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 класа-обгортки з розумним доступом до методів і полів класу

(перевантаження ->),

2. Реалізація псевдозмінних (забезпечення можливості використання лівосторонніх

функцій вигляду: f(x)=const). Спосіб реалізації дивитись в індивідуальному завданні.

3. Створення та використання файла бібліотеки (\*.LIB).

4. Повторне використання класів без їх перекомпіляції (ReUse).

5. Демонстрація поліморфізму.

2. Опис розв’язку.

Було створено клас Розумного покажчика для раціональних чисел. У класі реалізовано 1 закрите поля: покажчик на раціональне число. Також 1 конструктор, деструктор. Були перегруженні оператор "стрілочка" для доступу через адресу до полів класу, оператор круглих дужок для доступу до полів класу і оператори взяти та вивести з потоку.

3.Код розв’язку задачі

**Конструктор з параметрами:**

Вигляд:

PointRational(Rational\*)

Реалізація:

PointRational::~PointRational() {

if (ptr == nullptr) {

return;

}

std::cout << "Rational has deleted\n";

delete ptr;

}

Звернення: PointRational second(new Rational());

**Деконструктор**

Вигляд:

~PointRational();

Реалізація:

PointRational::~PointRational() {

if (ptr == nullptr) {

return;

}

std::cout << "Rational has deleted\n";

delete ptr;

}

Звернення:

При завершенні програми

**Перевантаження оператора ->:**

Вигляд:

Rational\* operator->();

Реалізація:

Rational\* PointRational::operator->() {

return ptr;

}

Звернення:

second->set\_numerator(200);

second->set\_denuminator(300);

**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:

first(0) = 30;

first(1) = 60;

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

Вигляд:

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

Реалізація:

std::istream& operator>>(std::istream& stream\_in, PointRational& ratioanl) {

stream\_in >> (\*ratioanl.ptr);

return stream\_in;

}

Звернення:

cin >> mali;

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

Вигляд:

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

Реалізація:

std::ostream& operator<<(std::ostream& stream\_out, PointRational& ratioanl) {

stream\_out << (\*ratioanl.ptr);

return stream\_out;

}

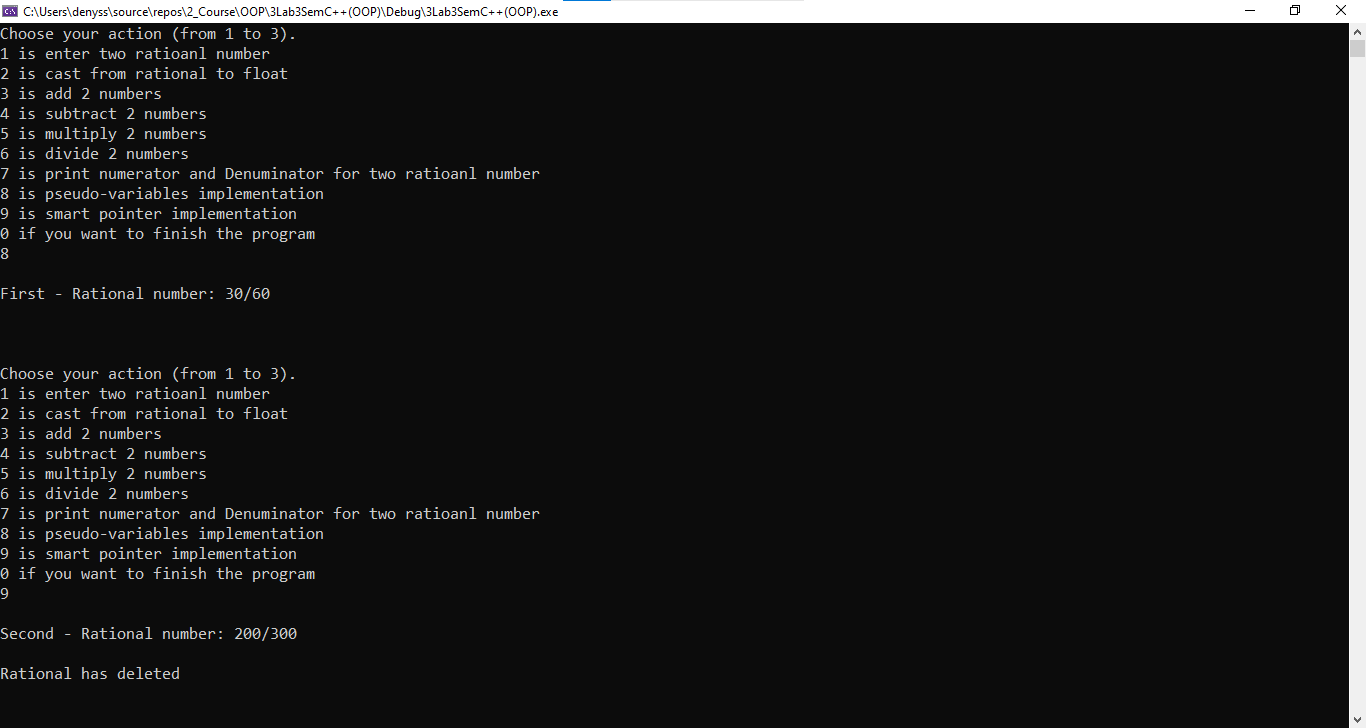
Звернення:

cout << "Second - " << second << endl;

4. Опис інтерфейсу (керівництво користувача)

На початку програми потрібно вибрати дію із запропвнованих, а для виходу ввести 0

5. Опис тестових прикладів



6. Аналіз помилок (опис усунення зауважень)

-