Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

# Лабораторная работа "АТД.Контейнеры."

Выполнил студент гр. РИС-24-3б

Жиряков Леонид Антонович

Проверил:

Доцент кафедры ИТАС   
Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь, 2024

Вариант 8.

Постановка задачи:

Общая:

1. Определить класс-контейнер.

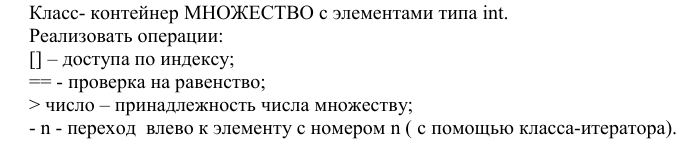
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.

3. Перегрузить операции, указанные в варианте.

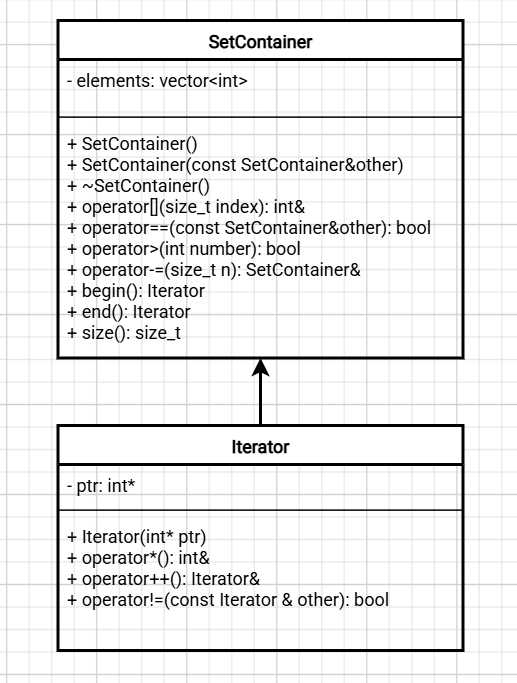
4. Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.

5. Написать тестирующую программу, иллюстрирующую выполнение операций.

Персональная:



UML – диаграмма:



Программная реализация

**Заголовочный файл**

#pragma once

#include <vector>

#include <iostream>

#include "Iterator.h" // Подключаем заголовок итератора

class SetContainer

{

public:

// Конструкторы и деструктор

SetContainer();

SetContainer(const SetContainer& other);

~SetContainer();

// Операторы

int& operator[](size\_t index);

bool operator==(const SetContainer& other) const;

bool operator>(int number) const;

SetContainer& operator-=(size\_t n);

friend std::ostream& operator<<(std::ostream& os, const SetContainer& set);

friend std::istream& operator>>(std::istream& is, SetContainer& set);

Iterator begin() { return Iterator(elements.data()); }

Iterator end() { return Iterator(elements.data() + elements.size()); }

size\_t size() const { return elements.size(); }

private:

std::vector<int> elements;

};

**Реализация методов**

#include "SetContainer.h"

#include <algorithm>

// Конструкторы

SetContainer::SetContainer() {}

SetContainer::SetContainer(const SetContainer& other) : elements(other.elements) {}

// Деструктор

SetContainer::~SetContainer() {}

// Оператор []

int& SetContainer::operator[](size\_t index)

{

return elements[index];

}

// Оператор равенства

bool SetContainer::operator==(const SetContainer& other) const

{

return elements == other.elements;

}

// Оператор большего

bool SetContainer::operator>(int number) const

{

return std::find(elements.begin(), elements.end(), number) != elements.end();

}

// Оператор -=

SetContainer& SetContainer::operator-=(size\_t n)

{

if (n < elements.size())

{

std::rotate(elements.begin(), elements.begin() + n, elements.end());

}

return \*this;

}

// Оператор вывода

std::ostream& operator<<(std::ostream& os, const SetContainer& set)

{

os << "{ ";

for (const auto& el : set.elements)

{

os << el << " ";

}

os << "}";

return os;

}

// Оператор ввода

std::istream& operator>>(std::istream& is, SetContainer& set)

{

int value;

while (is >> value)

{

set.elements.push\_back(value);

}

return is;

}

**Итератор**

#pragma once

class Iterator

{

public:

Iterator(int\* ptr) : ptr(ptr) {}

int& operator\*() { return \*ptr; }

Iterator& operator++() { ++ptr; return \*this; }

bool operator!=(const Iterator& other) const { return ptr != other.ptr; }

private:

int\* ptr;

};

**Функция main**

#include <iostream>

#include "SetContainer.h"

int main()

{

setlocale(LC\_ALL, "RU");

SetContainer set1, set2;

// Ввод элементов в первое множество

std::cout << "Введите целые числа для первого набора (заканчивайте нецелым числом): ";

std::cin >> set1;

set2 = set1;

// Вывод множеств

std::cout << "Первый набор: " << set1 << std::endl;

std::cout << "Второй набор: " << set2 << std::endl;

// Проверка на равенство множеств

if (set1 == set2)

{

std::cout << "Множества равны." << std::endl;

}

else

{

std::cout << "Множества не равны." << std::endl;

}

// Проверка на принадлежность элемента

int checkValue;

checkValue = 15;

if (set1 > checkValue)

{

std::cout << checkValue << " находится в первом сете." << std::endl;

}

else

{

std::cout << checkValue << " его нет в первом сете." << std::endl;

}

// Переход к элементу с индексом n

size\_t n;

n = 3;

if (n < set1.size())

{

std::cout << "Элемент в индексе " << n << ": " << set1[n] << std::endl; // Используем оператор доступа к индексу

}

else

{

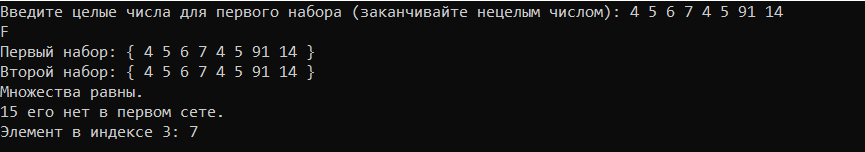
std::cout << "Индекс выходит за рамки допустимого." << std::endl;

}

return 0;

}

Результат работы:



Контрольные вопросы:

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

2. Привести примеры абстракции через параметризацию.

3. Привести примеры абстракции через спецификацию.

4. Что такое контейнер? Привести примеры.

5. Какие группы операций выделяют в контейнерах?

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

7. Что такое итератор?

8. Каким образом может быть реализован итератор?

9. Каким образом можно организовать объединение контейнеров?

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

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

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?

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

Ответы:

1. **Что такое абстрактный тип данных? Привести примеры АТД.**  
   Абстрактный тип данных (АТД) — это тип данных, определяемый через операции, которые могут выполняться над его объектами, без учета способа их представления. Примеры АТД: класс в языке C++, стек, очередь.
2. **Привести примеры абстракции через параметризацию.**  
   Абстракция через параметризацию может быть осуществлена с использованием параметров в функциях. Например, функция, которая принимает массив и его размер, или обобщенная функция, которая работает с различными типами данных.
3. **Привести примеры абстракции через спецификацию.**  
   Абстракция через спецификацию достигается, когда операции представляются как часть типа. Например, интерфейс для работы с контейнерами, который определяет операции добавления, удаления и доступа к элементам, но не указывает, как эти операции реализованы.
4. **Что такое контейнер? Привести примеры.**  
   Контейнер — это объект, который объединяет однотипные элементы. Примеры контейнеров: массивы, списки, стеки, очереди.
5. **Какие группы операций выделяют в контейнерах?**  
   В контейнерах выделяют следующие группы операций:
   * Операции доступа к элементам.
   * Операции добавления и удаления элементов.
   * Операции поиска элементов.
   * Операции объединения контейнеров.
   * Специальные операции, зависящие от вида контейнера.
6. **Какие виды доступа к элементам контейнера существуют? Привести примеры.**  
   Виды доступа к элементам контейнера:
   * Прямой доступ (например, a[10]).
   * Ассоциативный доступ (например, a["word"] в словаре).
   * Последовательный доступ (перемещение от элемента к элементу с помощью итератора).
7. **Что такое итератор?**  
   Итератор — это объект, который обеспечивает последовательный доступ к элементам контейнера и может реализовывать операции перемещения по элементам.
8. **Каким образом может быть реализован итератор?**  
   Итератор может быть реализован как часть класса-контейнера с набором методов для перемещения по элементам, или как отдельный класс с аналогичным интерфейсом, совместимым с массивами.
9. **Каким образом можно организовать объединение контейнеров?**  
   Объединение контейнеров может быть организовано различными способами:
   * Простое сцепление двух контейнеров.
   * Объединение упорядоченных контейнеров.
   * Объединение как объединение множеств или пересечение множеств.
   * Извлечение части элементов с помощью итераторов.
10. **Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?**  
    Контейнер, состоящий из элементов «ключ-значение», предоставляет ассоциативный доступ, где ключом является слово, а значением — его перевод.
11. **Как называется контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера?**  
    Такой контейнер называется стеком (Last In First Out - LIFO).
12. **Какой из объектов (a,b,c,d) является контейнером?**  
    d. int mas[100]; — это контейнер, так как он представляет собой массив фиксированного размера.
13. **Какой из объектов (a,b,c,d) не является контейнером?**  
    d. int mas; — это не контейнер, так как он представляет собой одну переменную, а не группу данных.
14. **Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?**  
    Доступ к элементам будет прямым, так как он осуществляется по индексу.
15. **Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера?**  
    Доступ к элементам будет последовательным, так как для перебора элементов потребуется перемещение от одного элемента к другому.

Ссылка:

**https://github.com/LeonidZhir/-**