МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра информационных систем управления**

**ЛАБОРАТОРНАЯ РАБОТА № 7**

По дисциплине «Объектно-ориентированное программирование»

На тему «Динамические структуры в языке C++»

|  |
| --- |
| Выполнил студент гр. Б8219 |
| О.В. Константинов |
|  |
| Проверил старший преподаватель |
| Г.Л. Берёзкина |
|  |
| (зачтено/не зачтено) |

г. Владивосток

2016

# Аннотация

Данный отчет подготовлен в рамках задания по дисциплине «Объектно-ориентированное программирование». Отчет призван систематизировать знания об использовании и реализации классов в языке C++. Отчет содержит список выполненных заданий с реализацией на языке C++ в среде программирования MS Visual Studio 2016.

# Задача 8

## Файл *“list\_lib.h”*

#pragma once

template <class T> class List {

private:

struct Element {

T data;

Element \*next;

Element \*prev;

};

Element \*Head;

Element \*Curr;

int length;

public:

int size() { return length; };

List() : Head(0), Curr(0), length(0) {};

~List() { for (int i = 0; i < length; i++) pop(); };

void init() { Curr = Head; };

void push(T data);

T pop();

void search(T adr) {

for (int i = 0; i < length; i++, next())

if (Curr->data == adr) return;

};

void loop();

void next() { Curr = Curr ? Curr->next : 0; };

void prev() { Curr = Curr ? Curr->prev : 0; };

T lp() { return Curr ? Curr->data : T(); };

};

template <class T> void List<T>::push(T data) {

Element \*inserted = new Element;

inserted->data = data;

if (!Curr) {

Head = inserted;

Curr = inserted;

Curr->next = inserted;

Curr->prev = inserted;

}

else {

inserted->next = Curr->next;

inserted->next->prev = inserted;

Curr->next = inserted;

inserted->prev = Curr;

}

length++;

Curr = inserted;

}

template <class T> T List<T>::pop() {

if (!Curr) return 0;

Element \*temp = Curr;

T tag = temp->data;

if (length == 1)

Head = Curr = 0;

else {

Curr->next->prev = Curr->prev;

Curr->prev->next = Curr->next;

Curr = Curr->prev;

}

if (temp == Head)

Head = Head->next;

length--;

delete temp;

return tag;

}

#include <iostream>

template <class T> void List<T>::loop() {

if (!Curr) return;

Element \*temp = Head;

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

std::cout << temp->data << " ";

temp = temp->next;

}

}

## Файл *“main.cpp”*

/\*

Задача 8. Написать и протестировать функции для включения, исключения и поиска элемента кругового списка для:

а) списка без заголовка;

б) списка с заголовком (заголовок может содержать некоторую информацию о списке, например, число элементов в списке).

\*/

#include "list\_lib.h"

#include <iostream>

using namespace std;

typedef bool(\*TEST\_FuncT)();

void TEST\_FUNCTION(TEST\_FuncT funcs[], int size) {

cout << "TEST COUNT " << size << endl;

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

cout << "TEST " << i + 1 << " ";

funcs[i]() ? cout << "OK" : cout << "FAIL";

cout << endl;

}

cout << "TEST IS FINISHED" << endl;

}

template<typename T> bool CompareList(List<T> &A, const T \*C, int size) {

#if \_DEBUG

cout << endl << "LIST(" << A.size() << ") = { ";

A.loop();

cout << "}" << endl;

cout << "ARRAY(" << size << ") = { ";

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

cout << C[i] << " ";

cout << "}" << endl;

#endif

if (A.size() != size)

return false;

A.init();

for (int i = 0; i < A.size(); i++, A.next())

if (A.lp() != C[i])

return false;

return true;

}

bool TEST\_ListDefault() {

List<int> L;

return L.size() == 0 && L.lp() == 0;

}

bool TEST\_ListPush() {

List<int> L;

int a[] = { 1, 2, 3, 4 };

L.push(1);

L.push(2);

L.push(3);

L.push(4);

return CompareList<int>(L, a, 4);

}

bool TEST\_ListPop() {

List<int> L;

int a[] = { 1, 2, 4 };

L.push(1);

L.push(2);

L.push(3);

L.push(4);

L.prev();

L.pop();

return CompareList<int>(L, a, 3);

}

bool TEST\_ListEmpty() {

List<int> L;

L.push(1);

L.push(2);

bool res = L.size() == 2;

L.pop();

L.pop();

return res && L.size() == 0;

}

bool TEST\_ListIterators() {

List<int> L;

L.push(1);

L.push(2);

L.push(3);

bool res = L.lp() == 3;

L.prev();

res = res && L.lp() == 2;

L.next();

L.next();

res = res && L.lp() == 1;

L.prev();

return res && L.lp() == 3;

}

bool TEST\_ListSearch() {

List<char> L;

L.push('1');

L.push('2');

L.push('3');

L.push('\*');

L.push('5');

bool res = L.lp() == '5';

L.search('@');

res = res && L.lp() == '5';

L.search('\*');

return res && L.lp() == '\*';

}

int main() {

TEST\_FuncT funcs[] = {

TEST\_ListDefault,

TEST\_ListPush,

TEST\_ListPop,

TEST\_ListEmpty,

TEST\_ListIterators,

TEST\_ListSearch

};

TEST\_FUNCTION(funcs, sizeof(funcs) / sizeof(TEST\_FuncT));

system("PAUSE");

return 0;

}