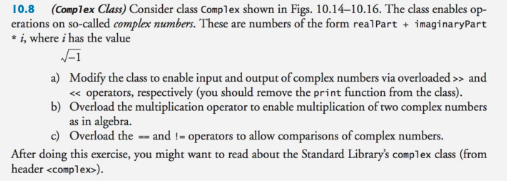
Name: Mukhammadnozim Islamov Class: COMSC-200 Lab: 2 Due Date: 02/22/2019

**PROBLEM – 1**



Answer:

**Complex.h**

// Fig. 10.14: Complex.h

// Complex class definition.

#ifndef COMPLEX\_H

#define COMPLEX\_H

#include<iostream>

class Complex

{

public:

explicit Complex( double = 0.0, double = 0.0 ); // constructor

Complex operator+( const Complex & ) const; // addition

Complex operator-( const Complex & ) const; // subtraction

void print() const; // output

friend std::ostream &operator<<(std::ostream &, const Complex &);

friend std::istream &operator>>(std::istream &, Complex &);

bool operator == (const Complex &) const;

bool operator != (const Complex &) const;

private:

double real; // real part

double imaginary; // imaginary part

}; // end class Complex

#endif

**Complex.cpp**

// Fig. 10.15: Complex.cpp

// Complex class member-function definitions.

#include <iostream>

#include "Complex.h" // Complex class definition

using namespace std;

// Constructor

Complex::Complex( double realPart, double imaginaryPart )

: real( realPart ),

imaginary( imaginaryPart )

{

// empty body

} // end Complex constructor

// addition operator

Complex Complex::operator+( const Complex &operand2 ) const

{

return Complex( real + operand2.real,

imaginary + operand2.imaginary );

} // end function operator+

// subtraction operator

Complex Complex::operator-( const Complex &operand2 ) const

{

return Complex( real - operand2.real,

imaginary - operand2.imaginary );

} // end function operator-

// display a Complex object in the form: (a, b)

void Complex::print() const

{

cout << '(' << real << ", " << imaginary << ')';

} // end function print

ostream &operator<<( ostream &output, const Complex &obj)

{

output << "( " << obj.real << ", " << obj.imaginary << " )";

return output;

}

istream &operator >> ( istream &input, Complex &obj)

{

cout << "Please enter the real part: ";

input >> obj.real;

cout << "Now enter the imaginary part: ";

input >> obj.imaginary;

return input;

}

bool Complex::operator==(const Complex &operand2) const

{

if (real == operand2.real && imaginary == operand2.imaginary)

return true;

else

return false;

}

bool Complex::operator!=(const Complex &operand2) const

{

return !(\*this == operand2);

}

**main.cpp**

#include<iostream>

#include "Complex.h"

using namespace std;

int main()

{

Complex c1(10,-3);

Complex c2;

cout<<"Imaginary numbers program."<<endl;

cout<<"Complex number with '<<' operator: "<<c1<<endl;

cout<<"Now let's try with '>>' operator."<<endl;

cin>>c2;

cout<<"Here is your complex number with '<<' operator: "<<c2<<endl;

cout<<"Now let's check them on equality."<<endl;

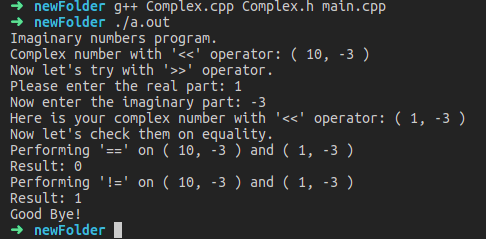
cout<<"Performing '==' on "<<c1<<" and "<<c2<<endl;

cout<<"Result: "<<(c1==c2)<<endl;

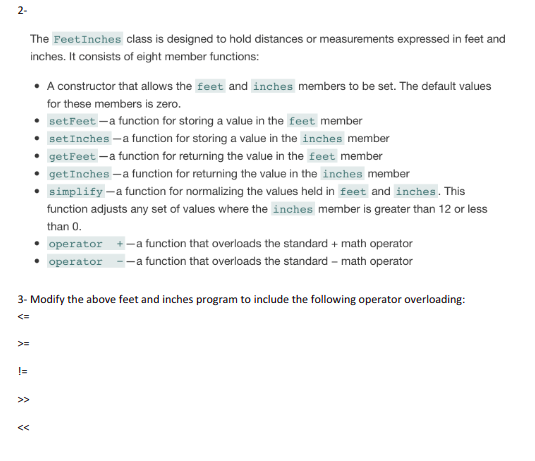
cout<<"Performing '!=' on "<<c1<<" and "<<c2<<endl;

cout<<"Result: "<<(c1!=c2)<<endl;

cout<<"Good Bye!"<<endl;

}

**PROBLEM-2 && PROBLEM-3**



Answer:

**FeetInches.h**

#ifndef FEETINCHES\_H

#define FEETINCHES\_H

#include<iostream>

class FeetInches

{

private:

int feet;

int inches;

public:

explicit FeetInches(int = 0, int = 0);

void setFeet(const int);

void setInches(const int);

int getFeet();

int getInches();

void simplify();

FeetInches operator+(const FeetInches &)const;

FeetInches operator-(const FeetInches &)const;

bool operator<=(const FeetInches &)const;

bool operator>=(const FeetInches &)const;

bool operator!=(const FeetInches &)const;

bool operator==(const FeetInches &)const;

friend std::ostream &operator<<(std::ostream &, const FeetInches &);

friend std::istream &operator>>(std::istream &, FeetInches &);

};

#endif

**FeetInches.cpp**

#include <iostream>

#include "FeetInches.h"

using namespace std;

FeetInches::FeetInches( int f, int i )

: feet( f ),

inches( i )

{

// empty body

} // end Complex constructor

void FeetInches::setFeet(const int f)

{

feet = f;

}

void FeetInches::setInches(const int i)

{

inches = i;

}

int FeetInches::getFeet()

{

return feet;

}

int FeetInches::getInches()

{

return inches;

}

void FeetInches::simplify()

{

feet += inches/12;

inches %= 12;

if (inches < 0 && inches > -12)

{

feet--;

inches = 12 + inches;

}

}

FeetInches FeetInches::operator+( const FeetInches &operand2 ) const

{

return FeetInches( feet + operand2.feet,

inches + operand2.inches );

} // end function operator+

// subtraction operator

FeetInches FeetInches::operator-( const FeetInches &operand2 ) const

{

return FeetInches( feet - operand2.feet,

inches - operand2.inches );

} // end function operator-

ostream &operator<<( ostream &output, const FeetInches &obj)

{

output << "( " << obj.feet << ", " << obj.inches << " )";

return output;

}

istream &operator >> ( istream &input, FeetInches &obj)

{

cout << "Please enter the Feet: ";

input >> obj.feet;

cout << "Now enter the Inches: ";

input >> obj.inches;

return input;

}

bool FeetInches::operator==(const FeetInches &operand2) const

{

if (feet == operand2.feet && inches == operand2.inches)

return true;

else

return false;

}

bool FeetInches::operator!=(const FeetInches &operand2) const

{

return !(\*this == operand2);

}

bool FeetInches::operator <=(const FeetInches &operand2) const

{

if ((feet \* 12 + inches) <= (operand2.feet \* 12 + operand2.inches))

return true;

else

return false;

}

bool FeetInches::operator >=(const FeetInches &operand2) const

{

if ((feet \* 12 + inches) >= (operand2.feet \* 12 + operand2.inches))

return true;

else

return false;

}

**main.cpp**

#include<iostream>

#include "FeetInches.h"

using namespace std;

int main()

{

FeetInches f1;

FeetInches f3;

int num;

cout<<"FeetInches Problem."<<endl;

cout<<"Setting the values of 2st variable, input the feet part:";

cin>>num;

f1.setFeet(num);

cout<<"Setting the values of 2st variable, input the inches part:";

cin>>num;

f1.setInches(num);

cout<<"The other values will be set with constructor: (1,10)"<<endl;

FeetInches f2(1,10);

cout<<"The values of 2nd variable, feet: "<<f2.getFeet()<<", inches: "<<f2.getInches()<<endl;

cout<<"Performing addition: ("<<f1.getFeet()<<","<<f1.getInches()<<") + ("<<f2.getFeet()<<","<<f2.getInches()<<")"<<endl;

f3 = f1+f2;

cout<<"The answer is: ("<<f3.getFeet()<<","<<f3.getInches()<<")"<<endl;

cout<<"Performing simplify of: ("<<f3.getFeet()<<","<<f3.getInches()<<")"<<endl;

f3.simplify();

cout<<"The answer is: ("<<f3.getFeet()<<","<<f3.getInches()<<")"<<endl;

cout<<"Performing subtruction: ("<<f1.getFeet()<<","<<f1.getInches()<<") - ("<<f2.getFeet()<<","<<f2.getInches()<<")"<<endl;

f3 = f1-f2;

cout<<"The answer is: ("<<f3.getFeet()<<","<<f3.getInches()<<")"<<endl;

cout<<"--------------------------------------------------------"<<endl;

cout<<"Now lets use operators '<<' and '>>'"<<endl;

cout<<"First Variable:"<<endl;

cin>>f1;

cout<<"Second Variable:"<<endl;

cin>>f2;

cout<<"First Variable:"<<f1<<endl;

cout<<"Second Variable:"<<f2<<endl;

cout<<f1<<" != "<<f2<< " ===> "<<(f1!=f2)<<endl;

cout<<f1<<" <= "<<f2<< " ===> "<<(f1<=f2)<<endl;

cout<<f1<<" >= "<<f2<< " ===> "<<(f1>=f2)<<endl;

}

