Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Обработка исключительных ситуаций.

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Токарев Павел Аркадьевич

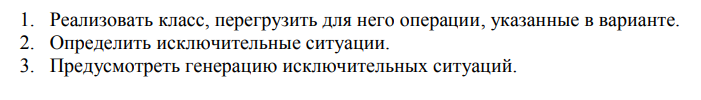
Проверил

Доцент кафедры ИТАС

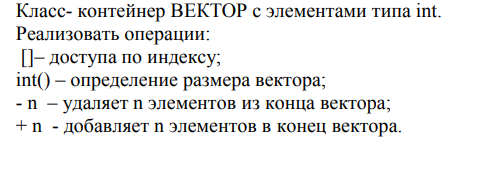
Полякова Ольга Андреевна

Г. Пермь-2023

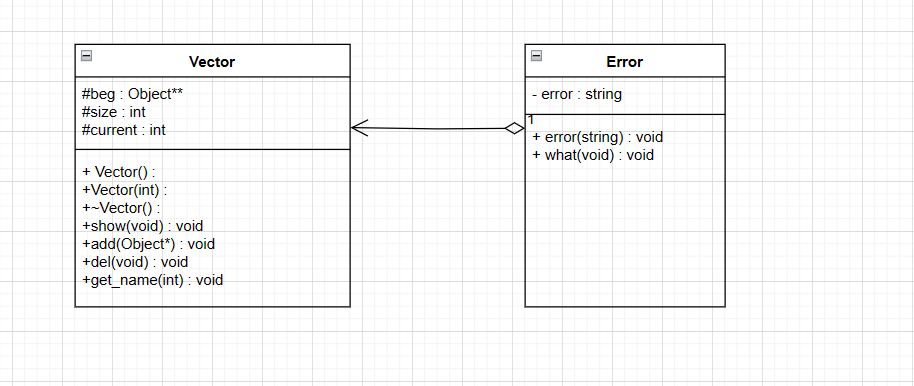
**Постановка задачи**

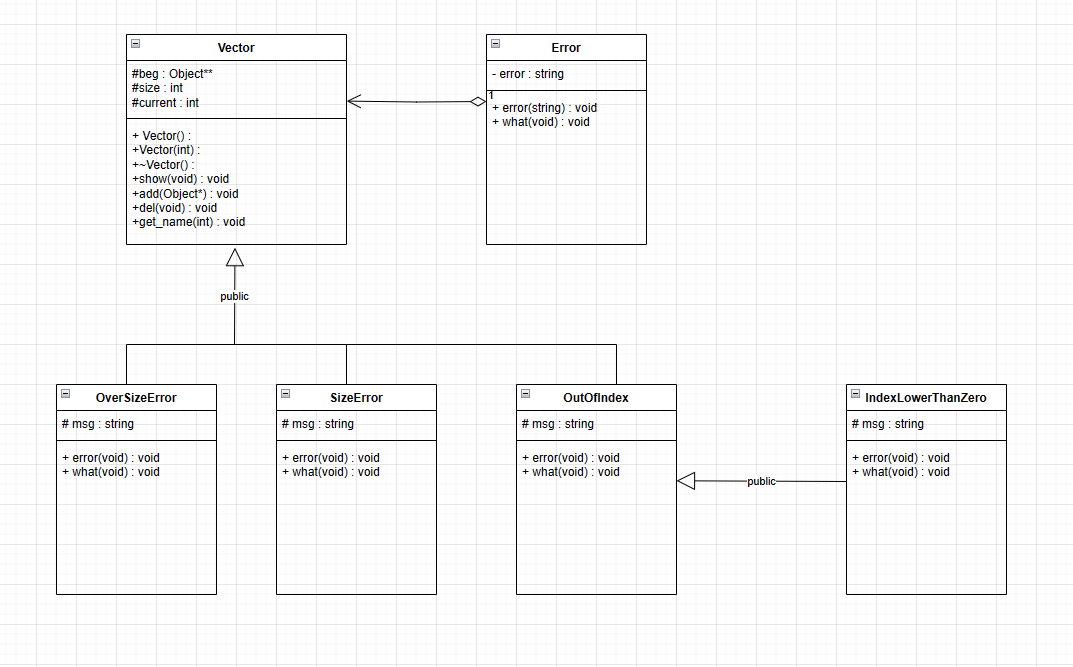


Вариант 2:



**Диаграмма UML**

****

****

**Код программы**

Error.h

#pragma once

#include<iostream>

using namespace std;

class Error {

public:

virtual void what() {

};

};

class OutOfIndex :public Error {

protected:

string msg;

public:

OutOfIndex() {

msg = "Ошибка индесации";

}

virtual void what() {

cout << msg << endl;

}

};

class SizeError :public Error {

protected:

string msg;

public:

SizeError() {

msg = "Ошибка в размере вектора";

}

virtual void what() {

cout << msg << endl;

}

};

class OverSizeError :public Error {

protected:

string msg;

public:

OverSizeError() {

msg = "Допустимый размер вектора превышен";

}

virtual void what() {

cout << msg << endl;

}

};

class EmptyVector :public SizeError {

protected:

string msg;

public:

EmptyVector() {

msg = "Удаление из пустого вектора невозможно";

}

virtual void what() {

cout << msg << endl;

}

};

class IndexLowerThanZero : public OutOfIndex {

protected:

string msg;

public:

IndexLowerThanZero() {

msg = "Индекс меньше нуля";

}

virtual void what() {

cout << msg << endl;

}

};

class IndexOutOfSize : public OutOfIndex {

protected:

string msg;

public:

IndexOutOfSize() {

msg = "Индекс больше текущего размера";

}

virtual void what() {

cout << msg << endl;

}

};

Vector.h

#pragma once

#include<iostream>

#include"Error.h"

using namespace std;

const int maxsize = 10;

class Vector

{

public:

Vector(int t);

Vector(const Vector& t);

~Vector();

int operator[](int index);

Vector& operator=(const Vector& t);

Vector operator+(const Vector& t);

Vector operator-(const Vector& t);

int operator() ();

void operator+(int n);

void operator-(int n);

friend ostream& operator<<(ostream& out, const Vector& t);

friend istream& operator>>(istream& in, Vector& t);

private:

int size;

int\* mass;

};

Vector.cpp

#include "Vector.h"

#include"Error.h"

#include<iostream>

using namespace std;

Vector::Vector(int sz) {

size = sz;

if (size > maxsize) {

throw OverSizeError();

}

mass = new int[sz];

for (int i = 0;i < sz;i++) {

mass[i] = 0;

}

}

Vector::Vector(const Vector& t) {

size = t.size;

mass = new int[size];

for (int i = 0;i < size;i++) {

mass[i] = t.mass[i];

}

}

Vector::~Vector() {

delete[] mass;

size = 0;

}

Vector& Vector::operator=(const Vector& t) {

size = t.size;

for (int i = 0;i < size;i++) {

mass[i] = t.mass[i];

}

return \*this;

}

int Vector::operator[](int index) {

if (index < 0) {

throw IndexLowerThanZero();

}

if (index >= size) {

throw IndexOutOfSize();

}

return mass[index];

}

Vector Vector::operator-(const Vector& t) {

Vector tmp(t.size);

for (int i = 0;i < t.size;i++) {

tmp.mass[i] = mass[i] - t.mass[i];

}

return tmp;

}

Vector Vector::operator+(const Vector& t) {

Vector tmp(t.size);

for (int i = 0;i < t.size;i++) {

tmp.mass[i] = mass[i] + t.mass[i];

}

return tmp;

}

int Vector::operator()() {

return size;

}

ostream& operator<<(ostream& out, const Vector& t) {

if (t.size == 0) {

throw EmptyVector();

}

for (int i = 0;i < t.size;i++) {

out << t.mass[i] << " ";

}

return out;

}

istream& operator>>(istream& in, Vector& t) {

for (int i = 0;i < t.size;i++) {

cout << (i + 1) << ":";

in >> t.mass[i];

}

return in;

}

void Vector::operator+(int n) {

if (n + size > maxsize) {

throw OverSizeError();

}

for (int i = 0;i < n;i++) {

cout << size << " : ";

cin >> mass[size];

size++;

}

}

void Vector::operator-(int n) {

if (size - n < 0) {

throw SizeError();

}

if (n > size) {

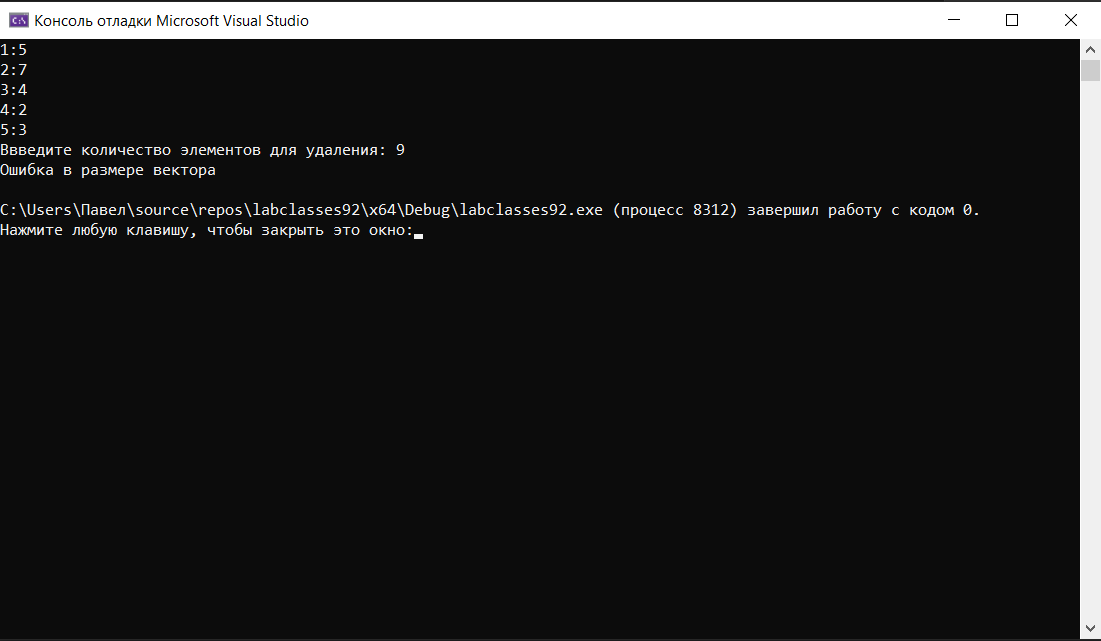
OverSizeError();

}

size -= n;

}

**Результат работы программы**

****

**Анализ результатов**

Удалось успешно реализовать обработку исключений. Программа успешно выполняет отлавливание исключений и выводит сообщение об ошибке успешно.

**Контрольные вопросы**

1.Что представляет собой исключение в С++?

Исключение – это объект, который система должна генерировать при возникновении исключительной ситуации. Генерация такого объекта и создает исключительную ситуацию

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

1) обнаружение аварийной ситуации (неизвестно как обрабатывать);

2) обработка аварийной ситуации (неизвестно, где она возникла).

3.Какой оператор используется для генерации исключительной ситуации?

Throw <выражение>;

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

Try catch. Если в Трай возникнет какая-либо ошибка, программа не остановит свое выполнение, и перейдет в блок кетч, и обработает ошибку.

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

Программа не остановит выполнения, и выполнит код тогда, когда в блоке трай появилась ошибка.

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

1) (тип имя)

2) (тип)

3) (…)

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

Для создания иерархии классов можно использовать класс Exeption.

8.Каким образом можно создать собственную иерархию исключений?

Унаследоваться от класса Exception.

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

Int и double

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

Любые

11.В какой части программы может генерироваться исключение?

Исключение может генерироваться где угодно.