**ЗВІТ**

про виконання лабораторної роботи № 2.5

«Конструктори та перевантаження операцій для класів з композицією»

з дисципліни

«Об’єктно-орієнтоване програмування»

студентки групи IK-11

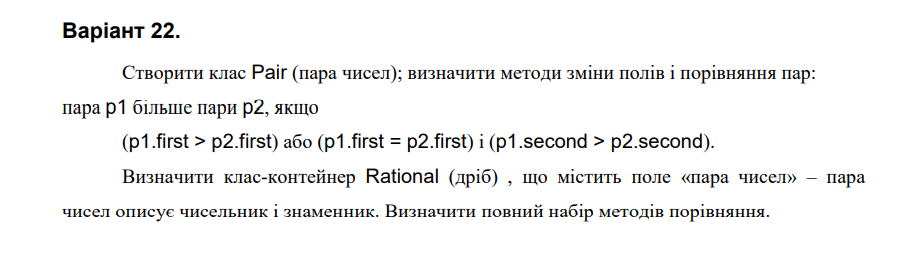
Ратушна Соломія Юріївна

**Мета роботи** : Освоїти використання конструкторів та перевантаження операцій для класів з композицією

**Умови завдання:** У всіх завданнях потрібно реалізувати по два-три класи. Один клас є основним, всі решту – допоміжні. Допоміжні класи мають бути визначені як незалежні. Об’єкти допоміжних класів мають використовуватися як поля основного класу

.

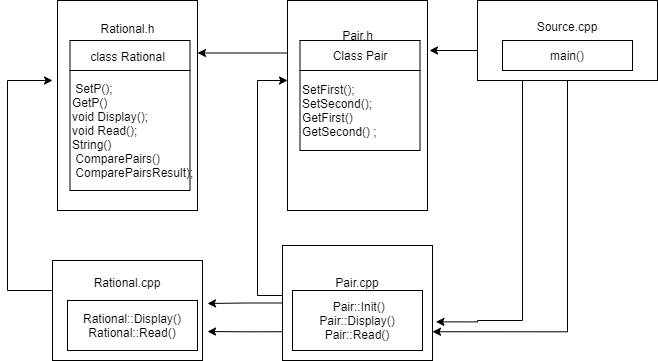
Умова варіанту:

****

**UML-діаграма класів:**

****

**Структурна схема програми:**

****

**Текст програми:**

//////////////////////////////////////////////////////////////////////////////

// Rational.h

// заголовний файл – визначення класу

#pragma once

#include "Pair.h"

class Rational

{

Pair p;

public:

int ComparePairs(const Pair p1, const Pair p2) const;

void ComparePairsResult(int result);

friend bool operator > (const Pair p1, const Pair p2);

friend bool operator < (const Pair p1, const Pair p2);

friend bool operator == (const Pair p1, const Pair p2);

};

//////////////////////////////////////////////////////////////////////////////

// Rational.cpp

// файл реалізації – реалізація методів класу

#include "Rational.h"

bool operator > (const Pair p1, const Pair p2)

{

if (p1.GetFirst() > p2.GetFirst() ||

(p1.GetFirst() == p2.GetFirst() && p1.GetSecond() > p2.GetSecond()))

return 1;

else

return 0;

}

bool operator < (const Pair p1, const Pair p2)

{

if (p1.GetFirst() < p2.GetFirst() ||

(p1.GetFirst() == p2.GetFirst() && p1.GetSecond() < p2.GetSecond()))

return 1;

else

return 0;

}

bool operator == (const Pair p1, const Pair p2)

{

if ((p1.GetFirst() == p2.GetFirst()) &&

(p1.GetSecond() == p2.GetSecond()))

return 1;

else

return 0;

}

int Rational::ComparePairs(const Pair p1, const Pair p2) const

{

if (p1 > p2)

return 1;

if (p1 < p2)

return 2;

if (p1 == p2)

return 3;

}

void Rational::ComparePairsResult(int result)

{

switch (result)

{

case 1:

cout << "\nP1 > P2" << endl;

break;

case 2:

cout << "\nP1 < P2" << endl;

break;

case 3:

cout << "\nP1 = P2" << endl;

break;

}

}

//////////////////////////////////////////////////////////////////////////////

// Source.cpp

// головний файл проекту – функція main

#include <iostream>

#pragma pack(1)

#include "Rational.h"

int main()

{

Pair a, b, c;

Rational f;

int result;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\nPair a " << endl;

cout << "(x/y)" << endl;

cin >> a;

cout << a;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\nPair b " << endl;

cout << "(x/y)" << endl;

cin >> b;

cout << b;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\nPair c " << endl;

cout << "(x/y)" << endl;

cin >> c;

cout << c;

cout << "\na and b" << endl;

result = f.ComparePairs(a, b);

f.ComparePairsResult(result);

cout << "\na and c" << endl;

result = f.ComparePairs(a, c);

f.ComparePairsResult(result);

cout << "\nb and c" << endl;

result = f.ComparePairs(b, c);

f.ComparePairsResult(result);

cout << "a++" << endl;

cout << a++ << endl;

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

cout << a-- << endl;

cout << "++a" << endl;

cout << ++a << endl;

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

cout << --a << endl;

cout << "Size of class is equal to " << sizeof(Rational) << endl;

return 0;

}

//////////////////////////////////////////////////////////////////////////////

// Pair.h

#include "Pair.h"

using namespace std;

void Pair::SetFirst(double value) { first = value; }

void Pair::SetSecond(double value) { second = value; }

double Pair::GetFirst() const { return first; }

double Pair::GetSecond() const { return second; }

Pair::Pair() { first = 0, second = 0; }

Pair::Pair(double first = 0, double second = 0)

{

this->first = first;

this->second = second;

}

Pair::Pair(const Pair& A)

{

first = A.GetFirst();

second = A.GetSecond();

}

Pair& Pair::operator = (const Pair& A)

{

first = A.first;

second = A.second;

return \*this;

}

ostream& operator << (ostream& out, const Pair& A)

{

out << string(A);

return out;

}

istream& operator >> (istream& in, Pair& A)

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Enter a first number of pair: "; in >> A.first;

cout << "Enter a second number of pair: "; in >> A.second;

return in;

}

Pair::operator string() const

{

stringstream ss;

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

ss << "First number of pair is equal to " << first << endl;

ss << "Second number of pair is equal to " << second << endl;

return ss.str();

}

Pair& Pair::operator ++ ()

{

first++;

return \*this;

}

Pair& Pair::operator -- ()

{

first--;

return \*this;

}

Pair Pair::operator ++ (int)

{

Pair t(\*this);

second++;

return t;

}

Pair Pair::operator -- (int)

{

Pair t(\*this);

second--;

return t;

}

//////////////////////////////////////////////////////////////////////////////

// Pair.h

#pragma once

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

class Pair

{

double first, second;

public:

void SetFirst(double value);

void SetSecond(double value);

double GetFirst() const;

double GetSecond() const;

Pair();

Pair(double, double);

Pair(const Pair&);

Pair& operator = (const Pair&);

friend ostream& operator << (ostream&, const Pair&);

friend istream& operator >> (istream&, Pair&);

operator string() const;

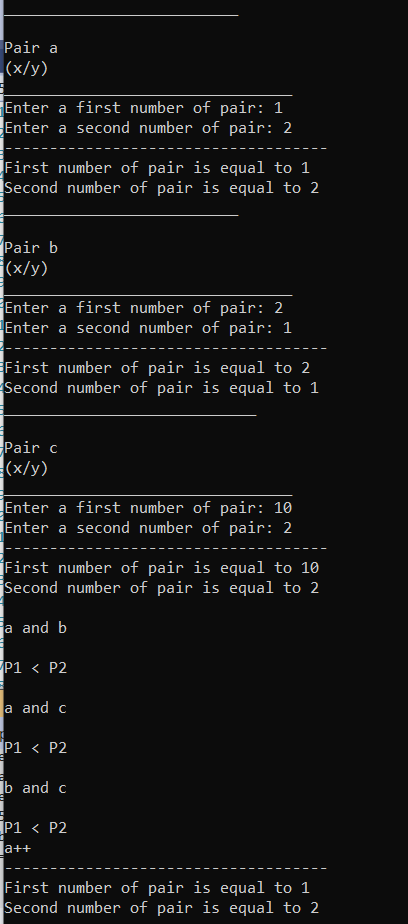
Pair& operator ++ ();

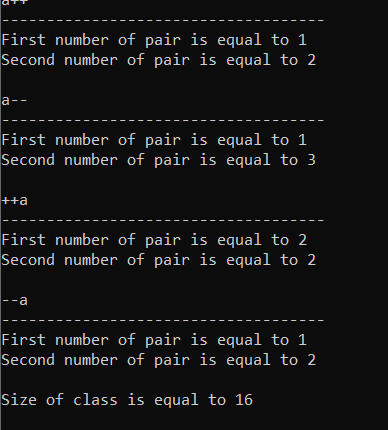
Pair& operator -- ();

Pair operator ++ (int);

Pair operator -- (int);

};





**Посилання на git-репозиторій з проектом:**

https://github.com/Solomia-Ratushna/oop1.6

**Результати unit-тесту:**

#include "../lab2.5/Pair.cpp"

#include "../lab2.5/Rational.h"

#include "../lab2.5/Rational.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(TestMethod1)

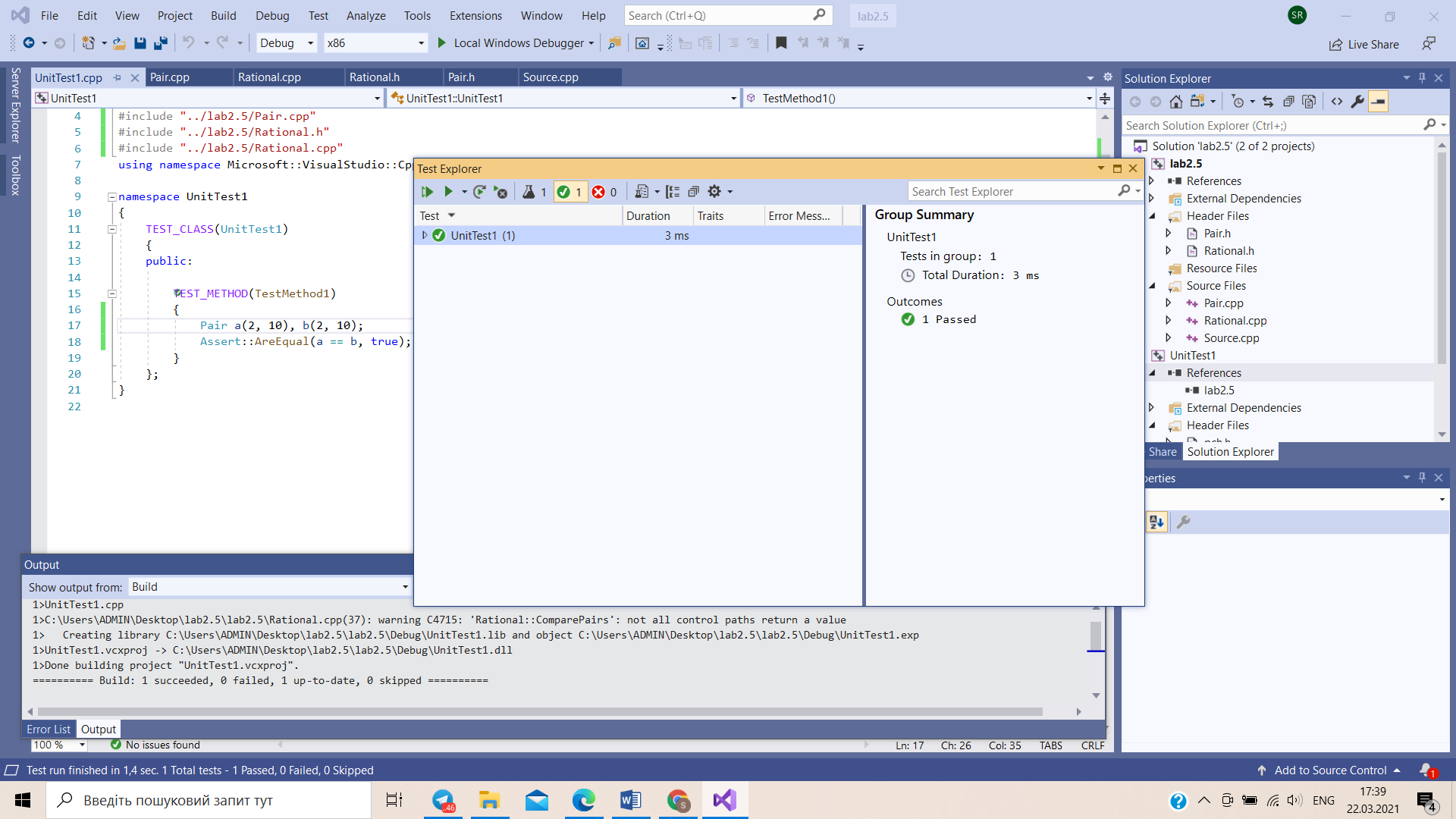
{

Pair a(2, 10), b(2, 10);

Assert::AreEqual(a == b, true);

}

};

}

**Висновки:** Я освоїла використання конструкторів та перевантаження операцій для класів з композицією