DHA SUFFA UNIVERSITY
Department of Computer Science
Object Oriented Programming
CS-1002L
Spring 2020

Lab 06

Operator Overloading - II

Objective:

To learn about overloading of assignment operators,subscript operator and functors

Assignment Operator Overloading

Assignment operator is overloaded to copy properties of one object to another.

Example 6.1

```
//FileName:Distance.h

#ifndef DISTANCE_H
#define DISTANCE_H
#include <iostream>
class Distance
{
            //Declaring data members
      private:
            int feet;
            int inches;
      public:
            //Declaring non-parameterized constructor
            Distance();
            //Declaring parameterized constructor
            Distance(int f, int i);
            //Declaring destructor
            ~Distance();
            //Declaring Assignment operator overloading
            Distance operator = (const Distance &D);
            void displayDistance();
};
#endif
```

```
//FileName:Distance.cpp

#include "Distance.h"
//Defining non parameterized constructor
Distance :: Distance()
{
```

```
        feet = 0;
        inches = 0;
}
//Defining parameterized constructor
Distance :: Distance(int f, int i)
{
        feet = f;
        inches = i;
}
//Defining Destructor
Distance ::~ Distance()
{
}

//Defining assignment operator overloading
Distance Distance :: operator = (const Distance &d )
{
        feet = d.feet;
        inches =d.inches;
        return d;
}
//Defining function
void Distance :: displayDistance()
{
        std::cout << "F: " << feet <<  " I:" <<  inches << std::endl;
}
```

```
//FileName:Main.cpp

#include"Distance.h"
#include <iostream>
int main()
{
   //Creating object of distance class
   Distance d1(1,2), d2(20,30);
   std::cout << "First Distance : "<< std::endl;
   d1.displayDistance();
   std::cout << "Second Distance :"<< std::endl;
   d2.displayDistance();
   //calling assignment operator overloading function
   d1 = d2;
   std::cout << "First Distance :"<< std::endl;
```

```
   d1.displayDistance();
   return 0;
}
```

Output

```
First Distance :
F: 1 I:2
Second Distance :
F: 20 I:30
First Distance :
F: 20 I:30
```

Overloading += operator

+= operator is overloaded to add and assign incremented values of properties of one object to another.

Example 6.2

```
//FileName:Distance.h

#ifndef DISTANCE_H
#define DISTANCE_H
#include <iostream>
class Distance
{
//Declaring data members
      private:
            int feet;
            int inches;
      public:
      //Declaring non-parameterized constructor
            Distance();
            //Declaring parameterized constructor
            Distance(int f, int i);
            //Declaring destructor
            ~Distance();
            //Declaring Compound Assignment operator overloading
            Distance operator += (const Distance &D);
            void displayDistance();
```

```
};
#endif
```

```
//FileName:Distance.cpp

#include"Distance.h"
//Defining non parameterized constructor
Distance :: Distance()
{
         feet = 0;
         inches = 0;
}
//Defining parameterized constructor
Distance :: Distance(int f, int i)
{
         feet = f;
         inches = i;
}
//Defining destructor
Distance :: ~Distance()
{
}
//Defining assignment operator overloading
Distance Distance :: operator += (const Distance &d )
{
         feet += d.feet;
         inches +=d.inches;
         return d;
}
//Defining function
void Distance :: displayDistance()
{
         std::cout << "F: " << feet <<  " I:"
         <<  inches <<  std::endl;
}
```

```
//FileName:Main.cpp
```

```
#include "Distance.h"
#include <iostream>
int main()
{
      //Creating object of distance class
   Distance d1(1,2), d2(20,30);
   std::cout << "First Distance : "<< std::endl;
   d1.displayDistance();
   std::cout << "Second Distance :"<< std::endl;
   d2.displayDistance();
   //calling assignment operator overloading function
   d1 += d2;
   std::cout << "First Distance :"<< std::endl;
   d1.displayDistance();
   return 0;
}
```

Output

```
First Distance :
F: 1 I:2
Second Distance :
F: 20 I:30
First Distance :
F: 21 I:32
```

Subscripting [ ] Operator Overloading

The subscript operator [] is overloaded to access array elements of object of a class

Example 6.3

```
//FileName:Marks.h

#ifndef MARKS_H
#define MARKS_H
#include <iostream>
//Defining class
class Marks
{
   //Declaring data members(instance members)
   private:
            int *numbers;
```

```
            int size;
   public:
            //Declaring parameterized constructor
            Marks(int);
            //Declaring [] operator overloading
            int& operator[](int i);
            // Declaring destructor
            ~Marks();
};
#endif
```

```
//FileName:Marks.cpp

#include "Marks.h"
//Defining parameterize constructor
Marks :: Marks(int s)
{
  size=s;
  numbers=new int[size];
}
Marks :: ~Marks
{
  delete []numbers;
}

//Defining [] overloading function
int& Marks :: operator[](int i)
{
  if( i >= size )
    {
            std::cout << "Index out of bounds";
            return numbers[0];
    }
  return numbers[i];
}
```

```
//FileName:Main.cpp
```

```
#include "Marks.h"
int main()
{
  int count;
  std::cout << "Enter Number Of Elements\n";
  std::cin >> count;
  Marks m1(count);
  int i;
  for(i = 0; i < count; ++i)
  {
      std::cin >> m1[i];
  }
        std::cout << "Value of marks[0] : " << m1[0] << std::endl;
        std::cout << "Value of marks[1] : " << m1[1] << std::endl;
        std::cout << "Value of marks[2] : " << m1[2] << std::endl;
        std::cout << "Value of marks[3] : " << m1[3] << std::endl;
        std::cout << "Value of marks[4] : " << m1[4] << std::endl;
        std::cout << "Value of marks[5] : " << m1[5] << std::endl;
        return 0;
}
```

Output

```
Enter Number Of Elements
5
10
20
30
40
50
Value of marks[0] : 10
Value of marks[1] : 20
Value of marks[2] : 30
Value of marks[3] : 40
Value of marks[4] : 50
Index out of boundsValue of marks[5] : 10
```

Functors

Functor or function object is a C++ class which defines the operator ( ). Functor let's you create objects which "looks like" functions.

Example 6.4

```
//FileName:Series.h

#ifndef SERIES_H
```

```
#define SERIES_H
#include <iostream>
//Defining class
class Series
{
      //Declaring data members (instance variables)
      private:
            int difference;
            int firstTerm;
      //Declaring Member functions
      public:
            Series(int , int);
            int operator()(int );
};
#endif
```

```
//FileName:Series.cpp

#include "Series.h"
//Defining constructor
Series :: Series(int a , int b ):difference(a),firstTerm(b)
{
}
 //Defining  () overloading
int Series :: operator()(int number)
{
      int answer=firstTerm+(number-1)*difference ;
      return answer;
}
```

```
//FileName:Main.cpp

#include "Series.h"
int main ()
{
  //Creating object of Series class
  // Sum = 1+(n-1)2
  Series s1(2,1);
  // Sum = 2+(n-1)2
  Series s2(2,2);
  double result1 = s1(9);
```

```
  double result2 = s2(10);
  std::cout << "Result  = " <<result1<<std::endl;
  std::cout << "Result  = " <<result2<<std::endl;
  return 0;
}
```

Output

```
Result  = 17
Result  = 20
```

Rule of Three

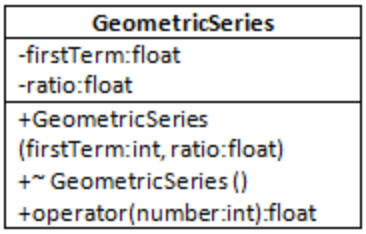
This rules states that its a good practise that if you are using any of the following constraints then explicitly define them

- Copy constructor
- Assignment Operator
- Destructor

Lab Tasks

Task 6.1

Implement following UML in C++



overload functors to calculate geometric series of a number.(Formula:S_n =a(r^n-1)/1-r)

Task 6.2

Implement following UML in C++

Square
-side[]:float
+Square() +~ Square() +operator=(object:Square):Square +showInfo():void +operator():float

Overload functors to calculate area of a square.(Formula:Area=side^2)
Overload assignment operator to copy one object of a square object to another

Task 6.3

Implement the following UML in cpp

Student
-id:string -name:string -sem:string -*courses:string
+Student(id:string,name:string,sem:string) +~Student() +operator[](int i):string& +viewInfo():void

Submission Instructions

1. 1. Number your solution folders as question number e.g. Q1, Q2, etc. (Q is in upper case)
2. 2. Every folder should contain three files (one header, one implementation and one driver)
3. 3. Create a new folder named cs152abc where abc is your 3 digit roll #. e.g. cs152111.
4. 4. Copy all the project folders into this folder.
5. 5. Now make sure a zip file named cs152abc.zip is created e.g. cs152111.zip
6. 6. Upload the assignment solution on LMS under the assignment named Lab 06 Assignment – XX, where XX is your section name