Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.9**

Дисциплина: «Информатика»

Тема: “ Объектно-ориентированное программирование.

Обработка исключительных ситуаций.”

Вариант 4

Выполнил:

Студент группы ИВТ-20-2Б Тедеев Алесандр Зурабович

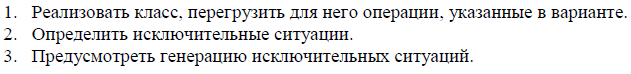
Проверила:

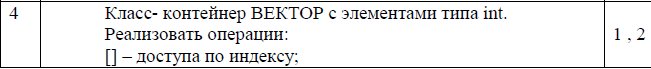
Доцент кафедры ИТАС

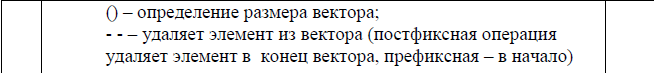
Полякова О.А.

Пермь, 2021

**Постановка задачи**







2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Vector с полем типа int – size, и указателем beg.

**1.2.** Организовать класс error с полем str типа string.

**1.3.** Организовать перегрузку оператора () дружественной классу Vector.

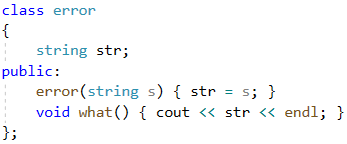
**1.4.** Организовать перегрузку оператора [] дружественной классу Vector.

**1.5.** Организовать перегрузку оператора = дружественной классу Vector.

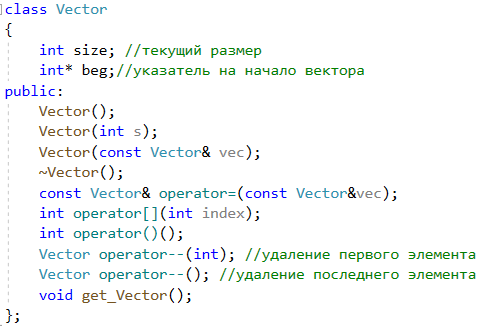
**1.6.** Организовать перегрузку оператора – (постфиксной и префиксной формы) дружественной классу Vector.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Класс error с полем str типа string для хранения строки ошибки в заголовочном файле error.h.



**2.2.** Класс Vector с полем типа int – size, и указателем beg в заголовочном файле Vector.h.



**2.3.** Объект класса Vector first – массив типа Vector.

Vector first(size);

3

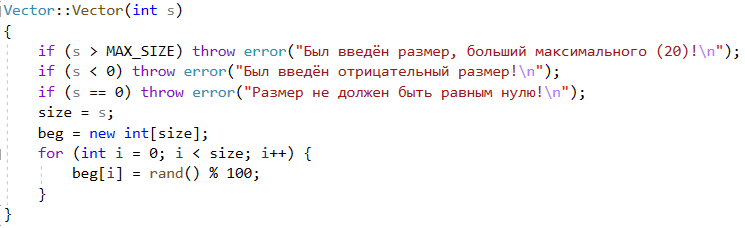
**2.4.** Переменная типа int: size, которая отвечает за размер массива. Перменная choice типа char отвечает за выбор пользователем варианта декрементирования массива.

int size;

char choice;

**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Данные инициализируются в конструкторе с параметром в классе Vector.



Добавлены 3 проверки на случаи:

1) введёный пользователем размер массива больше, чем допустимо возможный;

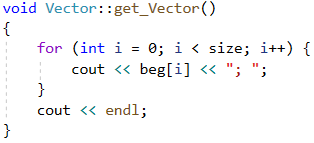
2) введёный пользователем размер массива является отрицательным;

3) введёный пользователем размер массива равен нулю, так как операции декремента приведут к отрицательности массива;

Указателю выделяется динамическая память и массив beg[] заполняется случайными числами в цикле for().

**4.** Для операций ввода и вывода использовались следующие операторы и функции:

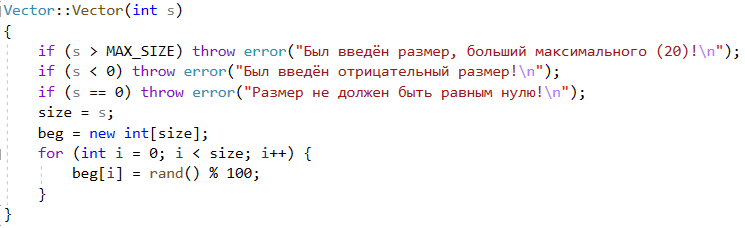
**4.1.** Для вывода массива класса Vector на консоль используется метод get\_Vector().



4

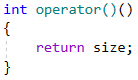
**4.2.** Для ввода массива класса Vector необходимо сначала ввести размер size, а затем в конструкторе сформировать динамический массив.



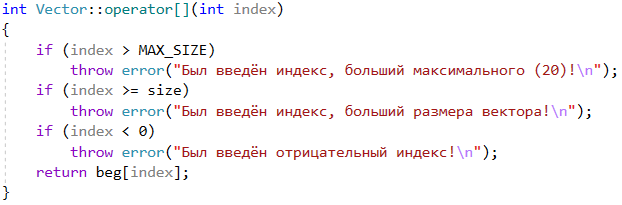


**5.** Поставленные задачи будут решены следующими действиями:

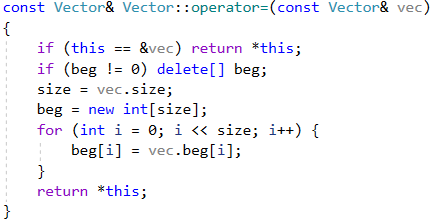
**5.1.** В перегрузке оператора () класса Vector производится вывод размерности объекта класса Vector.



**5.2.** В перегрузке оператора [] класса Vector производится вызов элемента множества по индексу объекта класса Vector.

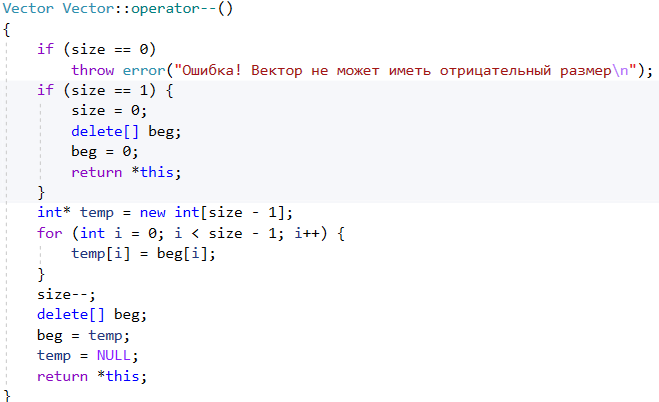


**5.3.** В перегрузке оператора = класса Vector производится вызов элемента множества по индексу объекта класса Vector.

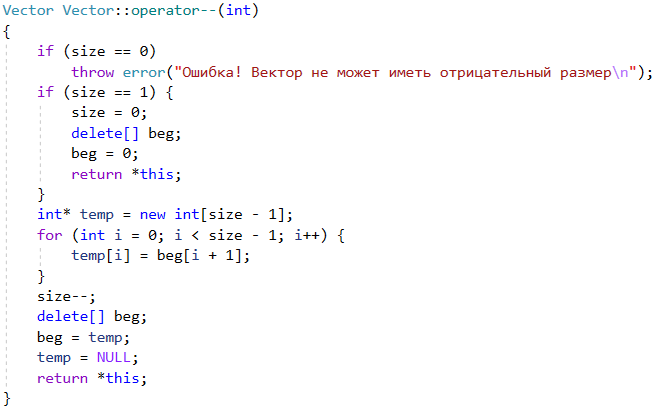


5

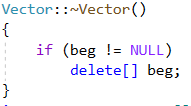
**5.4.** В перегрузке оператора -- класса Vector производится создание динамического массива с размерностью на 1 меньше и производится заполнение до size-1 элемента массива beg[]. Затем освобождается память, на которую указывает \*beg и передаётся память, на которую указывает \*temp.



В постфиксной форме оператора – производится похожи действия, однако необходимо удалить первый элемент, поэтому заполнение в цикле идёт для i+1 индекса для массива beg[].



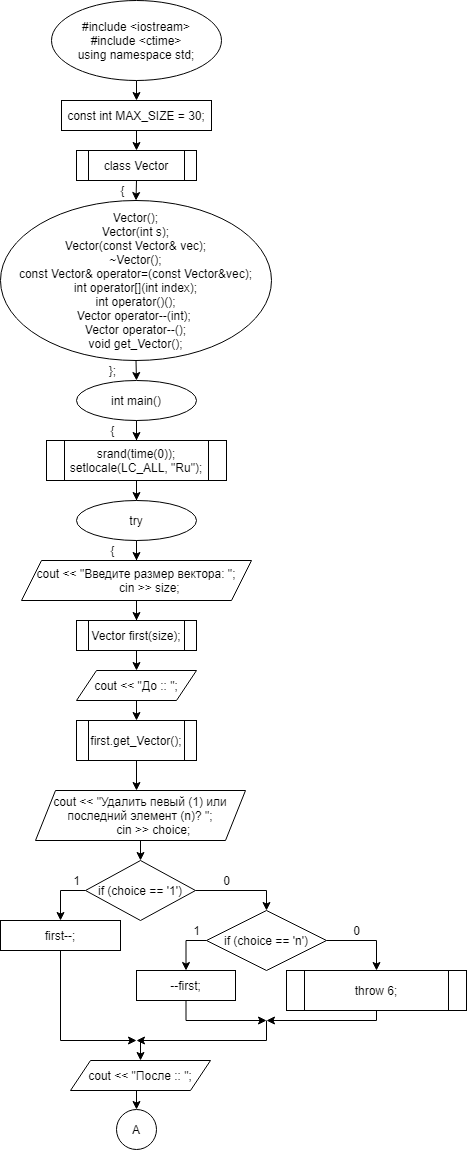
**5.5.** В деструкторе выполнятеся очистка памяти, если указатель не указывает на NULL.



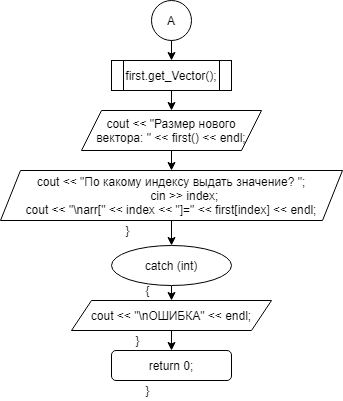
6

**Блок-схема**

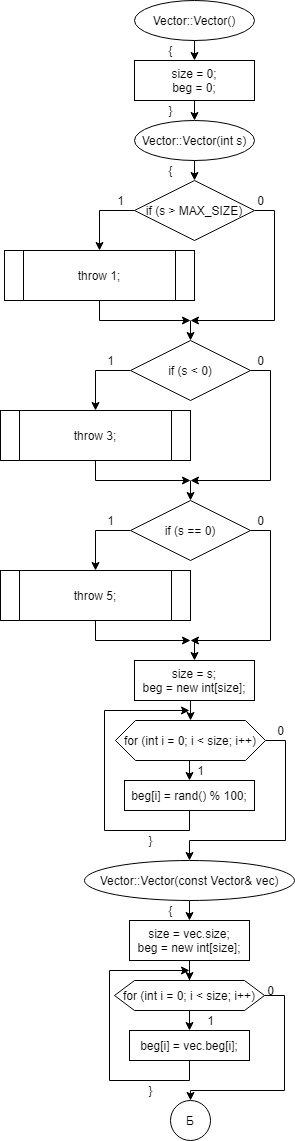
**Реализация 1-м способом**

****

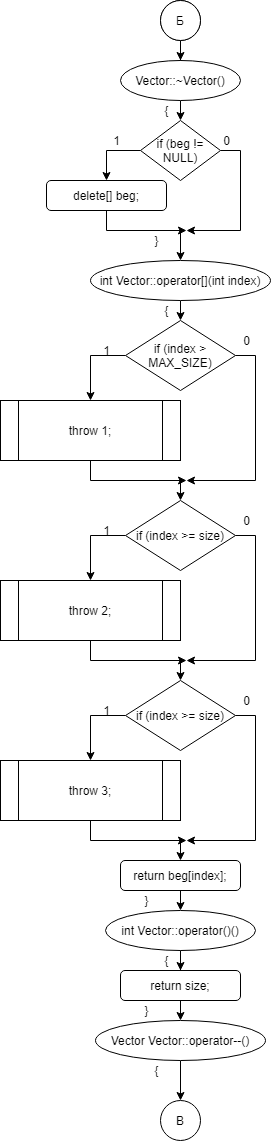
7



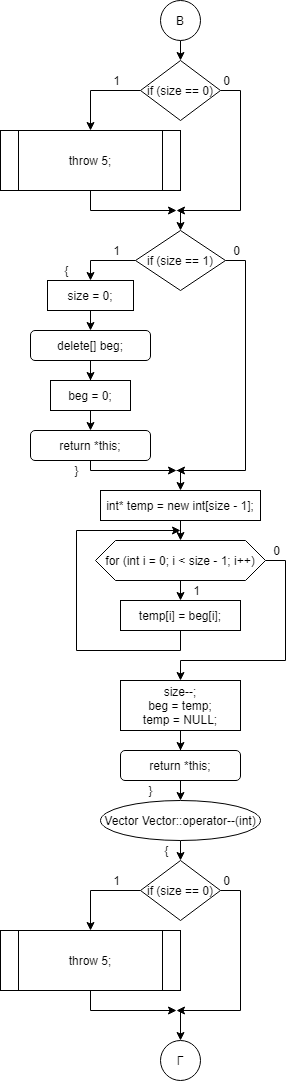
8



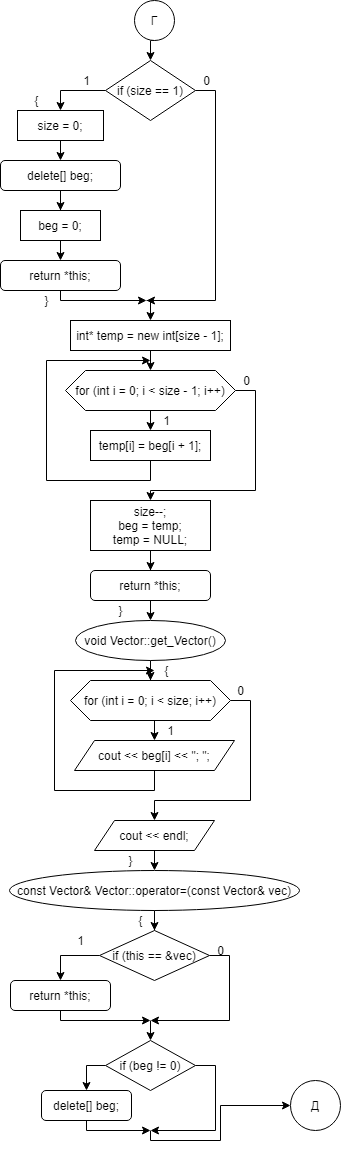
9



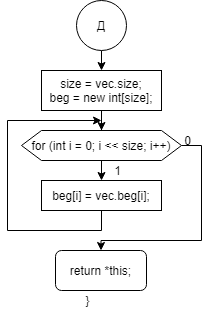
10



11

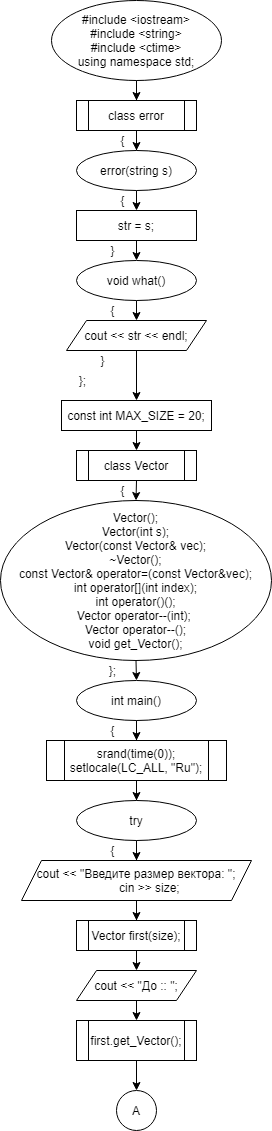
****

12

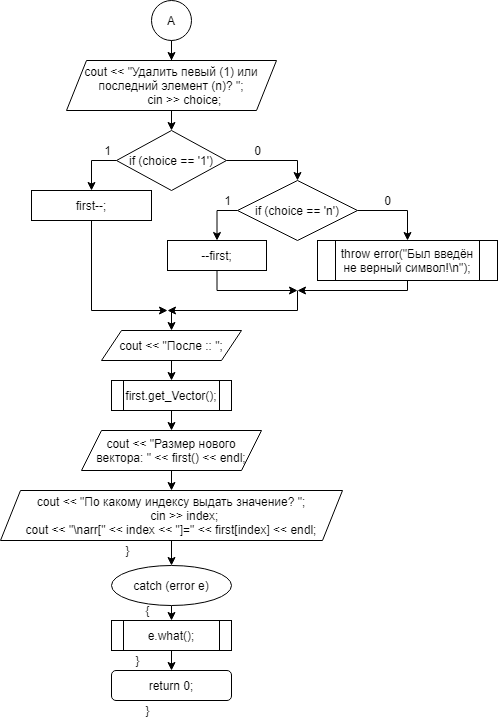


13

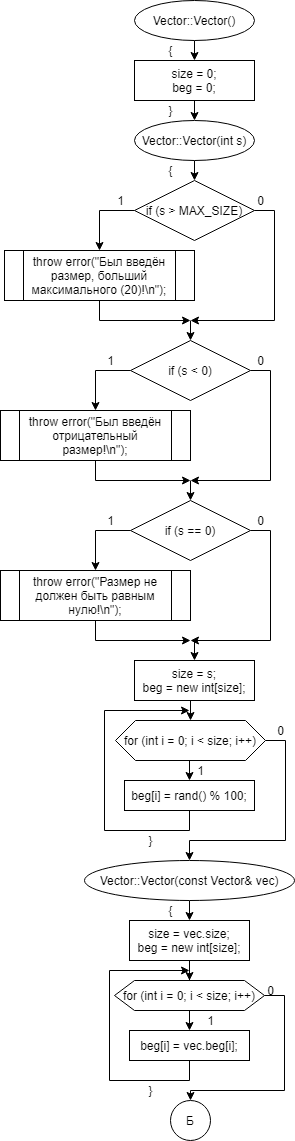
**Реализация 2-м способом**

****

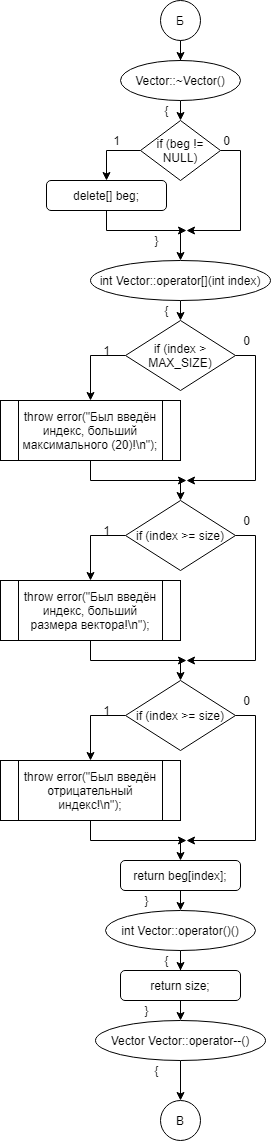
14

****

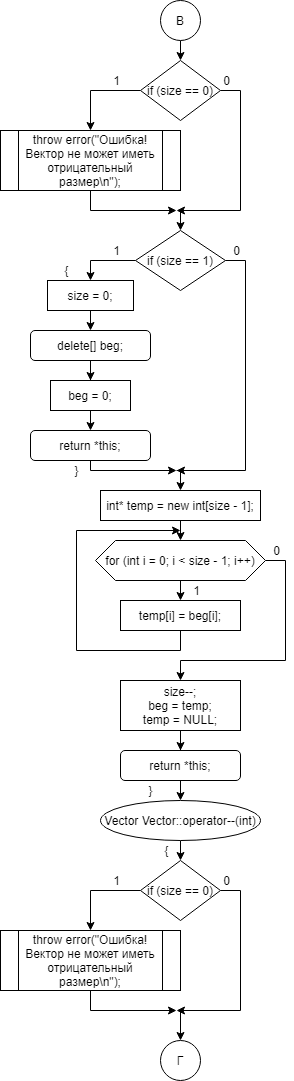
15



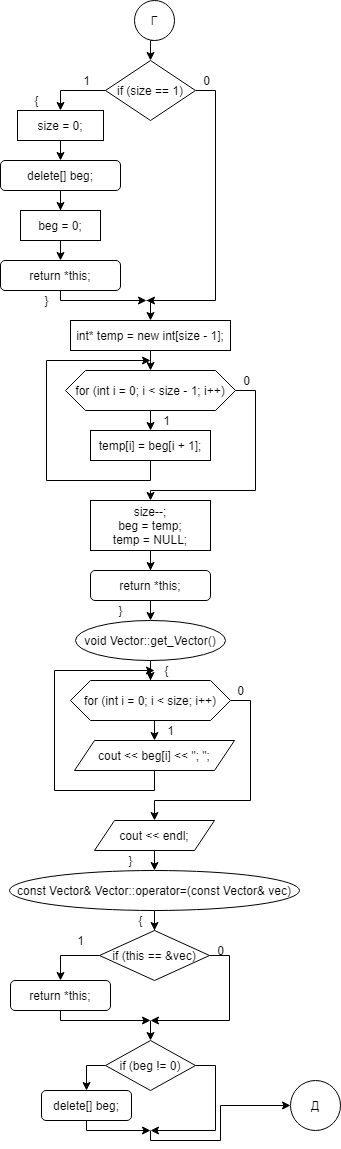
16

****

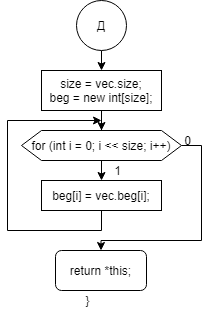
17

****

18



19



20

**Код**

**Реализация 1-м способом**

#include "Vector.h"

#include <ctime>

using namespace std;

int main()

{

srand(time(0));

setlocale(LC\_ALL, "Ru");

int size;

try {

cout << "Введите размер вектора: ";

cin >> size;

Vector first(size);

cout << "До :: ";

first.get\_Vector();

char choice;

cout << "Удалить певый (1) или последний элемент (n)? ";

cin >> choice;

if (choice == '1') {

first--;

}

else if (choice == 'n') {

--first;

}

else

throw 6;

cout << "После :: ";

first.get\_Vector();

cout << "Размер нового вектора: " << first() << endl;

int index;

cout << "По какому индексу выдать значение? ";

cin >> index;

cout << "\narr[" << index << "]=" << first[index] << endl;

}

catch (int) {

cout << "\nОШИБКА" << endl;

}

return 0;

}

21

#pragma once

#include <iostream>

using namespace std;

const int MAX\_SIZE = 30;

class Vector

{

int size; //текущий размер

int\* beg;//указатель на начало вектора

public:

Vector();

Vector(int s);

Vector(const Vector& vec);

~Vector();

const Vector& operator=(const Vector& vec);

int operator[](int index);

int operator()();

Vector operator--(int); //удаление первого элемента

Vector operator--(); //удаление последнего элемента

void get\_Vector();

};

22

#include "Vector.h"

Vector::Vector()

{

size = 0;

beg = 0;

}

Vector::Vector(int s)

{

if (s > MAX\_SIZE) throw 1;

if (s < 0) throw 3;

if (s == 0) throw 5;

size = s;

beg = new int[size];

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

beg[i] = rand()%100;

}

}

Vector::Vector(const Vector& vec)

{

size = vec.size;

beg = new int[size];

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

beg[i] = vec.beg[i];

}

}

Vector::~Vector()

{

if (beg!=NULL)

delete[] beg;

}

int Vector::operator[](int index)

{

if (index > MAX\_SIZE)

throw 1;

if (index >= size)

throw 2;

if (index < 0)

throw 3;

return beg[index];

}

int Vector::operator()()

{

return size;

}

Vector Vector::operator--()

{

if (size == 0)

throw 5;

if (size == 1) {

size = 0;

delete[] beg;

beg = 0;

return \*this;

}

int \*temp = new int[size - 1];

for (int i = 0; i < size - 1; i++) {

temp[i] = beg[i];

}

size--;

beg = temp;

temp = NULL;

return \*this;

}

Vector Vector::operator--(int)

23

{

if (size == 0)

throw 5;

if (size == 1) {

size = 0;

delete[] beg;

beg = 0;

return \*this;

}

int\* temp = new int[size - 1];

for (int i = 0; i < size-1; i++) {

temp[i] = beg[i+1];

}

size--;

beg = temp;

temp = NULL;

return \*this;

}

void Vector::get\_Vector()

{

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

cout << beg[i] << "; ";

}

cout << endl;

}

const Vector& Vector::operator=(const Vector& vec)

{

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

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

size = vec.size;

beg = new int[size];

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

beg[i] = vec.beg[i];

}

return \*this;

}

24

**Реализация 2-м способом**

#include "error.h"

#include "Vector.h"

#include <ctime>

int main()

{

srand(time(0));

setlocale(LC\_ALL, "Ru");

int size;

try {

cout << "Введите размер вектора: ";

cin >> size;

Vector first(size);

cout << "До :: ";

first.get\_Vector();

char choice;

cout << "Удалить певый (1) или последний элемент (n)? ";

cin >> choice;

if (choice == '1') {

first--;

}

else if (choice == 'n') {

--first;

}

else

throw error("Был введён не верный символ!\n");

cout << "После :: ";

first.get\_Vector();

cout << "Размер нового вектора: " << first() << endl;

int index;

cout << "По какому индексу выдать значение? ";

cin >> index;

cout << "\narr[" << index << "]=" << first[index] << endl;

}

catch (error e) {

e.what();

}

return 0;

}

25

#pragma once

#include <iostream>

#include <string>

using namespace std;

class error

{

string str;

public:

error(string s) { str = s; }

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

};

26

#pragma once

#include <iostream>

using namespace std;

const int MAX\_SIZE = 20;

class Vector

{

int size; //текущий размер

int\* beg;//указатель на начало вектора

public:

Vector();

Vector(int s);

Vector(const Vector& vec);

~Vector();

const Vector& operator=(const Vector&vec);

int operator[](int index);

int operator()();

Vector operator--(int); //удаление первого элемента

Vector operator--(); //удаление последнего элемента

void get\_Vector();

};

27

#include "Vector.h"

#include "error.h"

Vector::Vector()

{

size = 0;

beg = 0;

}

Vector::Vector(int s)

{

if (s > MAX\_SIZE) throw error("Был введён размер, больший максимального (20)!\n");

if (s < 0) throw error("Был введён отрицательный размер!\n");

if (s == 0) throw error("Размер не должен быть равным нулю!\n");

size = s;

beg = new int[size];

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

beg[i] = rand() % 100;

}

}

Vector::Vector(const Vector& vec)

{

size = vec.size;

beg = new int[size];

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

beg[i] = vec.beg[i];

}

}

Vector::~Vector()

{

if (beg != NULL)

delete[] beg;

}

int Vector::operator[](int index)

{

if (index > MAX\_SIZE)

throw error("Был введён индекс, больший максимального (20)!\n");

if (index >= size)

throw error("Был введён индекс, больший размера вектора!\n");

if (index < 0)

throw error("Был введён отрицательный индекс!\n");

return beg[index];

}

int Vector::operator()()

{

return size;

}

Vector Vector::operator--()

{

if (size == 0)

throw error("Ошибка! Вектор не может иметь отрицательный размер\n");

if (size == 1) {

size = 0;

delete[] beg;

beg = 0;

return \*this;

}

int\* temp = new int[size - 1];

for (int i = 0; i < size - 1; i++) {

temp[i] = beg[i];

}

size--;

delete[] beg;

beg = temp;

temp = NULL;

return \*this;

28

}

Vector Vector::operator--(int)

{

if (size == 0)

throw error("Ошибка! Вектор не может иметь отрицательный размер\n");

if (size == 1) {

size = 0;

delete[] beg;

beg = 0;

return \*this;

}

int\* temp = new int[size - 1];

for (int i = 0; i < size - 1; i++) {

temp[i] = beg[i + 1];

}

size--;

delete[] beg;

beg = temp;

temp = NULL;

return \*this;

}

void Vector::get\_Vector()

{

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

cout << beg[i] << "; ";

}

cout << endl;

}

const Vector& Vector::operator=(const Vector& vec)

{

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

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

size = vec.size;

beg = new int[size];

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

beg[i] = vec.beg[i];

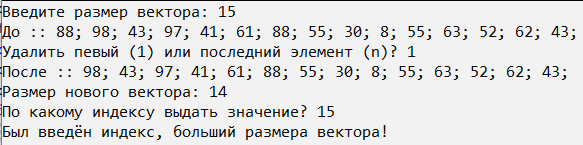
}

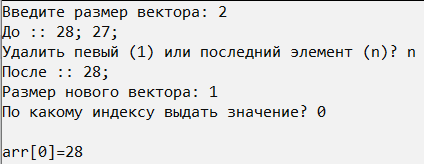
return \*this;

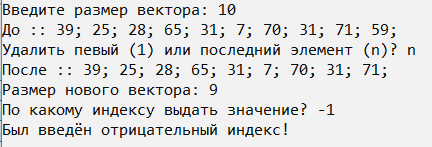
}

29

**Скриншоты**





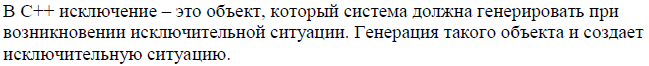


C:\Users\Alex\Desktop\Лаба18.9\sc4.PNG

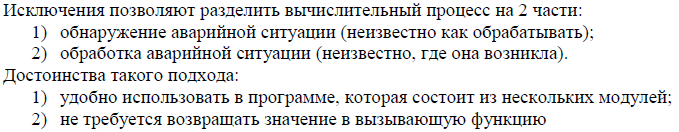
30

**Контрольные вопросы**

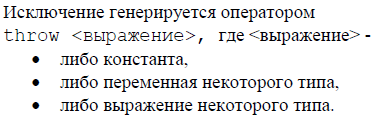


1. 

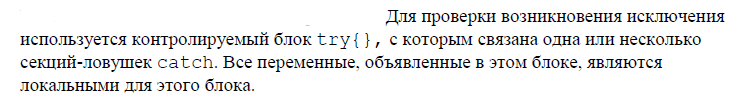


2. 



3. 

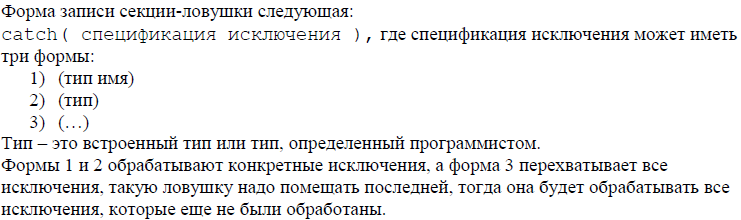


4. 



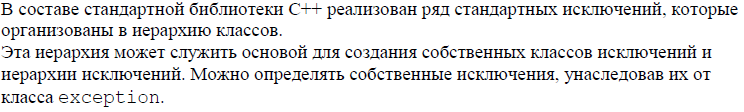
5. Для обработки исключений используется секция ловушка. catch()



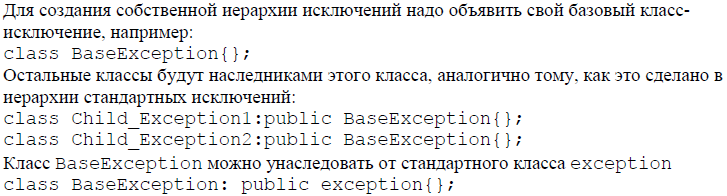


31



7. 



8. 



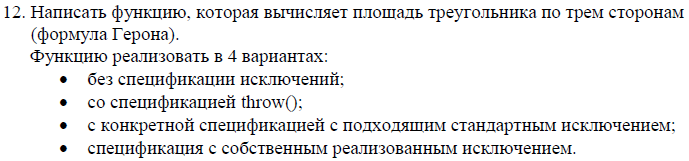
9. Функция генерирует те исключения, которые указаны



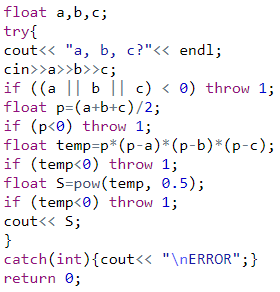
10. Функция не может генерировать исключения.



11.



32

12. 

33