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

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**Тема:** Лабораторная работа №6.1 по ООП

Семестр: 2

Выполнил студент ИВТ-23-2б:

Синицын Владислав Сергеевич

(дата, подпись)

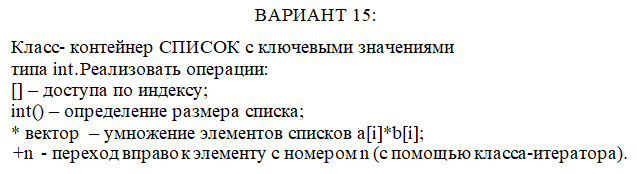
Проверила:

Полякова Ольга Андреевна

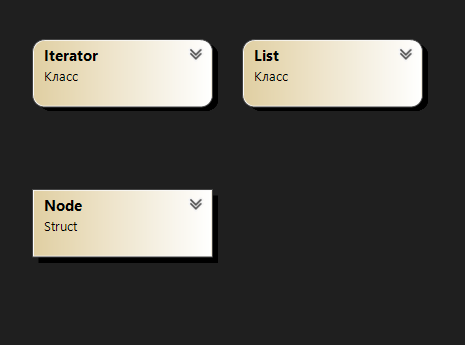
(дата, подпись)

Пермь 2024

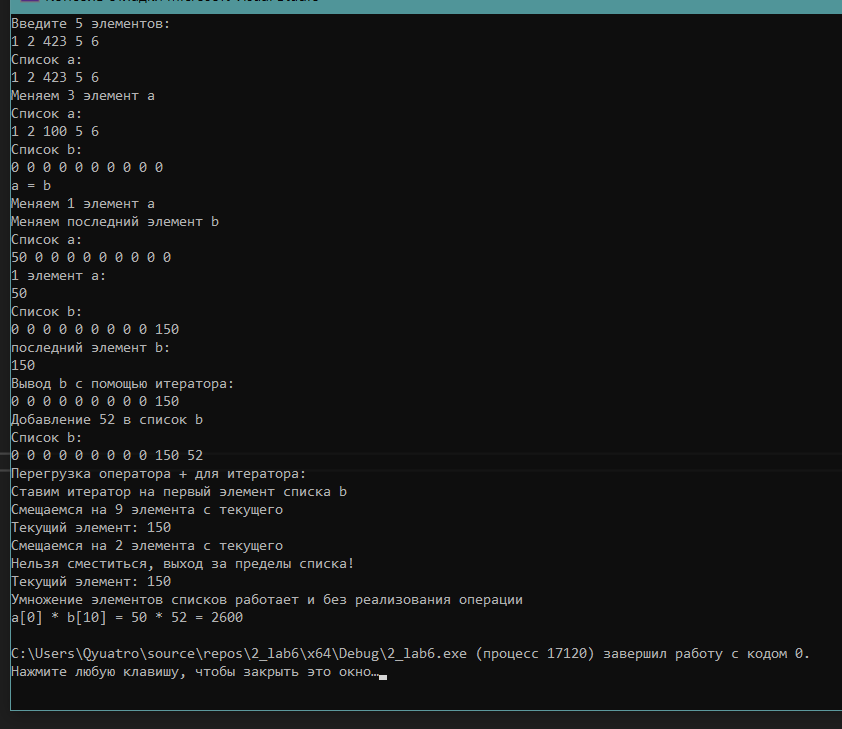
**Задание**



**UML-Диаграмма**



**Программное решение**

****

**List.h**

#pragma once

#include <string>

#include <iostream>

using namespace std;

struct Node {

int key;

Node\* next;

};

class Iterator {

private:

Node\* current;

friend class List;

public:

Iterator() {

current = NULL;

};

Iterator(Node\* node) {

current = node;

};

void operator ++ () {

current = current->next;

}

Node\* operator \*() const {

return current;

}

void operator + (int shift) {

Node\* tmp = current;

int i = 0;

while (i < shift && current != NULL && current->next != NULL) {

current = current->next;

i++;

}

if (i < shift) {

cout << "Нельзя сместиться, выход за пределы списка!\n";

current = tmp;

}

}

};

class List {

public:

List() {

n = 0;

head = new Node;

head->key = 0;

lastNd = head;

};

List(int);

~List();

List& operator = (List&);

int& operator[] (int);

int operator () ();

friend ostream& operator << (ostream&, List&);

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

Iterator first();

Iterator last();

void operator << (int);

private:

Node\* lastNd, \* current, \* head;

int n = 0;

Iterator beg, end;

};

**List.cpp**

#include "List.h"

List::List(int count) {

n = count;

head = new Node;

head->key = 0;

lastNd = head;

for (int i = 1; i < n; i++) {

current = new Node;

current->key = 0;

lastNd->next = current;

lastNd = current;

}

lastNd->next = NULL;

}

List::~List() {

lastNd = head;

while (lastNd != NULL) {

current = lastNd->next;

delete lastNd;

lastNd = current;

}

n = 0;

}

List& List::operator=(List& l) {

if (this != &l) {

if (this != 0) {

lastNd = head;

while (lastNd != NULL) {

current = lastNd->next;

delete lastNd;

lastNd = current;

}

n = 0;

}

lastNd = head = new Node;

l.lastNd = l.head->next;/////

lastNd->key = l.head->key;

while (l.lastNd != NULL) {

lastNd->next = new Node;

lastNd->next->key = l.lastNd->key;

l.lastNd = l.lastNd->next;

lastNd = lastNd->next;

}

lastNd->next = NULL;

n = l.n;

}

return \*this;

}

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

if (index < n) {

lastNd = head;

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

lastNd = lastNd->next;

}

return lastNd->key;

}

else {

cout << "Ошибка: Индекс за пределами диапазона!";

}

}

int List::operator () () {

return n;

}

Iterator List::first() {

beg.current = head;

return beg;

}

Iterator List::last() {

lastNd = head;

while (lastNd->next != NULL) {

lastNd = lastNd->next;

}

end.current = lastNd;

return end;

}

ostream& operator<<(ostream& out, List& l) {

if (l.n) {

l.lastNd = l.head;

while (l.lastNd != NULL) {

out << l.lastNd->key << ' ';

l.lastNd = l.lastNd->next;

}

}

else {

out << "Список пуст!";

}

return out;

}

istream& operator>>(istream& in, List& l) {

l.lastNd = l.head;

while (l.lastNd != NULL) {

in >> l.lastNd->key;

l.lastNd = l.lastNd->next;

}

return in;

}

void List::operator << (int number) {

if (head == NULL) {

head = new Node;

head->key = number;

head->next = NULL;

}

else {

lastNd = head;

while (lastNd->next != NULL) {

lastNd = lastNd->next;

}

lastNd->next = new Node;

lastNd->next->key = number;

lastNd = lastNd->next;

lastNd->next = NULL;

}

n += 1;

}

**2\_lab6.cpp**

#include <iostream>

using namespace std;

#include "List.h"

int main() {

system("chcp 1251 > NULL");

List a(5);

cout << "Введите 5 элементов:\n";

cin >> a;

cout << "Список a:\n";

cout << a << '\n';

cout << "Меняем 3 элемент а\n";

a[2] = 100;

cout << "Список a:\n";

cout << a << '\n';

List b(10);

cout << "Список b:\n";

cout << b << '\n';

a = b;

cout << "a = b\n";

cout << "Меняем 1 элемент а\n";

a[0] = 50;

cout << "Меняем последний элемент b\n";

b[b()-1] = 150;

cout << "Список a:\n";

cout << a << '\n';

cout << "1 элемент a:\n" << (\*(a.first()))->key << '\n';

cout << "Список b:\n";

cout << b << '\n';

cout << "последний элемент b:\n" << (\*(b.last()))->key << '\n';

cout << "Вывод b с помощью итератора:\n";

for (Iterator i = b.first(); \*i != NULL; ++i) {

cout << (\*i)->key << ' ';

}

cout << '\n';

cout << "Добавление 52 в список b\n";

b << 52;

cout << "Список b:\n";

cout << b << '\n';

cout << "Перегрузка оператора + для итератора:\n";

cout << "Cтавим итератор на первый элемент списка b\n";

Iterator i = b.first();

cout << "Смещаемся на 9 элемента с текущего\n";

i + 9;

cout << "Текущий элемент: " << (\*i)->key << '\n';

cout << "Смещаемся на 2 элемента с текущего\n";

i + 2;

cout << "Текущий элемент: " << (\*i)->key << '\n';

cout << "Умножение элементов списков работает и без реализования операции\na[0] \* b[10] = " << a[0] << " \* " << b[10] << " = " << a[0] \* b[10] << '\n';

return 0;

}

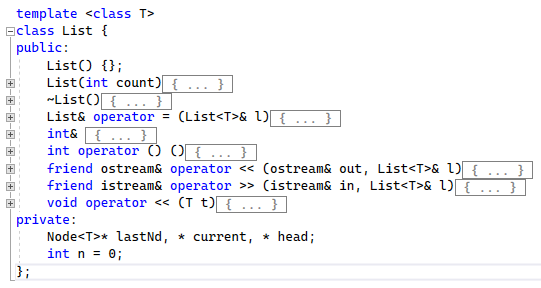
**Контрольные вопросы**

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. Примером абстрактного типа данных является класс в языке С++.

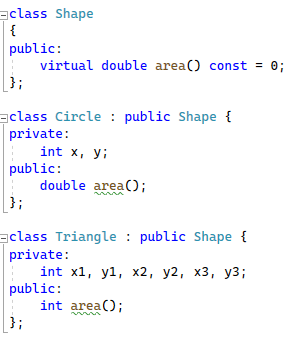
Пример абстрактного типа данных - класс Итератор

2. Привести примеры абстракции через параметризацию.



В этом примере класс «List» параметризован типом «T», который определяется при создании объекта класса. Это позволяет использовать один и тот же класс для работы с различными типами данных.

3. Привести примеры абстракции через спецификацию.



Класс Shape содержит чисто виртуальный метод area().

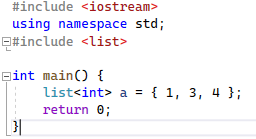
Класс Triangle является производным от класса Shape и реализует метод area(). Он содержит приватные поля x1, y1, x2, y2, x3, y3, которые используются для вычисления площади.

Класс Circle тоже является производным от класса Shape и реализует метод area(). Он содержит приватные поля x, y, которое используется для вычисления площади.

Оба класса «Rectangle» и «Circle» реализуют интерфейс, определенный в абстрактном классе «Shape», что позволяет использовать полиморфизм для работы с различными типами геометрических фигур.

4. Что такое контейнер? Привести примеры.

Контейнер – набор однотипных элементов.

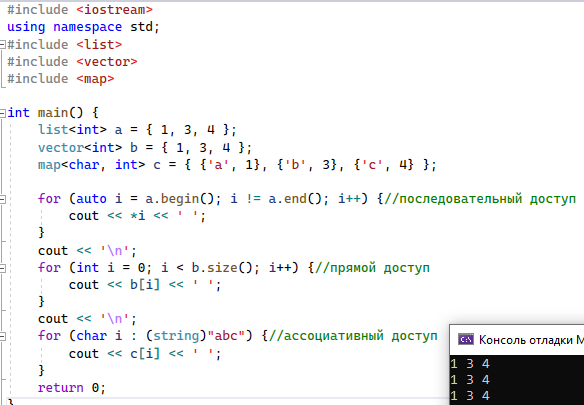


5. Какие группы операций выделяют в контейнерах?

1. Операции доступа к элементам
2. Операции добавления и удаления
3. Операции поиска
4. Операции объединения контейнеров

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

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



7. Что такое итератор?

Итератор – это объект, который обеспечивает последовательный доступ к элементам контейнера. Итератор может быть реализован как часть класса-контейнера в виде набора методов.

8. Каким образом может быть реализован итератор?

Как класс

9. Каким образом можно организовать объединение контейнеров?

1. Сложение множеств
2. Пересечение множеств
3. Вычитание множеств

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

По ключу

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

Стек

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];

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?

Прямой доступ.

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

Последовательный доступ.