

Курс

«Объектно-ориентированное программирование на C++»

Встреча №6

*Тема: Перегрузка операторов*

Задание 1.

Создайте класс Дробь (или используйте уже ранее создан- ный вами). Используя перегрузку операторов реализуйте для него арифметические операции для работы с дробями (операции +, -, \*, /).

Задание 2.

Создайте класс Complex (комплексное число) или исполь- зуйте уже созданный вами класс. Создайте перегруженные операторы для реализации арифметических операций для по работе с комплексными числами (операции +, -, \*, /).

Завдання 1

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include<Windows.h>

#include<math.h>

#include<string.h>

#include<stdlib.h>

#include<direct.h>

#include<io.h>

#include<time.h>

#include<conio.h>

using namespace std;

class Drib

{

private:

int chis;

int znamen;

public:

Drib()

{

chis = 0;

znamen = 0;

}

Drib(int chis, int znamen)

{

this->chis = chis;

this->znamen = znamen;

}

Drib(const Drib& obj)

{

this->chis = obj.chis;

this->znamen = obj.znamen;

}

void Set\_Drib()

{

cout << "Введіть чисельник дробу ->";

cin >> chis;

cout << "Введіть знаменник дробу ->";

cin >> znamen;

}

void Print\_Drib()

{

if (znamen == 0)

cout << "ERROR!\n";

else if (chis != 0)

cout << chis << "/" << znamen << endl;

else

cout << 0 << endl;

}

Drib Short\_Drib(Drib a)

{

for (int i = 2; i <= a.znamen; i++)

{

if (a.chis % i == 0 && a.znamen % i == 0)

{

a.chis = a.chis / i;

a.znamen = a.znamen / i;

i = 1;

}

}

return a;

}

Drib operator=(const Drib& obj)

{

this->chis = obj.chis;

this->znamen = obj.znamen;

return \*this;

}

Drib operator+(const Drib& obj)

{

Drib tmp;

if (this->znamen == obj.znamen)

{

tmp.chis = this->chis + obj.chis;

tmp.znamen = this->znamen;

return Short\_Drib(tmp);

}

else

{

this->chis = this->chis \* obj.znamen;

tmp.chis = obj.chis \* this->znamen;

tmp.znamen = this->znamen \* obj.znamen;

tmp.chis = tmp.chis + this->chis;

return Short\_Drib(tmp);

}

}

Drib operator-(const Drib& obj)

{

Drib tmp;

if (this->znamen == obj.znamen)

{

tmp.chis = this->chis - obj.chis;

tmp.znamen = this->znamen;

return Short\_Drib(tmp);

}

else

{

this->chis = this->chis \* obj.znamen;

tmp.chis = obj.chis \* this->znamen;

tmp.znamen = this->znamen \* obj.znamen;

tmp.chis = this->chis - tmp.chis;

return Short\_Drib(tmp);

}

}

Drib operator\*(const Drib& obj)

{

Drib tmp;

tmp.chis = this->chis \* obj.chis;

tmp.znamen = this->znamen \* obj.znamen;

return Short\_Drib(tmp);

}

Drib operator/(const Drib& obj)

{

Drib tmp;

tmp.chis = this->chis \* obj.znamen;

tmp.znamen = this->znamen \* obj.chis;

return Short\_Drib(tmp);

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Drib a(30, 15), b;

b = a;

b = b.Short\_Drib(b);

b.Print\_Drib();

return 0;

}

Завдання 2

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include<Windows.h>

#include<math.h>

#include<string.h>

#include<stdlib.h>

#include<direct.h>

#include<io.h>

#include<time.h>

#include<conio.h>

using namespace std;

class Complex

{

private:

float chislo;

float imag;

public:

Complex()

{

chislo = 0;

imag = 0;

}

Complex(float chislo, float imag)

{

this->chislo = chislo;

this->imag = imag;

}

Complex(const Complex& obj)

{

this->chislo = obj.chislo;

this->imag = obj.imag;

}

void Set\_Complex()

{

cout << "Введіть число ->";

cin >> chislo;

cout << "Введіть уявну одиницю(число)->";

cin >> imag;

}

void Print\_Complex()

{

cout << chislo << " ";

if (imag > 0)

cout << "+ " << imag << "i\n";

else if (imag < 0)

cout << "- " << (imag \* (-1)) << "i\n";

}

Complex operator=(const Complex& obj)

{

this->chislo = obj.chislo;

this->imag = obj.imag;

return \*this;

}

Complex operator+(const Complex& obj)

{

Complex tmp;

tmp.chislo = this->chislo + obj.chislo;

tmp.imag = this->imag + obj.imag;

return tmp;

}

Complex operator-(const Complex& obj)

{

Complex tmp;

tmp.chislo = this->chislo - obj.chislo;

tmp.imag = this->imag - obj.imag;

return tmp;

}

Complex operator\*(const Complex& obj)

{

Complex tmp;

tmp.chislo = (this->chislo \* obj.chislo) - (this->imag \* obj.imag);

tmp.imag = (this->chislo \* obj.imag) + (this->imag \* obj.chislo);

return tmp;

}

Complex operator/(const Complex& obj)

{

Complex tmp;

tmp.chislo = ((this->chislo \* obj.chislo) - (this->imag \* (-obj.imag))) / ((this->chislo \* obj.chislo) - (this->imag \* (-obj.imag)));

tmp.imag = ((this->chislo \* (-obj.imag)) + (this->imag \* obj.chislo)) / ((this->chislo \* obj.chislo) - (this->imag \* (-obj.imag)));

return tmp;

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Complex a(1, 3), b(2, 1);

(a / b).Print\_Complex();

return 0;

}