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 №6

По теме **«АТД. Контейнеры»**

Вариант №10

Выполнял:

студент группы РИС-24-1б

Морозова Н.С.

Проверял:

Доцент кафедры ИТАС

Полякова О.А.

Постановка задачи:

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.

Задача:

Класс-контейнер МНОЖЕСТВО с элементами типа int.

Реализовать операции:

[] доступа по индексу;

() определение размера множества;

- разность множеств;

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

Анализ задачи:

1. Создать в файле set\_value.h класс SetValue и описать его методы в файле set\_value.cpp.
2. Класс Iterator описан в файле set\_value.h перед классом SetValue и добавлены методы для работы с множествами через указатели.
3. Перегружены операции: [] доступа по индексу; () определение размера множества; - разность множеств; -- переход к предыдущему элементу.

Решение

Код

Файл Lab\_6.cpp

#include <iostream>

#include <Windows.h>

#include "set\_value.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

SetValue a(5); //создали множество из 5 элементов, заполненный нулями

cout << "Множество а: " << a << endl;

cout << "Заполните множество a" << endl;

cin >> a; //ввели с клавиатуры значения элементов множества

cout << a << endl; //вывели значения элементов

a[2] = 100; //присвоили новое значение элементу

cout << "Новое множество а: " << a << endl;

SetValue b(5);

cout << "Множество b: " << b << endl;

b = a; //присвоили множеству b множество a

cout << "Новое множество b: ";

cout << b << endl << endl;;

SetValue c(5);

cout << "Заполните множество c" << endl;

cin >> c;

c = c - a; //вычли из множества c множество a

cout << "Разность множеств c - a: " << c << endl;

cout << "Длина множества с = " << c() << endl << endl;

cout << "Разыменовываем последнее значение множества а: ";

cout << \*(a.last()) << endl;

//переменную типа Iterator устанавливаем на последний элемент множества а

Iterator i = a.last();

//операция декремент

--i;

cout << "Предыдущее значение, разыменовываем итератор: ";

cout << \*i << endl;

cout << "Выводим значения элеменов множества с помощью итератора: " << endl;

for (i = a.last(); i != a.first(); --i)

cout << \*i << endl;

cout << \*i;

return 0;

}

Файл set\_value.h

#pragma once

#include <iostream>

using namespace std;

class Iterator

{

friend class SetValue;//дружественный класс

public:

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

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

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

//перегруженные операции сравнения

bool operator==(const Iterator& it) { return elem == it.elem; }

bool operator!=(const Iterator& it) { return elem != it.elem; }

//перегруженная операция инкремент

void operator++() { ++elem; }

//перегруженная операция декремент

void operator--() { --elem; }

//перегруженная операция разыменования

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

};

class SetValue

{

int size; //размер множества

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

Iterator beg;//указатель на первый элемент вектора

Iterator end;//указатель на элемент следующий за последним

public:

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

//конструктор копирования

SetValue(const SetValue& a);

//деструктор

~SetValue();

//оператор присваивания

SetValue& operator=(const SetValue& a);

//операция доступа по индексу

int& operator[](int index);

//операция, возвращающая длину вектора

int operator()();

//операция разности множеств

friend SetValue operator-(const SetValue& a, const SetValue& b);

//перегруженные операции ввода-вывода

friend ostream& operator<<(ostream& out, const SetValue& a);

friend istream& operator>>(istream& in, SetValue& a);

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

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

};

Файл set\_value.cpp

#include "set\_value.h"

#include <iostream>

using namespace std;

//конструктор с параметрами

SetValue::SetValue(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 - 1];

}

//конструктор копирования

SetValue::SetValue(const SetValue& a)

{

size = a.size;

data = new int[size];

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

data[i] = a.data[i];

beg.elem = &data[0]; // Указываем на новые данные

end.elem = &data[size - 1]; // Указываем на новый конец

}

//деструктор

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

//операция присваивания

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

beg.elem = &data[0];

end.elem = &data[size];

return \*this;

}

//операция доступа по индексу

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

{

if (index < size)

return data[index];

else cout << "\nError! Index>size";

}

//операция для получения длины вектора

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

//операция для разности множеств

SetValue operator-(const SetValue& a, const SetValue& b)

{

if (a.size != b.size)

cout << "a.size != b.size" << endl;

SetValue temp(a.size);

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

{

bool flag = true;

for (int j = 0; j < b.size; ++j)

if (a.data[i] == b.data[j]) {

flag = false;

break;

}

if (flag)

temp.data[i] = a.data[i];

}

return temp;

}

//операции для ввода-выода

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

{

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

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

return out;

}

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

{

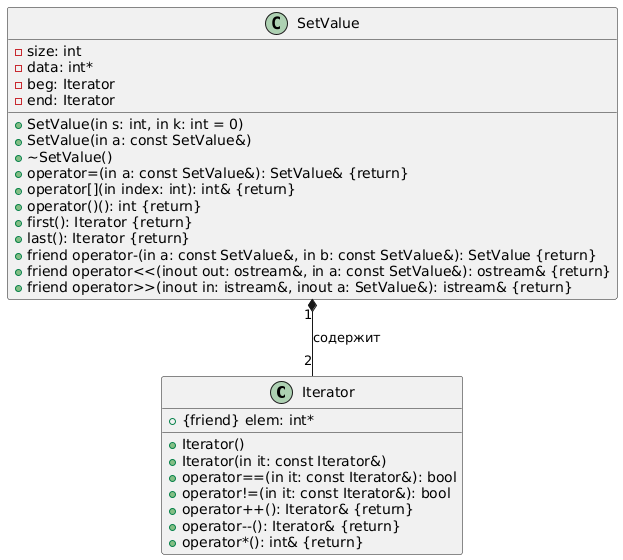
for (int i = 0; i < a.size; ++i)

in >> a.data[i];

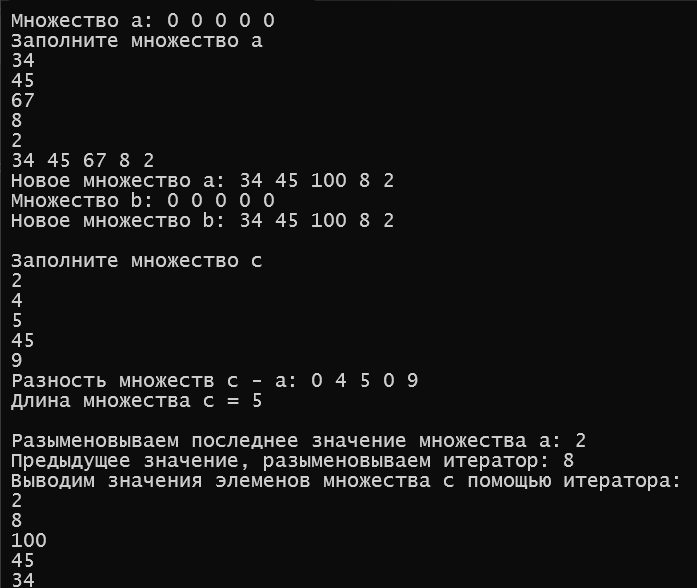
return in;

}

UML-диаграмма



Результат работы программы:



Контрольные вопросы:

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

АТД — это математическая модель для работы с данными, которая определяет:

1. Операции, которые можно выполнять с данными.
2. Поведение этих операций (без указания реализации).

Примеры АТД:

Стек: push, pop, peek.

Очередь: enqueue, dequeue.

Список: insert, remove, get.

Ассоциативный массив (словарь): put, get, delete.

1. Привести примеры абстракции через параметризацию.

Параметризация позволяет создавать обобщённые структуры данных. Например, шаблоны:

template <typename T>

class Stack { ... }; // Работает для любого типа T

1. Привести примеры абстракции через спецификацию.

Спецификация описывает что делает операция, но не как. Примеры:

1. АТД "Стек":

push(x): Добавляет x на вершину стека.

pop(): Удаляет и возвращает верхний элемент.

1. Функция sort(): результат - упорядоченный массив, способ сортировки не указан (может быть QuickSort, MergeSort и т.д.).
2. Что такое контейнер? Привести примеры.

Контейнер — объект, хранящий другие объекты (элементы). Примеры:

массив (int arr[10]), список (std::list), стек (std::stack), словарь (std::map).

1. Какие группы операций выделяют в контейнерах?
2. Инициализация: Создание контейнера.
3. Доступ: Чтение/изменение элементов (по индексу, ключу).
4. Вставка/удаление: insert, erase, push\_back.
5. Запросы: size, empty, find.
6. Обход: Итерация по элементам.
7. Какие виды доступа к элементам контейнера существуют? Привести примеры.
8. Прямой доступ (по индексу): array[5].
9. Последовательный доступ (линейный): linked list (переход к следующему элементу).
10. Доступ по ключу: map["key"].
11. Что такое итератор?

Итератор — объект для обхода элементов контейнера, абстрагирующий доступ к ним. Аналогия: указатель, но для любых структур данных.

1. Каким образом может быть реализован итератор?

class Iterator {

int\* ptr;

public:

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

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

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

};

1. Каким образом можно организовать объединение контейнеров?
   * + - 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.
       - Как объединение множеств, в новый контейнер попадают только те элементы, которые есть хотя бы в одном контейнере, операция коммутативна.
       - Как пересечение множеств, в новый контейнер попадают только те элементы, которые есть в обоих контейнерах, операция коммутативна.
       - 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.
2. Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?

По ключу: map["key"] = value; (пример: std::map в C++)

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

Это стек (std::stack).

1. Какой из объектов (a,b,c,d) является контейнером?
   1. int mas=10; переменная не хранит элементы
   2. int mas; переменная не хранит элементы
   3. struct {char name[30]; int age;} mas; структура
   4. **int mas[100]; массив - контейнер.**
2. Какой из объектов (a,b,c,d) не является контейнером?
   1. int a[]={1,2,3,4,5}; массив
   2. int mas[30]; массив
   3. struct {char name[30]; int age;} mas[30]; массив структур
   4. **int mas; обычная переменная**
3. Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?

Прямой доступ (по индексу за O(1)): arr[5].

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

Последовательный доступ: нужно итеративно переходить от узла к узлу.