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 №6**

Дисциплина: «Основы алгоритмизации и программирования»

Тема : "АТД. Контейнеры"

Выполнил работу:

Студент группы РИС-22-1Б

Батин В.В.

Проверил:

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи**

Вариант 5:

Класс - контейнер ВЕКТОР с элементами типа int.

Реализовать операции :

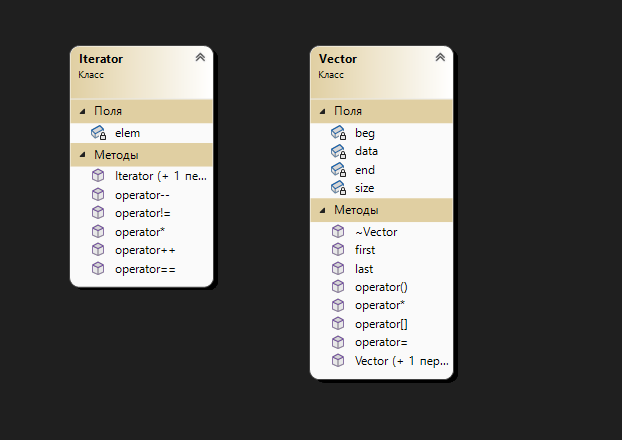
[] – доступа по индексу;

int() – определение размера вектора;

\*вектор – умножение элементов векторов a[i] \* b[i];

+n – переход вправо к элементу с номером n(с помощью класса - итератора).

**Диаграммы классов**

****

#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\*(Vector&);

//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;

int\* data;

Iterator beg;//указатель на первый элемент вектора

Iterator end;//указатель на элемент следующий за последним

};

**Определение компонентных функций**

#include "Vector.h"

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 умножения векторов

Vector Vector::operator\*(Vector& k)//\*вектор

{

Vector temp(size);

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

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

return temp;

}

//операция для получения длины вектора

int Vector::operator()()

{

int len = size;

return len;

}

//операции для ввода-выода

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

{

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

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

return out;

}

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

{

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

in >> a.data[i];

return in;

}

**Функция main**

#include <iostream>

#include "Vector.h"

using namespace std;

void main()

{

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

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

cout << "a > ";

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

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

a[2] = 0;//используя операцию [] присвоили новое значение элементу

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

cout << endl;

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

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

cout << "b > ";

cin >> b;

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

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

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

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

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

c = a \* b;//умножили а на b и присвоили это вектору c

cout << "a \* b : " << c << "\n";//вывели значения элементов вектора c

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

cout << endl;

//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

//разыменовываеи итератор и выводи его значение

cout << \*i << endl;

cout << endl;

//выводим значения элеменов вектора с помощью итератора

int n;

cout << "\nInput element's number: > "; cin >> n;

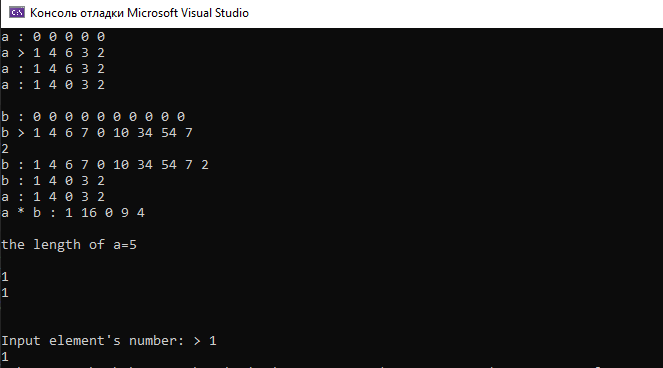
n--;

for (int j = 0; j < n; ++j) ++i;

cout << \*i;

}

**Результаты работы программы**

****

**Ответы на контрольные вопросы**

1. Что такое абстрактный тип данных? Привести примеры АТД.

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

Примеры: стек, очередь, список

2. Привести примеры абстракции через параметризацию.

функция, передаваемые параметры которой являются формальными, а фактические связываются с ними в момент использования такой абстракции.

Procedure Sort(Var A: TArray; N: Word);

3. Привести примеры абстракции через спецификацию.

позволяет абстрагироваться от процесса вычислений, описанных в теле процедуры, до уровня знаний лишь того, что данная процедура должна в итоге реализовать.

4. Что такое контейнер? Привести примеры.

Контейнер - набор однотипных элементов. Самый простой пример контейнера - массив.

5. Какие группы операций выделяют в контейнерах?

Среди всех операций контейнера можно выделить несколько типовых групп: • 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.

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

Доступ к элементам контейнера бывает: последовательный, прямой и ассоциативный.  
Прямой доступ — это доступ по индексу. Например, a[10] — требуется найти элемент контейнера с номером 10. В С++ нумерацию элементов контейнера принято начинать с нуля.  
Ассоциативный доступ также выполняется по индексу, но индексом будет являться не номер элемента, а его содержимое.   
При последовательном доступе осуществляется перемещение от элемента к элементу контейнера.

7. Что такое итератор?

Итератор - это объект, который обеспечивает доступ к его элементам, используя указатели.

8. Каким образом может быть реализован итератор?

Можно реализовать как класс или как часть класса.

9. 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.

10. Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?

Ассоциативный доступ.

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

Такой контейнер называется стеком.

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

Ответ: d

13. Какой из объектов (a,b,c,d) не является контейнером?  
a. int a[]={1,2,3,4,5};

b. int mas[30];  
c. struct {char name[30]; int age;} mas[30];  
d. int mas;

Ответ: d

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

Последовательный доступ.

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

Прямой доступ.