Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «АТД. Контейнеры»

Выполнил работу

Студент группы РИС-23-3Б

Шуракова А.А.

Проверил

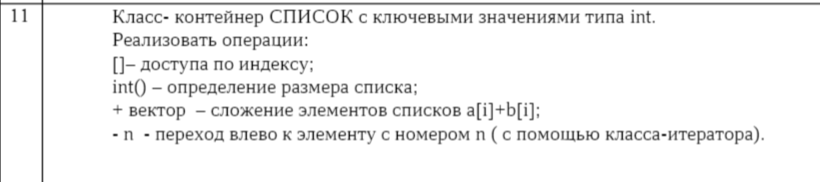
Доцент кафедры ИТАС

Полякова О.А.

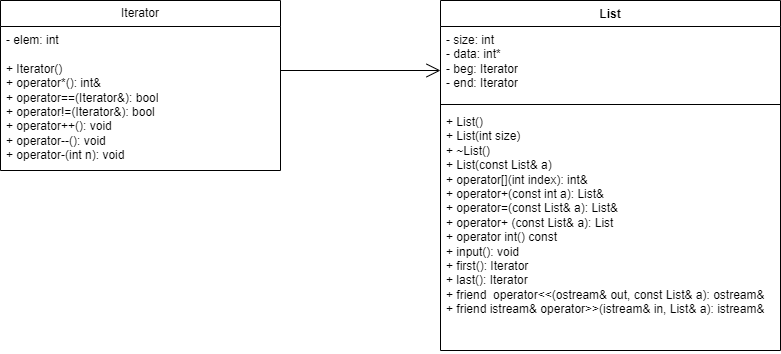
Г. Пермь-2024

**Постановка задачи:**

* Определить класс-контейнер.
* Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
* Перегрузить операции, указанные в варианте.
* Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.
* Написать тестирующую программу, иллюстрирующую выполнение операций.



**Uml диаграмма:**

****

**Код программы:**

**list.h**

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Iterator

{

friend class List;//дружественный класс

private:

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

public:

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

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

//перегруженная операция разыменования

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

//перегруженные операции сравнения

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

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

//перегруженная операция инкремент(переход к следующему элементу)

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

//перегруженная операция декремент(переход к предыдущему элементу)

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

//сдвиг итератора влево на n элементов

void operator-(int n);

};

class List

{

private:

int size;//размер вектора

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

Iterator beg, end;

public:

List();

List(int size);

//конструктор с параметрами

List(const List& a);

//деструктор

~List();

//оператор присваивания

List& operator=(const List& a);

//операция доступа по индексу

int& operator[](int index);

//

List& operator+(const int a) {};

//операция сложения списков

List operator+ (const List& a);

//операция приведения типа к int(возвращает размер списка)

operator int() const { return size; }

//Функция ввода элементво списка

void input();

//перегруженные операции ввода-вывода

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

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

Iterator first() { return beg; }

Iterator last() { return end; }

};

**list.cpp**

#include <iostream>

#include "list.h"

using namespace std;

bool Iterator::operator==(const Iterator& it) const {

return elem == it.elem;

}

bool Iterator::operator!=(const Iterator& it) {

return elem != it.elem;

}

List ::List():data(nullptr), size(0){}

List::List(int size) {

this->size = size;

data = new int[size];

beg.elem = &data[0];

end.elem = &data[size-1];

}

//конструктор копирования

List::List(const List& a)

{

size = a.size;

data = new int[size];

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

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

}

beg = a.beg;

end = a.end;

}

//деструктор

List::~List()

{

delete[]data;

data = 0;

}

//операция присваивания

List& List::operator=(const List& 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 = a.beg;

end = a.end;

return \*this;

}

//операция доступа по индексу

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

if (index >= 0 && index < size) {

return data[index];

}

cout << "Ошибка!";

return data[index];

}

//операция сложения списков

List List::operator+(const List& a) {

List result(size + a.size);

int i = 0;

for (; i < size; i++) {

result[i] = data[i];

}

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

result[i] = data[i] + a.data[j];

}

return result;

}

//оператор сдвига итераторов влево на n элементов

void Iterator::operator-(int n) {

for (int i = n; i>0; i--) {

--elem;

}

}

//функция ввода элементов списка

void List::input() {

cout << "Введите размер списка: ";

cin >> size;

data = new int[size];

cout << "Введите элементы списка: ";

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

cin >> data[i];

}

}

//операции для ввода-выода

ostream& operator<<(ostream& fout, const List& other) {

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

fout << other.data[i];

}

fout << endl;

return fout;

}

istream& operator>>(istream& fin, List& other) {

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

fin >> other.data[i];

}

return fin;

}

**lab6\_main.cpp**

#include <iostream>

#include "list.h"

using namespace std;

int main()

{

system("chcp 1251");

List list1, list2;

list1.input();

list2.input();

cout << "List1: " << list1 << endl;

cout << "List2: " << list2 << endl;

List list3 = list1 + list2;

cout << "list1 + list2 : " << list3 << endl;

cout << "Вывод размера list3: " << static\_cast<int>(list3) << endl;

cout << "Вывод элемента list3 с индексом 2 : " << list3[2] << endl;

cout << \*(list3.first()) << endl;

Iterator it = list3.first();

++it;

++it;

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

for (it = list3.first(); it != list3.last(); ++it) {

cout << \*it << "\t";

}

cout << endl;

//cout << list3[3] << endl;

cout << "Введите индекс для сдвига: ";

int n;

cin >> n;

it = list3.last();

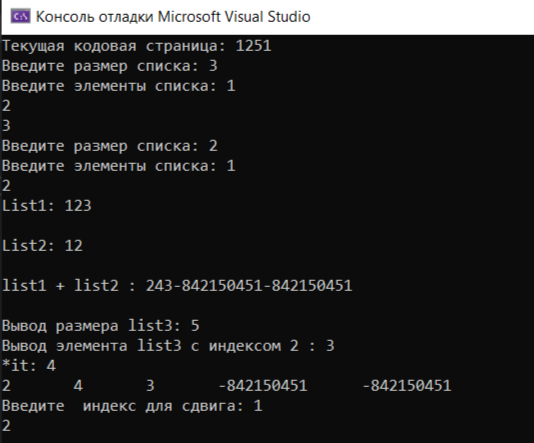
it - n;

cout << \*it << endl;

return 0;

}

**Результат работы программы:**



**Ответы на контрольные вопросы:**

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

АТД - тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов.

Примеры: стек, очередь, список

2. Привести примеры абстракции через параметризацию.

функция, передаваемые параметры которой являются формальными, а фактические связываются с ними в момент использования такой абстракции.

3. Привести примеры абстракции через спецификацию.

позволяет абстрагироваться от процесса вычислений, описанных в теле процедуры, до уровня знаний лишь того, что данная процедура должна в итоге реализовать.

4. Что такое контейнер? Привести примеры.

Контейнер - набор однотипных элементов. Самый простой пример контейнера - массив.

5. Какие группы операций выделяют в контейнерах?

Среди всех операций контейнера можно выделить несколько типовых групп:

• Операции доступа к элементам, которые обеспечивают и операцию замены значений элементов;  
• Операции добавления и удаления элементов или групп элементов;  
• Операции поиска элементов и групп элементов;  
• Операции объединения контейнеров;  
• Специальные операции, которые зависят от вида контейнера.

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

Доступ к элементам контейнера бывает: последовательный, прямой и ассоциативный.  
Прямой доступ — это доступ по индексу. Например, a[10] — требуется найти элемент контейнера с номером 10. В С++ нумерацию элементов контейнера принято начинать с нуля.  
Ассоциативный доступ также выполняется по индексу, но индексом будет являться не номер элемента, а его содержимое.   
При последовательном доступе осуществляется перемещение от элемента к элементу контейнера.

7. Что такое итератор?

Итератор - это объект, который обеспечивает доступ к его элементам, используя указатели.

8. Каким образом может быть реализован итератор?

Можно реализовать как класс или как часть класса.

9. Каким образом можно организовать объединение контейнеров?

- 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, которые есть хотя бы в одном контейнере, операция коммутативна.  
- Объединение контейнеров как пересечение множеств, в новый контейнер попадают только те элементы, которые есть в обоих контейнерах, операция коммутативна.

- Для контейнеров-множеств может быть еще реализована 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.

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

Ассоциативный доступ.

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

Такой контейнер называется стеком.

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

Ответ: d

13. Какой из объектов (a,b,c,d) не является контейнером?  
a. int a[]={1,2,3,4,5};

b. int mas[30];  
c. struct {char name[30]; int age;} mas[30];  
d. int mas;

Ответ: d

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

Последовательный доступ.

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

Прямой доступ.

**Ссылка на Git:**

https://github.com/Ananasic07/labs\_2sem/tree/main/класс%206