**Пермский национальный исследовательский политехнический университет**

Кафедра “Информационные технологии и автоматизированные системы”

**Лабораторная работа №26**

По дисциплине «Основы алгоритмизации и программирования»

**Тема:**

Обработка исключений.

Выполнила:

студентка группы ИВТ-19-1б

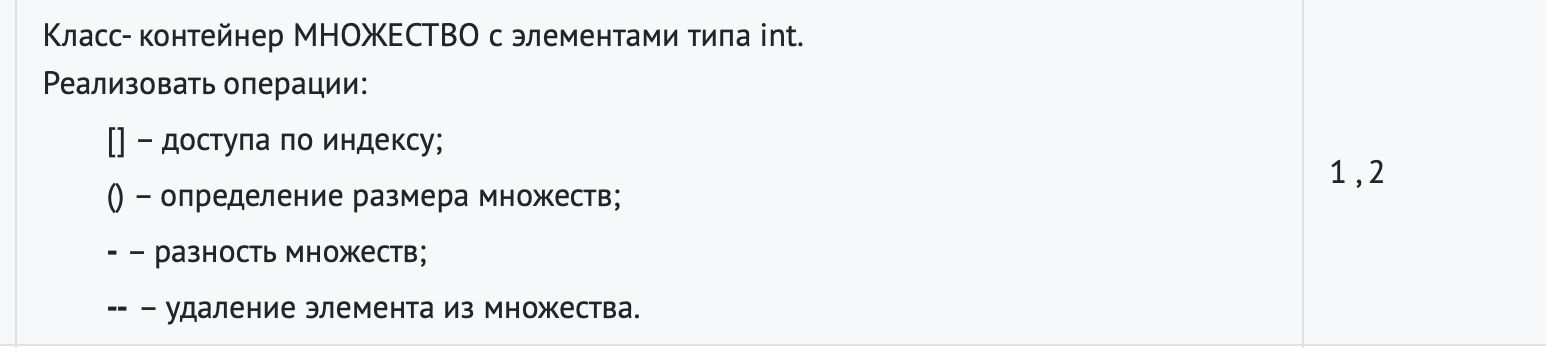
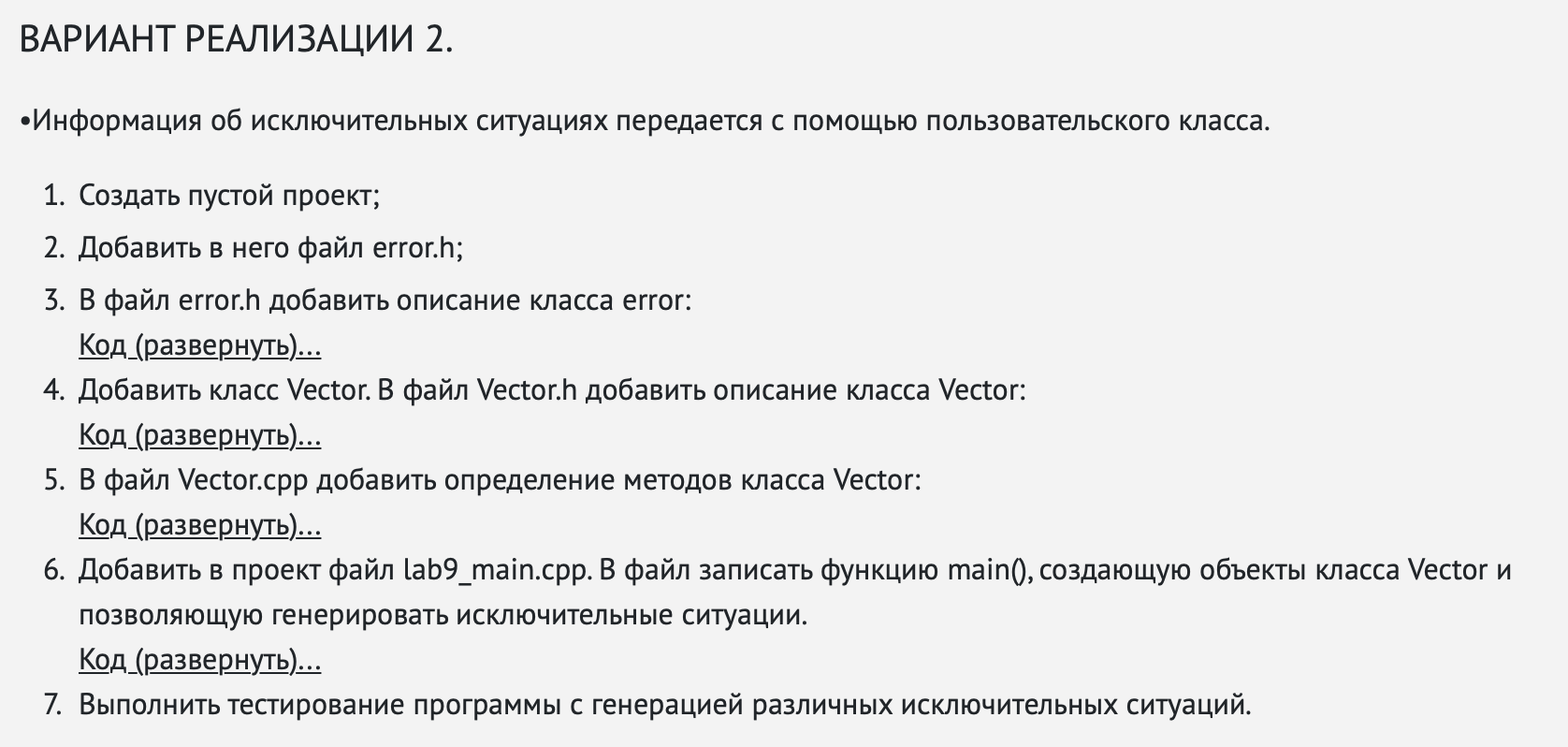
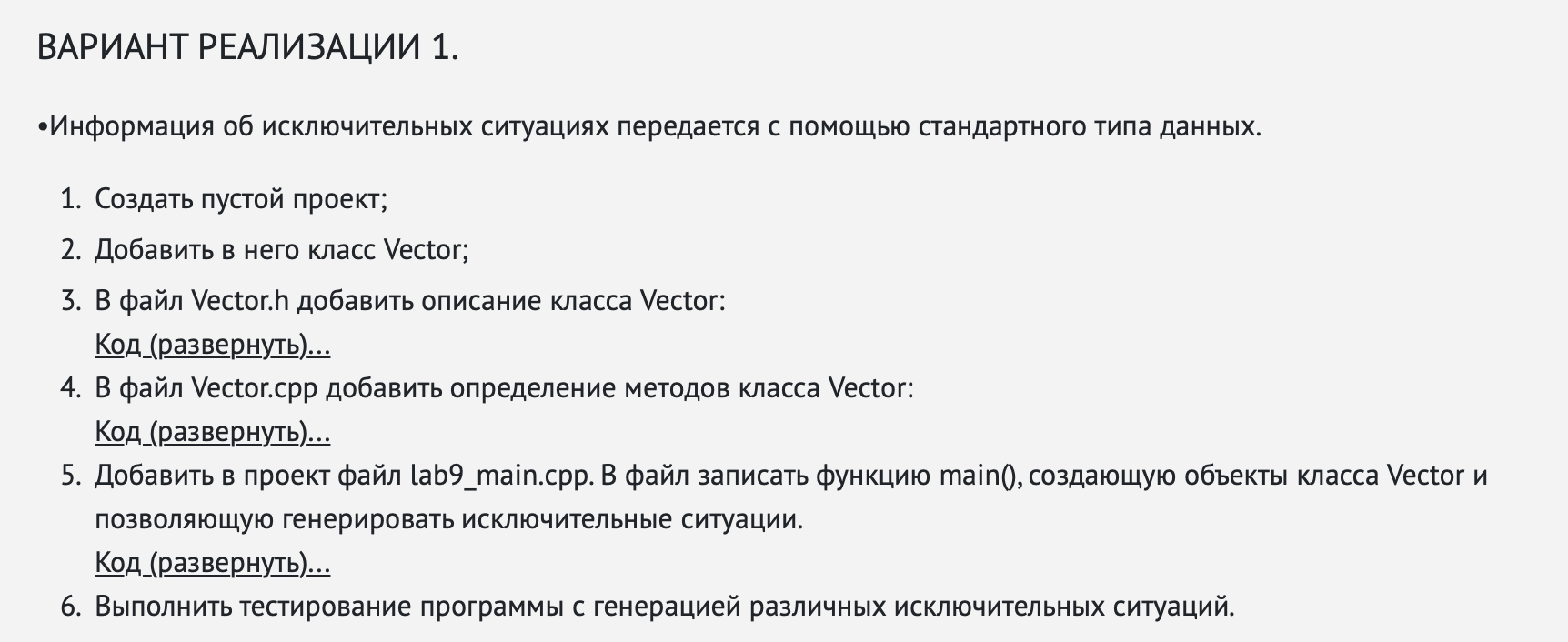
Еремеева Полина Алексеевна

Проверила:

доцент кафедры “ИТАС”

Полякова О.А.

Пермь, 2020

**Постановка задачи**

**Код**

Main.cpp

#include "Lots.hpp"

#include "Vector\_realiz2.hpp"

#include "Error.hpp"

#include <iostream>

**using** **namespace** std;

**int** main()

{

setlocale(LC\_ALL, "rus");

**int** a;

cout << "Выберите номер реализации(1 или 2): ";

cin >> a;

**int** mass[2] = { 1,2 };

**if** (a == 1)

{

**try**

{

Lots x(2, mass);

cout << "Элементы группы: ";

cout << x;

cout << "Введите индекс\n";

**int** i;

cin >> i;

cout << "Элемент с индексом " << i << "\n";

cout << x[i] << endl;

cout << "Вычитаем элемент!\n";

--x;

cout << x;

cout << "Вычитаем элемент!\n";

--x;

cout << x;

cout << "Вычитаем элемент!\n";

--x;

}

**catch** (**int**)

{

cout << "ERROR!!!" << endl;

}

**return** 0;

}

**else** **if** (a == 2)

{

**try**

{

Lots2 x(2, mass);

cout << "Элементы группы: ";

cout << x;

cout << "Введите индекс: ";

**int** i;

cin >> i;

cout << "Элемент с индексом " << i << "\n";

cout << x[i] << endl;

cout << "Вычитаем элемент!\n";

--x;

cout << x;

cout << "Вычитаем элемент!\n";

--x;

cout << x;

cout << "Вычитаем элемент!\n";

--x;

}

**catch** (error e)

{

e.what();

}

**return** 0;

}

**else**

{

cout << "Неверный номер!";

**return** 0;

}

}

Error.hpp

#pragma once

#include <string>

#include <iostream>

**using** **namespace** std;

**class** error

{

string str;

**public**:

error(string s){str=s;}

**void** what() { cout << str << endl; }

};

Vector\_realiz2.cpp

#include "Vector\_realiz2.hpp"

#include "Error.hpp"

Lots2::Lots2(**int** s)

{

**if** (s > MAX\_SIZE2) **throw** error("Vector length more than max size\n");

size = s;

beg = **new** **int**[s];

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

beg[i] = 0;

}

Lots2::Lots2(**const** Lots2& l)

{

size = l.size;

beg = **new** **int**[size];

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

{

beg[i] = l.beg[i];

}

}

Lots2::~Lots2()

{

**if** (beg != 0) **delete**[]beg;

}

Lots2::Lots2(**int** s, **int**\* mas)

{

**if** (s > MAX\_SIZE2) **throw** error("Vector length more than max size\n");

size = s;

beg = **new** **int**[size];

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

beg[i] = mas[i];

}

**const** Lots2& Lots2::**operator** =(**const** Lots2& l)

{

**if** (**this** == &l)**return** \***this**;

**if** (beg != 0) **delete**[]beg;

size = l.size;

beg = **new** **int**[size];

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

beg[i] = l.beg[i];

**return**\***this**;

}

ostream& **operator**<<(ostream& out, **const** Lots2& l)

{

**if** (l.size == 0) out << "Empty\n";

**else**

{

**for** (**int** i = 0; i < l.size; i++)

out << l.beg[i] << " ";

out << endl;

}

**return** out;

}

istream& **operator** >>(istream& in, Lots2& l)

{

**for** (**int** i = 0; i < l.size; i++)

{

cout << ">";

in >> l.beg[i];

}

**return** in;

}

**int** Lots2::**operator** [](**int** i)

{

**if** (i < 0)**throw** error("index < 0");

**if** (i >= size) **throw** error("index > size");

**return** beg[i];

}

Lots2 Lots2::**operator**-(**const** Lots2& v)

{

Lots2 temp(size);

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

{

temp.beg[i] = v.beg[i] - beg[i];

}

**return** temp;

}

Lots2 Lots2::**operator**--()

{

**if** (size == 0) **throw** error("Lots is empty");

**if** (size == 1)

{

size = 0;

**delete**[]beg;

beg = 0;

**return** \***this**;

}

Lots2 temp(size, beg);

**delete**[]beg;

size--;

beg = **new** **int**[size];

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

beg[i] = temp.beg[i];

**return**\***this**;

};

**int** Lots2::**operator**()()

{

**return** size;

}

Vector\_realiz2.hpp

#pragma once

#include <iostream>

**using** **namespace** std;

**const** **int** MAX\_SIZE2 = 30;

**class** Lots2

{

**int** size;

**int**\* beg;

**public**:

Lots2() { size = 0; beg = 0; }

Lots2(**int** s);

Lots2(**int** s, **int**\* mas);

Lots2(**const** Lots2& l);

~Lots2();

**const** Lots2& **operator**=(**const** Lots2& l);

**int** **operator**[](**int** i);

**int** **operator**()();

Lots2 **operator**-(**const** Lots2& l);

Lots2 **operator**--();

**friend** ostream& **operator**<<(ostream& out, **const** Lots2& v); **friend** istream& **operator**>>(istream& in, Lots2& v);

};

Lots.cpp

#include "Lots.hpp"

Lots::Lots(**int** s)

{

**if** (s > MAX\_SIZE) **throw** 1;

size = s;

beg = **new** **int**[s];

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

beg[i] = 0;

}

Lots::Lots(**const** Lots& l)

{

size = l.size;

beg = **new** **int**[size];

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

{

beg[i] = l.beg[i];

}

}

Lots::~Lots()

{

**if** (beg != 0) **delete**[]beg;

}

Lots::Lots(**int** s, **int**\* mas)

{

**if** (s > MAX\_SIZE) **throw** 1;

size = s;

beg = **new** **int**[size];

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

beg[i] = mas[i];

}

**const** Lots& Lots::**operator** =(**const** Lots& l)

{

**if** (**this** == &l)**return** \***this**;

**if** (beg != 0) **delete**[]beg;

size = l.size;

beg = **new** **int**[size];

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

beg[i] = l.beg[i];

**return**\***this**;

}

ostream& **operator**<<(ostream& out, **const** Lots& l)

{

**if** (l.size == 0) out << "Empty\n";

**else**

{

**for** (**int** i = 0; i < l.size; i++)

out << l.beg[i] << " ";

out << endl;

}

**return** out;

}

istream& **operator** >>(istream& in, Lots& l)

{

**for** (**int** i = 0; i < l.size; i++)

{

cout << ">";

in >> l.beg[i];

}

**return** in;

}

**int** Lots::**operator** [](**int** i)

{

**if** (i < 0)**throw** 2;

**if** (i >= size) **throw** 2;

**return** beg[i];

}

Lots Lots::**operator**-(**const** Lots& v)

{

Lots temp(size);

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

{

temp.beg[i] = v.beg[i] - beg[i];

}

**return** temp;

}

Lots Lots::**operator**--()

{

**if**(size==0) **throw** 5;

**if** (size == 1)

{

size = 0;

**delete**[]beg;

beg = 0;

**return** \***this**;

}

Lots temp(size, beg);

**delete**[]beg;

size--;

beg = **new** **int**[size];

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

beg[i] = temp.beg[i];

**return**\***this**;

};

**int** Lots::**operator**()()

{

**return** size;

}

Lots.hpp

#pragma once

#include <iostream>

**using** **namespace** std;

**const** **int** MAX\_SIZE = 30;

**class** Lots

{

**int** size;

**int**\* beg;

**public**:

Lots() { size = 0; beg = 0; }

Lots(**int** s);

Lots(**int** s, **int**\* mas);

Lots(**const** Lots& l);

~Lots();

**const** Lots& **operator**=(**const** Lots& l);

**int** **operator**[](**int** i);

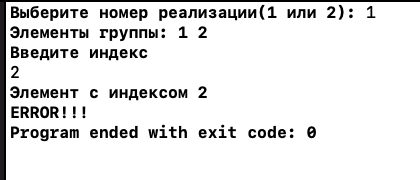
**int** **operator**()();

Lots **operator**-(**const** Lots& l);

Lots **operator**--();

**friend** ostream& **operator**<<(ostream& out, **const** Lots& v); **friend** istream& **operator**>>(istream& in, Lots& v);

};

**Тестирование**

