ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ

за курсом «Об’єктно-орієнтоване програмування»

студента групи ПЗ-21-2

Гудимова Дениса Сергійовича

кафедра математичного забезпечення ЕОМ, ДНУ

2022/2023 навч.р.

1. Постановка задачі:

1 Реалізація методів ініціалізації (конструктор по замовчуванню та конструктор з

параметрами), копіювання (конструктор копіювання).

2. Індексації (перевантаження []).

3. Присвоювання (перевантаження =).

4. Перевантаження потокового введення/виведення таким чином, що повинно працювати введення як з файлу, так і з консолі, виведення в файл та консоль. Демонструвати, як стан об’єкту може бути збережений на диск та відновлений зі збереженого стану.

Раціональні числа: різні форми зовнішнього представлення, точні обчислення.

Реалізувати клас для збереження раціонального числа (чисельник, знаменник).

Виконати перевантаження арифметичних операцій і операцій приведення типів.

Передбачити автоматичне перетворення раціонального числа в нескорочуваний дріб.

2. Опис розв’язку.

Було створено клас Раціональних чисел. У класі реалізовано 2 закриті поля: чисельник та знаменник та 3 закриті методи, а саме ввод, вівід та скорочення числа. Також 3 відкритих конструктори: без параметрів, з параметрами та копіювання. Були перегруженні оператор квадратних скобок[] для доступу до полів класу, оператор дорівнює= для присвоювання екзепляру класа іниший екзепляр, оператор круглих скобок() для приведення до типу **float**, а також оператори плюс+, минус-, помножити\*, поділити/ для арифметики раціональних чисел.

3.Код розв’язку задачі

**Конструктор без параметрів:**

Вигляд:

Rational();

Реалізація:

Rational::Rational() : numerator(0), denuminator(1) {};

Звернення:

Rational a;

Rational b;

Rational k;

**Конструктор з параметрами:**

Вигляд:

Rational(int, int);

Реалізація:

Rational::Rational(int num, int denum) {

if (num < 0 && denum < 0) {

num = -num;

denum = -denum;

}

numerator = num;

denuminator = denum;

reduce();

}

Звернення: Rational a2(10, 4);

**Конструктор копіювання:**

Вигляд:

Rational(Rational&);

Реалізація:

Rational::Rational(Rational& rat\_num) {

numerator = rat\_num.numerator;

denuminator = rat\_num.denuminator;

}

Звернення:

Rational b2(a2);

**Перевантаження оператора []:**

Вигляд:

int& operator[]( int) ;

Реалізація:

int& Rational::operator [] (int index) {

if (index == 0) {

return numerator;

}else if(index == 1){

return denuminator;

}

else {

cout << "Wrong index\n";

}

}

Звернення:

cout << "Numerator for first ratioanl number: " << a[0] << endl;

cout << "Denuminator for first ratioanl number: " << a[1] << endl << endl;

**Перевантаження оператора = :**

Вигляд:

Rational operator = (const Rational&) ;

Реалізація:

Rational Rational::operator=(const Rational& ratioanl) {

numerator = ratioanl.numerator;

denuminator = ratioanl.denuminator;

return \*this;

}

Звернення:

k = a - b;

k = a \* b;

k = a / b;

**Перевантаження оператора Float() для приведення типу до float:**

Вигляд:

operator float() const;

Реалізація:

Rational::operator float() const {

return static\_cast<float>(numerator) / denuminator;

}

Звернення:

float type\_cast;

type\_cast = a;

**Перевантаження оператора \* для арефмитичних дій:**

Вигляд:

Rational operator\*(const Rational&) const;

Реалізація:

Rational Rational::operator\*(const Rational& ratioanl) const {

int mult\_num, denum;

mult\_num = numerator \* ratioanl.numerator;

denum = denuminator \* ratioanl.denuminator;

Rational multiplication(mult\_num, denum);

return multiplication;

}

Звернення:

k = a \* b;

cout << "Multiplication result of a + b:" << k << endl;

**Перевантаження оператора + для арефмитичних дій:**

Вигляд:

Rational operator+(const Rational&) const;

Реалізація:

Rational Rational::operator+(const Rational& ratioanl) const {

int sum\_num,denum;

sum\_num = (numerator\* ratioanl.denuminator) + (ratioanl.numerator\* denuminator);

denum = denuminator \* ratioanl.denuminator;

Rational sum(sum\_num, denum);

return sum;

}

Звернення:

k = a + b;

cout << "Summa result of a + b:" << k << endl;

**Перевантаження оператора - для арефмитичних дій:**

Вигляд:

Rational operator-(const Rational&) const;

Реалізація:

Rational Rational::operator-(const Rational& ratioanl) const {

int dif\_num, denum;

dif\_num = (numerator \* ratioanl.denuminator) - (ratioanl.numerator \* denuminator);

denum = denuminator \* ratioanl.denuminator;

Rational difference(dif\_num, denum);

return difference;

}

Звернення:

k = a - b;

cout << "Difference result of a - b:" << k << endl;

**Перевантаження оператора / для арефмитичних дій:**

Вигляд:

Rational operator/(const Rational&) const;

Реалізація:

Rational Rational::operator/(const Rational& ratioanl) const {

int div\_num, denum;

div\_num = numerator \* ratioanl.denuminator;

denum = denuminator \* ratioanl.numerator;

Rational division(div\_num, denum);

return division;

}

Звернення:

k = a / b;

cout << "Division result of a - b:" << k << endl;

**Перевантаження оператора >> для введення у потік**

Вигляд:

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

Реалізація:

istream& operator>>(istream& stream\_in, Rational& ratioanl) {

cout << "Enter rational value (a/b): ";

stream\_in >> ratioanl.numerator;

stream\_in.ignore();

stream\_in >> ratioanl.denuminator;

if (ratioanl.numerator < 0 && ratioanl.denuminator < 0) {

ratioanl.numerator = -ratioanl.numerator;

ratioanl.denuminator = -ratioanl.denuminator;

}

ratioanl.reduce();

cout << endl;

return stream\_in;

}

Звернення:

cout << "Enter first ratioanl number:\n";

cin >> a;

cout << "\nEnter second ratioanl number:\n";

cin >> b;

**Перевантаження оператора << для виведення з потоку:**

Вигляд:

friend std::ostream& operator<<(std::ostream&, Rational&);

Реалізація:

ostream& operator<<(ostream& stream\_out, Rational& ratioanl) {

Звернення:

stream\_out << "Rational number: " << ratioanl.numerator << "/" << ratioanl.denuminator << endl;

return stream\_out;

}

Звернення:

cout << "Your first " << a;

cout << "Your second " << b;

**Метод для скорочення дробу:**

Вигляд:

void reduce();

Реалізація:

void Rational::reduce() {

int min, max;

numerator > denuminator ? (min = abs(denuminator), max = abs(numerator)) : (min = abs(numerator), max = abs(denuminator));

for (int delimeter = 1; delimeter <= min / 2; delimeter++) {

if (min % delimeter == 0 && max % (min / delimeter) == 0) {

numerator /= (min / delimeter);

denuminator /= (min / delimeter);

return;

}

}

}

Звернення:

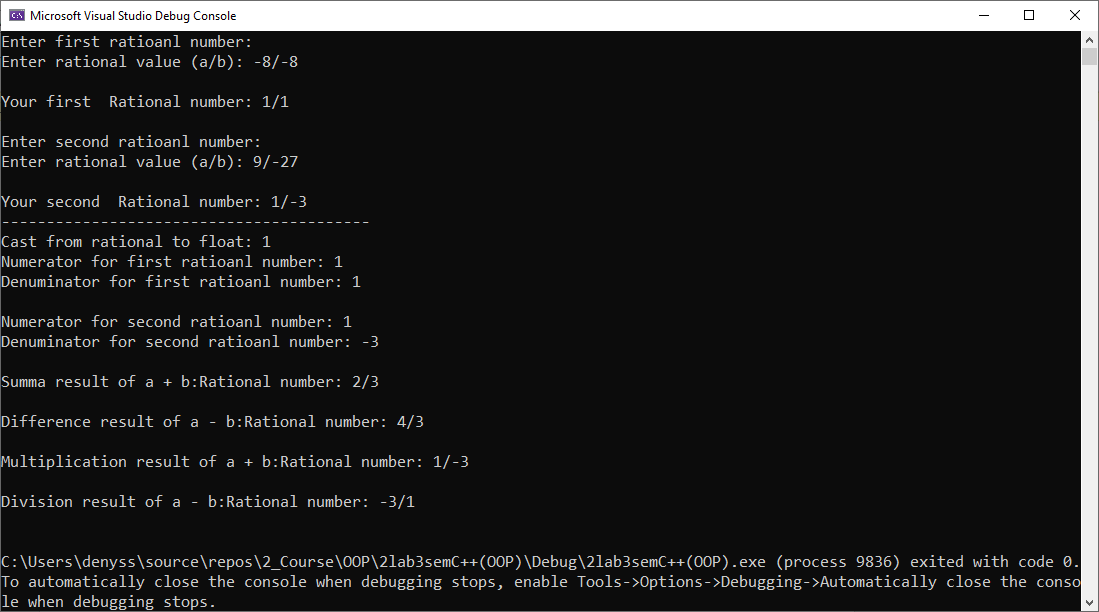
Відбувається при введені чи у конструкторі

reduce();

4. Опис інтерфейсу (керівництво користувача)

На початку програми потрібно ввести 2 раціональних числа

5. Опис тестових прикладів

При вводі відемного чисельника та знаменик Вони стають додатніми і якщо число можна скоротити то воно скоротиться6. Аналіз помилок (опис усунення зауважень)

-