentC++"; if ($?) { g++ helloworld.cpp -o helloworld } ; if ($?) { .\helloworld }
Hello World !
PS C:\Users\nayna\Desktop\Programing\classes\AssignmentC++>
```

The status bar at the bottom indicates the file is at line 2, column 1, using UTF-8 encoding and CRLF line endings. The system tray shows a temperature of 31°C and the date 9/2/2023.

Ans.

2> What is OOP? List OOP concepts.

Ans. Object Oriented Programming is a paradigm that supplies many concepts such as **inheritance, data binding, polymorphism etc.**

The programming paradigm where everything is represented as an object is known as truly object-oriented programming language. **Smalltalk** is considered as the first truly object-oriented programming language.

OOP Concepts are following below:

Object means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a method or paradigm to design a program

using classes and objects. It simplifies the software development and maintenance by supplying some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

--> **Object:**

Any entity that has a state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

--> **Class:**

Collection of objects is called class. It is a logical entity.

A Class in C++ is the foundational element that leads to Object-Oriented programming. A class instance must be created to access and use the user-defined data type's data

members and member functions. An object's class acts as its blueprint. Take the class of cars as an example. Even if different names and brands may be used for different cars, all of them will have some characteristics in common, such as four wheels, a speed limit, a range of miles, etc. In this case, the class of car is represented by the wheels, the speed limitations, and the mileage.

--> **Inheritance:**

When one object gets all the properties and behaviors of the parent object, i.e., known as inheritance. It supplies code reusability. It is used to achieve runtime polymorphism.

1. Sub class - Subclass or Derived Class refers to a class that receives properties from another class.
2. Super class - The term "Base Class" or "Super Class" refers to the class from which a subclass inherits its properties.
3. Reusability - As a result, when we wish to create a new class, but an existing class already holds some of the code we need, we can generate our new class from the old class thanks to inheritance. This allows us to use the fields and methods of the pre-existing class.

--> **Polymorphism:**

When **one task is performed in separate ways**, i.e., known as polymorphism. For example: to convince the customer differently, to draw something e.g., shape or rectangle etc.

Different situations may cause an operation to behave differently. The type of data used in the operation decides the behavior.

--> **Abstraction: Hiding internal details and showing functionality** is known as abstraction. Data abstraction is the process of exposing to the outside world only the information that is necessary while concealing implementation or background information. For example: phone call, we do not know the internal processing.

In C++, we use abstract class and interface to achieve abstraction.

--> **Encapsulation:**

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

Encapsulation is typically understood as the grouping of related pieces of information and data into a single entity. Encapsulation is the process of tying together data and the functions that work with it in object-oriented

programming. Look at a practical illustration of encapsulation: at a company, there are various divisions, including the sales division, the finance division, and the accounts division. All financial transactions are handled by the finance sector, which also supports records of all financial data. In a similar vein, the sales section is in charge of all tasks relating to sales and maintains a record of each sale. Now, a scenario could occur when, for some reason, a financial official requires all the information on sales for a specific month. Under the umbrella term "sales section," all of the employees who can influence the sales section's data are grouped together. Data abstraction or concealing is another side effect of encapsulation. In the same way that encapsulation hides the data. In the example, any other area cannot access any of the data from any of the sections, such as sales, finance, or accounts.

--> **Dynamic Binding** - In dynamic binding, a decision is made at runtime regarding the code that will be run in response to a function call. For this, C++ supports virtual functions.

--> **Advantage of OOPs over Procedure-oriented programming language:**

4. OOPs makes development and maintenance easier whereas in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.
5. OOPs provide data hiding whereas in Procedure-oriented programming language global data can be accessed from anywhere.
6. OOPs provide the ability to simulate real-world events much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

--> **Why do we need oops in C++?**

There were various drawbacks to the early methods of programming, as well as poor performance. The approach couldn't effectively address real-world issues since, like procedural-oriented programming, you couldn't reuse the

code within the program again, there was difficulty with global data access, and so on.

With the use of classes and objects, object-oriented programming makes code maintenance simple. Because inheritance allows for code reuse, the program is simpler because you do not have to write the same code repeatedly. Data hiding is also provided by ideas like encapsulation and abstraction.

--> Why is C++ a partial oop?

The object-oriented features of the C language were the primary motivation behind the construction of the C++ language.

The C++ programming language is categorized as a partial object-oriented programming language even though it supports OOP concepts, including classes, objects, inheritance, encapsulation, abstraction, and polymorphism.

1) The main function must always be outside the class in C++ and is required. This means that we may do without classes and objects and have a single main function in the application.

It is expressed as an object in this case, which is the first time Pure OOP has been violated.

2) Global variables are a feature of the C++ programming language that can be accessed by any other object within the program and are defined outside of it. Encapsulation is broken here. Even though C++ encourages encapsulation for classes and objects, it ignores it for global variables.

--> Overloading:

Polymorphism also has a subset known as overloading. An existing operator or function is said to be overloaded when it is forced to operate on a new data type.

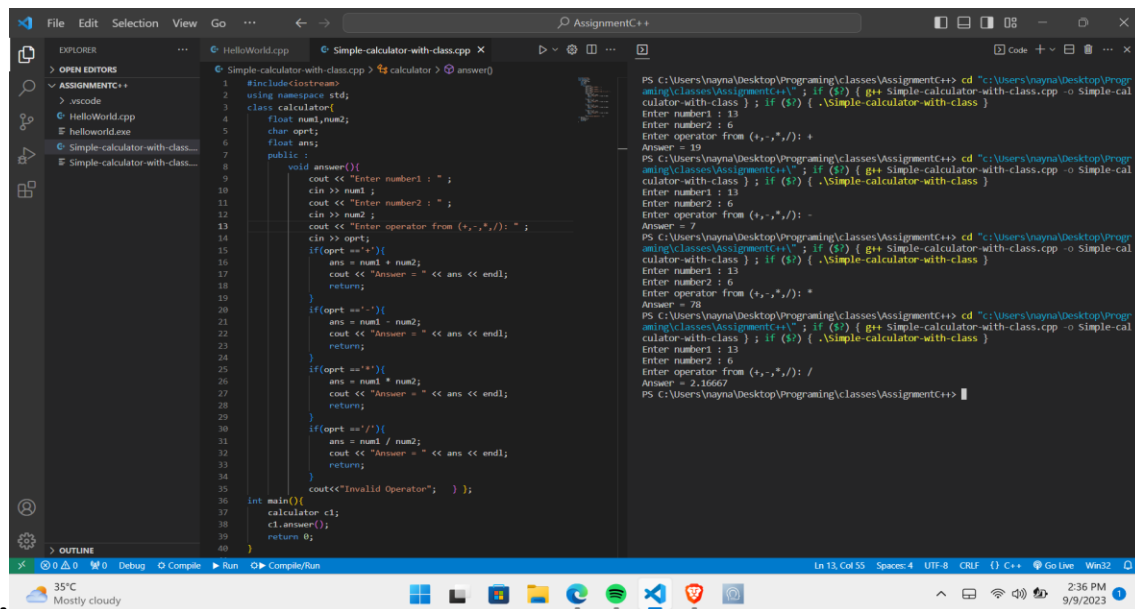
3> What is the difference between OOP and POP?

Ans.

OOP	POP
1. OOP takes a bottom-up approach in designing a program.	1. POP follows a top-down approach.
2. Program is divided into objects depending on the problems.	2. Program is divided into small chunks based on the functions.
3. Each object controls it's on data.	3. Each function contains different data.
4. Data hiding is possible in OOP.	4. No easy way for data hiding.
5. Inheritance is allowed in OOP.	5. No such concept of inheritance in POP.
6. Operator Overloading is allowed.	6. Operator Overloading is not allowed.
7. Focuses on security of the data irrespective of the algorithm.	7. Follows a systematic approach to solve the problem.
8. The main priority is data rather than functions in a program.	8. Functions are more important than data in a program.
9. The functions of the objects are linked via message passing.	9. Different parts of a program are interconnected via parameter passing.
10. Example of OOP is C++, Java, etc.,	10. Example of POP is Pascal, Fortran, etc.,

Module 4.2: Programming with C++

1> Write a program to create simple calculator using class.



```
1 #include<iostream>
2 using namespace std;
3 class calculator{
4     float num1,num2;
5     char opt;
6     float ans;
7 public:
8     void answer(){
9         cout << "Enter number1 : ";
10        cin >> num1 ;
11        cout << "Enter number2 : ";
12        cin >> num2 ;
13        cout << "Enter operator from (+,-,*,/): ";
14        cin >> opt;
15        if(opt == "+"){
16            ans = num1 + num2;
17            cout << "Answer = " << ans << endl;
18            return;
19        }
20        if(opt == "-"){
21            ans = num1 - num2;
22            cout << "Answer = " << ans << endl;
23            return;
24        }
25        if(opt == "*"){
26            ans = num1 * num2;
27            cout << "Answer = " << ans << endl;
28            return;
29        }
30        if(opt == "/"){
31            ans = num1 / num2;
32            cout << "Answer = " << ans << endl;
33            return;
34        }
35        cout<<"Invalid Operator";
36    };
37 int main(){
38     calculator c1;
39     c1.answer();
40     return 0;
41 }
```

The screenshot shows a C++ IDE with a file named 'Simple-calculator-with-class.cpp'. The code defines a 'calculator' class with private data members 'num1', 'num2', 'opt', and 'ans'. It has a public member function 'answer()' that prompts the user for two numbers and an operator, then performs the corresponding arithmetic operation and displays the result. The 'main' function creates an instance of the 'calculator' class and calls the 'answer()' method. The output window shows the program's execution, including prompts for input and the resulting calculations for addition, subtraction, multiplication, and division.

Ans.

2> Define a class to represent a bank account. Include the following members:

1. Data Member:

- Name of the depositor
- Account Number
- Type of Account
- Balance amount in the account

2. Member Functions:

- To assign values
- To deposit an amount
- To withdraw an amount after checking balance
- To display name and balance

```

#include<iostream>
using namespace std;

class Bank
{
    string name, account_type;
    int account_number;
    double balance, deposit_amount;

public:
    void assign_values()
    {
        cout << "Enter name of the depositor : " << endl;
        cin >> name;
        cout << "Enter account_number of the depositor : " << endl;
        cin >> account_number;
        cout << "Enter account_type of the depositor : " << endl;
        cin >> account_type;
        cout << "Enter balance of the depositor : " << endl;
        cin >> balance;
    }

    double deposit_amount()
    {
        cout << "Enter deposit amount : " << endl;
        cin >> deposit_amount;
        balance = balance + deposit_amount;
        return balance;
    }

    double withdraw_amount()
    {
        cout << "Enter withdraw amount : " << endl;
        cin >> withdraw_amount;
        balance = balance - withdraw_amount;
        return balance;
    }

    void display()
    {
        cout << "Enter name of the depositor : " << endl;
        cout << "Enter balance of the depositor : " << endl;
    }

};

int main()
{
    Bank b1;
    b1.assign_values();
    b1.deposit_amount();
    b1.withdraw_amount();
    b1.display();
    return 0;
}

```

```

PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++> cd "C:\Users\nayna\Desktop\Programming\classes\AssignmentC++\"; if ($?) { g++ Bank-account-using-class.cpp -o Bank-account-using-class }; if ($?) { .\Bank-account-using-class }
Enter name of the depositor :
Priyanka
Enter account_number of the depositor :
468978987
Enter account_type of the depositor :
Saving
Enter balance of the depositor :
30000
Enter deposit amount :
1000
Enter withdraw amount :
800
Enter name of the depositor : Priyanka
Enter balance of the depositor : 30200
PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++>

```

Ans.

3> Write a program to find the multiplication values and the cubic values using inline function.

```

#include<iostream>
using namespace std;

class Line
{
public:
    inline double mul(double p, double u)
    {
        return p*u;
    }
    inline double cube(double j)
    {
        return j*j*j;
    }
};

int main()
{
    Line n;
    double n1,n2;

    cout << endl << "Enter number1 : " << endl;
    cin >> n1;
    cout << endl << "Enter number2 : " << endl;
    cin >> n2;
    cout << endl << "Multiplication value is : " << n.mul(n1,n2) << endl;
    cout << endl << "Cube value is : " << n.cube(n2) << endl << endl;

    return 0;
}

```

```

PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++> cd "C:\Users\nayna\Desktop\Programming\classes\AssignmentC++\"; if ($?) { g++ Inline-function-with-cubic-values.cpp -o Inline-function-with-cubic-values }; if ($?) { .\Inline-function-with-cubic-values }
Enter number1 :
13
Enter number2 :
6
Multiplication value is : 78
Cube value is : 216
PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++>

```

Ans.

4> Write a program of Addition, Subtraction, Division, Multiplication using constructor.

```
File Edit Selection View Go Run ... J\ AssignmentC++ ...
Calculator with constructor.cpp
1 #include<iostream>
2 using namespace std;
3 class calculator
4 {
5     char optr;
6     float ans;
7 public:
8     calculator(float num1,float num2){
9         cout << "Enter operator from (+,-,*,/): " ;
10        cin >> optr;
11        if(optr == "+"){
12            ans = num1 + num2;
13            cout << "Answer = " << ans << endl;
14            return;
15        }
16        if(optr == "-"){
17            ans = num1 - num2;
18            cout << "Answer = " << ans << endl;
19            return;
20        }
21        if(optr == "*"){
22            ans = num1 * num2;
23            cout << "Answer = " << ans << endl;
24            return;
25        }
26        if(optr == "/"){
27            ans = num1 / num2;
28            cout << "Answer = " << ans << endl;
29            return;
30        }
31        cout<<"Invalid Operator";
32    }
33
34 int main(){
35     calculator c1(13,6);
36     return 0;
37 }
38
39 PS C:\Users\Ananya\Desktop\Programming\classes\Assignme
ntC++> cd "c:\Users\Ananya\Desktop\Programming\classes\
Assignme"
ntC++> g++ Calculator-with-construct
or.cpp -o Calculator-with-construct
or ; if ($?) {
    Enter operator from (+,-,*/): +
    Answer = 19
    PS C:\Users\Ananya\Desktop\Programming\classes\Assignme
ntC++> cd "c:\Users\Ananya\Desktop\Programming\classes\
Assignme"
ntC++> g++ Calculator-with-construct
or.cpp -o Calculator-with-construct
or ; if ($?) {
    Enter operator from (+,-,*/): -
    Answer = 7
    PS C:\Users\Ananya\Desktop\Programming\classes\Assignme
ntC++> cd "c:\Users\Ananya\Desktop\Programming\classes\
Assignme"
ntC++> g++ Calculator-with-construct
or.cpp -o Calculator-with-construct
or ; if ($?) {
    Enter operator from (+,-,*/): *
    Answer = 78
    PS C:\Users\Ananya\Desktop\Programming\classes\Assignme
ntC++> cd "c:\Users\Ananya\Desktop\Programming\classes\
Assignme"
ntC++> g++ Calculator-with-construct
or.cpp -o Calculator-with-construct
or ; if ($?) {
    Enter operator from (+,-,*/): /
    Answer = 2.16667
    PS C:\Users\Ananya\Desktop\Programming\classes\Assignme
ntC++>
Ln 4, Col 1  Spaces: 4  UTF-8  CRLF  C++  Go Live  Win32
35°C
3/20/2024
```

Ans.

5> Assume a class cricketer is declared. Declare a derived class batsman from cricketer. Data member of batsman. Total runs, Average runs and best performance. Member functions input data, calculate average runs, Display data (Single Inheritance).

```

1 // Single-level-inheritance.cpp
2 #include <iostream>
3 using namespace std;
4
5 class cricketer
6 {
7 public:
8     int runs;
9     float averageRuns;
10    string bestPerformance;
11
12    cricketer()
13    {
14        cout << endl << "This is best cricket match." << endl;
15        cout << endl << "The runs are : " << endl;
16        cin >> runs;
17        cout << endl << "The bestPerformance of cricket player "
18            << endl << endl;
19        cin >> bestPerformance;
20    }
21
22    class batsman : public cricketer
23    {
24 public:
25        batsman()
26        {
27            averageRuns = runs / 33;
28            cout << endl << "Runs are : " << runs << endl << endl;
29            cout << endl << "The averageRuns are : " << averageRuns << endl << endl;
30            cout << endl << "The bestPerformance of cricket player is : " <<
31                bestPerformance << endl << endl;
32        }
33    }
34
35    int main()
36    {
37        batsman obj;
38        return 0;
39    }

```

Ans.

6> Create a class person having members name and age. Derive a class student having member percentage. Derive another class teacher having member salary. Write necessary member function to initialize, read and write data. Write also Main function (Multiple Inheritance).

```

class Person
{
protected:
    int age;
    string name;
public:
    void get_person(const string &s)
    {
        name = s;
    }
};

class Student : public Person
{
protected:
    float percentage;
public:
    void get_student(const float p)
    {
        percentage = p;
    }
};

class Teacher : public Person, public Student
{
protected:
    float salary;
public:
    void get_teacher(const float s)
    {
        salary = s;
    }
};

int main()
{
    Teacher t1;
    t1.get_person("Priyanka");
    t1.get_student(88.9);
    t1.get_teacher(250000);
    return 0;
}
    
```

```

PS C:\Users\nayna\Desktop\Programing\classes\AssignmentC++> cd "c:\Users\nayna\Desktop\Programing\classes\AssignmentC++"; if ($?) { g++ Multiple-inheritance.cpp -o Multiple-inheritance }; if ($?) { .\Multiple-inheritance }

Age is : 26
Name is : Priyanka
Percentage is :88.9
Salary of the teacher is : 250000
PS C:\Users\nayna\Desktop\Programing\classes\AssignmentC++>
    
```

Ans.

7> Assume that the test results of a batch of students are stored in three different classes. Class Students are storing the roll number. Class Test stores the marks obtained in two subjects and class result contains the total marks obtained in the test. The class result can inherit the details of the marks obtained in the test and roll number of students (Multilevel Inheritance).

```

class Students
{
public:
    int rollNumber;
    Students()
    {
        cout << "Roll number : " << endl;
    }
};

class Test : public Students
{
public:
    float marks1, marks2;
    Test()
    {
        cout << "Enter marks of first subject : " << endl;
        cin >> marks1;
        cout << "Enter marks of second subject : " << endl;
        cin >> marks2;
    }
};

class Result : public Test
{
public:
    float totalMarks;
    Result()
    {
        totalMarks = marks1 + marks2;
        cout << "Roll number is : " << rollNumber << endl;
        cout << "TotalMarks is : " << totalMarks << endl;
    }
};

int main()
{
    Result obj;
    return 0;
}
    
```

```

PS C:\Users\nayna\Desktop\Programing\classes\AssignmentC++> cd "c:\Users\nayna\Desktop\Programing\classes\AssignmentC++"; if ($?) { g++ Multi-level-inheritance.cpp -o Multi-level-inheritance }; if ($?) { .\Multi-level-inheritance }

Enter marks of first subject :
89
Enter marks of second subject :
98
Roll number is : 1
TotalMarks is : 187
PS C:\Users\nayna\Desktop\Programing\classes\AssignmentC++>
    
```

Ans.

8> Write a program to Mathematic operation like Addition, Subtraction, Multiplication, Division Of two number using different parameters and Function Overloading.

```

Calculator-with-function-overloading.cpp
#include<iostream>
using namespace std;

class P
{
public:
    void add(int p,int u)
    {
        cout << endl << "This is add function with two argument " << endl;
        cout << "Addition of p and u is : " << p + u << endl << endl;
    }
    void sub(int p,double u)
    {
        cout << "This is get_n function with two arguments"<< endl;
        cout << "Subtraction of p and u is : " << p - u << endl << endl;
    }
    void mul(double p,int u)
    {
        cout << "This is get_n function with two argument " << endl;
        cout << "Multiplication of p and u is : " << p * u << endl << endl;
    }
    void div(double p,double u)
    {
        cout << "This is get_n function with two argument " << endl;
        cout << "Division of p and u is : " << p / u << endl << endl;
    }
};

int main()
{
    P p1;
    p1.add(13,6);
    p1.sub(13,6.05);
    p1.mul(13.05,4);
    p1.div(13.05,6.05);
    return 0;
}
    
```

```

AssignmentC++
PS C:\Users\nayna\Desktop\Programming\Classes\AssignmentC++> cd "C:\Users\nayna\Desktop\Programming\Classes\AssignmentC++" & if ($?) { g++ Calculator-with-function-overloading.cpp -o Calculator-with-function-overloading } ; if ($?) { .\Calculator-with-function-overloading }
This is add function with two argument
Addition of p and u is : 19
This is get_n function with two argument
Subtraction of p and u is : 6.99
This is get_n function with two argument
Multiplication of p and u is : 78.06
This is get_n function with two argument
Division of p and u is : 2.16473
    
```

Ans.

9> Write a program to calculate the area of circle, rectangle and triangle using Function Overloading.

- Rectangle: Area * breadth
- Triangle: $\frac{1}{2}$ *Area* breadth
- Circle: Pi * Area *Area

```

Area-with-function-overloading.cpp
#include<iostream>
using namespace std;

class P
{
public:
    void Rectangle(int l,int w)
    {
        cout << endl << "This is rectangle function with two argument " << endl;
        cout << "Area of rectangle is : " << l * w << endl << endl;
    }
    void Triangle(int l,double w)
    {
        cout << "This is triangle function with two argument"<< endl;
        cout << "Area of triangle is : " << 0.5*l*w << endl << endl;
    }
    void Circle(int r)
    {
        cout << "This is circle function with one argument " << endl;
        cout << "Area of circle is : " << 3.14*r*r << endl << endl;
    }
};

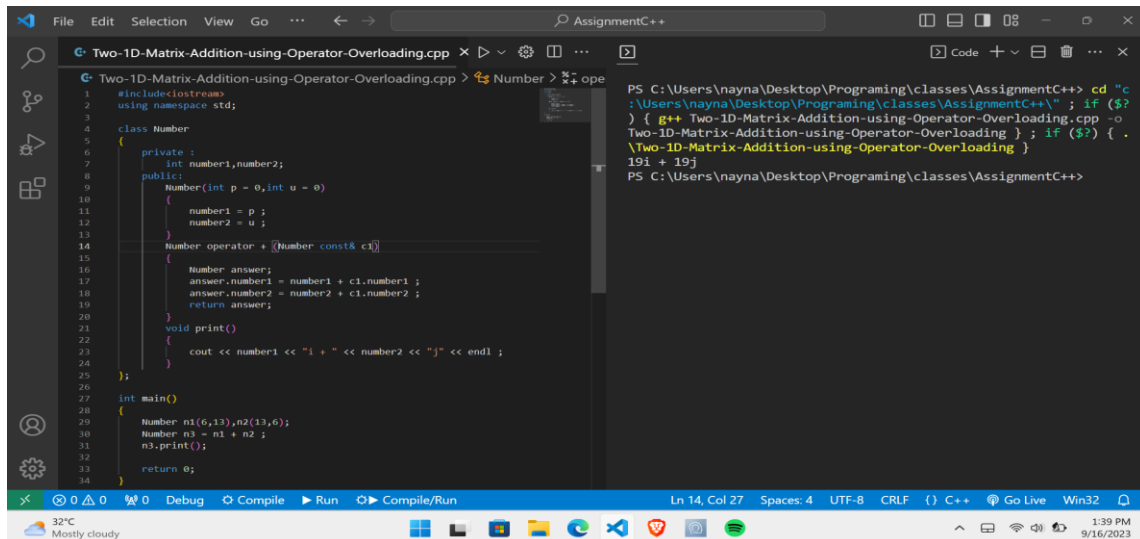
int main()
{
    P p1;
    p1.Rectangle(13,6);
    p1.Triangle(13,6.05);
    p1.Circle(6);
    return 0;
}
    
```

```

AssignmentC++
PS C:\Users\nayna\Desktop\Programming\Classes\AssignmentC++> cd "C:\Users\nayna\Desktop\Programming\Classes\AssignmentC++" & if ($?) { g++ Area-with-function-overloading.cpp -o Area-with-function-overloading } ; if ($?) { .\Area-with-function-overloading }
This is rectangle function with two argument
Area of rectangle is : 78
This is triangle function with two argument
Area of triangle is : 39.065
This is circle function with one argument
Area of circle is : 113.04
    
```

Ans.

10> Write a Program of Two 1D Matrix Addition using Operator Overloading.

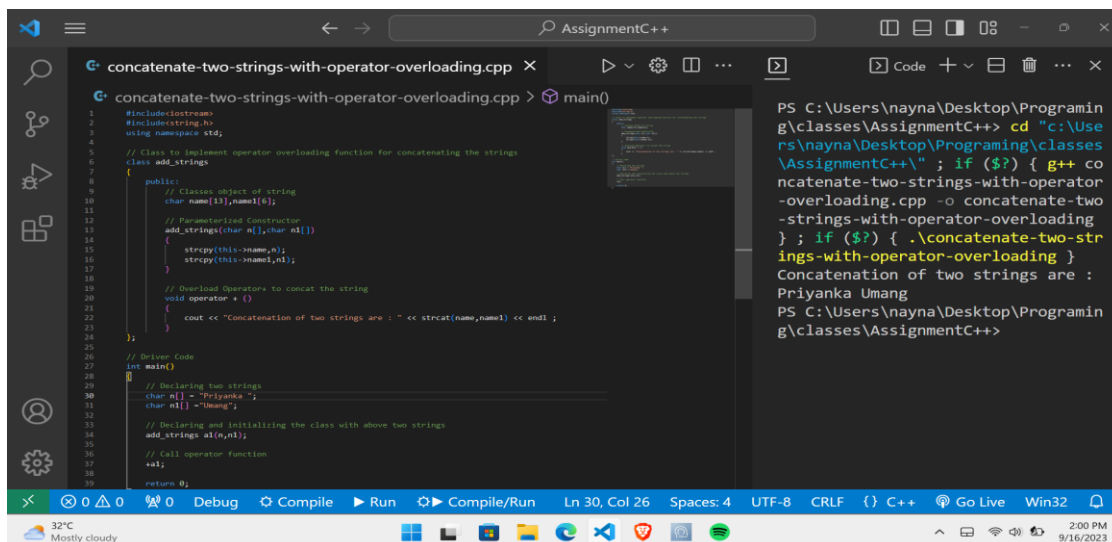


The screenshot shows a Visual Studio Code editor with a C++ file named 'Two-1D-Matrix-Addition-using-Operator-Overloading.cpp'. The code defines a 'Number' class with two private integers, 'number1' and 'number2'. It includes a constructor, a parameterized constructor, and an overloaded '+' operator that returns a new 'Number' object with the sum of the two numbers. A 'print()' method is also defined. The 'main()' function creates two 'Number' objects, 'n1' and 'n2', adds them, and prints the result.

```
1 #include<iostream>
2 using namespace std;
3
4 class Number
5 {
6 private :
7     int number1,number2;
8 public:
9     Number(int p = 0,int u = 0)
10    {
11        number1 = p ;
12        number2 = u ;
13    }
14    Number operator + (Number const& c1)
15    {
16        Number answer;
17        answer.number1 = number1 + c1.number1 ;
18        answer.number2 = number2 + c1.number2 ;
19        return answer;
20    }
21    void print()
22    {
23        cout << number1 << "i + " << number2 << "j" << endl ;
24    }
25 };
26
27 int main()
28 {
29     Number n1(6,13),n2(13,6);
30     Number n3 = n1 + n2 ;
31     n3.print();
32 }
33 return 0;
```

Ans.

11> Write a program to concatenate the two strings using Operator Overloading.

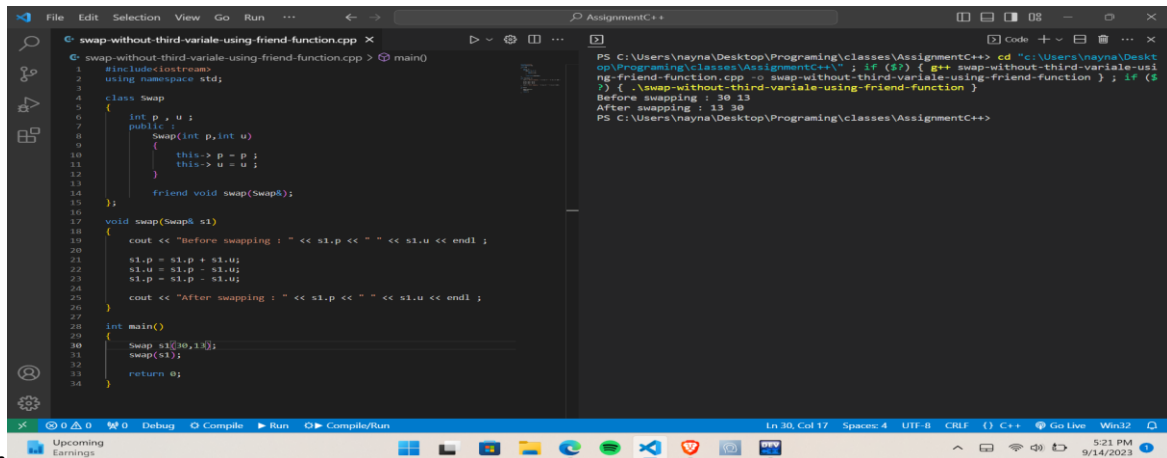


The screenshot shows a Visual Studio Code editor with a C++ file named 'concatenate-two-strings-with-operator-overloading.cpp'. The code defines a 'string' class with a character array 'name'. It includes a constructor, a parameterized constructor, and an overloaded '+' operator that concatenates two strings. A 'print()' method is also defined. The 'main()' function creates two 'string' objects, 's1' and 's2', concatenates them, and prints the result.

```
1 #include<iostream>
2 #include<string.h>
3 using namespace std;
4
5 // Class to implement operator overloading function for concatenating the strings
6 class add_strings
7 {
8 public:
9     // Classes object of string
10    char name[10],name1[6];
11
12    // Parameterized Constructor
13    add_strings(char n[],char n1[])
14    {
15        strcpy(this->name,n);
16        strcpy(this->name1,n1);
17    }
18
19    // Overload Operators to concat the string
20    void operator + ()
21    {
22        cout << "Concatenation of two strings are : " << strcat(name,name1) << endl ;
23    }
24 };
25
26 // Driver Code
27 int main()
28 {
29     // Declaring two strings
30     char n1[] = "Priyanka ";
31     char n2[] = "Umang";
32
33     // Declaring and Initializing the class with above two strings
34     add_strings a1(n1,n2);
35
36     // Call operator function
37     a1 + a1;
38 }
39 return 0;
```

Ans.

12> Write a program to swap the two numbers using friend function without using third variable.

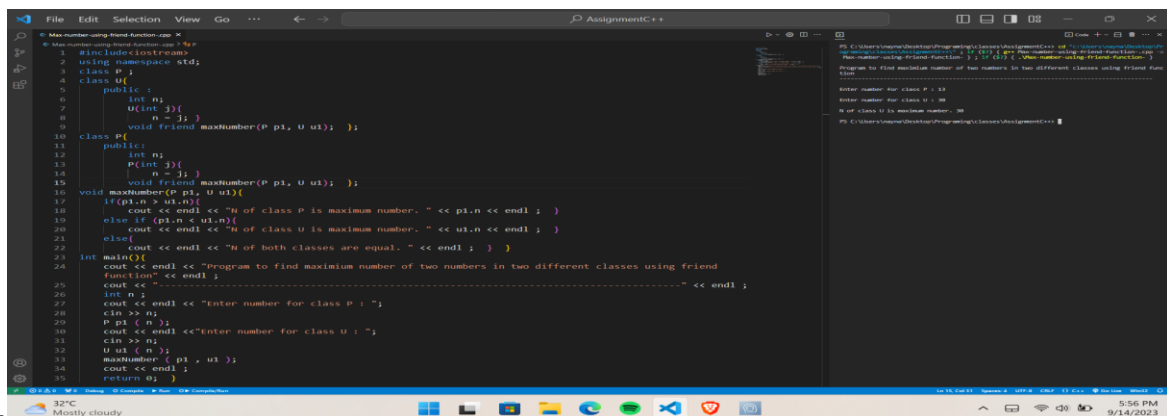


```
1 #include <iostream>
2 using namespace std;
3
4 class Swap
5 {
6     int p, u;
7     public:
8         Swap(int p, int u)
9         {
10             this->p = p;
11             this->u = u;
12         }
13
14         friend void swap(Swap&);
15 };
16
17 void swap(Swap& s1)
18 {
19     cout << "Before swapping : " << s1.p << " " << s1.u << endl;
20
21     s1.p = s1.p + s1.u;
22     s1.u = s1.p - s1.u;
23     s1.p = s1.p - s1.u;
24
25     cout << "After swapping : " << s1.p << " " << s1.u << endl;
26 }
27
28 int main()
29 {
30     Swap s1(10, 15);
31     swap(s1);
32
33     return 0;
34 }
```

PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++> cd "C:\Users\nayna\Desktop\Programming\classes\AssignmentC++"; if (\$?) { g++ swap-without-third-variable-using-friend-function.cpp -o swap-without-third-variable-using-friend-function }; if (\$?) { .\swap-without-third-variable-using-friend-function }; Before swapping : 10 15 After swapping : 15 10 PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++>

Ans.

13> Write a program to find the max number from given two numbers using friend function.



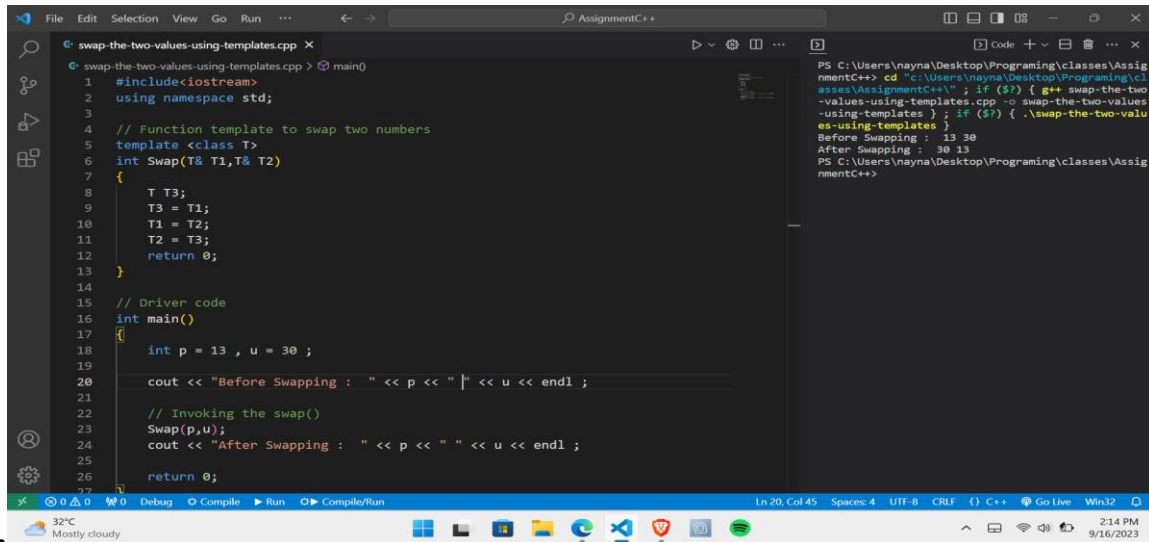
```
1 #include <iostream>
2 using namespace std;
3
4 class P
5 {
6     public:
7         P(int n)
8         {
9             n = n;
10         }
11         void friend maxNumber(P p1, U u1);
12 };
13
14 class U
15 {
16     public:
17         U(int n)
18         {
19             n = n;
20         }
21         void friend maxNumber(P p1, U u1);
22 };
23
24 void maxNumber(P p1, U u1)
25 {
26     if(p1.n > u1.n)
27         cout << endl << "n of class P is maximum number. " << p1.n << endl;
28     else if (p1.n < u1.n)
29         cout << endl << "n of class U is maximum number. " << u1.n << endl;
30     else
31         cout << endl << "n of both classes are equal. " << endl;
32 }
33
34 int main()
35 {
36     cout << endl << "Program to find maximum number of two numbers in two different classes using friend function" << endl;
37
38     int n;
39     cout << endl << "Enter number for class P : ";
40     cin >> n;
41     P p1(n);
42     cout << endl << "Enter number for class U : ";
43     cin >> n;
44     U u1(n);
45     maxNumber(p1, u1);
46     cout << endl;
47     return 0;
48 }
```

PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++> cd "C:\Users\nayna\Desktop\Programming\classes\AssignmentC++"; if (\$?) { g++ max-number-using-friend-function.cpp -o max-number-using-friend-function }; if (\$?) { .\max-number-using-friend-function }; Enter number for class P : 15 Enter number for class U : 10 n of class P is maximum number. 15 PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++>

Ans.

Module 4.3: C, C++ Templates

1> Write a program to swap the two values using templates.

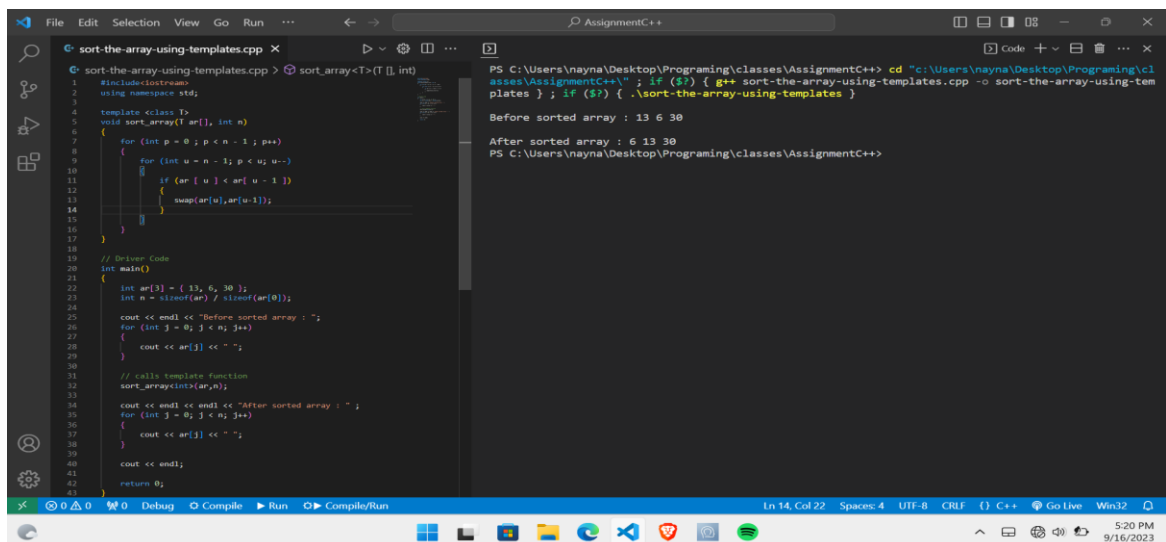


```
1 #include<iostream>
2 using namespace std;
3
4 // Function template to swap two numbers
5 template <class T>
6 int Swap(T& T1,T& T2)
7 {
8     T T3;
9     T3 = T1;
10    T1 = T2;
11    T2 = T3;
12    return 0;
13 }
14
15 // Driver code
16 int main()
17 {
18     int p = 13 , u = 30 ;
19
20     cout << "Before Swapping : " << p << " " << u << endl ;
21
22     // Invoking the swap()
23     Swap(p,u);
24     cout << "After Swapping : " << p << " " << u << endl ;
25
26     return 0;
27 }
```

PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++> cd "c:\Users\nayna\Desktop\Programming\classes\AssignmentC++\"; if (\$?) { g++ swap-the-two-values-using-templates.cpp -o swap-the-two-values-using-templates }; if (\$?) { .\swap-the-two-values-using-templates }
Before Swapping : 13 30
After Swapping : 30 13
PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++>

Ans.

2> Write a program to sort the array using templates.



```
1 #include<iostream>
2 using namespace std;
3
4 template <class T>
5 void sort_array(T ar[], int n)
6 {
7     for (int p = 0 ; p < n - 1 ; p++)
8     {
9         for (int u = n - 1 ; p < u ; u--)
10         {
11             if (ar[p] < ar[u - 1])
12             {
13                 swap(ar[u],ar[u-1]);
14             }
15         }
16     }
17 }
18
19 // Driver Code
20 int main()
21 {
22     int ar[3] = { 13, 6, 30 };
23     int n = sizeof(ar) / sizeof(ar[0]);
24
25     cout << endl << "Before sorted array : ";
26     for (int j = 0 ; j < n ; j++)
27     {
28         cout << ar[j] << " ";
29     }
30
31     // calls template function
32     sort_array<int>(ar,n);
33
34     cout << endl << endl << "After sorted array : ";
35     for (int j = 0 ; j < n ; j++)
36     {
37         cout << ar[j] << " ";
38     }
39
40     cout << endl;
41
42     return 0;
43 }
```

PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++> cd "c:\Users\nayna\Desktop\Programming\classes\AssignmentC++\"; if (\$?) { g++ sort-the-array-using-templates.cpp -o sort-the-array-using-templates }; if (\$?) { .\sort-the-array-using-templates }
Before sorted array : 13 6 30
After sorted array : 6 13 30
PS C:\Users\nayna\Desktop\Programming\classes\AssignmentC++>

Ans.

