ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа

Обработка исключительных ситуаций

Выполнил студент группы РИС-23-3Б

Мазунин М.А.

Проверила доцент кафедры ИТАС

О. А. Полякова

2024 г.

**1. Постановка задачи**

1. Реализовать класс, перегрузить для него операции, указанные в варианте

2. Определить исключительные ситуации

3. Предусмотреть генерацию исключительных ситуаций

**2. Анализ задачи**

Класс BasicError - Базовый класс иерархии исключений

Класс WrongSizeError - Класс ошибок неверного размера, наследуется от BasicError

Класс EmptyCollectionError - Класс ошибки пустого списка, наследуется от WrongSizeError

Класс OverflowError - Класс ошибки выхода за границы списка, наследуется от WrongSizeError

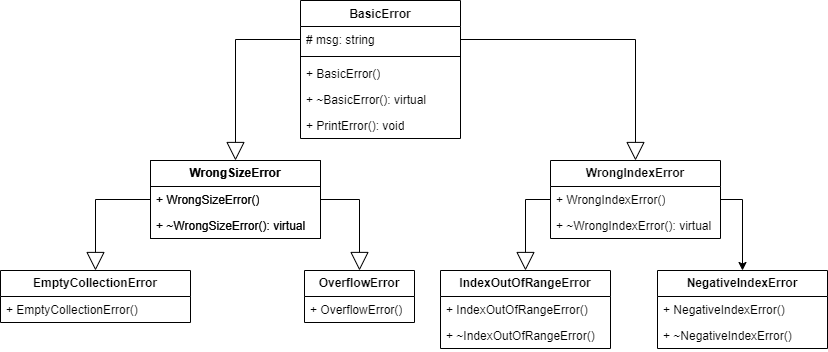
Класс WrongIndexError - Класс ошибок неверного индекса, наследуется от BasicError

Класс IndexOutOfRangeError - Класс ошибки выхода индекса за пределы списка, наследуется от WrongIndexError  
Класс NegativeIndexError - Класс ошибки отрицательного значения индекса, наследуется от WrongIndexError

Класс List - Класс-контейнер

**3. Разбор работы алгоритма**

UML диаграмма

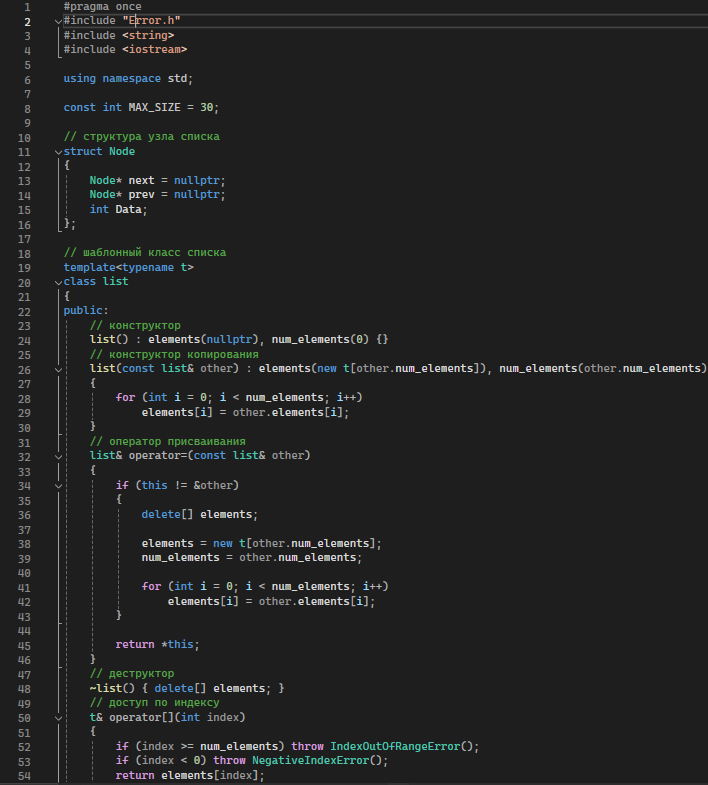


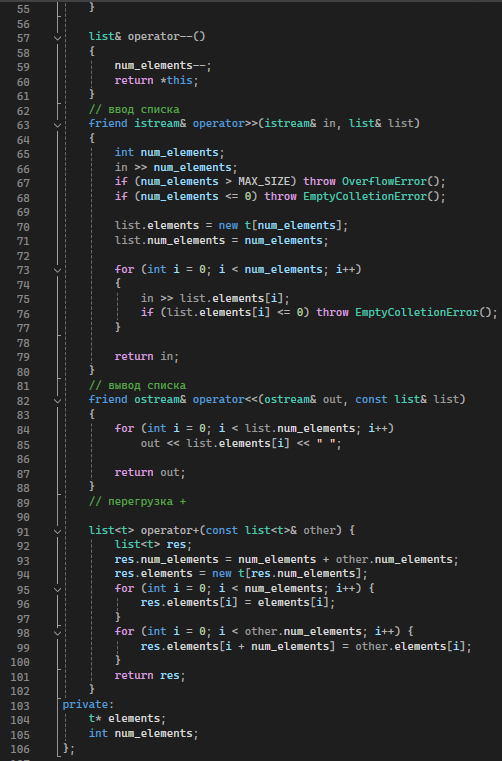
Код программы

Error.h

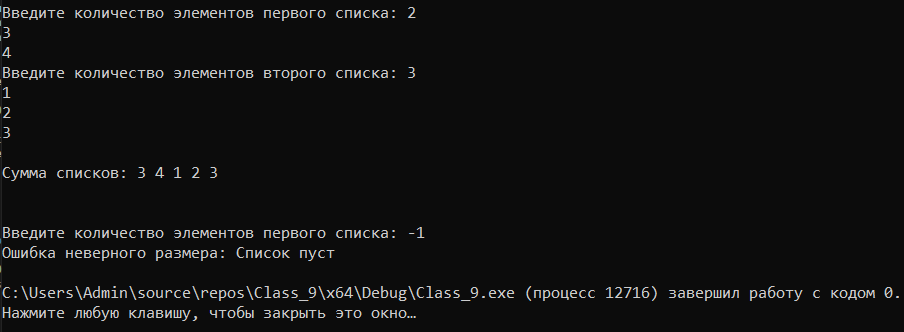


List.h





**4. Результаты работы программы**



**5. Контрольные вопросы**

1. Механизм обработки ошибок
2. Обнаружение исключения и его обработка
3. throw
4. Блок, в котором происходит проверка на наличие исключения
5. Секция-ловушка передаёт код ошибки обработчику
6. (Тип, имя); (Тип) ; (...)
7. <expection>
8. Создать свой базовый класс исключений
9. logic\_error, runtime\_error, bad\_cast
10. logic\_error, runtime\_error
11. Исключение может генерироваться в той части программы, которая обнаружила аномальную ситуацию

#include <iostream>

using namespace std;

class BasicError {

public:

BasicError() {}

virtual ~BasicError() {}

void PrintError() {

cout << "Error!" << endl;

}

};

int main()

{

while (true) {

try

{

double a, b, c, P, S;

cin >> a >> b >> c;

if (a < 0 || b < 0 || c < 0 || a + b < c || b + c < a || a + c < b)

{

throw BasicError();

}

P = (a + b + c) / 2;

S = sqrt(P \* (P - a) \* (P - b) \* (P - c));

cout << S;

}

catch (BasicError e) {

e.PrintError();

continue;

}

}

}

**6. Вывод**

Алгоритм выполняет поставленную задачу.

Данный проект можно найти на GitHub



