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 10

Выполнил:

Студент группы ИВТ-20-2Б Галинов О. Ю.

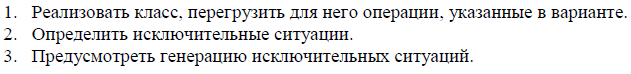
Проверила:

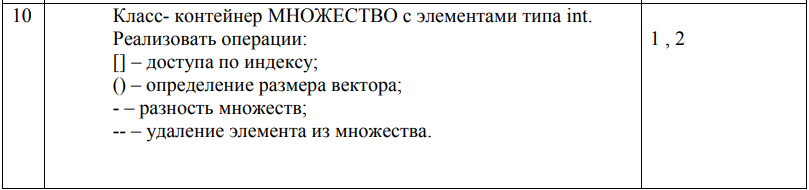
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**





**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Vector с полем типа int – \_size, и указателем data.

**1.2.** Организовать методе size() дружественной классу Vector.

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

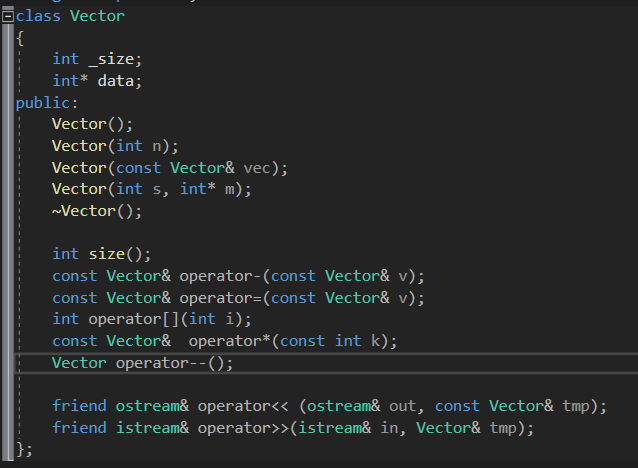
**1.4.** Организовать перегрузку оператора = дружественной классу Vector.

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

**1.6.** При вызове какого-либо действия и срабатывании ошибки (index>size, i>size и тд), вызывается throw с типом данных int, а затем в функции main() с помощью catch выводится сообщение об ошибке.

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

**2.3.** Класс Vector с полем типа int – \_size, и указателем data в заголовочном файле Vector.h.



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

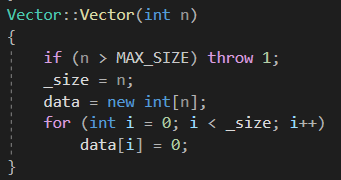
Vector A(size);

**2.4.** Переменная типа int: size, которая отвечает за размер массива.

int size;

**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:

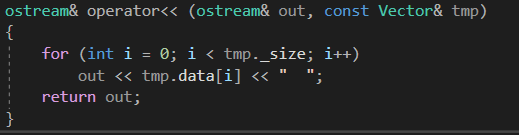
1) введёный пользователем размер массива больше, чем допустимо возможный;

2) индекс элемента меньше 0;

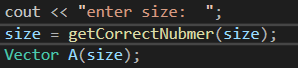
3) индекс элемента больше размера массива;

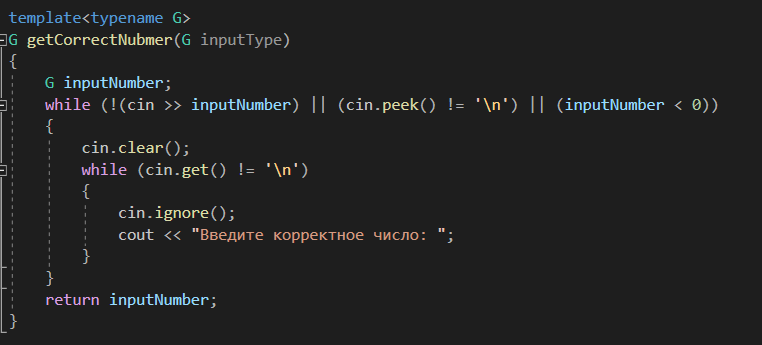
**4.** Для операций ввода и вывода использовались следующие операторы и функции:

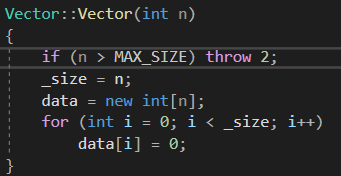
**4.1.** Для вывода массива класса Vector на консоль используется перегрузка оператора <<.



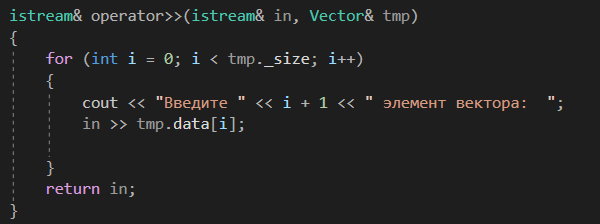
**4.2.** Для ввода массива класса Vector необходимо сначала ввести размер size, который выполнит проверку на ввод, а затем в конструкторе сформировать динамический массив.





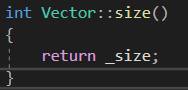


**4.3.** Кроме того ввод может осуществляться при помощи перегрузки оператора >>.

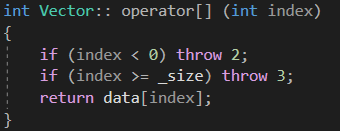
****

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

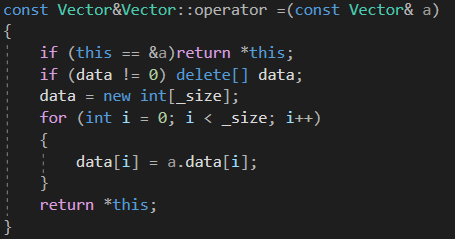
**5.1.** В методе size() класса Vector производится вывод размерности объекта класса Vector.



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

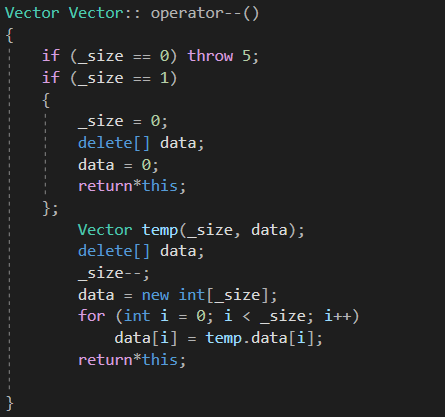


**5.3.** В перегрузке оператора = класса Vector производится вызов элемента множества по индексу объекта класса Vector.

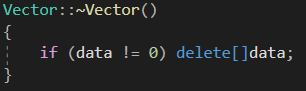


5

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

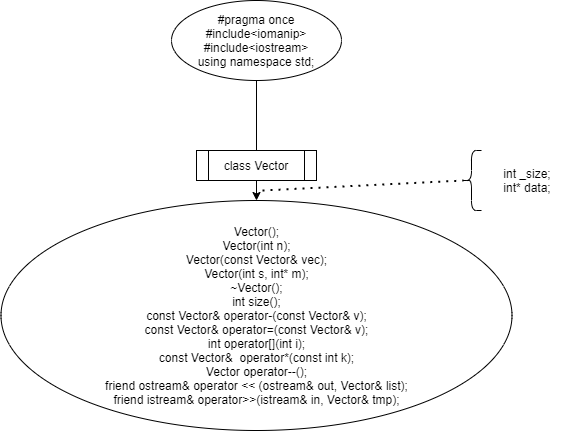


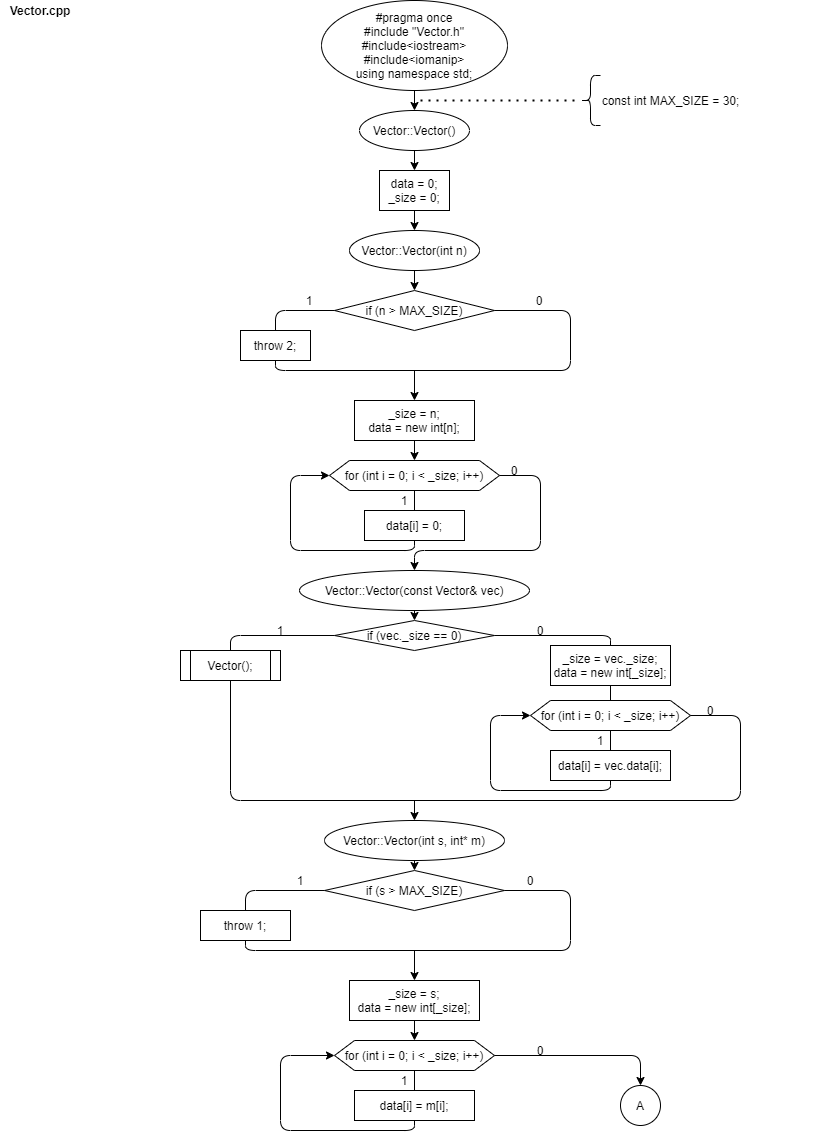
**5.5.** В деструкторе выполнятеся очистка памяти, если указатель не указывает на NULL.

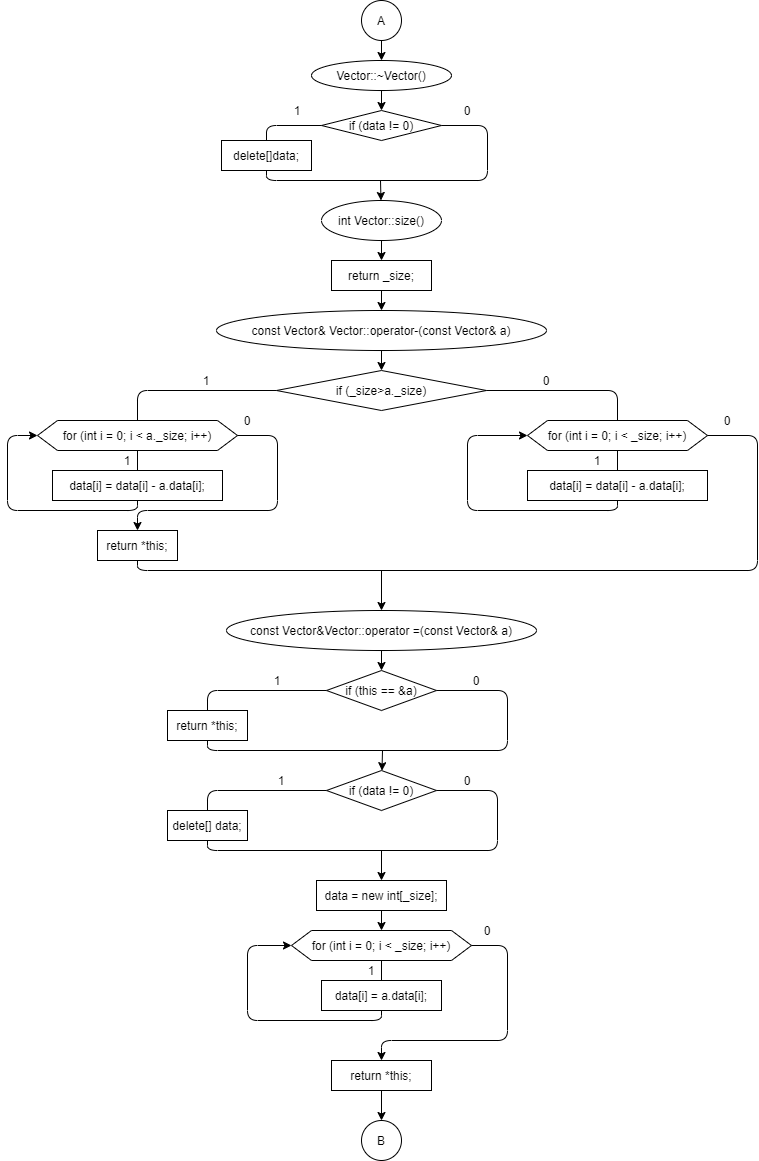


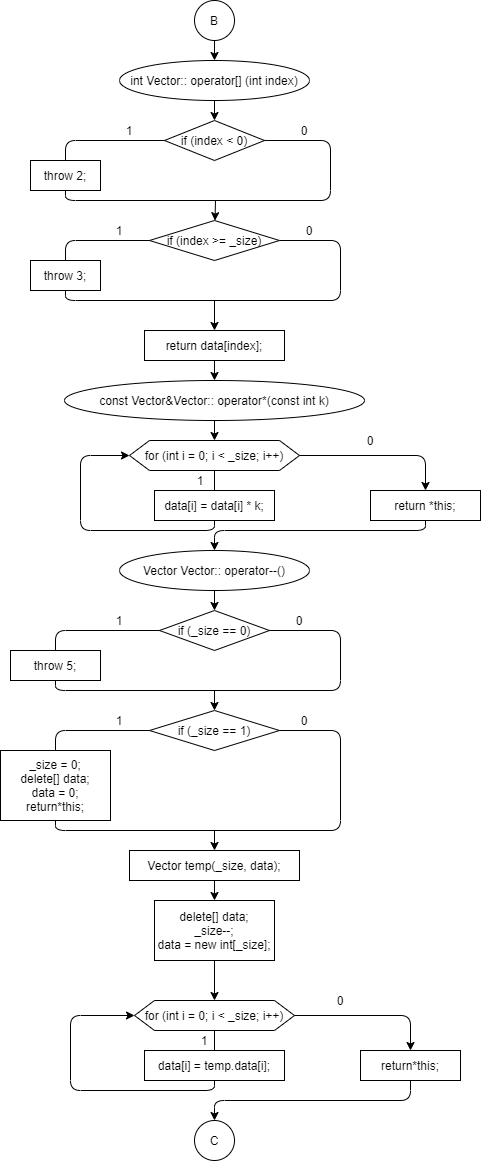
**Блок-схема**

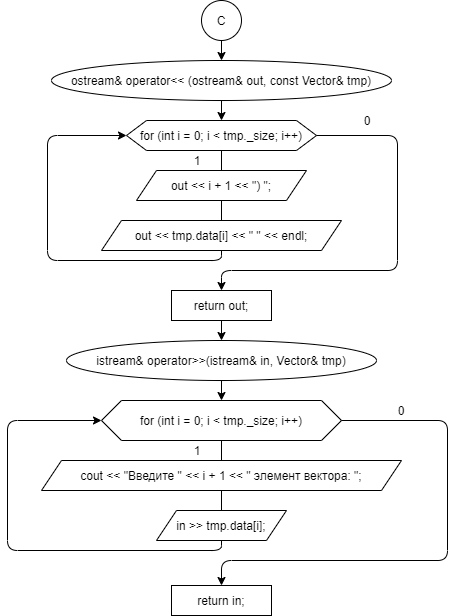
**Реализация 1-м способом**

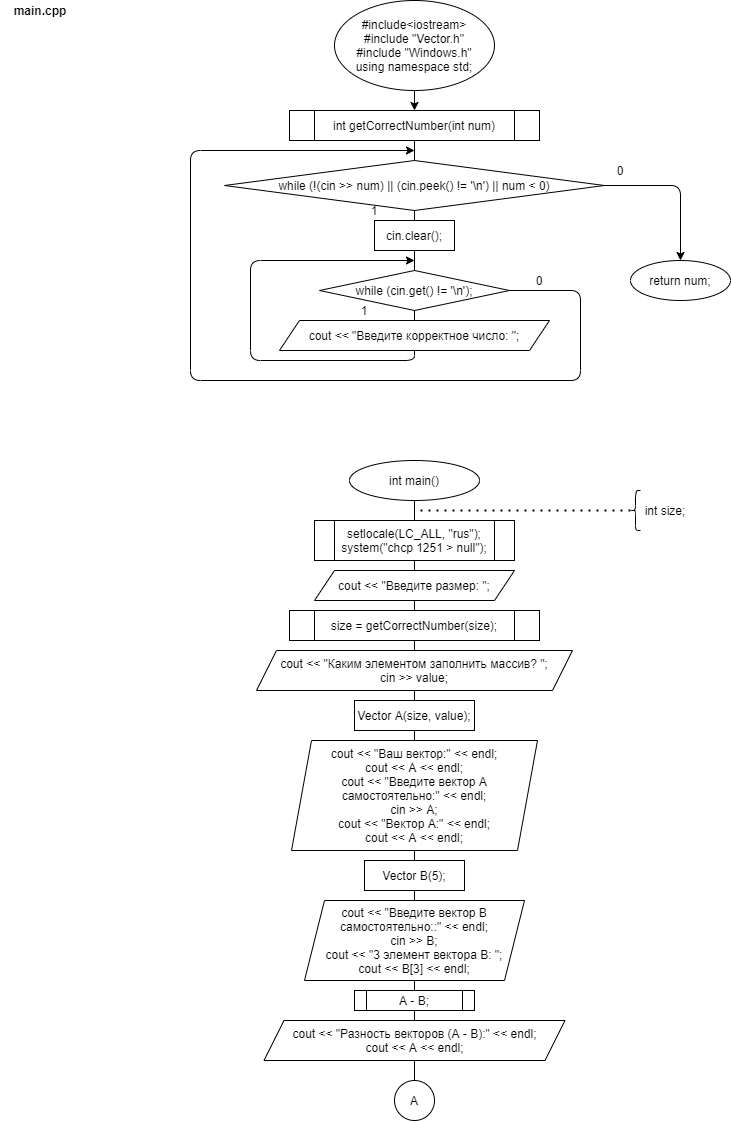
****

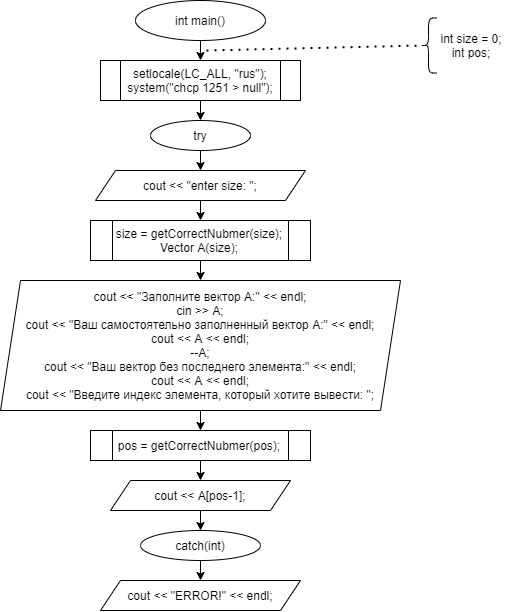












**Код**

**Реализация 1-м способом**

#include <iostream>

#include"Vector.h"

using namespace std;

#include <iostream>

using namespace std;

int getCorrectNumber(int num)

{

while (!(cin >> num) || (cin.peek() != '\n') || num < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "Введите корректное число: ";

}

return num;

}

int main()

{

setlocale(LC\_ALL, "rus");

system("chcp 1251 > null");

try

{

int size = -1;

int pos = -1;

cout << "enter size: ";

size = getCorrectNumber(size);

Vector A(size);

cout << "Заполните вектор А:" << endl;

cin >> A;

cout << "Ваш самостоятельно заполненный вектор А:" << endl;

cout << A << endl;

--A;

cout << "Ваш вектор без последнего элемента:" << endl;

cout << A << endl;

cout << "Введите индекс элемента, который хотите вывести: ";

pos = getCorrectNumber(pos);

cout << A[pos-1];

}

catch(int)

{

cout << "ERROR!" << endl;

}

}

#pragma once

#include<iomanip>

#include<iostream>

using namespace std;

class Vector

{

int \_size;

int\* data;

public:

Vector();

Vector(int n);

Vector(const Vector& vec);

Vector(int s, int\* m);

~Vector();

int size();

const Vector& operator-(const Vector& v);

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

int operator[](int i);

const Vector& operator\*(const int k);

Vector operator--();

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

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

};

#pragma once

#include "Vector.h"

#include<iostream>

#include<iomanip>

using namespace std;

const int MAX\_SIZE = 30;

//construct and destruct

Vector::Vector()

{

data = 0;

\_size = 0;

}

Vector::Vector(int n)

{

if (n > MAX\_SIZE) throw 2;

\_size = n;

data = new int[n];

for (int i = 0; i < \_size; i++)

data[i] = 0;

}

Vector::Vector(const Vector& vec)

{

if (vec.\_size == 0)

Vector();

else

{

\_size = vec.\_size;

data = new int[\_size];

for (int i = 0; i < \_size; i++)

data[i] = vec.data[i];

}

}

Vector::Vector(int s, int\* m)

{

if (s > MAX\_SIZE) throw 1;

\_size = s;

data = new int[\_size];

for (int i = 0; i < \_size; i++)

data[i] = m[i];

}

Vector::~Vector()

{

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

}

//METHODS

int Vector::size()

{

return \_size;

}

const Vector& Vector::operator-(const Vector& a)

{

if (\_size>a.\_size)

{

for (int i = 0; i < a.\_size; i++)

{

data[i] = data[i] - a.data[i];

}

return \*this;

}

else

{

for (int i = 0; i < \_size; i++)

{

data[i] = data[i] - a.data[i];

}

}

}

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

{

if (this == &a)

return \*this;

if (data != 0)

delete[] data;

data = new int[\_size];

for (int i = 0; i < \_size; i++)

{

data[i] = a.data[i];

}

return \*this;

}

int Vector:: operator[] (int index)

{

if (index < 0) throw 2;

if (index >= \_size) throw 3;

return data[index];

}

const Vector&Vector:: operator\*(const int k)

{

for (int i = 0; i < \_size; i++)

data[i] = data[i] \* k;

return \*this;

}

Vector Vector:: operator--()

{

if (\_size == 0) throw 5;

if (\_size == 1)

{

\_size = 0;

delete[] data;

data = 0;

return\*this;

}

Vector temp(\_size, data);

delete[] data;

\_size--;

data = new int[\_size];

for (int i = 0; i < \_size; i++)

data[i] = temp.data[i];

return\*this;

}

//overloads

ostream& operator<< (ostream& out, const Vector& tmp)

{

for (int i = 0; i < tmp.\_size; i++)

{

out << i + 1 << ") ";

out << tmp.data[i] << " " << endl;

}

return out;

}

istream& operator>>(istream& in, Vector& tmp)

{

for (int i = 0; i < tmp.\_size; i++)

{

cout << "Введите " << i + 1 << " элемент вектора: ";

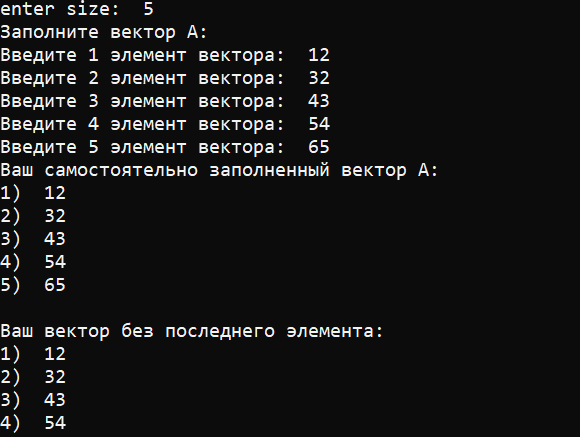
in >> tmp.data[i];

}

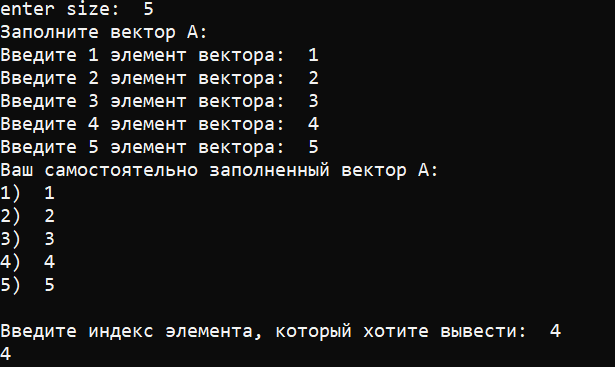
return in;

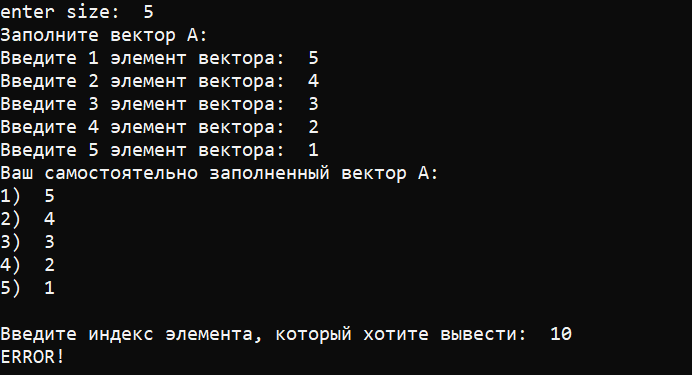
}

**Скриншоты**



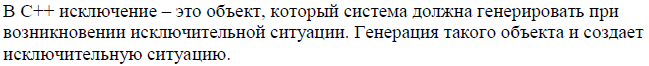




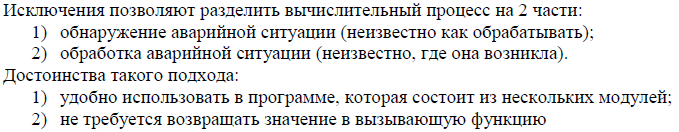


**Контрольные вопросы**

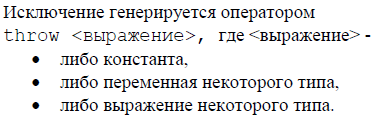


1. 

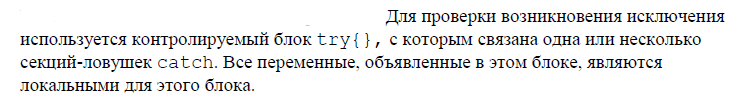


2. 



3. 

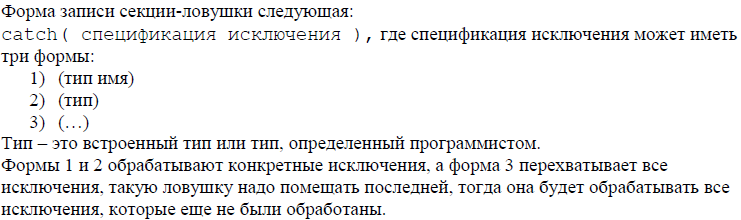


4. 

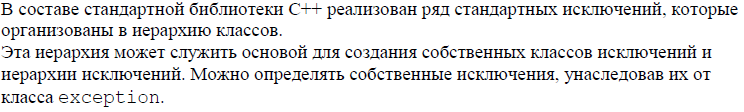


5. Для обработки исключений используется секция ловушка. catch()

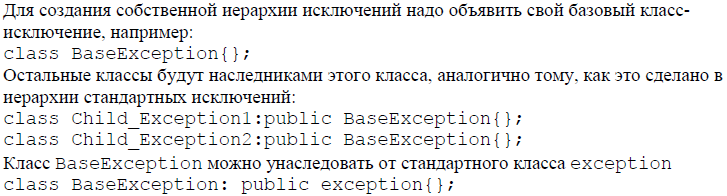






7. 



8. 



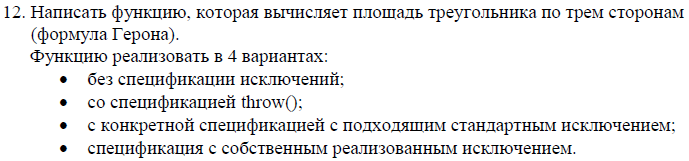
9. Функция генерирует те исключения, которые указаны



10. Функция не может генерировать исключения.



11.



12. 