Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №6

Тема: «АТД. Контейнеры»

Выполнил

Студент группы РИС-22-2б

Арапов М.В.

Проверил доц. Кафедры ИТАС

Полякова О. А.

Пермь 2023

# Постановка задачи

Класс- контейнер ВЕКТОР с элементами типа int.

Реализовать операции:

[] - доступа по индексу;

+ вектор - сложение элементов векторов а[i]+b[i];

+ число - добавляет константу ко всем элементам вектора;

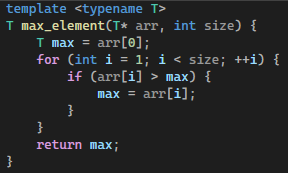
-- - переход к предыдущему элементу ( с помощью класса-итератора).

**Контрольные вопросы**

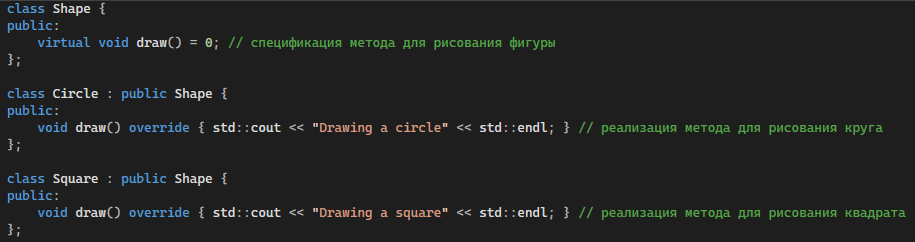
1. Что такое абстрактный тип данных? Привести примеры АТД.

Абстрактный тип данных (АТД) - это тип данных, определенный не через его внутреннюю реализацию, а через набор операций, которые можно выполнять над этим типом данных. Примеры АТД: стек, очередь, список, дерево и т.д.

1. Привести примеры абстракции через параметризацию.



1. Привести примеры абстракции через спецификацию.



4. Что такое контейнер? Привести примеры.

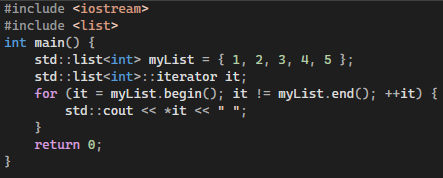
Контейнер - это объект, который содержит другие объекты и предоставляет методы для добавления, удаления и доступа к этим объектам. Примеры: массив, список, стек, очередь, хэш-таблица и т.д.

5. Какие группы операций выделяют в контейнерах?

Группы операций в контейнерах: операции добавления и удаления элементов, операции доступа к элементам, операции поиска элементов, операции сортировки и т.д.

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

Пример доступа к элементам списка с помощью итератора:



7. Что такое итератор?

Итератор - это объект, который позволяет перебирать элементы контейнера последовательно, без доступа к его внутренней реализации.

8. Каким образом может быть реализован итератор?

Итератор может быть реализован как указатель на элемент контейнера, либо как отдельный объект, который хранит информацию о текущей позиции в контейнере.

9. Каким образом можно организовать объединение контейнеров?

Объединение контейнеров можно организовать с помощью операций объединения, пересечения и разности множеств.

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

Контейнер, состоящий из элементов "ключ-значение" (например, ассоциативный массив или словарь), предоставляет доступ к элементам по ключу.

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

Контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера, называется стеком.

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

a. int mas=10;

b. 2. int mas;

c. 3. struct {char name[30]; int age;} mas;

**d. 4. int mas[100];**

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

a. int a[]={1,2,3,4,5};

**b. 2. int mas[30];**

c. 3. struct {char name[30]; int age;} mas[30];

d. 4. int mas;

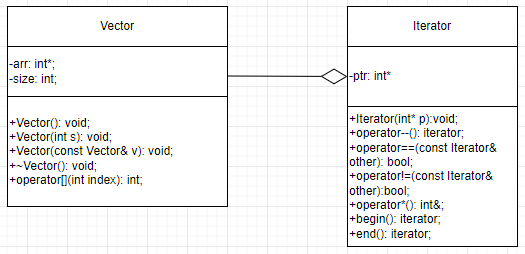
14. Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?

Доступ к элементам контейнера будет осуществляться по индексу (для массива).

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

Доступ к элементам контейнера будет осуществляться с помощью итератора или указателя на текущий элемент списка.

UML таблица



Код программы

#include <iostream>

#include <cstring>

using namespace std;

class Vector {

private:

int\* arr;

int size;

public:

// Конструкторы

Vector() {

arr = nullptr;

size = 0;

}

Vector(int s) {

arr = new int[s];

size = s;

}

Vector(const Vector& v) {

size = v.size;

arr = new int[size];

memcpy(arr, v.arr, sizeof(int) \* size);

}

// Деструктор

~Vector() {

if (arr != nullptr) {

delete[] arr;

}

}

// Оператор присваивания

Vector& operator=(const Vector& v) {

if (this != &v) {

if (arr != nullptr) {

delete[] arr;

}

size = v.size;

arr = new int[size];

memcpy(arr, v.arr, sizeof(int) \* size);

}

return \*this;

}

// Оператор доступа по индексу

int operator[](int index) const {

return arr[index];

}

// Оператор сложения двух векторов

Vector operator+(const Vector& v) const {

Vector result(size);

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

result.arr[i] = arr[i] + v.arr[i];

}

return result;

}

// Оператор добавления числа к вектору

Vector operator+(int number) const {

Vector result(size);

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

result.arr[i] = arr[i] + number;

}

return result;

}

// Оператор перехода к предыдущему элементу

class Iterator {

private:

int\* ptr;

public:

Iterator(int\* p) {

ptr = p;

}

Iterator operator--() {

--ptr;

return \*this;

}

bool operator==(const Iterator& other) const {

return ptr == other.ptr;

}

bool operator!=(const Iterator& other) const {

return ptr != other.ptr;

}

int& operator\*() {

return \*ptr;

}

};

Iterator begin() {

return Iterator(arr);

}

Iterator end() {

return Iterator(arr - 1);

}

// Оператор ввода

friend istream& operator>>(istream& in, Vector& v) {

in >> v.size;

v.arr = new int[v.size];

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

in >> v.arr[i];

}

return in;

}

// Оператор вывода

friend ostream& operator<<(ostream& out, const Vector& v) {

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

out << v.arr[i] << " ";

}

return out;

}

};

int main() {

Vector v1, v2, v3, v4;

int num;

cout << "Enter the size of the first vector: "; // вводить n+1 чисел. первое число - "размер", остальные - "наполнение"

cin >> v1;

cout << "Enter the size of the second vector: "; // вводить n+1 чисел. первое число - "размер", остальные - "наполнение"

cin >> v2;

cout << "First vector: " << v1 << endl;

cout << "Second vector: " << v2 << endl;

cout << "Enter a number to add to the vectors: ";

cin >> num;

v3 = v1 + num;

cout << "Result of adding " << num << " to the first vector: " << v3 << endl;

v4 = v2 + num;

cout << "Result of adding " << num << " to the second vector: " << v4 << endl;

v3 = v1 + v2;

cout << "Result of adding the first and second vectors: " << v3 << endl;

return 0;

}