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;

}

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.fist: " << \*(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";

return 0;

}

**UML диаграмма:**

**Скриншот результата выполнения программы**

**Ответы на 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?
13. int mas = 10;
14. 2. int mas;
15. 3. struct {char name[30]; int age;} mas;
16. 4. int mas[100];
17. Какой из объектов (a, b, c, d) не является контейнером?
18. int a[] = {1, 2, 3, 4, 5};
19. 2. int mas[30];
20. 3. struct {char name[30]; int age;} mas[30];
21. 4. int mas;
22. Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?
23. Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера?