**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №5**

Выполнил: студент группы ПИ-3-21

Жайлообек уулу Улукбек

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

Написать программу на C++, которая реализует класс **Int** для работы с целыми числами. Класс должен перегружать арифметические операторы (**+**, **-**, **\***, **/**) для выполнения операций над объектами класса **Int** и числами. Также необходимо реализовать операторы инкремента (**++**) в префиксной и постфиксной форме. При выполнении арифметических операций и инкремента необходимо проверять на переполнение типа **int**. В случае переполнения должно генерироваться исключение. Программа должна продемонстрировать использование класса **Int** через выполнение различных арифметических операций и операций инкремента.

**Исходный код программы на C++:**

#include <iostream>

#include <limits>

using namespace std;

class Int {

private:

int value; // Целочисленное значение

public:

Int(int value) : value(value) {} // Конструктор для инициализации значения

// Перегрузка оператора сложения

Int operator+(const Int& other) const { //явное приведение типов данных

long long result = static\_cast<long long>(this->value) + other.value; // Используем long long для предотвращения переполнения

if (result > numeric\_limits<int>::max() || result < numeric\_limits<int>::min()) {

throw overflow\_error("Integer overflow");

}

return Int(static\_cast<int>(result));

}

Int operator-(const Int& other) const {

long long result = static\_cast<long long>(this->value) - other.value;

if (result > numeric\_limits<int>::max() || result < numeric\_limits<int>::min()) {

throw overflow\_error("Integer overflow");

}

return Int(static\_cast<int>(result));

}

// Перегрузка оператора умножения

Int operator\*(const Int& other) const {

long long result = static\_cast<long long>(this->value) \* other.value;

if (result > numeric\_limits<int>::max() || result < numeric\_limits<int>::min()) {

throw overflow\_error("Integer overflow");

}

return Int(static\_cast<int>(result));

}

// Перегрузка оператора деления

Int operator/(const Int& other) const {

if (other.value == 0) {

throw invalid\_argument("Division by zero");

}

return Int(this->value / other.value);

}

// Перегрузка оператора префиксного инкремента

Int& operator++() {

\*this = \*this + Int(1); // Используем перегруженный оператор сложения

return \*this;

}

// Перегрузка оператора постфиксного инкремента

Int operator++(int) {

Int temp = \*this;

++(\*this); // Используем перегруженный оператор префиксного инкремента

return temp;

}

// геттер

int getValue() const {

return value;

}

};

int main() {

Int a(2147483647); // Максимальное значение int

Int b(1);

try {

Int result = a + b; // Попытка сложения, которое превысит максимальное значение int

cout << "сумма " << result.getValue() << endl;

}

catch (overflow\_error& e) {

cout << "Результат выходит за границы типа int: " << e.what() << endl;

}

return 0;

}

**Тесты:**

****

****

# **Задание №2**

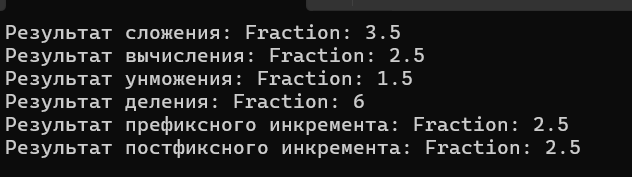
**Постановка задачи:**

Написать программу на C++, которая реализует класс **Fraction** для работы с дробями. Класс должен перегружать арифметические операторы (**+**, **-**, **\***, **/**) для выполнения операций над объектами класса **Fraction** и числами. Также необходимо реализовать операторы инкремента (**++**) в префиксной и постфиксной форме. При выполнении арифметических операций необходимо проверять деление на ноль и генерировать исключение в случае деления на ноль. Программа должна продемонстрировать использование класса **Fraction** через выполнение различных арифметических операций и операций инкремента.

**Исходный код программы на C++:**

#include <iostream>

**Тесты:**



# **Задание №3**

**Постановка задачи:**

Написать программу на C++, которая реализует класс **Rectangle** для работы с прямоугольниками и квадратами. Класс должен иметь метод **rect\_area()**, который перегружается двумя способами: для вычисления площади прямоугольника (при передаче двух аргументов - длины и ширины) и для вычисления площади квадрата (при передаче одного аргумента - стороны квадрата). Программа должна продемонстрировать использование класса **Rectangle** через вычисление площади прямоугольника и квадрата.

**Исходный код программы на C++:**

\*/

#include <iostream>

class Rectangle {

private:

double length;

double width;

public:

// Конструктор

Rectangle(double l, double w) : length(l), width(w) {}

// Метод для вычисления площади прямоугольника (первая версия)

double rect\_area(double l, double w) {

return l \* w;

}

// Перегрузка метода для вычисления площади прямоугольника (вторая версия)

double rect\_area(double side) {

return side \* side;

}

// Метод для вывода информации о прямоугольнике

void display() {

std::cout << "Length: " << length << ", Width: " << width << std::endl;

}

};

int main() {

// Создание объекта прямоугольника

Rectangle rect(5.0, 3.0);

// Вычисление площади прямоугольника с передачей обоих размеров

double area1 = rect.rect\_area(5.0, 3.0);

std::cout << "Площадь прямоугольника: " << area1 << std::endl;

// Вычисление площади квадрата с передачей только одного размера

double area2 = rect.rect\_area(4.0);

std::cout << "Площадь квадрата: " << area2 << std::endl;

return 0;

}}

**Тесты:**

