МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Факультет: Электротехнический

Кафедра: «Информационные технологии и автоматизированные системы» (ИТАС)

Направление: Разработка информационных систем (РИС)

**О Т Ч Ё Т**   
**о лабораторной работе №6**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «АТД. Контейнеры»

Выполнил:

Студент группы РИС-24-3б

Суяргулов А.О.

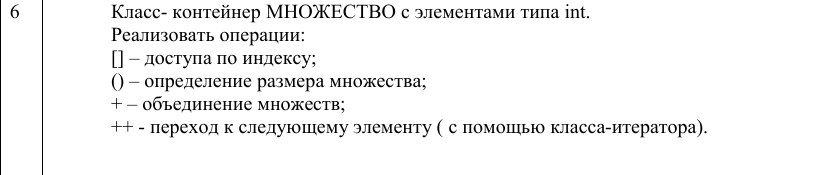
Проверил:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2025

ВВЕДЕНИЕ



ОСНОВНАЯ ЧАСТЬ

**Диаграмма классов:**

**Код:**

**Main:**

#include <iostream>

#include <vector>

// Класс для представления множества целых чисел

class Mnojestvo {

private:

// Вектор для хранения элементов множества

std::vector<int> элементы;

// Метод для проверки, содержится ли элемент в множестве

bool содержит(int элемент) const {

for (size\_t i = 0; i < элементы.size(); ++i) {

if (элементы[i] == элемент) {

return true;

}

}

return false;

}

public:

// Конструктор по умолчанию

Mnojestvo() {}

// Конструктор с начальными элементами

Mnojestvo(const std::vector<int>& начальныеЭлементы) {

for (int элемент : начальныеЭлементы) {

добавить(элемент);

}

}

// Метод для добавления элемента в множество

void добавить(int элемент) {

if (!содержит(элемент)) {

элементы.push\_back(элемент);

}

}

// Оператор доступа по индексу

int& operator[](size\_t индекс) {

if (индекс >= элементы.size()) {

throw std::out\_of\_range("Индекс за пределами диапазона");

}

return элементы[индекс];

}

// Оператор для получения размера множества

size\_t operator()() const {

return элементы.size();

}

// Оператор для объединения множеств

Mnojestvo operator+(const Mnojestvo& другоеМножество) const {

Mnojestvo результат = \*this;

for (size\_t i = 0; i < другоеМножество.элементы.size(); ++i) {

результат.добавить(другоеМножество.элементы[i]);

}

return результат;

}

// Класс итератора для перебора элементов множества

class Итератор {

private:

Mnojestvo\* множество;

size\_t индекс;

public:

// Конструктор итератора

Итератор(Mnojestvo\* множество, size\_t начальныйИндекс) : множество(множество), индекс(начальныйИндекс) {}

// Оператор для получения текущего элемента

int operator\*() const {

return множество->элементы[индекс];

}

// Оператор для перехода к следующему элементу

Итератор& operator++() {

++индекс;

return \*this;

}

// Оператор для сравнения итераторов на неравенство

bool operator!=(const Итератор& другойИтератор) const {

return (множество != другойИтератор.множество) || (индекс != другойИтератор.индекс);

}

// Оператор для сравнения итераторов на равенство

bool operator==(const Итератор& другойИтератор) const {

return (множество == другойИтератор.множество) && (индекс == другойИтератор.индекс);

}

};

// Метод для получения итератора начала множества

Итератор начало() {

return Итератор(this, 0);

}

// Метод для получения итератора конца множества

Итератор конец() {

return Итератор(this, элементы.size());

}

// Метод для вывода элементов множества

void вывести() const {

std::cout << "{ ";

for (size\_t i = 0; i < элементы.size(); ++i) {

std::cout << элементы[i];

if (i < элементы.size() - 1) {

std::cout << ", ";

}

}

std::cout << " }" << std::endl;

}

// Метод для удаления элемента из множества

void удалить(int элемент) {

for (size\_t i = 0; i < элементы.size(); ++i) {

if (элементы[i] == элемент) {

элементы.erase(элементы.begin() + i);

return; // Удаляем только первый найденный элемент

}

}

}

};

int main() {

setlocale(LC\_ALL, "RU");

// Создаем два множества

Mnojestvo множество1({ 1, 2, 3, 4, 5 });

Mnojestvo множество2({ 3, 5, 6, 7, 8 });

// Выводим множества

std::cout << "Множество 1: ";

множество1.вывести();

std::cout << "Множество 2: ";

множество2.вывести();

// Получаем элемент по индексу

std::cout << "Элемент с индексом 2 в множестве 1: " << множество1[2] << std::endl;

// Получаем размер множества

std::cout << "Размер множества 1: " << множество1() << std::endl;

// Объединяем множества

Mnojestvo множество3 = множество1 + множество2;

std::cout << "Множество 1 + Множество 2: ";

множество3.вывести();

// Итерируемся по объединенному множеству

std::cout << "Итерируемся по множеству 3: ";

for (Mnojestvo::Итератор it = множество3.начало(); it != множество3.конец(); ++it) {

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

}

std::cout << std::endl;

// Удаляем элемент из множества

множество3.удалить(3);

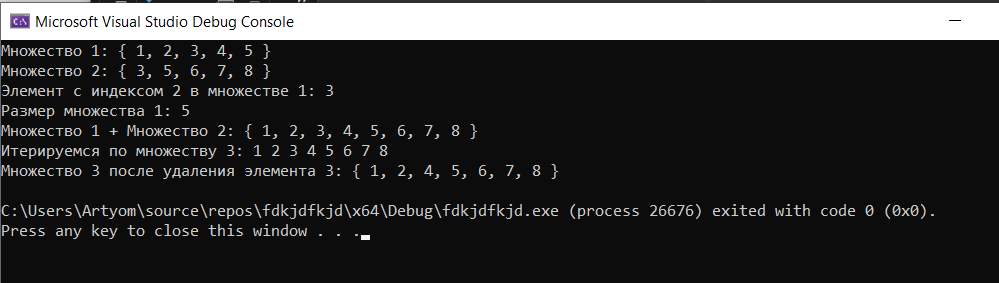
std::cout << "Множество 3 после удаления элемента 3: ";

множество3.вывести();

return 0;

}

**Результат работы кода:**

****

**Контрольные вопросы:**

**Вот ответы на вопросы из изображения:**

1. Что такое абстрактный тип данных? Привести примеры АТД. Абстрактный тип данных (АТД) - это тип данных, который определяется через операции, которые можно над ним выполнять, без указания конкретной реализации. Примеры: Список, Стек, Очередь, Дерево.

2. Привести примеры абстракции через параметризацию. Параметризация - это когда тип данных определяется с использованием другого типа данных в качестве параметра.

3. Привести примеры абстракции через спецификацию. Спецификация определяет поведение типа данных без указания конкретной реализации.

4. Что такое контейнер? Привести примеры. Контейнер - это структура данных, которая хранит коллекцию объектов одного или разных типов. Примеры: массив, список, стек, очередь, дерево, хеш-таблица.

5. Какие группы операций выделяют в контейнерах? Основные группы операций: добавление элемента, удаление элемента, поиск элемента, доступ к элементу, проверка на пустоту, получение размера контейнера, обход (итерация) по элементам.

6. Какие виды доступа к элементам контейнера существуют? Привести примеры. Доступ может быть:

\* По индексу (массив).

\* По ключу (словарь, хеш-таблица).

\* Последовательный (список, итератор).

7. Что такое итератор? Итератор - это объект, который позволяет последовательно перебирать элементы контейнера без раскрытия его внутренней структуры.

8. Каким образом может быть реализован итератор? Итератор обычно реализуется как отдельный класс, который хранит указатель на текущий элемент контейнера и предоставляет методы для перемещения к следующему элементу, проверки, есть ли следующий элемент, и получения значения текущего элемента.

9. Каким образом можно организовать объединение контейнеров? Можно использовать:

\* Вложение одного контейнера в другой (например, список списков).

\* Контейнер, хранящий указатели на другие контейнеры.

\* Создание нового контейнера, который содержит копии элементов из других контейнеров (операция объединения/конкатенации).

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

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

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

\* d. int mas[100];

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

\* d. int mas;

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

15. Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера? Последовательный доступ, требуется перебор элементов от начала списка (в худшем случае O(n)).

**Ссылка на GitHub:**

<https://github.com/Drazda1488/oop_6lab>