Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Тестирование кода из лекции»

Выполнил:

Студент 1 курса 8 группы

Лужецкий Владислав Константинович

Проверил:

Белодед Николай Иванович

2024, Минск

#include <iostream>

#include <Windows.h>

using namespace std;

struct node { // структура для узла списка

int value; //значение текущего элемента

node\* next; //указатель на следующий элемент

};

int main()

{

setlocale(0, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

node\* phead, \* t; //указатель на 1 элемент списка, указатель для временного хранения

phead = new (node); //выделение памяти

t = phead; //присваиваем буф. указателю адрес 1 элемента списка

(\*t).value = 1; //присваиваем текущему элементу значение 1

(\*t).next = new(node); //выделение памяти

t = (\*t).next; //сдвигаем на следующий элемент

(\*t).value = 2; //присваиваем текущему элементу значение 2

(\*t).next = new(node); //выделение памяти

t = (\*t).next; //сдвигаем на следующий элемент

(\*t).value = 6; //присваиваем текущему элементу значение 6

(\*t).next = new(node); //выделение памяти

t = (\*t).next; //сдвигаем на следующий элемент

(\*t).value = 17; //присваиваем текущему элементу значение 17

(\*t).next = new(node); //выделение памяти

(\*t).next = NULL;; //следующий элемент равен 0, конец списка

for (t = phead; t != NULL; t = (\*t).next) //цикл начинающийся с 1 элемента списка и до окончания списка(=NULL)

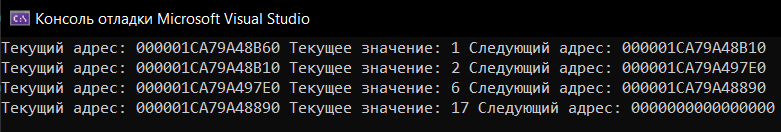
{

cout << "Текущий адрес: " << t << " Текущее значение: " << (\*t).value << " Cледующий адрес: " << (\*t).next << endl; //вывод адреса и значения текущего элемента и адреса след. эл.

}

return 0;

}



#include <iostream>

#include <Windows.h>

using namespace std;

class Spisok //объявление класса для списка

{

private: //фрагмент кода для приватных функций

struct node

{

int elem; // Значение элемента списка

node\* sled; // Указатель на следующий элемент списка

} \*phead; // Указатель на 1 эл. списка

public: // фрагмент кода для общедоступных функций

Spisok() // конструктор списка

{

phead = new(node); // выделение памяти

(\*phead).sled = NULL; // Инициализация указателя следующего элемента

}

~Spisok() // деструктор списка

{

delete phead; // Освобождение памяти, занятой 1 эл списка

}

void POSTROENIE(); // Прототип для построения списка

void VYVOD(); // Прототип для вывода списка

void OCHISTKA(); // Прототип для очистки списка

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Spisok A; // Создание объекта класса Spisok

A.POSTROENIE(); // Вызов ф-ции POSTROENIE для построения списка

A.VYVOD(); // Вызов ф-ции VYVOD для вывода списка

A.OCHISTKA(); // Вызов ф-ции OCHISTKA для очистки списка

cout << '\n';

return 0;

}

void Spisok::POSTROENIE()

{

node\* t; // Вспомогательный указатель

int el; // Переменная для ввода эл. списка

t = phead; // Начинаем с 1 эл. списка

cout << "Введите элементы списка:\n";

cin >> el; // Ввод первого элемента списка

while (el != 0)

{

(\*t).sled = new(node); // Выделение памяти для нового элемента списка

t = (\*t).sled; // Переход к следующему элементу

(\*t).elem = el; // Присвоение введенного значения текущему элементу

(\*t).sled = NULL; // Установка указателя следующего элемента == NULL

cin >> el; // Ввод следующего элемента списка

}

}

void Spisok::VYVOD()

{

node\* t; // Вспомогательный указатель

t = (\*phead).sled; // Начинаем с первого элемента списка

cout << "Список: \n";

while (t)

{

cout << (\*t).elem << ' '; // Вывод значения текущего элемента

t = (\*t).sled; // Переход к следующему элементу

}

cout << '\n';

}

void Spisok::OCHISTKA()

{

node\* q, \* q1; // Вспомогательные указатели для освобождения памяти

q = phead; // Начинаем с 1 эл. списка

q1 = (\*q).sled; // Сдвигаем на 1 эл. дальше

while (q1)

{

q = q1; // Переход к следующему элементу

q1 = (\*q1).sled; // Указатель на следующий элемент

delete q; // Освобождение памяти текущего элемента

}

}

