**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

Алтынбеков Даурен

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;

public:

Int(int val = 0) : value(val) {}

Int operator+(const Int& other) const {

long double result = static\_cast<long double>(value) + other.value;

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

throw overflow\_error("Arithmetic overflow occurred.");

}

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

}

Int operator-(const Int& other) const {

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

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

throw overflow\_error("Arithmetic overflow occurred.");

}

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

}

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

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

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

throw overflow\_error("Arithmetic overflow occurred.");

}

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

}

Int operator/(const Int& other) const {

if (other.value == 0) {

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

}

return Int(value / other.value);

}

Int operator++(int) {

Int temp(\*this);

++(\*this);

return temp;

}

Int& operator++() {

long double result = static\_cast<long double>(value) + 1;

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

throw overflow\_error("Arithmetic overflow occurred.");

}

value = static\_cast<int>(result);

return \*this;

}

friend ostream& operator<<(ostream& os, const Int& num) {

os << num.value;

return os;

}

};

int main() {

try {

Int a(2147483646);

Int b(1);

Int c = a + b;

cout << "Result of addition: " << c << endl;

Int d = a \* b;

cout << "Result of multiplication: " << d << endl;

Int e = ++a;

cout << "Prefix increment: " << e << endl;

Int f = b++;

cout << "Postfix increment: " << f << endl;

/\*Int g = Int(numeric\_limits<int>::max()) + Int(1);

cout << "Result of addition (should cause overflow): " << g << endl;\*/

}

catch (const exception& e) {

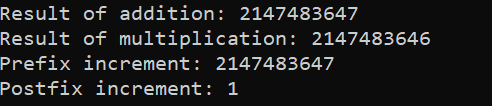
cerr << "Exception caught: " << e.what() << endl;

}

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>

using namespace std;

class Fraction {

private:

double chislo;

public:

Fraction(double value = 0) : chislo(value) {}

Fraction operator+(const Fraction& other) const {

return Fraction(chislo + other.chislo);

}

Fraction operator-(const Fraction& other) const {

return Fraction(chislo - other.chislo);

}

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

return Fraction(chislo \* other.chislo);

}

Fraction operator/(const Fraction& other) const {

if (other.chislo == 0) {

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

}

return Fraction(chislo / other.chislo);

}

Fraction operator+(double value) const {

return Fraction(chislo + value);

}

Fraction operator-(double value) const {

return Fraction(chislo - value);

}

Fraction operator\*(double value) const {

return Fraction(chislo \* value);

}

Fraction operator/(double value) const {

if (value == 0) {

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

}

return Fraction(chislo / value);

}

Fraction& operator++() {

chislo++;

return \*this;

}

Fraction operator++(int) {

Fraction temp(\*this);

++(\*this);

return temp;

}

friend ostream& operator<<(ostream& os, const Fraction& frac) {

os << frac.chislo;

return os;

}

};

int main() {

try {

Fraction a(3.0 / 4), b(2.0 / 5);

Fraction c = a + b;

cout << "Result of addition: " << c << endl;

Fraction d = a \* 2.0;

cout << "Result of multiplication: " << d << endl;

Fraction e = ++a;

cout << "Prefix increment: " << e << endl;

Fraction f = b++;

cout << "Postfix increment: " << f << endl;

}

catch (const exception& e) {

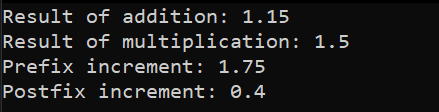
cerr << "Exception caught: " << e.what() << endl;

}

return 0;

}

**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>

using namespace std;

class Rectangle {

private:

double length;

double width;

public:

Rectangle(double l = 0, double w = 0) : length(l), width(w) {}

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

return l \* w;

}

double rect\_area() {

return length \* width;

}

double rect\_area(double side) {

return side \* side;

}

};

int main() {

Rectangle rect(4, 6);

cout << "Area of rectangle: " << rect.rect\_area() << endl;

double side = 5;

cout << "Area of square: " << rect.rect\_area(side) << endl;

return 0;

}

**Тесты:**

