National University of Computer and Emerging Sciences



Assignment 05

Object Oriented Programming

|  |  |
| --- | --- |
| Name | Muhammad Zain |
| Roll No. | 19F-0228 |
| INSTRUCTOR | Dr Danish Shehzad |
| Semester | Spring 2020 |

Task 1:

# Question:

(RationalNumbers) Create a class Rational Number (fractions) with the following capabilities: a) Create a constructor that prevents a 0 denominator in a fraction, reduces or simplifies fractions that are not in reduced form and avoids negative denominators. b) Overload the addition, subtraction, multiplication and division operators for this class. c) Overload the relational and equality operators for this class.

# Source Code:

#include<iostream>

using namespace std;

class Rational

{

float Numerator;

float Denominator;

public:

Rational() {

cout << "This is Rational Class Constructor" << endl;

Numerator != 0;

Denominator != 0;

}

void input()

{

cout << "\tInput Rational Number "<<endl;

cout << "\n\nInput value of Numerator " << endl;

cin >> Numerator;

cout << "Input value of Denominator "<<endl;

cin >> Denominator;

while (Denominator <= 0)

{

cout << "Inavild input :-(" << endl;

cout<< "Input Again ";

cin >> Denominator;

}

}

void print1()

{

cout << "\nRationel Number before Updation " << endl;

cout << Numerator << "/" << Denominator << endl;

}

Rational operator ++()//prefix

{

Numerator++;

Denominator++;

return \*this;

}

Rational operator ++(int)

// for postfix we have to put one as argument

//ek us na khud la lena obj 2sra usko dena parna argument pass kr k

{

Numerator++;

Denominator++;

return \*this;

}

Rational operator --()//prefix

{

Numerator--;

Denominator--;

return \*this;

}

Rational operator --(int)

// for postfix we have to put one as argument

//ek us na khud la lena obj 2sra usko dena parna argument pass kr k

{

Numerator--;

Denominator--;

return \*this;

}

Rational operator !()

{

int swap;

swap = Numerator;

Numerator = Denominator;

Denominator = swap;

return \*this;

}//this to return whole function values

void Print2()

{

cout << "Rationel Number After Updation " << endl;

cout << Numerator << "/" << Denominator << endl;

}

};

int main()

{

Rational obj;

obj.input();

obj.print1();

int choice;

while(int x=1)

{

cout << "Press -1 to Exit "<<endl << endl;

cout << "Press 1 to call ++ prefix " << endl;

cout << "Press 2 to call ++ postfix " << endl;

cout << "Press 3 to call -- prefix " << endl;

cout << "Press 4 to call -- postfix " << endl;

cout << "Press 5 to call ! operator" << endl<<endl;

cin >> choice;

//system("cls");

if (choice == 1)

{

++obj;

obj.Print2();

}

else if (choice == 2)

{

obj++;

obj.Print2();

}

else if (choice == 3)

{

--obj;

obj.Print2();

}

else if (choice == 4)

{

obj--;

obj.Print2();

}

else if (choice == 5)

{

!obj;

obj.Print2();

}

else if (choice == -1)

{

break;

}

}

system("pause>0");

}

# Snips:

A screenshot of a cell phone on a table

Description automatically generatedA screenshot of a cell phone on a table

Description automatically generated

Task 2

# Question:

Polynomial) Develop class Polynomial. The internal representation of a Polynomial is an array of terms. Each term contains a coefficient and an exponent, e.g., the term 2x 4 has the coefficient 2 and the exponent 4. Develop a complete class containing proper constructor and destructor functions as well as set and get functions The class should also provide the following overloaded operator capabilities: a) Overload the assignment operator to assign one Polynomial to another. b) Overload the multiplication operator (\*) to multiply two Polynomials. c) Overload the addition assignment operator (+=), subtraction assignment operator (-=), and multiplication assignment operator (\*=).

# Source code:

#include<iostream>

#include<math.h>

using namespace std;

class Polynomial

{

private:

int Array[100];

int x;

public:

Polynomial() {}

//simple constructor

void getData(int size)

{

for (int i = 0; i <= size; i++)

{

cout << "Enter the " << i + 1 << " Value : " ;

cin >> Array[i];

}cout << endl;

}

void disp(int size)

{

int dummy;

dummy = size;

for (int i = 0; i <= size; i++)

{

if (i == 0)

{

cout << Array[i] << "x^" << size;

}

else if (i < size && i>0)

{

cout << " + " << Array[i] << "x^" << --dummy;

}

else if (i == size)

{

cout << " + " << Array[i];

}

}

cout << endl;

}

Polynomial operator +(Polynomial obj\_second)

{

Polynomial t;

for (int i = 0; i <= 3; i++)

{

t.Array[i] = Array[i] + obj\_second.Array[i];

}

return t;

}

Polynomial operator +=(Polynomial obj\_second)

{

Polynomial t;

for (int i = 0; i <= 3; i++)

{

t.Array[i] = Array[i] + obj\_second.Array[i];

}

return t;

}

Polynomial operator -(Polynomial obj\_second)

{

Polynomial t;

for (int i = 0; i <= 3; i++)

{

t.Array[i] = Array[i] - obj\_second.Array[i];

}

return t;

}

Polynomial operator -=(Polynomial obj\_second)

{

Polynomial t;

for (int i = 0; i <= 3; i++)

{

t.Array[i] = Array[i] - obj\_second.Array[i];

}

return t;

}

Polynomial operator \*(Polynomial obj\_second)

{

Polynomial t;

for (int i = 0; i <= 3; i++)

{

t.Array[i] = Array[i] \* obj\_second.Array[i];

}

return t;

}

Polynomial operator \*=(Polynomial obj\_second)

{

Polynomial t;

for (int i = 0; i <= 3; i++)

{

t.Array[i] = Array[i] \* obj\_second.Array[i];

}

return t;

}

Polynomial operator ==(Polynomial obj\_second)

{

Polynomial t;

for (int i = 0; i <= 3; i++)

{

t.Array[i] = Array[i];

Array[i]=obj\_second.Array[i];

obj\_second.Array[i] = t.Array[i];

}

return t;

}

void out\_for\_multiplication(int size)

{

int dummy;

dummy = size;

for (int i = 0; i <= size; i++)

{

if (i == 0)

{

cout << Array[i] << "x^" << dummy+dummy;

}

else if (i < size && i>0)

{

int temp;

temp = --dummy;

cout << " + " << Array[i] << "x^" << temp+temp;

}

else if (i == size)

{

cout << " + " << Array[i];

}

}

cout << endl;

}

};

int main()

{

int y;

cout << "Enter the size of Array" << endl;

cin >> y;

Polynomial obj, obj2;

cout << "Enter the Values of First Array" << endl;

obj.getData(y);

cout << "Enter the Values of Second Array" << endl;

obj2.getData(y);

cout << endl << endl;

cout << "Expression of 1st Polynomial : ";

obj.disp(y);

cout << "Expression of 2nd Polynomial : ";

obj2.disp(y);

cout << endl << endl;

Polynomial obj3;//for storage of addition

obj3 = obj2 + obj;

cout << "Sum with + = ";

obj3.disp(y);

Polynomial obj4;//for storage of subtraction

obj4 = obj2 - obj;

cout << "Difference with - = ";

obj4.disp(y);

cout << endl;

Polynomial obj5;//for storage of addition

obj5 = obj2 += obj;

cout << "Sum with -= = ";

obj5.disp(y);

cout << endl;

Polynomial obj6;//for storage of subtraction

obj6 = obj2 -= obj;

cout << "Difference with -= = ";

obj4.disp(y);

cout << endl;

Polynomial obj7;//for storage of Multiplication

obj7 = obj2 \* obj;

cout << "Multiplication with \* = ";

obj7.out\_for\_multiplication(y);

cout << endl;

Polynomial obj8;//for storage of Multiplication

obj8 = obj2 \*= obj;

cout << "Multiplication with \*= = ";

obj8.out\_for\_multiplication(y);

cout << endl;

Polynomial obj9;//for storage of subtraction

obj9 = obj2 == obj;

cout << "Assinging 1 to 2 with -= = ";

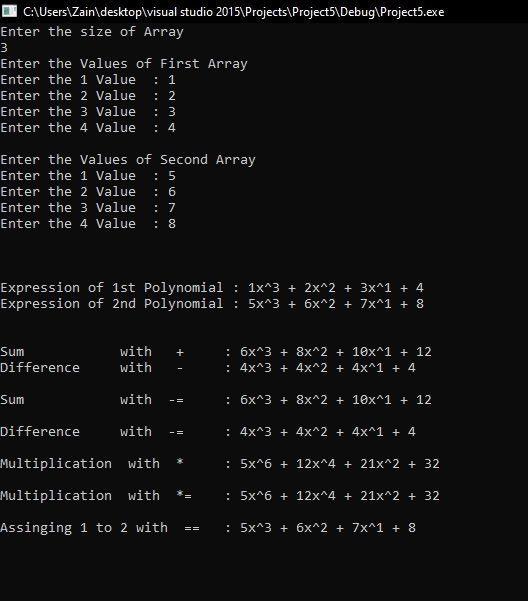
obj9.disp(y);

cout << endl;

system("Pause>0");

}

# Snip:



Task 3

# Source code:

#include <iostream>

using namespace std;

class Complex

{

private:

double Real;

double Imginary;

public:

Complex()

{

Real = 0.00;

Imginary = 0.00;

}

void getData()

{

cout << "Input Real Part : ";

cin >> Real;

cout << "Input Imginaryinary Part : ";

cin >> Imginary;

}

void disp()

{

cout << " Complex number = " << Real << "+" << Imginary << "i" << endl;

}

Complex operator + (Complex obj\_2)

{

Complex t;

t.Real = Real + obj\_2.Real;

t.Imginary = Imginary + obj\_2.Imginary;

return t;

}

Complex operator - (Complex obj\_2)

{

Complex t;

t.Real = Real - obj\_2.Real;

t.Imginary = Imginary - obj\_2.Imginary;

return t;

}

Complex operator \* (Complex obj\_2)

{

Complex t;

t.Real = Real \* obj\_2.Real;

t.Imginary = Imginary \* obj\_2.Imginary;

return t;

}

Complex operator / (Complex obj\_2)

{

Complex t;

t.Real = Real / obj\_2.Real;

t.Imginary = Imginary / obj\_2.Imginary;

return t;

}

bool operator ==(Complex &obj\_2)

{

if (Real == obj\_2.Real && Imginary == obj\_2.Imginary)

{

return true;

}

else

return false;

}

bool operator !=(Complex &obj\_2)

{

if (Real == obj\_2.Real && Imginary == obj\_2.Imginary)

{

return false;

}

else

return true;

}

};

int main()

{

Complex c1, c2, c3, c4, c5, c6;

cout << "Input first Complex number" << endl;

c1.getData();

c1.disp();

cout << "Input second complex number" << endl;

c2.getData();

c2.disp();

cout << endl << endl << endl;

c3 = c1 + c2;

cout << "Operator + overloading " << endl;

c3.disp();

cout << endl;

c4 = c1 - c2;

cout << "Operator - overloading " << endl;

c4.disp();

cout << endl;

c5 = c1\*c2;

cout << "Operator \* overloading " << endl;

c5.disp();

cout << endl;

c6 = c1 / c2;

cout << "Operator \ overloading " << endl;

c6.disp();

cout << endl;

cout << "Operator == overloading " << endl;

if (c1 == c2 == true)

{

cout << "Both are Equal." << endl;

}

else

{

cout << "Not Equal." << endl;

}

cout << "Operator !=overloading " << endl;

if (c1 != c2 == true)

{

cout << "Not Equal." << endl;

}

else

{

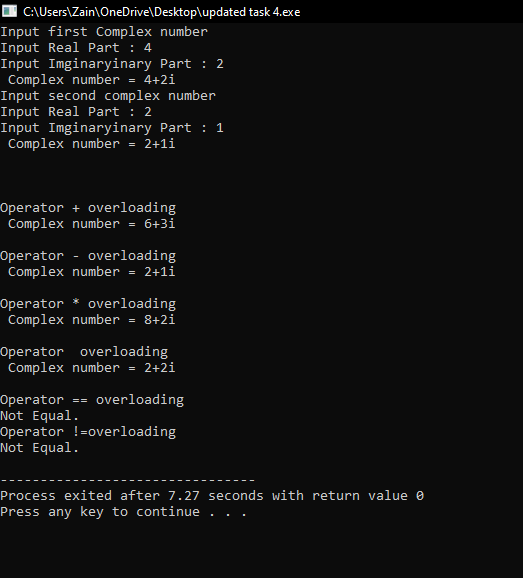
cout << "Equal." << endl;

}

return 0;

}

# Snip:



}}Task 4

# Question:

(Arrays) Develop a class to implement a class for Array with following functionalities

a) Range checking

b) Array assignment

c) Arrays that know their size

d) Outputting/inputting entire arrays with << and >>

# Source Code:

#include<iostream>

using namespace std;

class Array {

private:

int \*array;

int size;

public:

Array() {}

int &operator[](int check)

{

if (check < size)

{

cout << "Array is in Bound" << endl;

return array[check];

}

else

{

cout << "Out of bound " << endl;

}

}

int getsize()

{

return size;

}

void operator==(Array& obj)

{

cout << "Enter the size of Array " << endl;

cin >> obj.size;

cout << "Input the Array of " << obj.size << " element" << endl << endl;

obj.array = new int[obj.size];

for (int j = 0; j<obj.size; j++)

{

cout << "Input the " << j + 1 << " Element of array : ";

cin >> obj.array[j];

}

}

friend ostream &operator <<(ostream& o, Array&);

friend istream &operator >> (istream& i, Array&);

};

istream & operator >> (istream &in, Array &obj\_in)

{

cout << "Enter the size of Array " << endl;

cin >> obj\_in.size;

cout << "Input the Array of " << obj\_in.size << " element" << endl << endl;

obj\_in.array = new int[obj\_in.size];

for (int j = 0; j<obj\_in.size; j++)

{

cout << "Input the " << j + 1 << " Element of array : ";

in >> obj\_in.array[j];

}

return in;

}

ostream& operator <<(ostream &out, Array &obj\_o)

{

cout << endl << endl;

cout << "Outputing using << operator " << endl;

for (int j = 0; j < obj\_o.size; j++)

{

out << j + 1 << " Element of array : ";

out << obj\_o.array[j] << endl;

}

return out;

}

int main()

{

int pos, value;

Array obj;

cout << "Inputing the array using >> operator " << endl;

cin >> obj;

cout << obj;

cout << "Assinging the values by using == operator : ";

obj.operator==(obj);

cout << obj;

cout << "latest size of array" << endl;

cout<<obj.getsize();

cout << "To check the Range in latest entries input the index : ";

cin >> pos;

obj[pos] = 10;//just to check the position if exists

system("pause>0");

}

# Snip:

A screenshot of a cell phone

Description automatically generated

Task 5

# Question:

(Calendar Class) Defining Operator (+=) and Operator (-=) to Add or Subtract Days in the Calendar Given an Integer Input.

# Source Code:

#include<iostream>

using namespace std;

class Day\_Change {

protected:

int day;

int month;

int year;

public:

Day\_Change() {}

Day\_Change(int x, int y, int z)

{

day = x;

month = y;

year = z;

}

friend Day\_Change operator +=(Day\_Change&, int);

friend Day\_Change operator -=(Day\_Change&, int);

void output()

{

cout << "Date : " << day << " - " << month << " - " << year << endl;

}

};

Day\_Change operator +=(Day\_Change&obj, int input)//jo user input kray ga k itna plus krna

{

return Day\_Change(obj.day + input, obj.month, obj.year);

}

Day\_Change operator -=(Day\_Change&obj, int input)//jo user input kray ga k itna plus krna

{

return Day\_Change(obj.day - input, obj.month, obj.year);

}

int main()

{

int Day, Month, Year, plus, minus;

cout << "Input Day " << endl;

cin >> Day;

cout << "Input Month" << endl;

cin >> Month;

cout << "Input Year " << endl;

cin >> Year;

cout << endl << endl;

cout << "Enter number of Days to Add " << endl;

cin >> plus;

Day\_Change Obj\_1(Day, Month, Year);//parametrized constructor

Day\_Change Obj\_2 = Obj\_1 += plus;

Obj\_2.output();

cout << endl << endl;

cout << "Enter number of Days to Subtract " << endl;

cin >> minus;

Day\_Change Obj\_3 = Obj\_2 -= minus;//in d2

Obj\_3.output();

system("pause>0");

}

# Snip:

