Пермский национальный исследовательский политехнический университет

**Лабораторная работа №6**

**Дисциплина: Информатика**

**Вариант № 9**

**"АТД. Контейнеры"**

Выполнил: Зайченко Никита Геннадьевич

Проверила: Доцент кафедры ИТАС Полякова О. А.

Пермь 2022

**Содержание отчета**

1) Постановка задачи (общая и конкретного варианта).

2) Описание класса-контейнера.

3) Определение компонентных функций.

4) Описание класса-итератора и его компонентных функций

5) Функция main().

6) Объяснение результатов работы программы.

7) Ответы на контрольные вопросы.

**Постановка задачи (общая и конкретного варианта)**

1) Создание консольного приложения, состоящего из нескольких файлов в системе

программирования Visual Studio.

2) Реализация класса-контейнера.

3) Класс- контейнер МНОЖЕСТВО с элементами типа int.

Реализовать операции:

[] – доступа по индексу;

!= - проверка на неравенство;

< число – принадлежность числа множеству;

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

**Описание класса-контейнера.**

class Set

{

public:

Set(int s, int k = 0); //конструктор с параметрами: выделяет память под s элементов и заполняет их значением k

Set(const Set& a); //конструктор с параметрами

~Set(); //деструктор

Set& operator=(const Set& a); //оператор присваивания

int& operator[](int index); //операция доступа по индексу

Set operator+(const int k); //операция для добавление константы

int operator()(); //операция, возвращающая длину множества

bool operator<(int value) const; //проверка принадлежности числа множеству

bool operator!=(const Set& ); // проверка на неравенство

friend ostream& operator<<(ostream& out, const Set& a); //перегруженные операции вывода

friend istream& operator>>(istream& in, Set& a); //перегруженные операции ввода

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

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

private:

int size; //размер множества

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

Iterator beg; //указатель на первый элемент множества

Iterator end; //указатель на элемент следующий за последним

};

****

**Определение компонентных функций.**

Set::Set(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

Set::Set(const Set& 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

Set::~Set() { delete[]data; data = 0; }

//операция присваивания

Set& Set::operator=(const Set& 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];

return \*this;

beg = a.beg;

end = a.end;

return \*this;

}

//операция доступа по индексу

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

{

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

else cout << "\nОшибка индекс больше размера множества";

}

//операция для добавления константы

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

{

Set temp(size);

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

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

return temp;

}

//операция для получения длины вектора

int Set::operator ()() { return size; }

//проверка принадлежности числа множеству

bool Set::operator<(int value) const

{

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

if (data[i] >= value) {

return data[i] == value;

}

}

return false;

}

// проверка на неравенство

bool Set::operator!=(const Set& a) { return data != a.data; }

//операции для ввода-выода

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

{

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

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

return out;

}

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

{

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

in >> a.data[i];

return in;

}

**Описание класса-итератора и его компонентных функций**

class Iterator

{

friend class Set;//дружественный класс

public:

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

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

int& operator \*() const { return\*elem; } //перегруженная операция разыменования

Iterator operator+(int n) const { auto copy = \*this; advance(copy.elem, n); return copy; } //переход вправо к элементу с номером n

private:

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

};

**Функция main().**

setlocale(LC\_ALL, "Russian");

int n, k, l, j, s;

Set a(5);

cout << "Множнство a: " << endl;

cout << a << endl;

cin >> a;

cout << "Множнство a: " << endl;

cout << a << endl;

cout << "Введите элемент котрый хотите заменить: " << endl;

cin >> n;

cout << "Введите новое значение: " << endl;

cin >> k;

a[n - 1] = k;

cout << "Множнство a: " << endl;

cout << a << endl;

Set b(10);

cout << "Множество b из 10 элементов: " << endl;

cout << b << endl;

b = a;

cout << "Присвоили множеству b значения вектора a: " << endl;

cout << b << endl;

Set c(10);

c = b + 100;

cout << "Cоздали множество c из 10 элементов и увеличили на 100: " << endl;

cout << c << endl;

cout << endl << "Длина множества a = " << a() << endl;

cout << endl << "Проверяем неравенство первого и второго элемента множества a: ";

cout << endl << a[0] << " != " << a[1] << endl;

if (a[0] != a[1]) { cout << "Элементы не равны" << endl; }

else { cout << "Элементы равны" << endl; }

cout << endl << "Введите элимент для проверки принодлежности множеству: ";

cin >> l;

if (a < l) { cout << "Элементы принадлежит множеству a" << endl; }

else { cout << "Элементы не принадлежит множеству a" << endl; }

cout << "Возвращаем первый элемент множеству a" << endl;

cout << "Первый элемент: " << \*(a.first()) << endl;

Iterator i = a.first();

cout << "Переход вправо к элементу с номером n, введите n: " << endl;

cin >> j;

j = j - 1;

Iterator d;

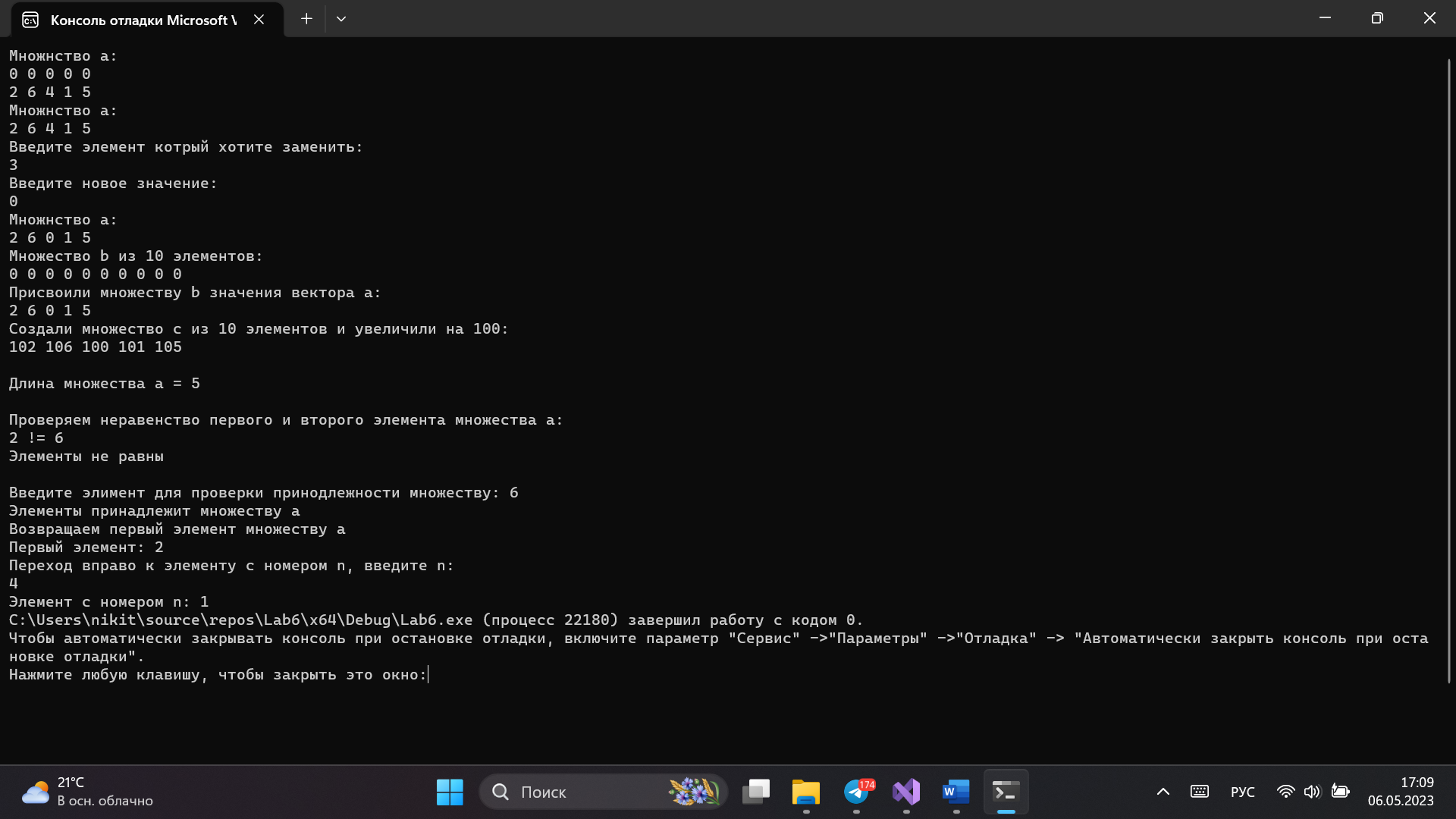
d = i+j;

cout << "Элемент с номером n: " << \*d;

}

**Объяснение результатов работы программы.**

Программа создает и оперирует множествами (Set) - пользовательским классом, реализующим работу с динамическим массивом. Все операции выполняются с помощью перегруженных операторов класса Set, что упрощает использование и снижает количество кода при работе с множествами и элементами.



**Ответы на контрольные вопросы**

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

Абстрактный тип данных (АТД) - это математическая модель, описывающая структуру данных и операции над ней, независимо от конкретной реализации. Примеры АТД: стек, очередь, список, дерево, граф.

**2. Привести примеры абстракции через параметризацию.**

template<typename T>

class Vector {

private:

T\* arr;

int size;

public:

Vector() {

arr = new T[100];

size = 100;

}

**3. Привести примеры абстракции через спецификацию.**

class Animal {

public:

virtual void make\_sound() = 0; // чисто виртуальная функция

};

**4. Что такое контейнер? Привести примеры.**

Контейнер - это объект, содержащий данные в определенном формате и предоставляющий операции для их хранения и управления. Примеры контейнеров: массив, списки, стек, очередь, ассоциативный массив и др.

**5. Какие группы операций выделяют в контейнерах?**

Группы операций в контейнерах: добавление и удаление элементов, доступ к элементам, перебор элементов, сортировка и т.д.

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

Виды доступа к элементам контейнера: доступ по индексу (для массивов, векторов), итераторы (способ обхода элементов контейнера), доступ по ключу (ассоциативные массивы).

**7. Что такое итератор?**

Итератор - это объект, предоставляющий способ обхода элементов контейнера.

**8. Каким образом может быть реализован итератор?**

Реализация итератора может быть осуществлена через указатель на текущий элемент или через класс-обертку, инкапсулирующий текущий элемент и способ его получения/изменения.

**9. Каким образом можно организовать объединение контейнеров?**

Объединение контейнеров можно организовать через создание нового контейнера и копирование элементов из двух исходных контейнеров в него.

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

Контейнер, состоящий из элементов «ключ-значение», предоставляет доступ к элементам по ключу, который является уникальным идентификатором элемента.

**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];**

Контейнером является объект d - массив.

**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;**

Объект d не является контейнером, так как это обычный статический массив.

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

Доступ к элементам контейнера, реализованного как динамический массив с операцией доступа по индексу, будет осуществляться через индексацию массива.

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

Доступ к элементам контейнера, реализованного как линейный список, будет осуществляться через обход списка по итераторам.