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**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: АТД. Контейнеры

Выполнил работу

Студент группы РИС-22-1Б

Лихачев Д.А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Постановка задачи**





**Диаграммы классов**



**Код программы**

**Array.h**

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Iterator

{

friend class Array;

private:

int\* elem;

public:

Iterator();

Iterator(const Iterator& it);

bool operator==(const Iterator& it) const;

bool operator!=(const Iterator& it);

void operator++();

void operator--();

void operator+=(int n);

int& operator \*();

};

class Array

{

private:

int size;

int\* data;

Iterator beg;

Iterator end;

public:

Array();

Array(int size);

Array(const Array& other);

~Array();

int& operator[](int index);

bool operator!=(const Array& a) const;

bool operator<(int) const;

friend ostream& operator<<(ostream& fout, const Array& other);

friend istream& operator>>(istream& fin, Array& other);

Iterator first();

Iterator last();

};

**Array.cpp**

#include <iostream>

#include "Array.h"

using namespace std;

Iterator::Iterator() {

elem = nullptr;

}

Iterator::Iterator(const Iterator& it) {

elem = it.elem;

}

bool Iterator::operator==(const Iterator& it) const {

return elem == it.elem;

}

bool Iterator::operator!=(const Iterator& it) {

return elem != it.elem;

}

void Iterator::operator++() {

++elem;

}

void Iterator::operator--() {

--elem;

}

void Iterator::operator+=(int n) {

/\*while (\*elem != n) {

++elem;

}

\*/

for (int i = 1; i < n; ++i) {

++elem;

}

}

int& Iterator::operator \*() {

return \*elem;

}

Array::Array() {

size = 0;

data[size] = {};

beg.elem = &data[0];

end.elem = &data[size];

}

Array::Array(int size) {

this->size = size;

data = new int[size];

cout << "Enter array" << endl;

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

cout << "Enter " << i + 1 << " element: ";

cin >> data[i];

}

beg.elem = &data[0];

end.elem = &data[size];

}

Array::Array(const Array& other) {

size = other.size;

data = new int[size];

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

data[i] = other.data[i];

}

beg = other.beg;

end = other.end;

}

Array::~Array() {

delete[]data;

data = 0;

}

int& Array::operator[](int index) {

if (index >= 0 && index < size) {

return data[index];

}

cout << "Error out of range";

}

bool Array::operator!=(const Array& a) const {

return (size == a.size) && (data == a.data);

}

bool Array::operator<(int x) const {

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

if (data[i] = x) return 1;

}

return 0;

}

Iterator Array::first() {

return beg;

}

Iterator Array::last() {

return end;

}

ostream& operator<<(ostream& fout, const Array& other) {

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

fout << other.data[i];

}

fout << endl;

return fout;

}

istream& operator>>(istream& fin, Array& other) {

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

fin >> other.data[i];

}

return fin;

}

**Lab6\_main.cpp**

#include <iostream>

#include "Array.h"

using namespace std;

int main()

{

Array arr1(7);

cout << \*(arr1.first()) << endl;

Iterator it = arr1.first();

++it;

++it;

cout << "\*it: " << \*it << endl;

for (it = arr1.first(); it != arr1.last(); ++it) {

cout << \*it << "\t";

}

cout << endl;

cout << arr1[3] << endl;

cout << "Enter index for it: ";

int n;

cin >> n;

it = arr1.first();

it += n;

cout << \*it << endl;

cout << "Enter number for check: ";

cin >> n;

if (arr1 < n) {

cout << "Number in array";

}

else {

cout << "Number out of array";

}

return 0;

}

**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.