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**

**о работе по информатике**

Семестр: 2

На тему: «ООП.Стек»

Выполнил студент ИВТ-22-2б:

Казанцев Антон Васильевич

(дата, подпись)

Проверила:

Полякова Ольга Андреевна

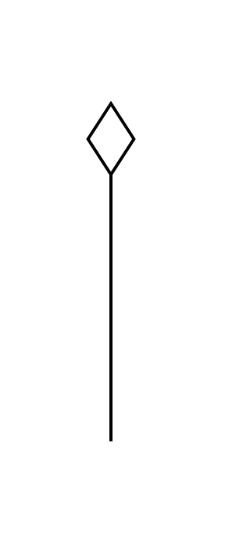
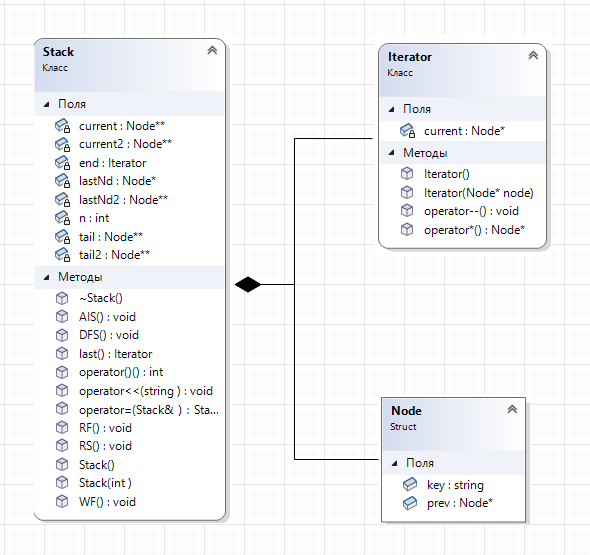
(дата, подпись)

Пермь 2023

**Задание**

15. записи в линейном списке содержат ключевое поле типа\* char(строка символов ) .Сформировать однонаправленный список. Удалить элемент с заданным ключом. Добавить К элементов перед элементом с заданньм номером.

**UML-Диаграмма**

****

**Программное решение**

***Stack.h***

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Node {

string key;

Node\* prev;

};

class Iterator {

private:

Node\* current;

friend class Stack;

public:

Iterator() {};

Iterator(Node\* node) {

current = node;

};

void operator -- () {

current = current->prev;

}

Node\* operator \*() const {

return current;

}

};

class Stack {

private:

Node\* lastNd, \* current, \* tail, \* lastNd2, \* current2, \* tail2;

int n = 0;

Iterator end;

public:

Stack() {};

Stack(int);

~Stack();

Stack& operator = (Stack&);

int operator () ();

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

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

Iterator last();

void operator << (string);

void AIS() {//AddIntoStack

int l;

cout << "Введите номер элемента: ";

cin >> l;

lastNd2 = tail;

tail = tail->prev;

lastNd2->prev = NULL;

for (int i = 1; i < n - l + 1; i++) {//Перемещаем все элементы во стек 2 до нужного номера включительно

current2 = tail;

tail = tail->prev;

current2->prev = lastNd2;

lastNd2 = current2;

}

tail2 = lastNd2;

cout << "Введите количество элементов: ";

cin >> l;

n += l;

lastNd = tail;

for (int i = 0; i < l; i++) {//Вводим новые K элементов в стек 1

current = new Node;

cout << "Введите " << i + 1 << " элемент: ";

cin >> current->key;

current->prev = lastNd;

lastNd = current;

}

tail = lastNd;

while (tail2 != NULL) {//возвращаем обратно списанные элементы

current = tail2;

tail2 = tail2->prev;

current->prev = lastNd;

lastNd = current;

}

tail = lastNd;

}

void DFS() {//DeleteFromStack

string l;

cout << "Введите ключ элемента: ";

cin >> l;

current = NULL;

lastNd = tail;

bool replace = false;

if (lastNd->key == l) {//Если нужный элемент является последним, то просто удаляем его из стека

tail = lastNd->prev;

delete lastNd;

replace = true;

n--;

}

else {

lastNd2 = lastNd;

lastNd = lastNd->prev;

lastNd2->prev = NULL;

while (lastNd != NULL) {//переписывем все элементы из стека 1 в стек 2, кроме удаляемого

current2 = lastNd;

lastNd = lastNd->prev;

if (current2->key != l) {

current2->prev = lastNd2;

lastNd2 = current2;

}

else {

delete current2;

n--;

}

}

tail = lastNd2;//замена стека 1 стеком 2, теперь стек 1 отражённый и не содержит в себе элемента с заданным ключом

lastNd2 = NULL;

while (tail != NULL) {//переписываем все элементы из стека 1 в стек 2, таким образом возвращая исходный стек

current2 = tail;

tail = tail->prev;

current2->prev = lastNd2;

lastNd2 = current2;

}

tail = lastNd2;//замена стека 1 стеком 2

}

}

void WF() {//WriteFile

ofstream F("F.txt");

F << ' ' << n;

lastNd2 = tail;

tail = tail->prev;

lastNd2->prev = NULL;

while (tail != NULL) {//переписываем все элементы из стека 1 в стек 2, таким образом переворачивая исходный стек

current2 = tail;

tail = tail->prev;

current2->prev = lastNd2;

lastNd2 = current2;

}

tail = lastNd2;//замена стека 1 стеком 2

lastNd2 = NULL;

while (tail != NULL) {//записываем элементы в файл и восстанавливаем исходный стек в стек 2

F << ' ' << tail->key;

current2 = tail;

tail = tail->prev;

current2->prev = lastNd2;

lastNd2 = current2;

}

tail = lastNd2;//заменяем стек 1 на стек 2

F.close();

}

void RF() {//ReadFile

ifstream F("F.txt");

F >> n;

lastNd = NULL;

while (F.get() && !F.eof()) {

current = new Node;

F >> current->key;

current->prev = lastNd;

lastNd = current;

}

tail = lastNd;

F.close();

}

void RS() {//RemoveStack

lastNd = tail;

while (lastNd != NULL) {

current = lastNd->prev;

delete lastNd;

lastNd = current;

}

n = 0;

}

};

***Stack.cpp***

#include "Stack.h"

Stack::Stack(int count) {

n = count;

lastNd = NULL;

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

current = new Node;

current->key = "";

current->prev = lastNd;

lastNd = current;

}

tail = lastNd;

}

Stack::~Stack() {

this->RS();

}

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

if (this != &l) {

if (this != 0) {

this->RS();

}

lastNd = new Node;

l.lastNd = l.tail->prev;

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

lastNd->prev = NULL;

while (l.lastNd != NULL) {

current = new Node;

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

l.lastNd = l.lastNd->prev;

current->prev = lastNd;

lastNd = current;

}

tail = lastNd;

n = l.n;

lastNd2 = NULL;

while (tail != NULL) {//восстанавливаем исходный стек в стек 2

current2 = tail;

tail = tail->prev;

current2->prev = lastNd2;

lastNd2 = current2;

}

tail = lastNd2;//заменяем стек 1 на стек 2

}

return \*this;

}

int Stack::operator () () {

return n;

}

Iterator Stack::last() {

end.current = tail;

return end;

}

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

if (l.n > 0) {

l.lastNd2 = l.tail;

l.tail = l.tail->prev;

l.lastNd2->prev = NULL;

while (l.tail != NULL) {//переписываем все элементы из стека 1 в стек 2, таким образом переворачивая исходный стек

l.current2 = l.tail;

l.tail = l.tail->prev;

l.current2->prev = l.lastNd2;

l.lastNd2 = l.current2;

}

l.tail = l.lastNd2;//замена стека 1 стеком 2

l.lastNd2 = NULL;

while (l.tail != NULL) {//выводим элементы и восстанавливаем исходный стек в стек 2

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

l.current2 = l.tail;

l.tail = l.tail->prev;

l.current2->prev = l.lastNd2;

l.lastNd2 = l.current2;

}

l.tail = l.lastNd2;//заменяем стек 1 на стек 2

}

else {

out << "Стек пуст!";

}

return out;

}

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

l.lastNd = l.tail;

while (l.lastNd != NULL) {

in >> l.lastNd->key;

l.lastNd = l.lastNd->prev;

}

l.lastNd2 = NULL;

while (l.tail != NULL) {//восстанавливаем исходный стек в стек 2

l.current2 = l.tail;

l.tail = l.tail->prev;

l.current2->prev = l.lastNd2;

l.lastNd2 = l.current2;

}

l.tail = l.lastNd2;//заменяем стек 1 на стек 2

return in;

}

void Stack::operator << (string str) {

if (tail == NULL) {

tail = new Node;

tail->key = str;

tail->prev = NULL;

}

else {

lastNd = new Node;

lastNd->key = str;

lastNd->prev = tail;

tail = lastNd;

}

n += 1;

}

***LabaOOP6S.cpp***

#include <iostream>

using namespace std;

#include "Stack.h"

int main() {

system("chcp 1251 > NUSS");

Stack a(3);

cout << "Введите 3 элемента: ";

cin >> a;

cout << "Стек a:\n";

cout << a << '\n';

cout << "Выполнение команды добавления элементов перед элементом с заданным номером\n";

a.AIS();

cout << "Стек a:\n";

cout << a << '\n';

cout << "Выполнение команды удаления элемента с заданным ключом\n";

a.DFS();

cout << "Стек a:\n";

cout << a << '\n';

cout << "Записываем стек в файл\n";

a.WF();

cout << "Удаляем стек\n";

a.RS();

cout << "Проверям что стек пуст:\n";

cout << "Стек a:\n";

cout << a << '\n';

cout << "Считываем стек из файла:\n";

a.RF();

cout << "Стек a:\n";

cout << a << '\n';

cout << "Длина стека а:\n";

cout << a() << '\n';

cout << "Последний элемент стека a:\n";

cout << (\*(a.last()))->key << '\n';

cout << "Вывод стека через итератор:\n";

Stack b;//b - это стек, обратный стеку a

for (Iterator i = a.last(); \*i != NULL; --i) {

b << (\*i)->key;

}

cout << "Стек a:\n";

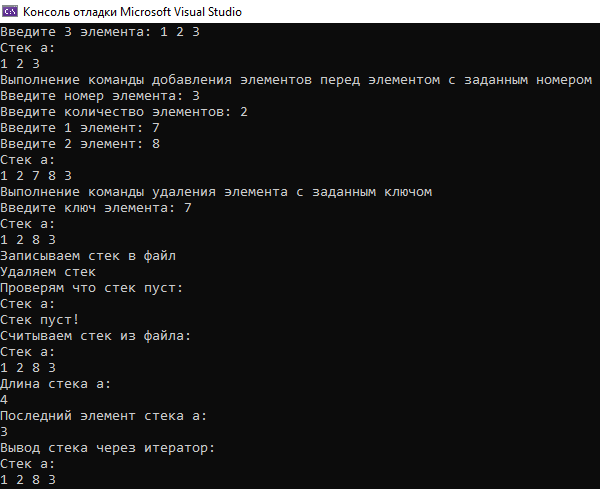
for (Iterator i = b.last(); \*i != NULL; --i) {

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

}

return 0;

}



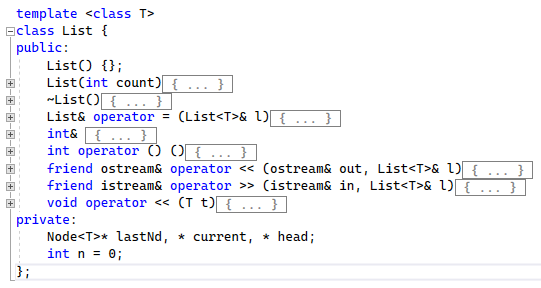
**Контрольные вопросы**

*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++.

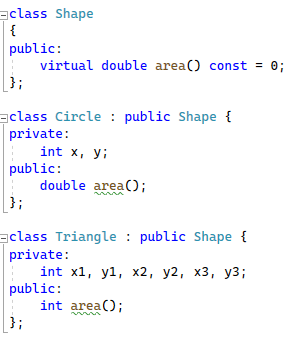
Пример абстрактного типа данных - класс Итератор

*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().

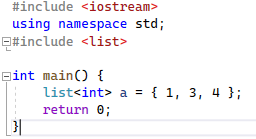
Класс Triangle является производным от класса Shape и реализует метод area(). Он содержит приватные поля x1, y1, x2, y2, x3, y3, которые используются для вычисления площади.

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.

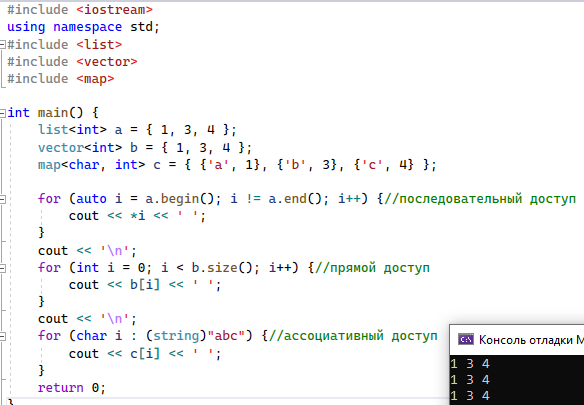


*5. Какие группы операций выделяют в контейнерах?*

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.