МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа № 9:**

"Обработка исключительных ситуаций."

Выполнила: студентка гр.РИС-23-3Б

Мокроусова Ангелина Андреевна

Проверила: доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2024

**Постановка задачи:**

Общая часть:

* Реализовать класс, перегрузить для него операции, указанные в варианте.
* Определить исключительные ситуации.
* Предусмотреть генерацию исключительных ситуаций.

Задание варианта:

Класс-контейнер СПИСОК с ключевыми значениями типа int.

Реализовать операции:

[] – доступа по индексу

+ вектор – добавление списка b к списку а (a + b)

Вариант реализации: 1, 2

**Анализ задачи:**

Создание иерархии классов ошибок для обработки различных типов ошибок, таких как неверный размер, пустой список, переполнение, неверный индекс и т.д. Классы ошибок наследуются друг от друга, что позволяет обрабатывать ошибки на разных уровнях абстракций.

Класс списка (список) реализует шаблонный список с методами доступа по индексу, операторами сложения списков и добавлением элемента в начало списка. Также в класседолжны быть реализованы операторы ввода и вывода для таблиц.

В функции main() осуществляется ввод двух списков, их составление, добавление элемента в начало списка, а также доступ к элементу по индексу с обработкой возможных ошибок. В случае возникновения ошибки выводится сообщение.

**Код на С++ с комментариями:**

// error.h

#pragma once

#include "list.h"

#include <string>

#include <iostream>

using namespace std;

class BasicError {

public:

BasicError() {}

virtual ~BasicError() {}

void PrintError() {

cout << msg << endl;

}

protected:

string msg;

};

class WrongSizeError : public BasicError {

public:

WrongSizeError() { msg = "Ошибка неверного размера: "; }

virtual ~WrongSizeError() {}

};

class EmptyColletionError : public WrongSizeError {

public:

EmptyColletionError() { msg += "Список пуст"; }

};

class OverflowError : public WrongSizeError {

public:

OverflowError() { msg += "Список не может содержать более 30 элементов"; }

};

class WrongIndexError : public BasicError {

public:

WrongIndexError() { msg = "Ошибка индекса: "; }

virtual ~WrongIndexError() {}

};

class IndexOutOfRangeError : public WrongIndexError {

public:

IndexOutOfRangeError() { msg += "Индекс больше размера списка"; }

~IndexOutOfRangeError() {}

};

class NegativeIndexError : public WrongIndexError {

public:

NegativeIndexError() { msg += "Индекс отрицательный"; }

~NegativeIndexError() {}

};

// list.h

#pragma once

#include "error.h"

#include <string>

#include <iostream>

using namespace std;

const int MAX\_SIZE = 30;

// структура узла списка

struct Node

{

Node\* next = nullptr;

Node\* prev = nullptr;

int Data;

};

// шаблонный класс списка

template<typename t>

class list

{

public:

// конструктор

list() : elements(nullptr), num\_elements(0) {}

// конструктор копирования

list(const list& other) : elements(new t[other.num\_elements]), num\_elements(other.num\_elements)

{

for (int i = 0; i < num\_elements; i++)

elements[i] = other.elements[i];

}

// оператор присваивания

list& operator=(const list& other)

{

if (this != &other)

{

delete[] elements;

elements = new t[other.num\_elements];

num\_elements = other.num\_elements;

for (int i = 0; i < num\_elements; i++)

elements[i] = other.elements[i];

}

return \*this;

}

// деструктор

~list() { delete[] elements; }

// доступ по индексу

t& operator[](int index)

{

if (index >= num\_elements) throw IndexOutOfRangeError();

if (index < 0) throw NegativeIndexError();

return elements[index];

}

list& operator--()

{

num\_elements--;

return \*this;

}

// ввод списка

friend istream& operator>>(istream& in, list& list)

{

int num\_elements;

in >> num\_elements;

if (num\_elements > MAX\_SIZE) throw OverflowError();

if (num\_elements == 0) throw EmptyColletionError();

list.elements = new t[num\_elements];

list.num\_elements = num\_elements;

for (int i = 0; i < num\_elements; i++)

in >> list.elements[i];

return in;

}

// вывод списка

friend ostream& operator<<(ostream& out, const list& list)

{

for (int i = 0; i < list.num\_elements; i++)

out << list.elements[i] << " ";

return out;

}

// перегрузка +

list<t> operator+(const list<t>& other) {

list<t> res;

res.num\_elements = num\_elements + other.num\_elements;

res.elements = new t[res.num\_elements];

for (int i = 0; i < num\_elements; i++) {

res.elements[i] = elements[i];

}

for (int i = 0; i < other.num\_elements; i++) {

res.elements[i + num\_elements] = other.elements[i];

}

return res;

}

// перегрузка +

list<t> operator+(t num) {

list<t> res;

res.num\_elements = num\_elements + 1;

res.elements = new t[res.num\_elements];

res.elements[0] = num;

for (int i = 1; i < num\_elements + 1; i++) {

res.elements[i] = elements[i - 1];

}

return res;

}

private:

t\* elements;

int num\_elements;

};

// lab9\_main.cpp

#include "list.h"

#include "error.h"

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "ru");

int a = 0;

while (true) {

try

{

// создание списков

list<int> list1;

list<int> list2;

cout << "===============================================================================================" << endl;

// ввод списков

cout << "Введите количество элементов первого списка: ";

cin >> list1;

cout << "Введите количество элементов второго списка: ";

cin >> list2;

cout << "===============================================================================================" << endl;

// сложение списков

list<int> result = list1 + list2;

cout << "Сумма списков: " << result << endl;

cout << "===============================================================================================" << endl;

// сложение списка с числом

cout << "Введите элемент, который нужно добавить в начало списка: ";

cin >> a;

result = result + a;

cout << "Список после добавления элемента: " << result << endl;

cout << "===============================================================================================" << endl;

cout << "Введите индекс нужного элемента: ";

cin >> a;

cout << "Нужный элемент: " << result[a] << endl;

}

catch (IndexOutOfRangeError e) {

e.PrintError();

continue;

}

catch (NegativeIndexError e) {

e.PrintError();

continue;

}

catch (BasicError& err)

{

err.PrintError();

char c;

cout << "Хотите начать заново? (1/0): ";

cin >> c;

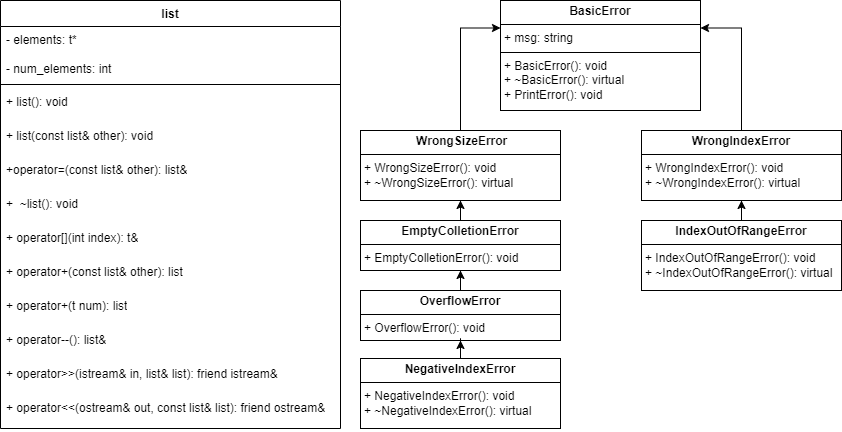
if (c == '0') break;

}

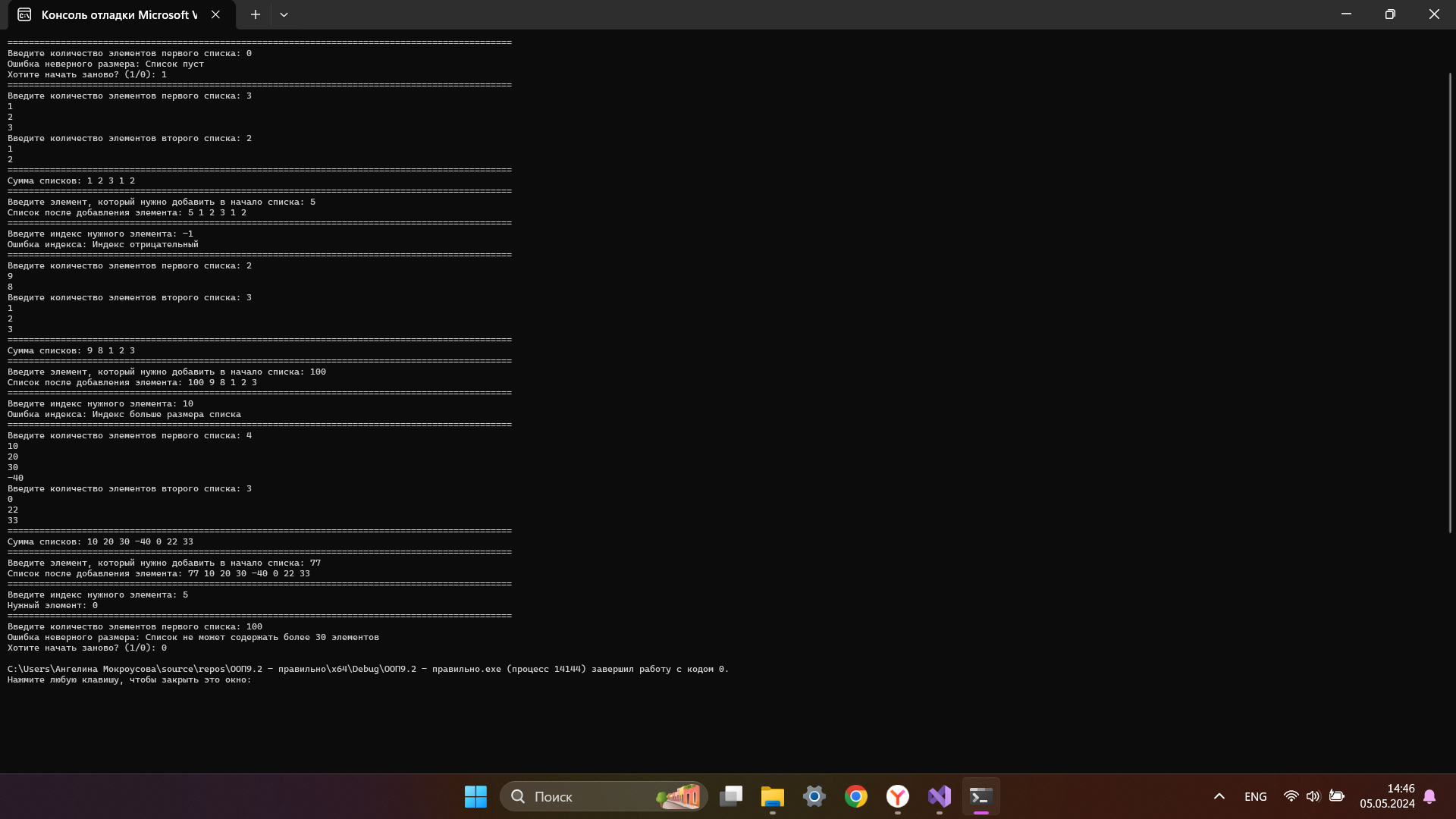
}

}

**UML диаграмма:**

****

**Скриншот работы программы:**



**Вывод:**

Мною была аписана программа, которая успешно выполняет поставленную перед ней задачу. В процессе написания был повторен и закреплен на практике пройденный ранее материал.

**Ответы на вопросы:**

1. Что представляет собой исключение в C++

Что делать, если возникла ошибка в программе и как обрабатывать эту ошибку.

1. На какие части исключения позволяют разделить вычислительный процесс? Достоинства такого подхода?

На генерацию исключения и обработку исключения. Достоинства - улучшение читаемости кода и обработку ошибок в специальных блоках.

1. Какой оператор используется для генерации исключительной ситуации?

Оператор throw.

1. Что представляет собой контролируемый блок? Для чего он нужен?

Контролируемый блок - это участок кода, в котором может произойти исключение. Нужен для обработки исключений.

1. Что представляет собой секция-ловушка? Для чего она нужна?

Секция-ловушка - это блок кода, который отлавливает и обрабатывает исключения. Нужна для обработки ошибок.

1. Какие формы может иметь спецификация исключения в секции ловушке? В каких ситуациях используются эти формы?

Может содержать конкретные типы исключений или ключевое слово throw. Используются для указания того, какие исключения будет перехватывать секция-ловушка.

1. Какой стандартный класс можно использовать для создания собственной иерархии исключений?

std::exception.

1. Каким образом можно создать собственную иерархию исключений?

Наследоваться от класса std::exception и переопределить метод what().

1. Если спецификация исключений имеет вид: void f1()throw(int,double); то какие исключения может прождать функция f1()?

Функция f1() может принимать исключения типа int и double.

1. Если спецификация исключений имеет вид: void f1()throw(); то какие исключения может прождать функция f1()?

Функция f1() не принимает никаких исключений.

1. В какой части программы может генерироваться исключение?

Исключение может генерироваться в любой части программы, но обычно в контролируемом блоке.

1. Написать функцию, которая вычисляет площадь треугольника по трем сторонам (формула Герона).

// Без спецификации исключений

double calculateTriangleArea(double a, double b, double c) {

double s = (a + b + c) / 2;

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

// Со спецификацией throw()

double calculateTriangleAreaThrow(double a, double b, double c) {

if (a <= 0 || b <= 0 || c <= 0) {

throw "Invalid side length";

}

double s = (a + b + c) / 2;

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

// С конкретной спецификацией с подходящим стандартным исключением

double calculateTriangleAreaStdException(double a, double b, double c) {

if (a <= 0 || b <= 0 || c <= 0) {

throw std::invalid\_argument("Invalid side length");

}

double s = (a + b + c) / 2;

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

// Спецификация с собственным реализованным исключением

class InvalidSideLengthException : public std::exception {

public:

const char\* what() const throw() {

return "Invalid side length";

}

};

double calculateTriangleAreaCustomException(double a, double b, double c) {

if (a <= 0 || b <= 0 || c <= 0) {

throw InvalidSideLengthException();

}

double s = (a + b + c) / 2;

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

**Скриншот с GitHab:**

