Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Тестирование кода с лекции»

Выполнила:

Студент(ка) 1 курса 7 группы

Подшиваленко Диана Игоревна

Проверил:

Белодед Николай Иванович

2024, Минск

#include <iostream>

#include <Windows.h>

using namespace std;

struct node { // структура для узла списка

int value; // информационное поле для хранения переменной

node\* next; // поле ссылки н следующий элемент

};

int main()

{

setlocale(0, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

node\* phead, \* t; // создаем указатель на первый элемент списка и буферный указатель

phead = new (node); // выделяем для него память

t = phead; // присваиваем буферному указателю адрес первого элемента списка

(\*t).value = 1; // записываем в информационное поле первого элемента значение 1

(\*t).next = new(node); // выделяем память для следующего элемента

t = (\*t).next; // присваиваем буферному указателю адрес следующего элемента, под который мы выделили память

(\*t).value = 2; // записываем в информационное поле второго элемента значение 2

(\*t).next = new(node); // выделяем память для следующего элемента

t = (\*t).next; // присваиваем буферному указателю адрес следующего элемента, под который мы выделили память

(\*t).value = 6;// записываем в информационное поле третьего элемента значение 6

(\*t).next = new(node);// выделяем память для следующего элемента

t = (\*t).next;// присваиваем буферному указателю адрес следующего элемента, под который мы выделили память

(\*t).value = 17;// записываем в информационное поле четвертого элемента значение 17

(\*t).next = new(node);// выделяем память для следующего элемента

(\*t).next = NULL;;// последнему указателю передаем значение NULL для указание на конец списка

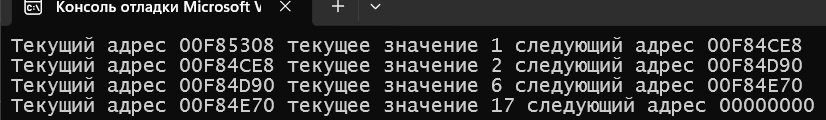
for (t = phead; t != NULL; t = (\*t).next) { // в цикле, начиная с первого элемента и, пока не дойдем да последнего

cout << "Текущий адрес " << t << " текущее значение " << (\*t).value << " следующий адрес " << (\*t).next << '\n'; // выводим текущий адрес, текущее значение, следующий адрес

}

return 0;

}

Результат выполнения:

#include <iostream>

#include <Windows.h>

using namespace std;

class Spisok { // объявляем класс для списка

private: // часть кода для приватных функций

struct node { // описываем структуру для узлов списка

int elem; // информационное поле для элементов

node\* sled; // поле для ссылки на следующий элемент списка

} \*phead; // создаем указатель phead типа node

public: // часть кода для публичных(общедоступных функций)

Spisok() { // создаем конструктор списка

phead = new(node); // выделяем память под указатель на список

(\*phead).sled = NULL; // и присваиваем ссылке на следующий элемент значение NULL

}

~Spisok() { // деструктор списка

delete phead; // очищение(освобождение) выделенной памяти

}

void POSTROENIE(); // прототип функции построения списка

void VYVOD(); // прототип функции вывода списка

void OCHISTKA(); // прототип функции очистки

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Spisok A; // создаем объект А класса Spisok

A.POSTROENIE(); // запускаем функцию построения списка

A.VYVOD(); // запускаем функцию вывода

A.OCHISTKA(); // запускаем функцию очистки

cout << '\n';

return 0;

}

void Spisok::POSTROENIE() { // функция для построения списка

node\* t; // создаем буферный указатель типа node

int el; // создаем буферную переменнуб для считывания элементов с клавиатуры

t = phead; // присваиваем буферному указателю адрес первого элемента списка

cout << "Введите элементы списка\n"; // просим ввести элементы списка

cin >> el; // читаем элемент

while (el != 0) { // в цикле: пока элемент не равен 0(это будет признак окончания ввода)

(\*t).sled = new(node); // выделяем память под следующий элемент списка

t = (\*t).sled; // и переносим наш буферный указатель на этот элемент

(\*t).elem = el; // записываем в информационное поле считанное значение

(\*t).sled = NULL; // в указатель на следующий элемент заносим значение NULL, что будет указывать на конец списка

cin >> el; // считываем следующий элемент

}

}

void Spisok::VYVOD() { // функция для вывода списка

node\* t; // создаем буферный указатель

t = (\*phead).sled; // и передаем адрес 2 элемента(так как phead ссылается на первый элемент, а мы обращаемся к его следующему элементу)

cout << "Список \n";

while (t) { // в цикле: пока буферный указатель не дойдет до конца списка(пока t не будет указывать на NULL)

cout << (\*t).elem << ' '; // выводим информационное поле текущего элемента

t = (\*t).sled; // передаем в указатель адрес следующего элемента списка

}

cout << '\n';

}

void Spisok::OCHISTKA() { // функция очистки

node \*q, \*q1; // создаем два буферных указателя

q = phead; // присваиваем первому указателю адрес первого элемента списка

q1 = (\*q).sled; // второму указателю присваиваем адрес следующего элемента списка

while (q1) { // в цикле: пока буферный указатель не дойдет до конца списка(пока q1 не будет указывать на NULL)

q = q1; // передаем в указатель q адрес q1

q1 = (\*q1).sled; // указателю q1 присваиваем адрес следующего элемента

delete q; // очищаем(освобождаем) память, на которую указывает q

}

}

Результат выполнения:

