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 № 18.6**

Дисциплина: Информатика

Тема: «Объектно-ориентированное программирование. АТД. Контейнеры»

Вариант 1

Выполнил работу:

студент группы РИС-20-1Б

Азмагулов Артём Вадимович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь

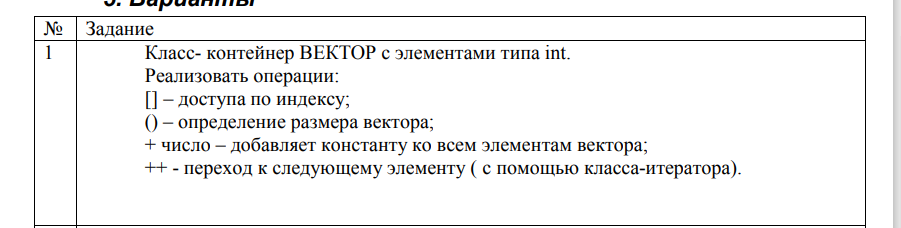
2021

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Реализация класса-контейнера.

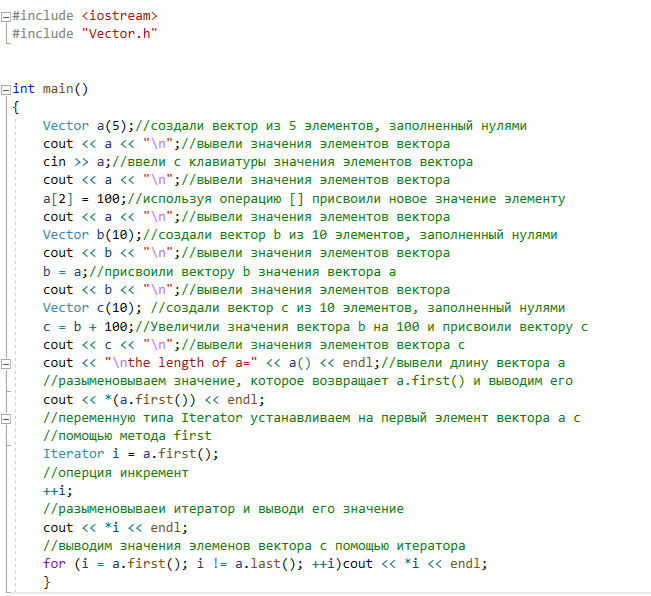
**Постановка задачи**

1. Определить класс-контейнер.
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.
5. Написать тестирующую программу, иллюстрирующую выполнение операций.

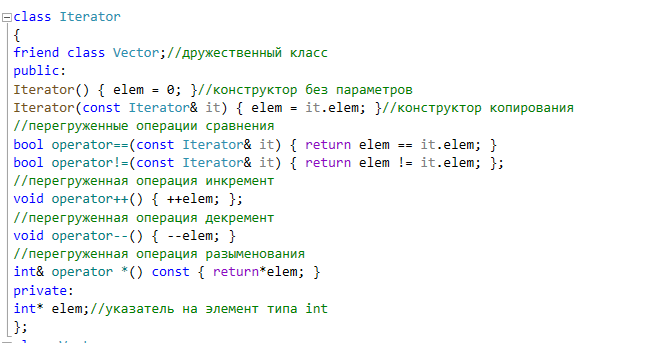


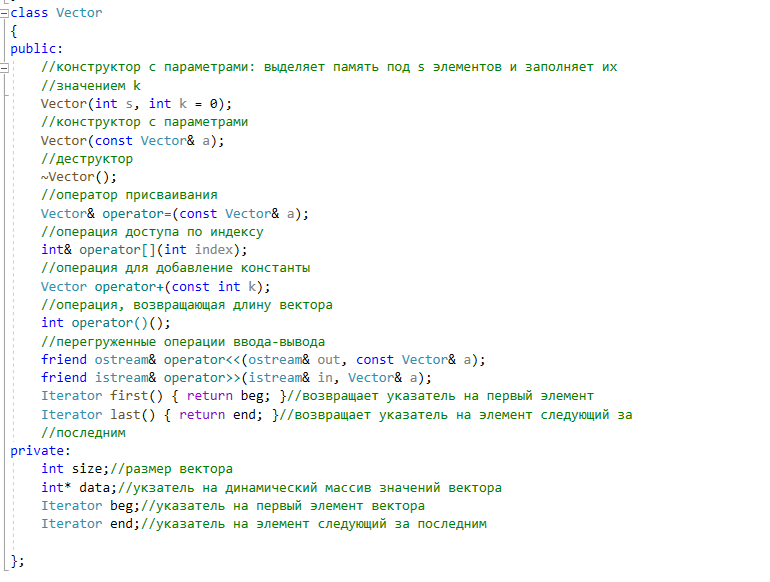
**Анализ задачи**

1. Необходимые действия
2. Продемонстрировать работу всех перегруженных функций в главной функции



2)Создать класс Iterator



3) Создать класс вектор

1. Программный код

Vector.h

#pragma once

#include <iostream>

using namespace std;

class Iterator

{

friend class Vector;//дружественный класс

public:

Iterator() { elem = 0; }//конструктор без параметров

Iterator(const Iterator& it) { elem = it.elem; }//конструктор копирования

//перегруженные операции сравнения

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

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

//перегруженная операция инкремент

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

//перегруженная операция декремент

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

//перегруженная операция разыменования

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

private:

int\* elem;//указатель на элемент типа int

};

class Vector

{

public:

//конструктор с параметрами: выделяет память под s элементов и заполняет их

//значением k

Vector(int s, int k = 0);

//конструктор с параметрами

Vector(const Vector& a);

//деструктор

~Vector();

//оператор присваивания

Vector& operator=(const Vector& a);

//операция доступа по индексу

int& operator[](int index);

//операция для добавление константы

Vector operator+(const int k);

//операция, возвращающая длину вектора

int operator()();

//перегруженные операции ввода-вывода

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

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

Iterator first() { return beg; }//возвращает указатель на первый элемент

Iterator last() { return end; }//возвращает указатель на элемент следующий за

//последним

private:

int size;//размер вектора

int\* data;//укзатель на динамический массив значений вектора

Iterator beg;//указатель на первый элемент вектора

Iterator end;//указатель на элемент следующий за последним

};

18.6.cpp

// 18.6.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include "Vector.h"

int main()

{

Vector a(5);//создали вектор из 5 элементов, заполненный нулями

cout << a << "\n";//вывели значения элементов вектора

cin >> a;//ввели с клавиатуры значения элементов вектора

cout << a << "\n";//вывели значения элементов вектора

a[2] = 100;//используя операцию [] присвоили новое значение элементу

cout << a << "\n";//вывели значения элементов вектора

Vector b(10);//создали вектор b из 10 элементов, заполненный нулями

cout << b << "\n";//вывели значения элементов вектора

b = a;//присвоили вектору b значения вектора a

cout << b << "\n";//вывели значения элементов вектора

Vector c(10); //создали вектор c из 10 элементов, заполненный нулями

c = b + 100;//Увеличили значения вектора b на 100 и присвоили вектору c

cout << c << "\n";//вывели значения элементов вектора c

cout << "\nthe length of a=" << a() << endl;//вывели длину вектора a

//разыменовываем значение, которое возвращает a.first() и выводим его

cout << \*(a.first()) << endl;

//переменную типа Iterator устанавливаем на первый элемент вектора а с

//помощью метода first

Iterator i = a.first();

//оперция инкремент

++i;

//разыменовываеи итератор и выводи его значение

cout << \*i << endl;

//выводим значения элеменов вектора с помощью итератора

for (i = a.first(); i != a.last(); ++i)cout << \*i << endl;

}

};

Vector.cpp

#include "Vector.h"

#include <iostream>

using namespace std;

Vector::Vector(int s, int k)

{

size = s;

data = new int[size];

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

data[i] = k;

beg.elem = &data[0];

end.elem = &data[size];

}

//конструктор копирования

Vector::Vector(const Vector& a)

{

size = a.size;

data = new int[size];

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

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

beg = a.beg;

end = a.end;

}

//деструктор

Vector::~Vector()

{

delete[]data;

data = 0;

}

//операция присваивания

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

{

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

size = a.size;

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

data = new int[size];

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

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

beg = a.beg;

end = a.end;

return \*this;

}

//операция доступа по индексу

int& Vector::operator[](int index)

{

if (index < size) return data[index];

else cout << "\nError! Index>size";

}

//операция для добавления константы

Vector Vector::operator+(const int k)//+k

{

Vector temp(size);

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

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

return temp;

}

//операция для получения длины вектора

int Vector::operator ()()

{

return size;//len();

}

//операции для ввода-выода

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

{

for (int i = 0; i < a.size/\*len()\*/; ++i)

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

return out;

}

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

{

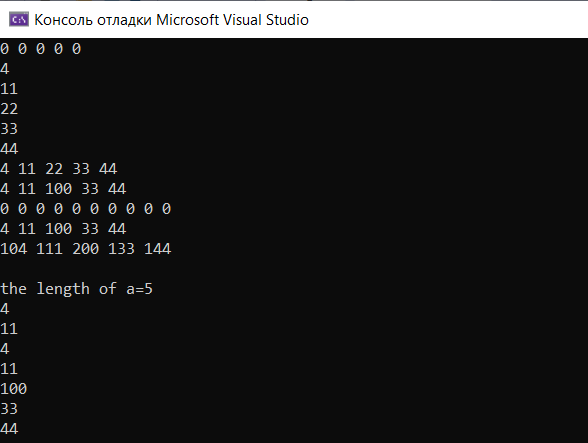
for (int i = 0; i <a.size /\*a.len()\*/; ++i)

in >> a.data[i];

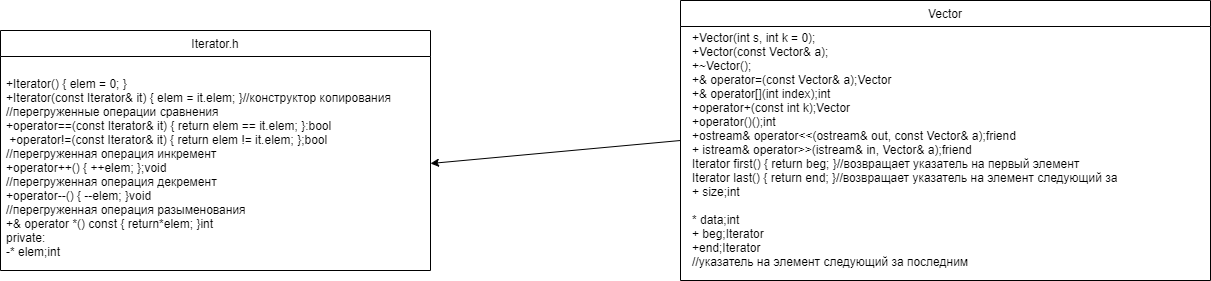
return in;

}

1. Демонстрация работы программы



1. Диаграммы классов



1. Контрольные вопросы
2. Что такое абстрактный тип данных? Привести примеры АТД.

- тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов.

- в качестве контейнера можно привести класс

1. Привести примеры абстракции через параметризацию.

- в качестве параметра можно передавать разные переменные в одну и ту же функцию

1. Привести примеры абстракции через спецификацию.

- реализация операторов сложения и вычитания в теле класса

1. Что такое контейнер? Привести примеры.

- набор однотипных элементов

- в качестве примера можно привести массивы

1. Какие группы операций выделяют в контейнерах?

* Операции доступа к элементам, которые обеспечивают и операцию замены значений элементов;
* Операции добавления и удаления элементов или групп элементов;
* Операции поиска элементов и групп элементов;
* Операции объединения контейнеров;
* Специальные операции, которые зависят от вида контейнера.

1. Какие виды доступа к элементам контейнера существуют? Привести примеры.

- последовательный, прямой и ассоциативный.

1. Что такое итератор?

- это объект, который обеспечивает последовательный доступ к элементам контейнера

1. Каким образом может быть реализован итератор?

- Итератор можно реализовать как класс, представляющий такой же набор операций.

1. Каким образом можно организовать объединение контейнеров?

* Простое сцепление двух контейнеров: в новый контейнер попадают сначала элементы первого контейнера, потом второго, операция не коммутативна.
* Объединение упорядоченных контейнеров, новый контейнер тоже будет упорядочен, операция коммутативна.
* Объединение контейнеров как объединение множеств, в новый контейнер попадают только те элементы, которые есть хотя бы в одном контейнере, операция коммутативна.
* Объединение контейнеров как пересечение множеств, в новый контейнер попадают только те элементы, которые есть в обоих контейнерах, операция коммутативна.
* Для контейнеров-множеств может быть еще реализована операция вычитания, в контейнер попадают только те элементы первого контейнера, которых нет во втором, операция не коммутативна.
* Извлечение части элементов из контейнера и создание нового контейнера. Эта операция может быть выполнена с помощью конструктора, а часть контейнера задается двумя итераторами.

1. Какой доступ к элементам предоставляет контейнер, состоящий из элементов

«ключ-значение»?

- ассоциативный

1. Как называется контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера?

- стек

1. Какой из объектов (a,b,c,d) является контейнером?
   * + int mas=10;
     + 2. int mas;
     + 3. struct {char name[30]; int age;} mas;
     + 4. int mas[100];

- 4

1. Какой из объектов (a,b,c,d) не является контейнером?

* int a[]={1,2,3,4,5};
* 2. int mas[30];
* 3. struct {char name[30]; int age;} mas[30];
* 4. int mas;

- 4

1. Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?

- прямой

1. Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера?

- последовательный