**ЗВІТ**

про виконання лабораторної роботи № 2.6

«Конструктори та перевантаження операцій для класів з вкладеними класами. Обчислення кількості об’єктів»

з дисципліни

«Об’єктно-орієнтоване програмування»

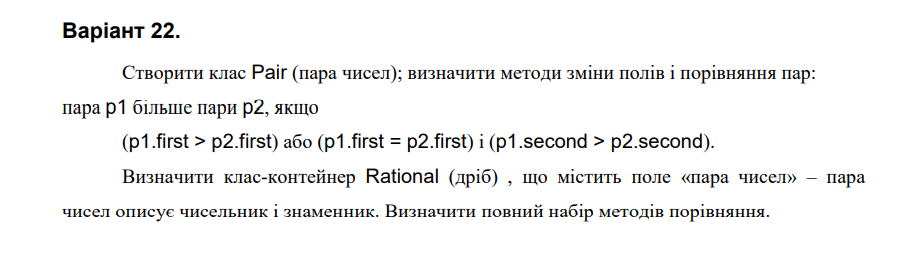
студентки групи IK-11

Ратушна Соломія Юріївна

**Мета роботи** : Освоїти використання конструкторів та перевантаження операцій для вкладених класів.

**Умови завдання:**

Виконати завдання свого варіанту Лабораторної роботи № 1.6 (Вкладені класи) з конструкторами і перевантаженням операцій (тобто, Лабораторної роботи № 2.5), використовуючи конструкцію вкладеного класу замість

****

**UML-діаграма класів:**

****

**Структурна схема програми:**

****

**Текст програми:**

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

// Rational.cpp

#include "Rational.h"

//Class Pair

int Rational::Pair::counter = 0;

int Rational::Pair::getCounter()

{

return Rational::Pair::counter;

}

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

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

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

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

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

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

{

this->first = first;

this->second = second;

Rational::Pair::counter++;

}

Rational::Pair::Pair(const Pair& A)

{

first = A.GetFirst();

second = A.GetSecond();

Rational::Pair::counter++;

}

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

{

first = A.first;

second = A.second;

return \*this;

}

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

{

out << string(A);

return out;

}

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

{

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

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

return in;

}

Rational::Pair::operator string() const

{

stringstream ss;

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

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

return ss.str();

}

Rational::Pair& Rational::Pair::operator ++ ()

{

first++;

return \*this;

}

Rational::Pair& Rational::Pair::operator -- ()

{

first--;

return \*this;

}

Rational::Pair Rational::Pair::operator ++ (int)

{

Pair t(\*this);

second++;

return t;

}

Rational::Pair Rational::Pair::operator -- (int)

{

Pair t(\*this);

second--;

return t;

}

//Class Rational

int Rational::counter = 0;

int Rational::getCounter()

{

return Rational::counter;

}

void Rational::SetP(double first, double second) { Pair(first, second); }

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

: p(first, second)

{

counter++;

}

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

{

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

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

return 1;

else

return 0;

}

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

{

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

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

return 1;

else

return 0;

}

bool operator == (const Rational::Pair p1, const Rational::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;

}

}//////////////////////////////////////////////////////////////////////////////

// Rational.h

// заголовний файл – визначення класу

#pragma once

#include <sstream>

#include <iostream>

#include <string>

using namespace std;

class Rational

{

public:

class Pair

{

double first, second;

static int counter;

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);

static int getCounter();

};

Rational(double, double);

void SetP(double first, double second);

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);

static int getCounter();

private:

Pair p;

static int counter;

};//////////////////////////////////////////////////////////////////////////////

// Source.cpp

// головний файл проекту – функція main

#include <iostream>

#pragma pack(1)

#include "Rational.h"

int main()

{

Rational::Pair a;

cout << "\n1 object of class Pair = " << Rational::Pair::getCounter();

Rational::Pair b;

cout << "\n2 objects of class Pair = " << Rational::Pair::getCounter();

Rational::Pair c;

cout << "\n3 objects of class Pair = " << Rational::Pair::getCounter();

cout << "\n0 objects of class Fraction = " << Rational::getCounter();

Rational f(0, 0);

cout << "\n1 object of class Fraction = " << Rational::getCounter();

int result;

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;

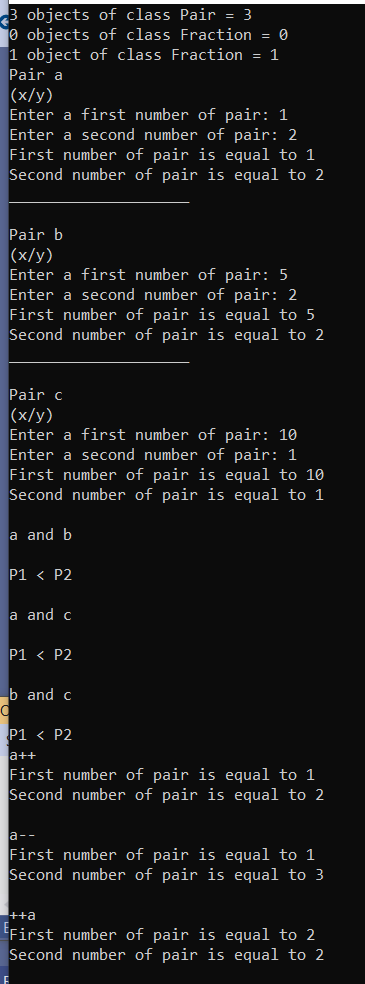
cout << "--a" << endl;

cout << --a << endl;

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

return 0;

}

****

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

**https://github.com/Solomia-Ratushna/oop1.5**

**Результати unit-тесту:**

#include "pch.h"

#include "CppUnitTest.h"

#include "../lab2.6/Rational.h"

#include "../lab2.6/Rational.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(TestMethod1)

{

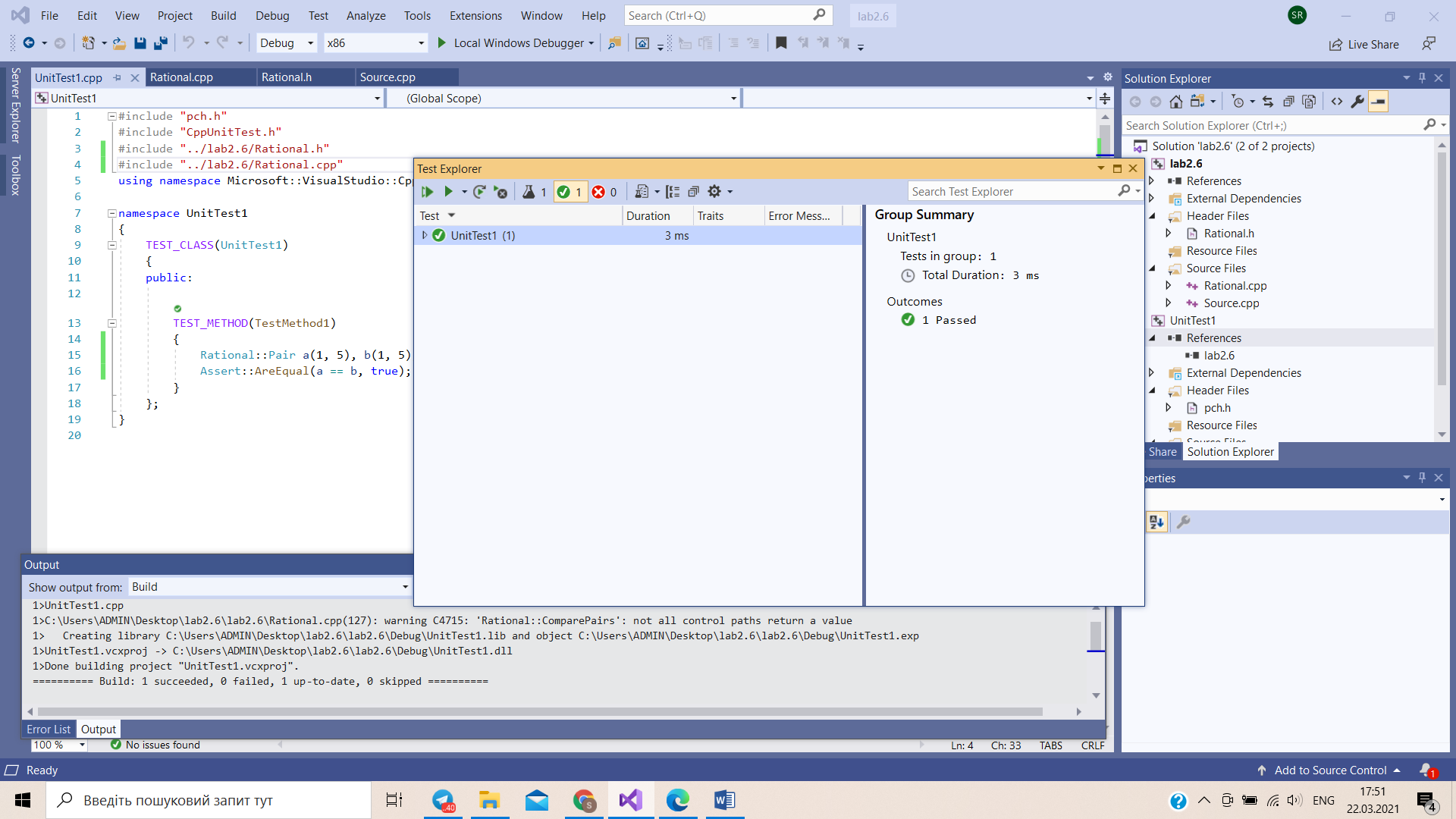
Rational::Pair a(1, 5), b(1, 5);

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

}

};

}



**Висновки:** Яосвоїла використання конструкторів та перевантаження операцій для вкладених класів.