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 №18.6**

Дисциплина: «Информатика»

Тема: “ Объектно-ориентированное программирование.

АДФ. Контейнеры.”

Вариант 10

Выполнил:

Студент группы ИВТ-20-2Б Галинов О.Ю.

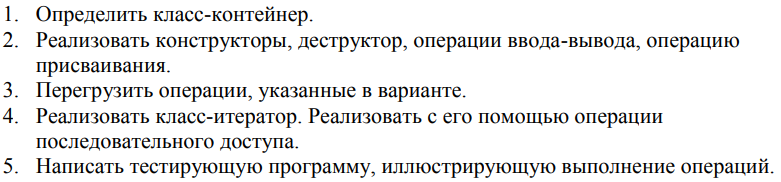
Проверила:

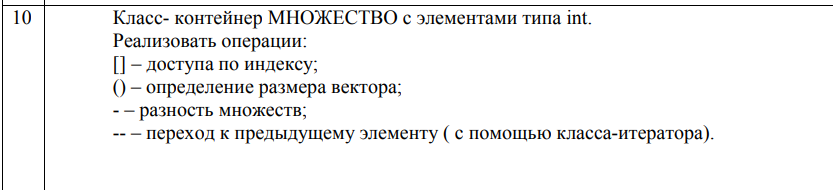
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**





**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Vector с полями типа int – \_size, указателем data, а также переменные beg, end – объекты класса Iterator.

**1.2.** Организовать класс Iterator, дружественный классу Vector, с полем типа int\*elem.

**1.3.** Организовать метод, возвращающий размер массива size().

**1.4.** Организовать перегрузку оператора [] для класса Vector.

**1.5.** Организовать перегрузку оператора - для класса Vector.

**1.6.** Организовать перегрузку оператора <<, дружественной классу Vector.

**1.7.** Организовать перегрузку оператора -- для класса Iterator.

**1.8.** Организовать перегрузку оператора ++ для класса Iterator.

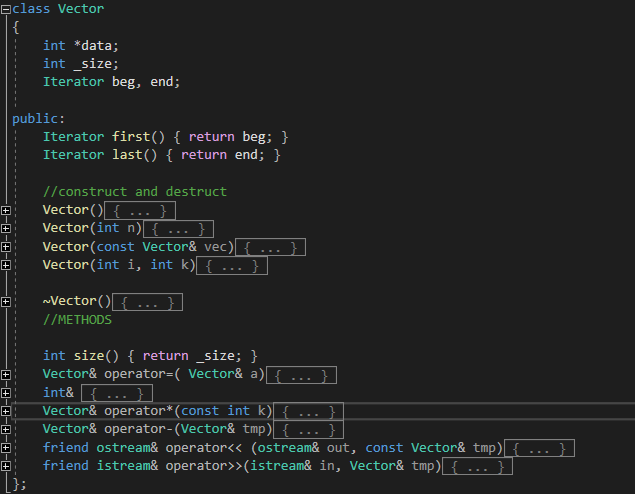
**1.9.** Организовать перегрузку оператора == для класса Iterator.

**1.10.** Организовать перегрузку оператора != для класса Iterator.

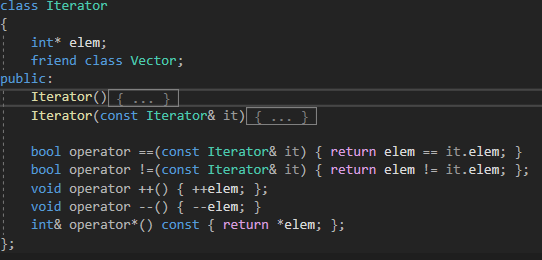
**1.11.** Организовать перегрузку оператора \* для класса Iterator.

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

**2.1.** Класс Vector с полями типа int – size, указателем data, а также переменные beg, end – объекты класса Iterator.



**2.2.** Класс Iterator, дружественный классу Vector, с полем типа int\*elem.



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

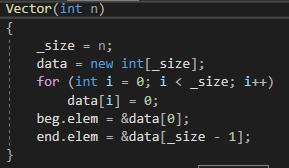
Vector A(size, value);

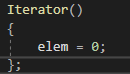
**2.4.** Переменная типа int: size, которая отвечает за размер массива, переменная типа int value, которая используется как временная для заполнения всего массива value.



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

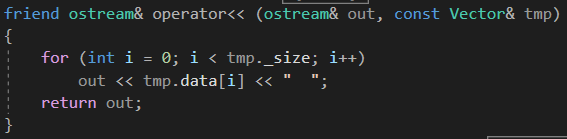
**3.1.** Данные инициализируются в конструкторе с параметром в классе Vector, Iterator.



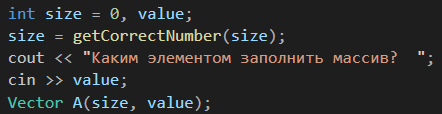


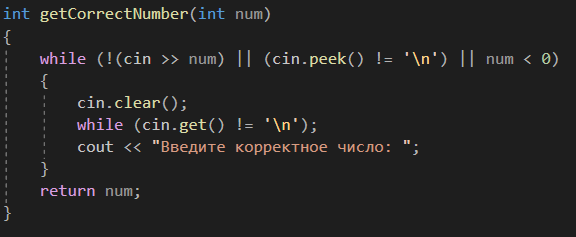
**4.** Для операций ввода и вывода использовались следующие операторы и функции:

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

cout << A << endl; 

**4.3.** Для ввода массива класса Vector необходимо сначала ввести размер size и значение, которым заполняется массив – value, а затем в конструкторе сформировать динамический массив.





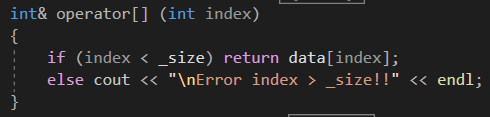
В функции getCorrectNumber() выполняются проверки, исключающие ввод символов, дробных и отрицательных чисел.

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

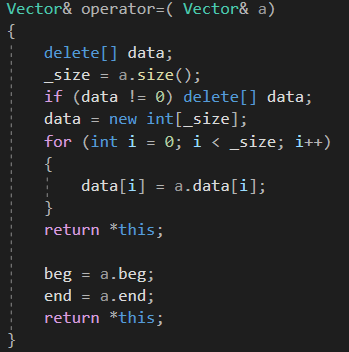
**5.1.** В методе size() класса Vector производится вывод размерности объекта класса Vector.



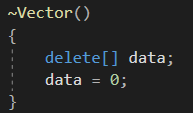
**5.2.** В перегрузке оператора [] класса Vector производится вызов элемента множества по индексу объекта класса Vector.



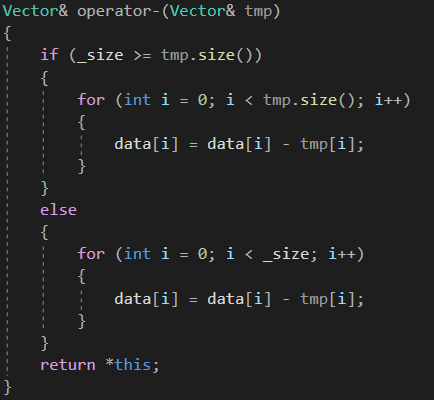
**5.3.** В перегрузке оператора = класса Vector производится вызов элемента множества по индексу объекта класса Vector.



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



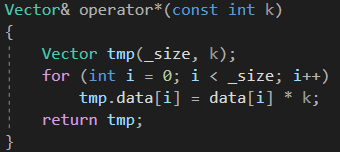
**5.5.** В перегрузке оператора - класса Vector производится вычитание соответствующих элементов Вектора, в зависимости от их размеров, чей размер меньше – такого размера и будет результирующий Вектор.



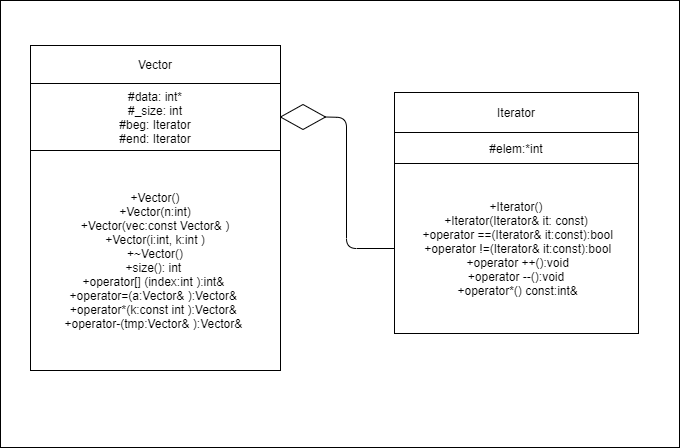
**5.6.** Методы first() и last() возвращают значения первого и последнего элемента массива соответственно.



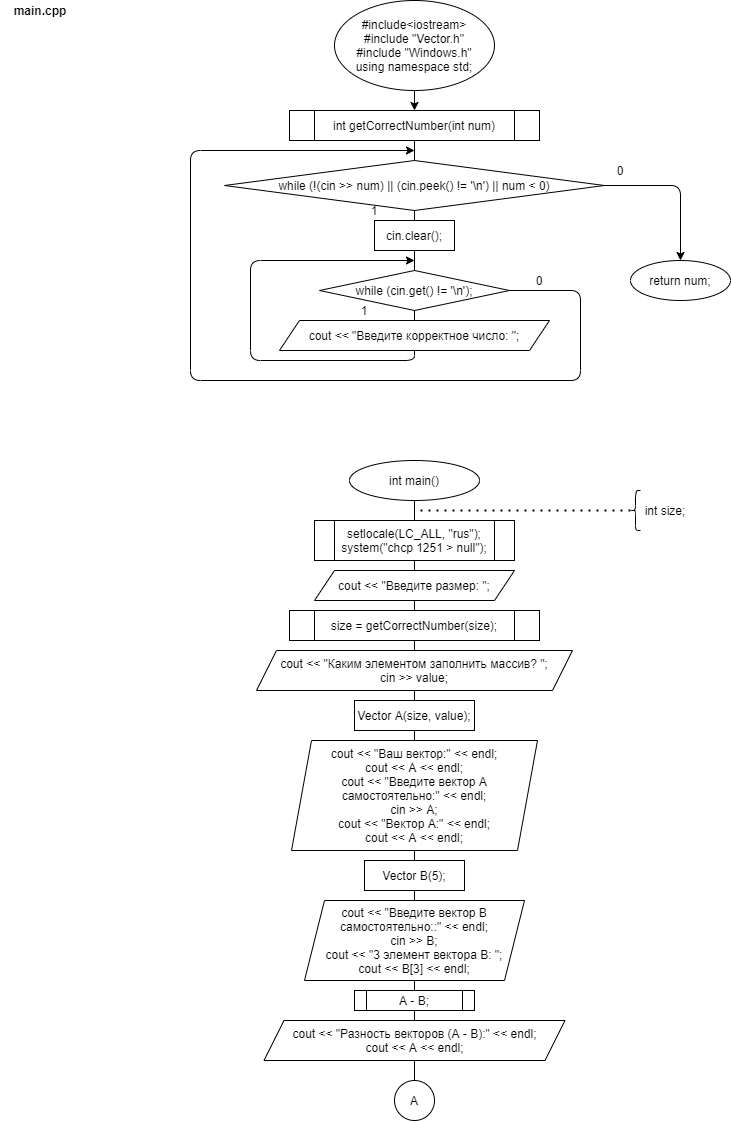
**5.7.** В перегрузке оператора \* выполняется умножение на введенное число.

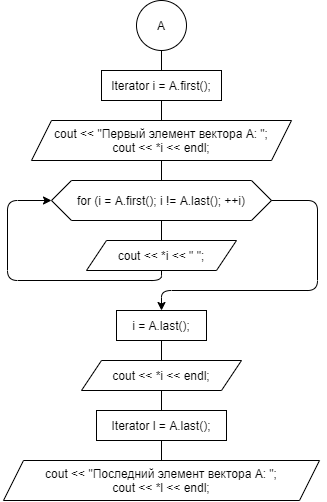


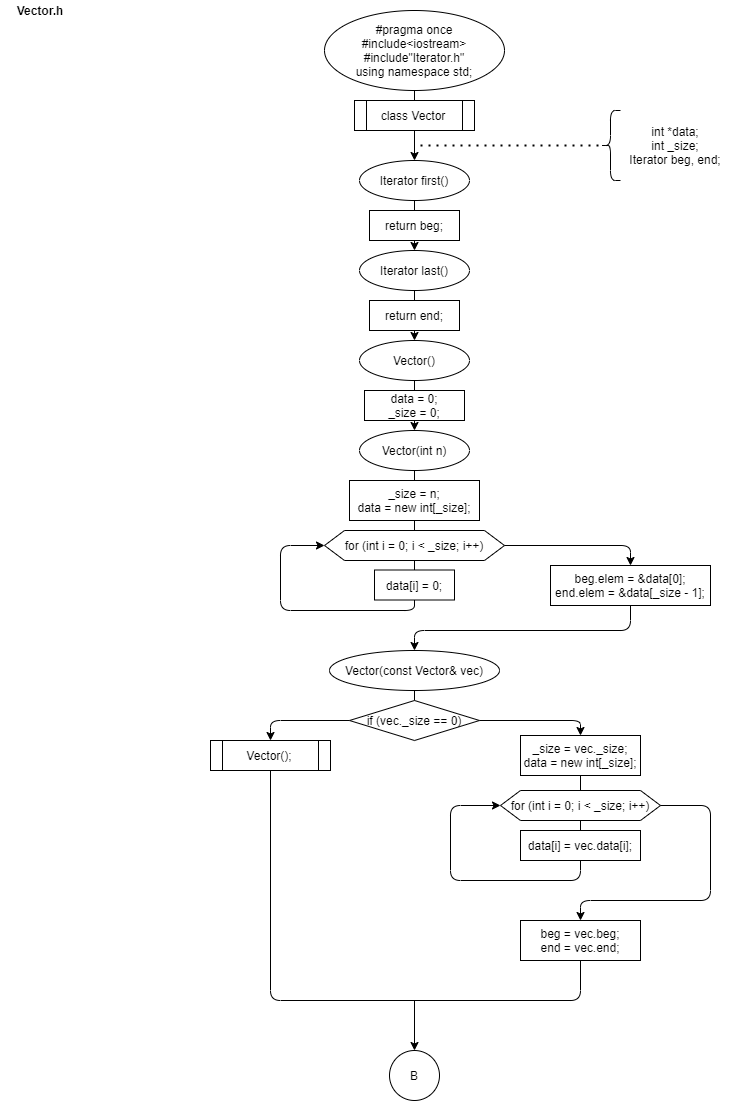
**Диаграммы**

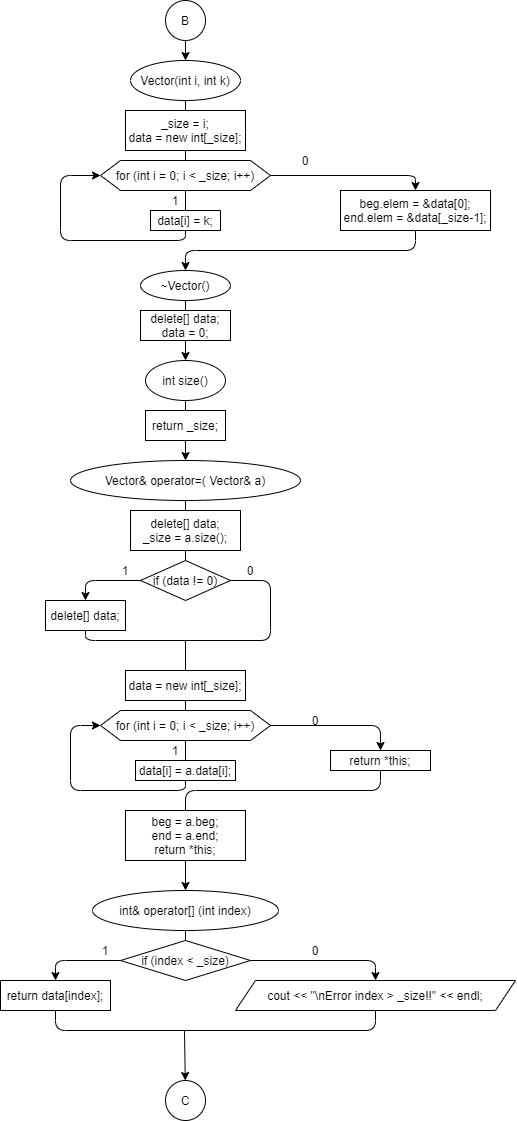


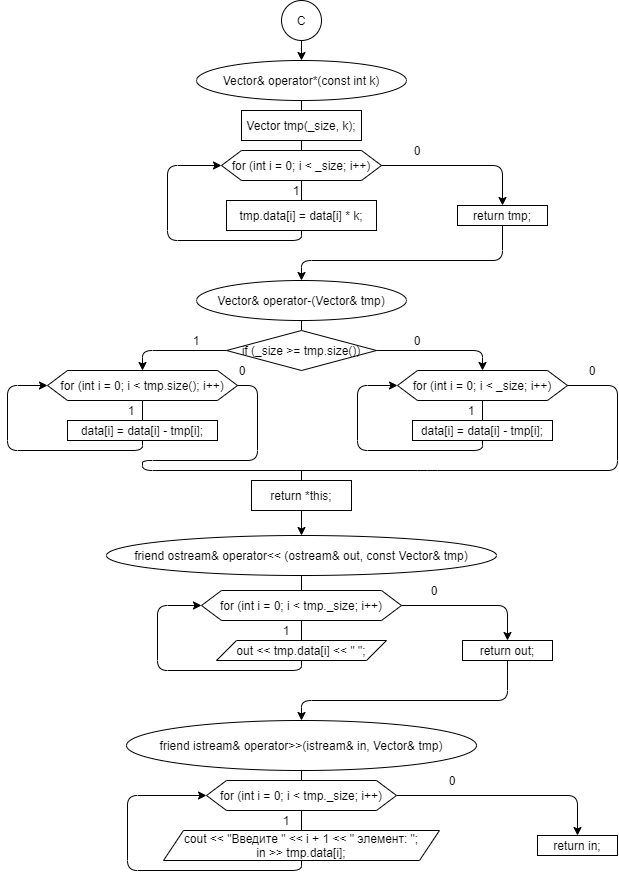
**Блок-схема**

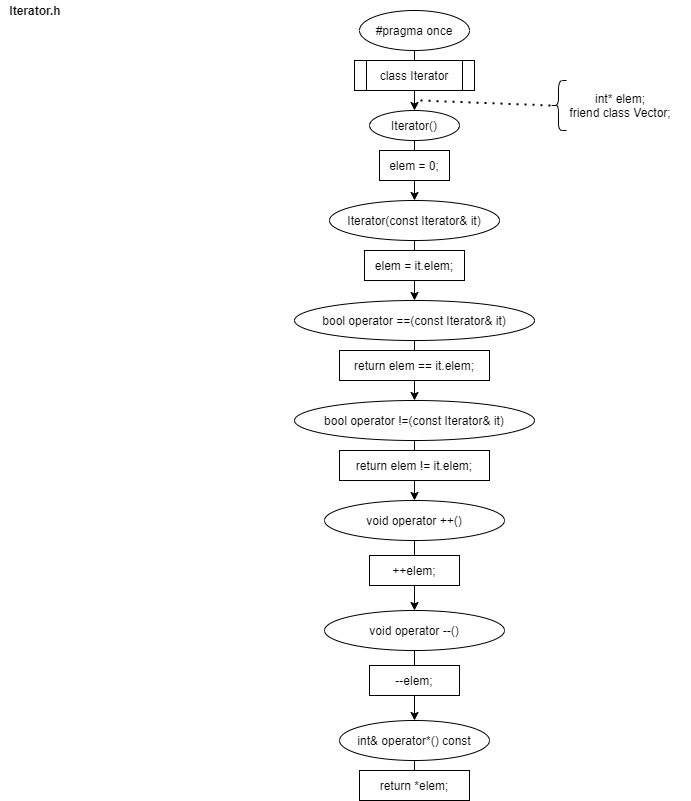












**Код**

#include <iostream>

#include "Vector.h"

#include<Windows.h>

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");

cout << "Введите размер: ";

int size = 0, value;

size = getCorrectNumber(size);

cout << "Каким элементом заполнить массив? ";

cin >> value;

Vector A(size, value);

cout << "Ваш вектор:" << endl;

cout << A << endl;

cout << "Введите вектор A самостоятельно:" << endl;

cin >> A;

cout << "Вектор A:" << endl;

cout << A << endl;

Vector B(5);

cout << "Введите вектор В самостоятельно::" << endl;

cin >> B;

cout << "3 элемент вектора В: ";

cout << B[3] << endl;

A - B;

cout << "Разность векторов (А - В):" << endl;

cout << A << endl;

Iterator i = A.first();

cout << "Первый элемент вектора A: ";

cout << \*i << endl;

//for (i = A.first(); i != A.last(); ++i)

// cout << \*i << " ";

i = A.last();

cout << \*i << endl;

Iterator l = A.last();

cout << "Последний элемент вектора А: ";

cout << \*l << endl;

}

#pragma once

class Iterator

{

int\* elem;

friend class Vector;

public:

Iterator()

{

elem = 0;

};

Iterator(const Iterator& it)

{

elem = it.elem;

}

bool operator ==(const Iterator& it) { return elem == it.elem; }

bool operator !=(const Iterator& it) { return elem != it.elem; };

void operator ++() { ++elem; };

void operator --() { --elem; }

int& operator\*() const { return \*elem; };

};

#pragma once

#include<iostream>

#include"Iterator.h"

using namespace std;

class Vector

{

int \*data;

int \_size;

Iterator beg, end;

public:

Iterator first() { return beg; }

Iterator last() { return end; }

//construct and destruct

Vector()

{

data = 0;

\_size = 0;

}

Vector(int n)

{

\_size = n;

data = new int[\_size];

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

data[i] = 0;

beg.elem = &data[0];

end.elem = &data[\_size - 1];

}

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];

}

beg = vec.beg;

end = vec.end;

}

Vector(int i, int k)

{

\_size = i;

data = new int[\_size];

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

data[i] = k;

beg.elem = &data[0];

end.elem = &data[\_size-1];

}

~Vector()

{

delete[] data;

data = 0;

}

//METHODS

int size()

{ return \_size; }

Vector& operator=( Vector& a)

{

delete[] data;

\_size = a.size();

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

data = new int[\_size];

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

{

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

}

return \*this;

beg = a.beg;

end = a.end;

return \*this;

}

int& operator[] (int index)

{

if (index < \_size) return data[index];

else cout << "\nError index > \_size!!" << endl;

}

Vector& operator\*(const int k)

{

Vector tmp(\_size, k);

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

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

return tmp;

}

Vector& operator-(Vector& tmp)

{

if (\_size >= tmp.size())

{

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

{

data[i] = data[i] - tmp[i];

}

}

else

{

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

{

data[i] = data[i] - tmp[i];

}

}

return \*this;

}

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

{

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

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

return out;

}

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

{

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

{

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

in >> tmp.data[i];

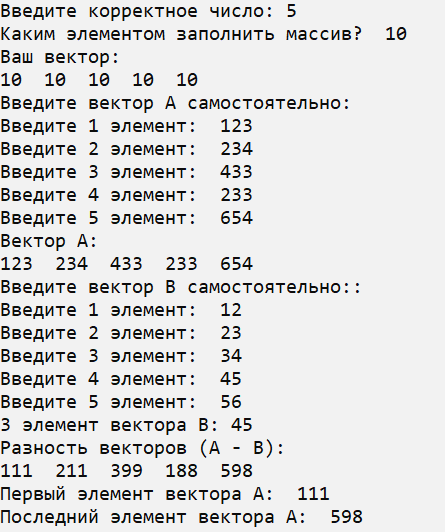
}

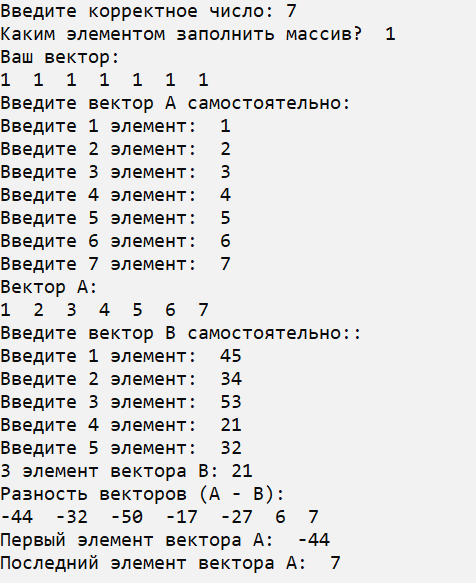
return in;

}

};

**Скриншоты**





**Контрольные вопросы**



1. 





2. 

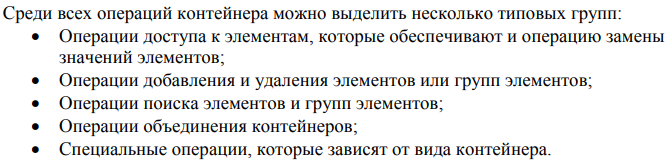


3. 

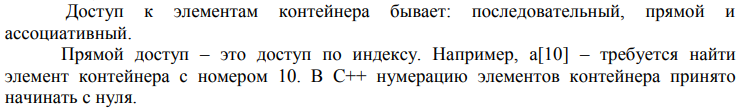


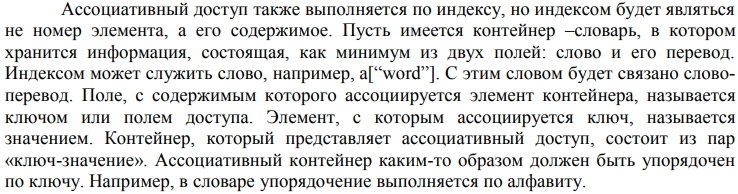
4. 

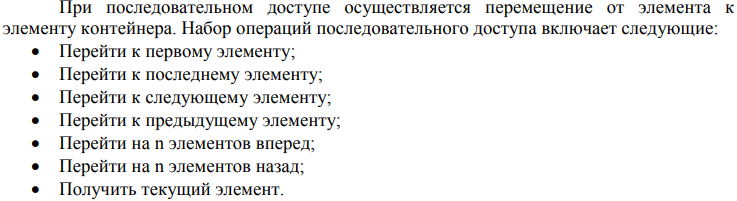


5. 

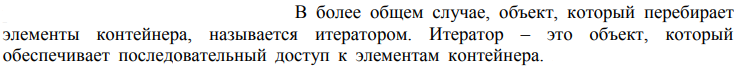


6. 



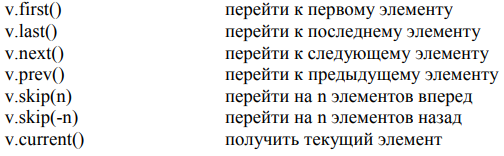




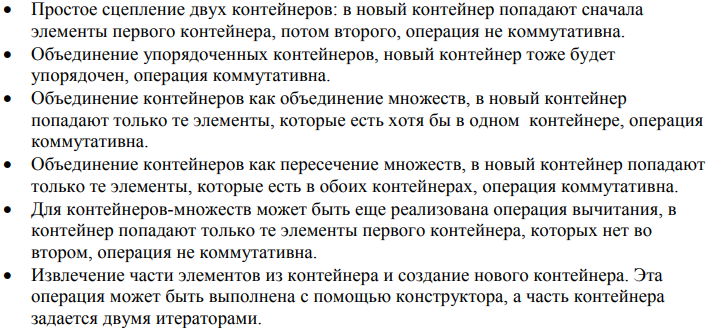
7. 



8. Итератор может быть реализован как часть класса-контейнера в виде набора методов:





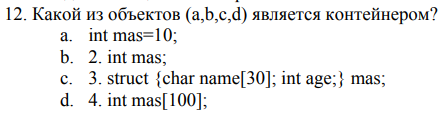
9. 



10. Ассоциативный доступ.



11. Стек.



12. d.





13. d.



14. Прямой доступ.



15. Последовательный доступ.