Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Тема: «Лабораторная работа №6»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Полякова О. А.

Пермь 2023

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.

**Вариант 8:**

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 класс-итератора).

**Исходный код программы:**

**Файл “Set.h”:**

#include <iostream>

using namespace std;

class Iterator {

friend class Set;

public:

Iterator() {

elem = 0;

}

Iterator(const Iterator& it) {

elem = it.elem;

}

bool operator ==(const Iterator &it) {

return elem == it.elem;

}

bool operator !=(const Iterator &it) {

return elem != it.elem;

}

void operator++() {

++elem;

}

void operator--() {

--elem;

}

int& operator \*() const {

return \*elem;

}

void operator +(int n) {

elem += n;

}

private:

int\* elem;

};

class Set {

public:

Set(int s);

Set();

~Set();

Set& operator |=(int n);

Set& operator =(const Set& other);

bool operator !=(const Set& other);

bool operator <(int key) const;

int& operator [](int index);

friend ostream& operator <<(ostream& out, const Set& other);

Iterator first() {

return begin;

}

Iterator last() {

return end;

}

private:

int\* elements;

int size;

Iterator begin;

Iterator end;

};

**Файл “Set.cpp”:**

#include "Set.h"

Set::Set(int s) {

size = s;

elements = new int[size];

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

elements[i] = 0;

begin.elem = &elements[0];

end.elem = &elements[size];

}

Set::Set() {

size = 0;

elements = new int[size];

begin.elem = &elements[0];

end.elem = &elements[size];

}

Set::~Set() {

delete[] elements;

size = 0;

}

Set& Set::operator =(const Set& other) {

if (this == &other)

return \*this;

size = other.size;

if (elements != 0)

delete[] elements;

elements = new int[size];

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

elements[i] = other.elements[i];

begin = other.begin;

end = other.end;

return \*this;

}

bool Set::operator !=(const Set& other) {

bool flag;

if (size != other.size)

return false;

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

flag = (elements[i] == other.elements[i]);

return flag;

}

bool Set::operator <(int key) const {

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

if (key == elements[i])

return true;

return false;

}

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

if (index < size)

return elements[index];

else

throw out\_of\_range("Index doesn't exist");

}

Set& Set::operator |=(int key) {

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

if (key == elements[i])

return \*this;

int i, k;

for (k = 0; k < size && elements[k] <= key; k++);

if (k == size) {

elements[size] = key;

}

else {

for (i = size; i > k; i--)

elements[i] = elements[i - 1];

elements[k] = key;

}

end.elem = &elements[size];

size++;

return \*this;

}

ostream& operator <<(ostream& out, const Set& set) {

out << "[ ";

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

out << set.elements[i] << " ";

out << "]\n";

return out;

}

**Файл “Lab\_6.cpp”:**

#include "Set.cpp"

using namespace std;

int main() {

Set s1, s2;

(((s1 |= 2) |= 3) |= 1) |= 1;

(((s2 |= 2) |= 1) |= 3);

cout << "s1: " << s1;

cout << "s2: " << s2;

if (s1<3)

cout << "s1 < 3\n";

else

cout << "s1 != 3\n";

if (s1<4)

cout << "s2 < 4\n";

else

cout << "s2 != 4\n";

if (s1 != s2)

cout << "s1 != s2\n";

else

cout << "s1 == s2\n";

cout << "s1[2]: " << s1[2] << "\n";

Set s;

((s |= 2) |= 5) |= 10;

cout << "s: " << s;

cout << "s.first: " << \*(s.first()) << "\n";

Iterator i = s.first();

++i;

cout << \*i << "\n";

cout << "s.last: " << \*(s.last()) << "\n";

i = s1.first();

cout << "s1.first: " << \*i << "\n";

i+2;

cout << "i: " << \*i << "\n";

cout << "s: ";

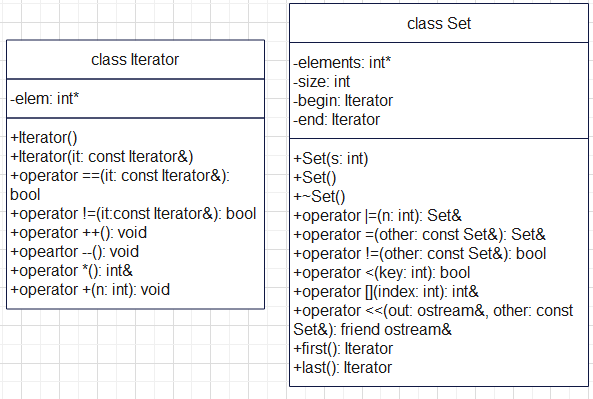
for (i = s.first(); i != s.last(); ++i)

cout << \*i << " ";

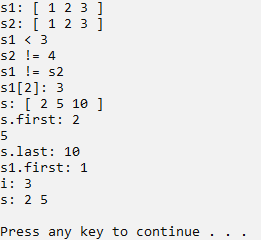
return 0;

}

**UML диаграмма:**



**Скриншот результата выполнения программы**



**Ответы на вопросы**

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

**Ответ:** Абстрактный тип данных - тип данных, определяемы только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов.

1. Привести примеры абстракции через параметризацию.

**Ответ:**

int Array[100];

int max\_value = GetMaxValue(a);

int max\_index = GetMaxIndex(a);

1. Привести примеры абстракции через спецификацию.

**Ответ:**

1. Что такое контейнер? Привести примеры.

**Ответ: Контейнер** - набор однотипных элементов. Пример: встроенные массивы в C++.

Контейнер - это объект. Имя контейнера - это имя переменной.

1. Какие группы операций выделяют в контейнерах?

**Ответ:**

1. Операции доступа к элементам, которые обеспечивают и операцию замены значений элементов;
2. Операции добавления и удаления элементов или групп элементов;
3. Операции поиска элементов и групп элементов;
4. Операции объединения контейнеров;
5. Специальные операции, которые зависят от вида контейнера.
6. Какие виды доступа к элементам контейнера существуют? Привести примеры.

**Ответ:** Доступ к элементам контейнера бывает: последовательный, прямой и ассоциативный.

1. Что такое итератор?

**Ответ:** Итератор - это объект, который обеспечивает последовательный доступ к элементам контейнера.

1. Каким образом может быть реализован итератор?

**Ответ:** Итератора может быть реализован как часть класса-контейнера в виде набора методов:

v.first() перейти к первому элементу

v.last() перейти к последнему элементу

v.next() перейти к следующему элементу

v.prev() перейти к предыдущему элементу

1. Каким образом можно организовать объединение контейнеров?

**Ответ:** Объединение контейнеров может быть реализовано в разных вариантах:

* 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 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 задаётся двумя итераторами.

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

**Ответ:** Доступ к элементам осуществляется по ключу.

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

**Ответ:** Стек.

1. Какой из объектов (a, b, c, d) является контейнером?
2. int mas = 10;
3. 2. int mas;
4. 3. struct {char name[30]; int age;} mas;
5. 4. int mas[100];

**Ответ:** Объект по буквой d, так как массив является контейнером.

1. Какой из объектов (a, b, c, d) не является контейнером?
2. int a[] = {1, 2, 3, 4, 5};
3. 2. int mas[30];
4. 3. struct {char name[30]; int age;} mas[30];
5. 4. int mas;

**Ответ:** Объект под буквой d, так как это целочисленная переменная.

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

**Ответ:** Доступ к элементам будет прямой, через указатель.

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

**Ответ:** Доступ к элементам будет последовательный, так как чтобы достичь заданное значение, нужно будет пройтись по предыдущим.