**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

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

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Дисциплина «**Объектно-Ориентированное Программирование**»

**Отчет**

**Лабораторная работа №5**

Выполнил: студент группы ПИ-2-21

Газиев Давид

Проверил: Мусабаев Э. Б.

**Бишкек 2024**

**Задание 1**

Создайте и откомпилируйте класс **Int**. Перегрузите четыре бинарных целочисленных арифметических операции (+, -, \*, /) и унарные операции постфиксной и префиксной форм инкремента с помощью внутренней операторной функции так, чтобы их можно было использовать для операций с объектами класса **Int**.

Если результат какой-либо из них выходит за границы типа **int** (в 32-битной системе), имеющее значения от **2 147 483 648** до **-2 147 483 648**, то операция должна послать сообщение об ошибке и завершить программу. Для выявления ошибки арифметического переполнения используйте концепцию **исключения**.

**Код программы**

#include <iostream>

#include <stdexcept>

using namespace std;

class Int {

private:

long double value;

public:

Int(long double val = 0) : value(val) {}

Int operator+(const Int& other) const {

long double result = value + other.value;

if (result > INT\_MAX || result < INT\_MIN) {

throw overflow\_error("Выполнение оператора сложения\nВыход за границы целого числа!");

}

return Int(result);

}

Int operator-(const Int& other) const {

long double result = value - other.value;

if (result > INT\_MAX || result < INT\_MIN) {

throw overflow\_error("Выполнение оператора вычитания\nВыход за границы целого числа!");

}

return Int(result);

}

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

long double result = value \* other.value;

if (result > INT\_MAX || result < INT\_MIN) {

throw overflow\_error("Выполнение оператора умножения\nВыход за границы целого числа!");

}

return Int(result);

}

Int operator/(const Int& other) const {

if (other.value == 0) {

throw runtime\_error("Деления на ноль!");

}

long double result = value / other.value;

if (result > INT\_MAX || result < INT\_MIN) {

throw overflow\_error("Выполнение оператора деления\nВыход за границы целого числа!");

}

return Int(result);

}

Int& operator++() {

if (value >= INT\_MAX) {

throw overflow\_error("Выполнение унарного оператора сложения\nВыход за границы целого числа!");

}

++value;

return \*this;

}

Int operator++(int) {

Int temp = \*this;

++(\*this);

return temp;

}

Int& operator--() {

if (value <= INT\_MIN) {

throw overflow\_error("Выполнение унарного оператора вычитания\nВыход за границы целого числа!");

}

--value;

return \*this;

}

Int operator--(int) {

Int temp = \*this;

--(\*this);

return temp;

}

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

out << num.value;

return out;

}

};

int main() {

setlocale(LC\_ALL, "ru");

try {

Int a = LLONG\_MAX;

Int b = 1;

Int sum = a + b;

cout << "Сумма: " << sum << endl;

Int diff = a - b;

cout << "Разность: " << diff << endl;

Int prod = a \* b;

cout << "Произведение: " << prod << endl;

Int div = a / b;

cout << "Деление: " << div << endl;

Int c = INT\_MAX - 1000;

cout << "c до постфиксного инкремента: " << c++ << endl;

cout << "c после постфиксного инкремента: " << c << endl;

cout << "c до префиксного инкремента: " << ++c << endl;

Int d = INT\_MIN + 1000;

cout << "d до постфиксного декремента: " << d-- << endl;

cout << "d после постфиксного декремента: " << d << endl;

cout << "d до префиксного декремента: " << --d << endl;

}

catch (const exception& e) {

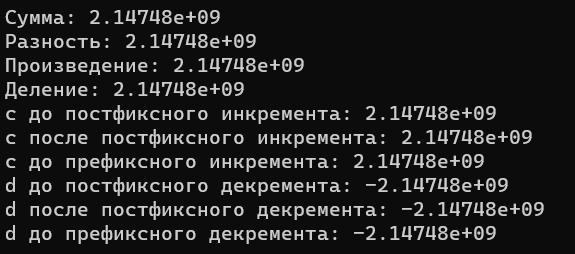
cerr << "Ошибка: " << e.what() << endl;

}

return 0;

}

**Проверка выполнения**

****

**Блок схема**



**Задание 2**

Опишите класс **fraction**, у которого есть одно закрытое целочисленное поле **chislo** типа **double**. Перегрузите для этого класса арифметические операции **сложения**, **вычитания**, **умножения** и **деления** так, чтобы они могли оперировать как с объектами класса, так и с числами (то есть выполнять, например, не только действие 3/4 +2/5, но и 1/2 + 4 или 2\* 5/6). Также перегрузите унарную операцию инкремента в префиксной или постфиксной форме увеличения дроби. Продемонстрируйте работу класса. Используйте конструктор по умолчанию и конструктор с одни аргументом для инициализации поля класса.

**Код программы**

#include <iostream>

#include <stdexcept>

using namespace std;

class Fraction {

private:

double value;

public:

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

Fraction operator+(const Fraction& other) const {

return Fraction(value + other.value);

}

Fraction operator-(const Fraction& other) const {

return Fraction(value - other.value);

}

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

return Fraction(value \* other.value);

}

Fraction operator/(const Fraction& other) const {

if (other.value == 0) {

throw runtime\_error("Деление на ноль!");

}

return Fraction(value / other.value);

}

friend Fraction operator+(double num, const Fraction& frac) {

return Fraction(frac.value + num);

}

friend Fraction operator-(double num, const Fraction& frac) {

return Fraction(frac.value - num);

}

friend Fraction operator\*(double num, const Fraction& frac) {

return Fraction(frac.value \* num);

}

friend Fraction operator/(double num, const Fraction& frac) {

if (num == 0) {

throw runtime\_error("Деление на ноль!");

}

return Fraction(frac.value / num);

}

Fraction& operator++() {

value++;

return \*this;

}

Fraction operator++(int) {

Fraction temp = \*this;

++(\*this);

return temp;

}

void display() const {

cout << value;

}

};

int main() {

setlocale(LC\_ALL, "ru");

Fraction a(3.0);

Fraction b(0.5);

cout << "Начальные значения:\na = ";

a.display();

cout << "\nb = ";

b.display();

cout << "\n\n";

Fraction c = a + b; cout << "a + b = ";

c.display();

cout << endl;

Fraction d = a - 2;

cout << "a - 2 = ";

d.display();

cout << endl;

Fraction e = 2 \* b;

cout << "2 \* b = ";

e.display();

cout << endl;

Fraction f = a / 4;

cout << "a / 4 = ";

f.display();

cout << endl;

Fraction g = ++c;

cout << "++c = ";

g.display();

cout << endl;

Fraction h = d++;

cout << "d++ = ";

h.display();

cout << endl;

Fraction j = a / 1 + 15;

cout << "a / 1 + 15 = ";

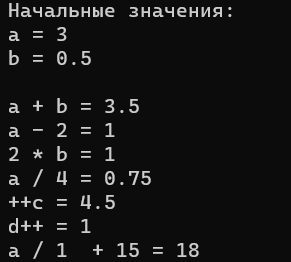
j.display();

cout << endl;

return 0;

}

**Проверка выполнения**

****

**Блок схема**



**Задание 3**

Создать класс, в котором перегружается метод **rect\_area()**.

Этот метод возвращает площадь прямоугольника. В этой программе метод **rect\_area()** перегружается двумя способами. В первом — методу передаются оба размера фигуры. Эта версия используется для прямоугольника. Однако, в случае квадрата необходимо задавать только один аргумент, поэтому вызывается вторая версия метода **rect\_area().**

**Код программы**

#include <iostream>

using namespace std;

class Rectangle {

private:

double length;

double width;

public:

Rectangle(double len = 0, double wid = 0) : length(len), width(wid) {}

double rect\_area(double len, double wid) {

return len \* wid;

}

double rect\_area(double side) {

return side \* side;

}

};

int main() {

setlocale(LC\_ALL, "ru");

Rectangle rect;

double length, width;

cout << "Введите длину прямоугольника: ";

cin >> length;

cout << "Введите ширину прямоугольника: ";

cin >> width;

cout << "Площадь прямоугольника: " << rect.rect\_area(length, width) << endl;

double side;

cout << "Введите длину стороны квадрата: ";

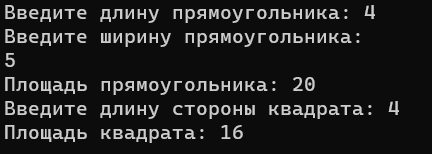
cin >> side;

cout << "Площадь квадрата: " << rect.rect\_area(side) << endl;

return 0;

}

**Проверка выполнения**

****

**Блок схема**

