INDEX

Lab-8

Program to demonstrate the concept of destructor.

Code

```cpp
// Program to demonstrate the concept of destructor

#include <iostream>
using namespace std;

class Point
{
private:
    int x, y;

public:

    // Constructor
    Point(int a, int b)
    {
        x = a;
        y = b;
    }

    ~Point()
    {
        cout<<"x: "<<x<<endl;
        cout<<"y: "<<y<<endl;
    }
};

int main()
{
    Point p(1, 3);
    return 0;
}
```

Output

```
Loading personal and system profiles took 1887ms.
(base) PS D:\CUH\4th Sem\OOP with C++\Lab> cd "d:\CUH\4th Sem\OOP with C++\Lab\11. Destructor\" ; if ($?) { g++ Destructor.cpp -o Destructor } ; if ($?) { .\Destructor }
x: 1
y: 3
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/11.%20Destructor

Lab-9

Create the class TIME to store time in hours and minutes. Write a friend function to add two TIME objects.

Code

```cpp
// Write a class TIME to store time in hours and minutes. Write a friend function to add two TIME objects.

#include <iostream>
using namespace std;

class Time
{
private:
    int hours, minutes;

public:
    void input(int h, int min)
    {
        hours = h;
        minutes = min;
    }

    friend Time Add(Time, Time);

    void display()
    {
        cout<<"Time: "<<hours<<":"<<minutes<<endl;
    }
};

Time Add(Time obj1, Time obj2)
{
    Time obj3;
    obj3.hours = obj1.hours + obj2.hours;
    obj3.minutes = obj1.minutes + obj2.minutes;

    return obj3;
}

int main()
{
    Time t1, t2, t3;

    t1.input(10, 36);
    t2.input(3, 13);
    t3 = Add(t1, t2);

    t3.display();

    return 0;
}
```

Output

```
(base) PS D:\CUH\4th Sem\OOP with C++\Lab\11. Destructor> cd "d:\CUH\4th Sem\OOP with C++\Lab\7. Time\" ; if ($?) { g++ Time.cpp -o Time } ; if ($?) { .\Time }
Time: 13:49
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/7.%20Time

Lab-10

Create two classes DM and DB. DM stores the distance in meters and centimeters and DB stores the distance in feet and inches. Write a program to add object of DM with the object of DB class.

Code

```
// Create two classes DM and DB. DM stores the distance in meters and centimeters a

#include <iostream>
using namespace std;

class DM
{
public:
    int meters;
    int centimeters;

    DM(int m, int cm)
    {
        meters = m;
        centimeters = cm;

        while (centimeters > 99)
        {
            centimeters -= 100;
            meters++;
        }
    }

    void display()
    {
        cout<<"Meters: "<<meters<<endl;
        cout<<"Centimeters: "<<centimeters<<endl;
    }
};

class DB
{
public:
    int feet;
    int inches;

    DB(int ft, int in)
    {
        feet = ft;
        inches = in;

        while (inches > 11)
        {
            inches -= 12;
            feet++;
        }
    }
```

```
49      void display()
50      {
51          cout<<"Feet: "<<feet<<endl;
52          cout<<"Inches: "<<inches<<endl;
53      }
54  };
55
56  DM add_DB_to_DM(DM a, DB b)
57  {
58      DM result(a.meters, a.centimeters);
59
60      result.meters += 0.3048*b.feet;
61      result.centimeters += 2.54*b.inches;
62
63      while (result.centimeters > 99)
64      {
65          result.centimeters -= 100;
66          result.meters++;
67      }
68
69      return result;
70  }
71
72  int main()
73  {
74      DM obj1(8, 54);
75      DB obj2(5, 6);
76
77      obj1.display();
78      obj2.display();
79
80      DM result = add_DB_to_DM(obj1, obj2);
81      result.display();
82
83      return 0;
84  }
```

Output

```
Meters: 8
Centimeters: 54
Feet: 5
Inches: 6
Meters: 9
Centimeters: 69
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/8.%20Distance%20Conversion

Lab-11

Program to overload unary operator.

Code

```cpp
// Program to overload unary operator

#include <iostream>
using namespace std;

class Distance
{
private:
    int meters, centimeters;

public:
    Distance()
    {
        meters = 0;
        centimeters = 0;
    }

    Distance(int m, int cm)
    {
        meters = m;
        centimeters = cm;
    }

    // Displays Distance
    void display()
    {
        cout<<meters<<" m "<<centimeters<<" cm"<<endl;
    }

    Distance operator- ()
    {
        meters = -meters;
        centimeters = -centimeters;

        return Distance(meters, centimeters);
    }
};

int main()
{
    Distance d1(11, 10), d2(5, 1);

    -d1;
    d1.display();

    -d2;
    d2.display();

    return 0;
}
```

Output

```
(base) PS D:\CUH\4th Sem\OOP with C++\Lab\8. Distance Conversion> cd "d:
\CUH\4th Sem\OOP with C++\Lab\16. Overload Unary Operator\" ; if ($?) {
g++ UnaryOperator.cpp -o UnaryOperator } ; if ($?) { .\UnaryOperator }
-11 m -10 cm
-5 m -1 cm
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/16.%20Overload%20Unary%20Operator

Lab-12

Program to overload binary operator.

Code

```cpp
// Program to overload binary operator

#include <iostream>
using namespace std;

class Distance
{
private:
    int meters, centimeters;

public:
    Distance()
    {
        meters = 0;
        centimeters = 0;
    }

    Distance(int m, int cm)
    {
        meters = m;
        centimeters = cm;
    }

    // Displays Distance
    void display()
    {
        cout<<meters<<" m "<<centimeters<<" cm"<<endl;
    }

    Distance operator+ (Distance const &obj)
    {
        Distance result;

        result.meters = meters + obj.meters;
        result.centimeters = centimeters + obj.centimeters;

        return result;
    }
};

int main()
{
    Distance d1(10, 5), d2(2, 4);
    Distance d3 = d1 + d2;
    d3.display();

    return 0;
}
```

Output

```
(base) PS D:\CUH\4th Sem\OOP with C++\Lab\16. Overload Unary Operator> cd "d:\CUH\4th Sem\OOP with C++\Lab\
17. Overload Binary Operator\" ; if ($?) { g++ BinaryOperator.cpp -o BinaryOperator } ; if ($?) { .\BinaryO
perator }
12 m 9 cm
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/17.%20Overload%20Binary%20Operator

Lab-13

Program to show multiple inheritance.

Code

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

class A
{
public:
    A()
    {
        cout<<"Constructor of A\n";
    }
};

class B
{
public:
    B()
    {
        cout<<"Constructor of B\n";
    }
};

class C: public B, public A
{
public:
    C()
    {
        cout << "Constructor of C\n";
    }
};

int main()
{
    C c;
    return 0;
}
```

Output

```
(base) PS D:\CUH\4th Sem\OOP with C++\Lab\17. Overload Binary Operator> cd "d:\CUH\4th Sem\OOP with C++\Lab
\12. Multiple Inheritance\" ; if ($?) { g++ multipleInheritance.cpp -o multipleInheritance } ; if ($?) { .\
multipleInheritance }
Constructor of B
Constructor of A
Constructor of C
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/12.%20Multiple%20Inheritance

Lab-14

Program to show Multi-Level Inheritance.

Code

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

class A
{
public:
    void FunctionA()
    {
        cout<<"This is in class A";
    }
};

// Derived class (child)
class Child: public A
{

};

// Derived class (grandchild)
class GrandChild: public Child
{
};

int main()
{
    GrandChild myObj;
    myObj.FunctionA();
    return 0;
}
```

Output

```
(base) PS D:\CUH\4th Sem\OOP with C++\Lab\12. Multiple Inheritance> cd "d:\CUH\4th Sem\OOP with C++\Lab\13.
 Multi-level Inheritance\" ; if ($?) { g++ multiLevelInheritance.cpp -o multiLevelInheritance } ; if ($?) {
 .\multiLevelInheritance }
This is in class A
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/13.%20Multi-level%20Inheritance

Lab-15

Program to show hybrid inheritance.

Code

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

class A
{
public:
    A()
    {
        cout<<"This is constructor A\n";
    }
};

class B: public A
{
public:
    B()
    {
        cout<<"This is constructor B\n";
    }
};

class C
{
public:
    void display()
    {
        cout<<"This is member function of C\n";
    }
};

class D: public B, public C
{
public:
    D()
    {
        cout<<"Hybrid class D\n";
    }
};

int main()
{
    D d;
    d.display();

    return 0;
}
```

Output

```
(base) PS D:\CUH\4th Sem\OOP with C++\Lab\13. Multi-level Inheritance> cd "d:\CUH\4th Sem\OOP with C++\Lab\14. Hybrid Inheritance\" ; if ($?) { g++ hybridInheritance.cpp -o hybridInheritance } ; if ($?) { .\hybridInheritance }
This is constructor A
This is constructor B
Hybrid class D
This is member function of C
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/14.%20Hybrid%20Inheritance

Lab-16

Program to show run time polymorphism using virtual function.

Code

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

class Shape
{
public:
        virtual void calculate()
        {
                cout<<"Area of your Shape ";
        }
};

class Rectangle : public Shape
{
public:
        int width, height, area;

        void calculate()
        {
                cout<<"Enter Width of Rectangle: ";
                cin>>width;

                cout<<"Enter Height of Rectangle: ";
                cin>>height;

                area = height * width;
                cout << "Area of Rectangle: " << area << "\n";
        }
};

class Square : public Shape {
public:
        int side, area;

        void calculate()
        {
                cout << "Enter one side your of Square: ";
                cin >> side;

                area = side * side;
                cout << "Area of Square: " << area;
        }
};

int main()
{
    Rectangle r;
        r.calculate();

    Square sq;
        sq.calculate();

        return 0;
}
```

Output

```
Enter Width of Rectangle: 10
Enter Height of Rectangle: 20
Area of Rectangle: 200
Enter one side your of Square: 23
Area of Square: 529
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/18.%20Polymorphism