

ASSIGNMENT OPERATOR OVERLOADING

1. Create a class FLOAT that contains one float data member .Overload all the four arithmetic operators so that they operate on the objects of FLOAT.

```
#include <iostream>
#include <iomanip>
using namespace std;

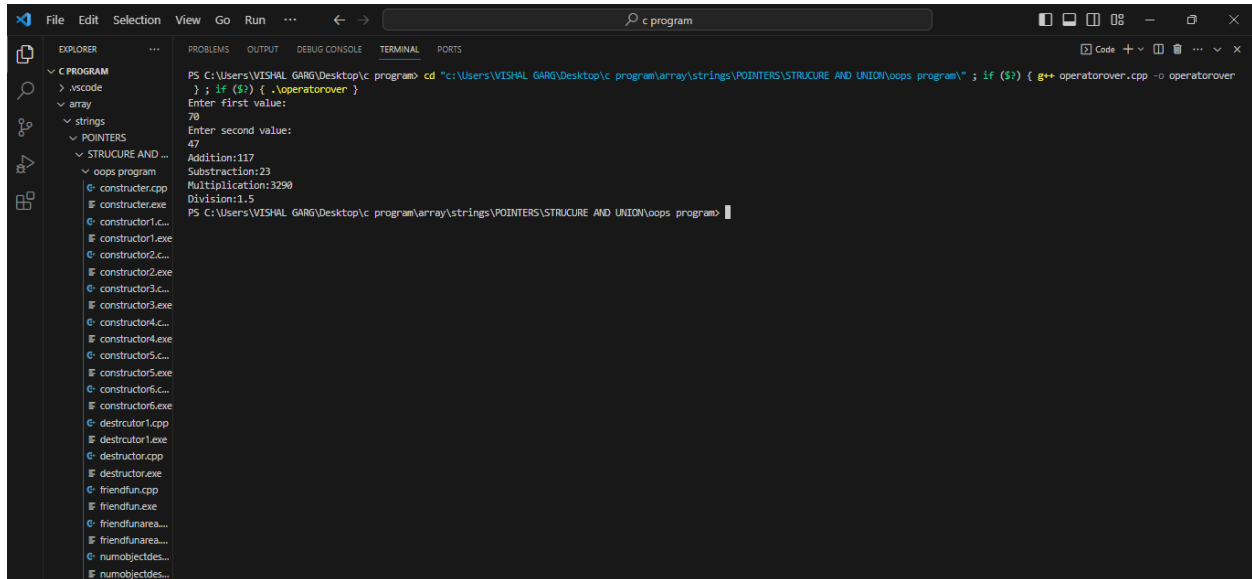
class Float
{
public:
    float f;
    void operator+(Float ob)
    {
        float result = f + ob.f;
        cout << "Addition:" << result << endl;
    }
    void operator-(Float ob)
    {
        float result = f - ob.f;
        cout << "Subtraction:" << result << endl;
    }
    void operator*(Float ob)
    {
        float result = f * ob.f;
        cout << "Multiplication:" << result << endl;
    }
    void operator/(Float ob)
    {
        float result = f / ob.f;
        cout << "Division:" << setprecision(2) << result << endl;
    }
};

int main()
{
    Float obj1, obj2;
    cout << "Enter first value:" << endl;
    cin >> obj1.f;
    cout << "Enter second value:" << endl;
    cin >> obj2.f;
    obj1 + obj2;
```

```

obj1 - obj2;
obj1 *obj2;
obj1 / obj2;
return 0;
}

```



2. Define a class string. Overload ==operator to compare 2 strings.

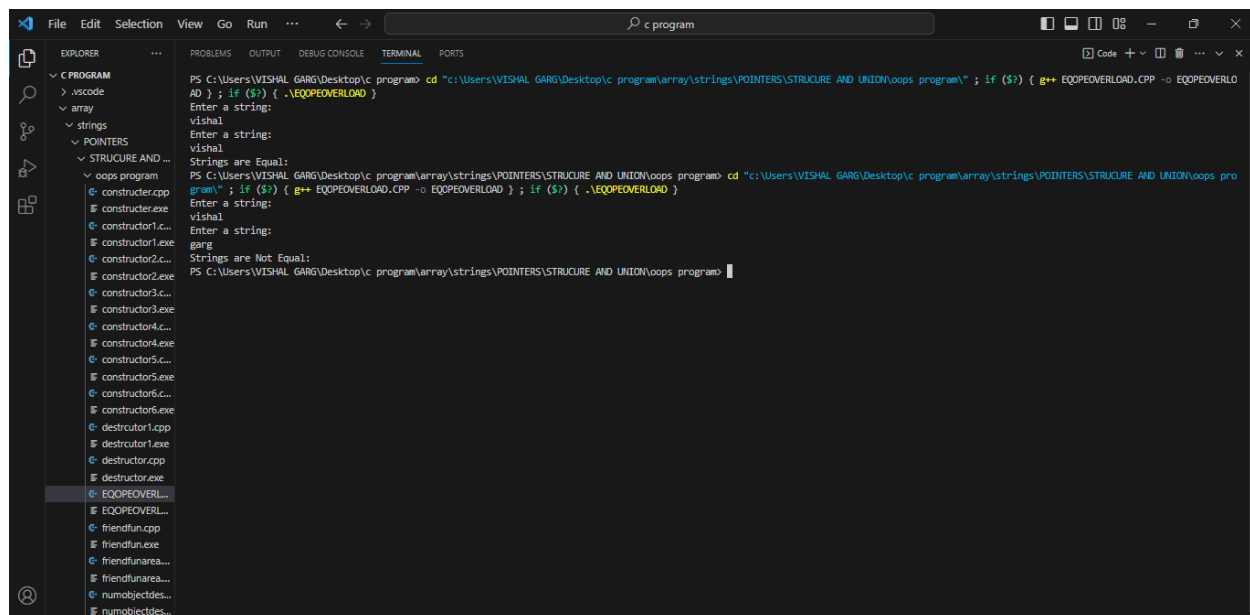
```

#include <iostream>
#include <string.h>
using namespace std;
class String
{
    char str[20];

public:
    void display()
    {
        cout << "Enter a string:" << endl;
        cin >> str;
    }
    int operator==(String x)
    {
        if (strcmp(str, x.str) == 0)
            return 1;
        else
            return 0;
    }
}

```

```
};
int main()
{
    String s1, s2;
    s1.display();
    s2.display();
    if (s1 == s2)
        cout << "Strings are Equal:" << endl;
    else
        cout << "Strings are Not Equal:" << endl;
    return 0;
}
```



3. Create a Complex class that has real(int) and img(int) as member data, and has getData and showData functions. Then also overload the following operators for Complex class. =, ==, +, ++, --,

```
#include <iostream>
using namespace std;
class Complex
{
private:
    int real;
    int imag;
```

```

public:
    Complex(int r = 0, int i = 0) : real(r), imag(i) {}
    void getData()
    {
        cout << "Real = " << real << ", Imaginary = " << imag << endl;
    }
    void showData() const
    {
        cout << "Complex Number = " << real << " + " << imag << "i" << endl;
    }
    Complex &operator=(const Complex &other)
    {
        real = other.real;
        imag = other.imag;
        return *this;
    }
    bool operator==(const Complex &other) const
    {
        return (real == other.real && imag == other.imag);
    }
    Complex operator+(const Complex &other) const
    {
        Complex result;
        result.real = real + other.real;
        result.imag = imag + other.imag;
        return result;
    }
    Complex &operator++()
    {
        ++real;
        ++imag;
        return *this;
    }
    Complex &operator--()
    {
        --real;
        --imag;
        return *this;
    }
};

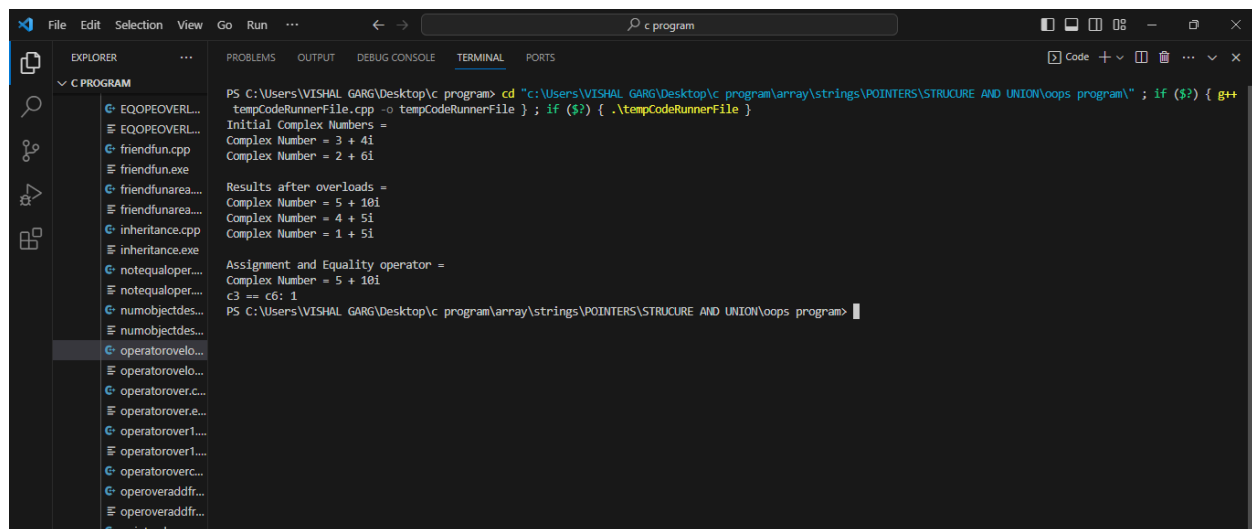
int main()
{
    Complex c1(3, 4);
    Complex c2(2, 6);

```

```

    cout << "Initial Complex Numbers = " << endl;
    c1.showData();
    c2.showData();
    Complex c3 = c1 + c2;
    Complex c4 = ++c1;
    Complex c5 = --c2;
    cout << "\nResults after overloads = " << endl;
    c3.showData();
    c4.showData();
    c5.showData();
    Complex c6 = c3;
    cout << "\nAssignment and Equality operator = " << endl;
    c6.showData();
    cout << "c3 == c6: " << (c3 == c6) << endl;
    return 0;
}

```



The screenshot shows a C++ IDE with the following components:

- EXPLORER:** A list of files in the project, including `EQOPEOVERL...`, `EQOPEOVERL...`, `friendfun.cpp`, `friendfun.exe`, `friendfunarea...`, `friendfunarea...`, `inheritance.cpp`, `inheritance.exe`, `notequaloper...`, `notequaloper...`, `numobjectdes...`, `numobjectdes...`, `operatorovelo...`, `operatorovelo...`, `operatorover.c...`, `operatorover.e...`, `operatorover1...`, `operatorover1...`, `operatoroverc...`, `operoveraddfr...`, `operoveraddfr...`, and `pointerclass.cpp`.
- TERMINAL:** The output of the program, showing the initial complex numbers, results after overloads, and the assignment and equality operator results.

The terminal output is as follows:

```

PS C:\Users\VISHAL GARG\Desktop\c program> cd "c:\Users\VISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program\" ; if ($?) { g++
tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Initial Complex Numbers =
Complex Number = 3 + 4i
Complex Number = 2 + 6i

Results after overloads =
Complex Number = 5 + 10i
Complex Number = 4 + 5i
Complex Number = 1 + 5i

Assignment and Equality operator =
Complex Number = 5 + 10i
c3 == c6: 1
PS C:\Users\VISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program>

```

4. Write a C++ program to overload '!' operator using friend function

```

#include <iostream>
using namespace std;
class Complex
{
public:
    float real, imag;
    friend bool operator!=(const Complex &lhs, const Complex &rhs);
};
bool operator!=(const Complex &lhs, const Complex &rhs)
{

```

```

    if (lhs.real != rhs.real || lhs.imag != rhs.imag)
    {
        cout << "Numbers are unequal:" << endl;
        return true;
    }

    else
        cout << "Numbers are Equal:" << endl;
    return false;
}

int main()
{
    Complex c1,
        c2;
    cout << "Enter two complex number:" << endl;
    cin >> c1.real >> c1.imag >> c2.real >> c2.imag;
    c1 != c2;
    return 0;
}

```

The screenshot shows the Visual Studio IDE with the 'notequaloper.cpp' file open. The terminal window displays the following output:

```

PS C:\Users\VISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program> cd "c:\Users\VISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program\" ; if ($?) { g++ notequaloper.cpp -o notequaloper } ; if ($?) { .\notequaloper }
Enter two complex number First Enter Real part then Imaginary part:
10
20
10
20
Numbers are Equal:
PS C:\Users\VISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program> cd "c:\Users\VISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program\" ; if ($?) { g++ notequaloper.cpp -o notequaloper } ; if ($?) { .\notequaloper }
Enter two complex number First Enter Real part then Imaginary part:
10
20
20
20
Numbers are unequal:
PS C:\Users\VISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program>

```

5. Read a value of distance from one object and add with a value in another object using friend function.

```

#include <iostream>
using namespace std;
class Ftinch;
class Mtrcm
{
    int mtr, cm;
}

```

```

public:
    Mtrcm()
    {
        mtr = 0;
        cm = 0;
    }
    Mtrcm(int x, int y)
    {
        mtr = x;
        cm = y;
    }
    void display()
    {
        cout << mtr << "m" << cm << "cm";
    }
    friend Mtrcm add(Mtrcm, Ftinch);
};

class Ftinch
{
    int ft, in;

public:
    Ftinch(int x, int y)
    {
        ft = x;
        in = y;
    }
    void display()
    {
        cout << ft << "ft" << in << "in";
    }
    friend Mtrcm add(Mtrcm, Ftinch);
};

Mtrcm add(Mtrcm d1, Ftinch d2)
{
    int t_cm = (d1.mtr * 100 + d1.cm) + (d2.ft * 12 + d2.in) * 2.54;
    int mtr, cm;
    mtr = t_cm / 100;
    cm = t_cm % 100;
    Mtrcm d3(mtr, cm);
    return d3;
}

int main()
{

```

```

Mtrcm obj1(4, 15);
Ftinch obj2(5, 10);
Mtrcm obj3 = add(obj1, obj2);
cout << "T_DISTANCE-1:\n";
obj1.display();
cout << "\nT_DISTANCE-2:\n";
obj2.display();
cout << "\nT_Total\n";
obj3.display();
}

```

The screenshot shows the Visual Studio Code interface with a C++ program open. The Explorer pane on the left shows a project named 'C PROGRAM' with various files. The main editor displays the following C++ code:

```

PS C:\Users\VIISHAL GARG\Desktop\c program> cd "c:\Users\VIISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program\" ; if ($?) { g++ operoveraddfrnd.cpp -o operovera
ddfrnd } ; if ($?) { .\operoveraddfrnd }
T_DISTANCE-1:
4m15cm
T_DISTANCE-2:
5ft10in
T_Total
5m02cm
PS C:\Users\VIISHAL GARG\Desktop\c program\array\strings\POINTERS\STRUCTURE AND UNION\oops program>

```

The output of the program is displayed in the terminal, showing the results of the calculations for T_DISTANCE-1, T_DISTANCE-2, and T_Total.