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**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №9**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Обработка исключительных ситуаций

Вариант 6.

Выполнил работу

Студент группы РИС-22-1Б

Поважный В. Е.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

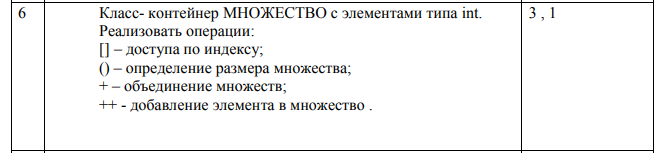
г. Пермь-2023

**Введение**

Для выполнения лабораторной работы требуется сделать отчет программы. Создание программ – отличный способ практики программирования.

**Постановка задачи**

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



**Код программы**

**ооп 9.cpp**

#include "Many.h"

#include "Error.h"

#include <iostream>

using namespace std;

int main()

{

try

{

Many x(2);

Many y;

cout << x;

cout << "Nomer" << endl;

int i;

cin >> i;

cout << x[i] << endl;

y = x + 3;

cout << y;

--x;

cout << x;

--x;

cout << x;

--x;

}

catch (Error& e)

{

e.what();

}

return 0;

}

**error.h**

**#pragma** once

#include <string>

#include <iostream>

using namespace std;

class Error

{

public:

virtual void what() {};

};

class IndexError : public Error

{

protected:

string msg;

public:

IndexError()

{

msg = "Index Error\n";

}

virtual void what()

{

cout << msg;

}

};

class SizeError : public Error

{

protected:

string msg;

public:

SizeError()

{

msg = "size error\n";

}

virtual void what()

{

cout << msg;

}

};

class MaxSizeError : public SizeError

{

protected:

string msg\_;

public:

MaxSizeError()

{

SizeError();

msg\_ = "size>MAXSIZE\n";

}

virtual void what()

{

cout << msg << msg\_;

}

};

class EmptySizeError : public SizeError

{

protected:

string msg\_;

public:

EmptySizeError()

{

SizeError();

msg\_ = "Vector is empty\n";

}

virtual void what()

{

cout << msg << msg\_;

}

};

class IndexError1 : public IndexError

{

protected:

string msg\_;

public:

IndexError1()

{

IndexError();

msg\_ = "index <0\n";

}

virtual void what()

{

cout << msg << msg\_;

}

};

class IndexError2 : public IndexError

{

protected:

string msg\_;

public:

IndexError2()

{

IndexError();

msg\_ = "index>size\n";

}

virtual void what()

{

cout << msg << msg\_;

}

};

**Many.h**

**#pragma** once

#include <iostream>

#include <fstream>

using namespace std;

const int MAX\_SIZE = 20;

class Many

{

public:

int size;

int\* beg;

Many()

{

size = 0;

beg = 0;

}

Many(int s);

Many(int s, int\* mas);

Many(const Many& v);

~Many();

const Many& operator=(const Many& v);

int operator[](int i);

Many operator+(int a);

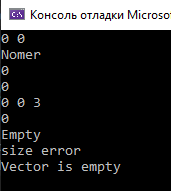
Many operator--();

friend ostream& operator<<(ostream& out, const Many& v);

friend istream& operator>>(istream& in, const Many& v);

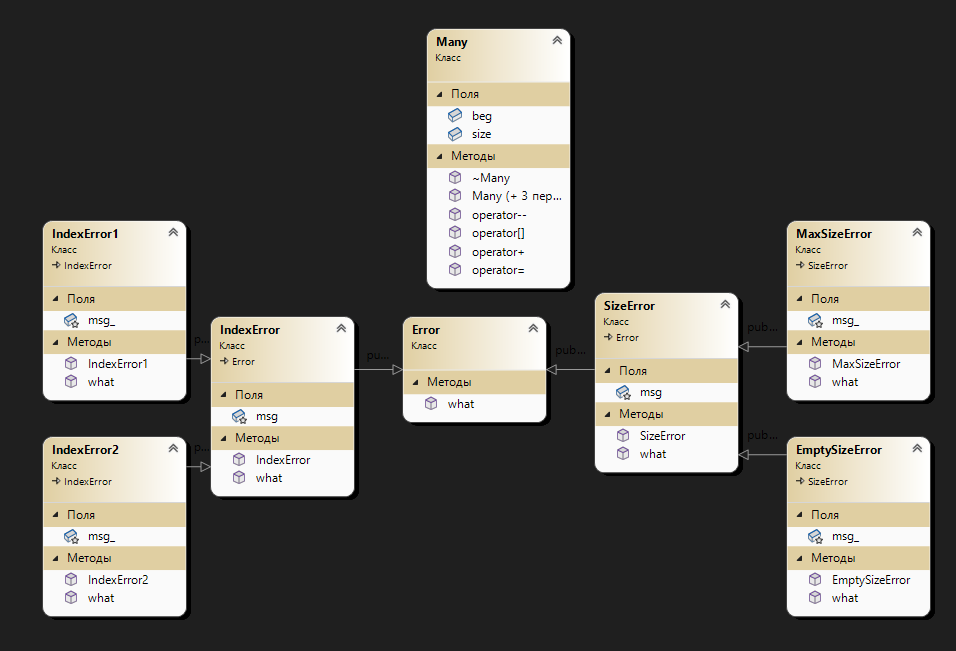
};

**Вывод программы**

****

***Рисунок 1 – вывод программы***

**UML диаграмма**

****

***Рисунок 2 – UML диаграмма***

**Вывод**

Программа выполняет свою задачу.

**Заключение**

Для решения задачи потребовались знания языка программирования, а конкретнее C++. Программа выполняет те условия, что были указаны в постановке задачи и работает без проблем.