МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет: Информационных технологий

Кафедра: Программной инженерии

Тестирование кодов

По дисциплине” Основы Алгоритмизации и Программирования”

***Выполнил:*** *студент 1 курса 8 группы*

**Специальности:** *ПИ Федорович В.Г.*

**Проверил:** преподаватель *Белодед Николай Иванович*

Минск

2024

Навигация:

1. Введение………………………………………..3
2. Код Тестирования 1………………………..4
3. Код Тестирования 2………………………..5
4. Вывод………………………………………………7

Введение:

В данном реферате предоставлены коды, данные вами на лекции с заданием тестирования.

Они же представлен с комментариями, выводом адресов узлов(элементов списка). – Код 1

Доработан и прокомментирован –Код 2,

Так же в конце вы можете найти вывод о плюсах списка.

Код -1

#include<iostream>

using namespace std;

// Определение структуры

struct node

{

int value; // информация, содержащееся в узле

node\* next; // Указатель на след. узел

};

void main() {

setlocale(0, "");

int i;

node\* phead, \* t; // Указатели на голову списка и временный указатель

// Выделение памяти под 1-ый узел и инициализация указателей

phead = new(node);

t = phead;

// Инициализация содержимого первого узла

(\*t).value = 1;

(\*t).next = new(node); // Выделение памяти под след. узел

t = (\*t).next; // Перемещение указателя на следующий узел

// Инициализация значений второго узла

(\*t).value = 2;

(\*t).next = new(node);

t = (\*t).next;

// Инициализация значений третьего узла

(\*t).value = 6;

(\*t).next = new(node);

t = (\*t).next;

// Инициализация значений четвертого узла

(\*t).value = 17;

(\*t).next = new(node);

(\*t).next = NULL; // Установка указателя на следующий узел в NULL, тк это последний узел

// Вывод содержимого всех узлов

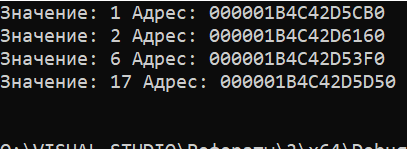
for (t = phead; t != NULL; t = (\*t).next)

{

cout << "Значение: " << (\*t).value << " "; cout << "Адрес: " << t << endl; // Вывод значения и адреса текущего узла

}cout << endl;

}



Код - 2

#include<iostream>

using namespace std;

class Spisok

{

private:

struct node

{

int elem; // Значение элемента

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

} \*phead; // Указатель на начало списка

public:

// Конструктор класса, инициализирующий начало списка

Spisok() {phead = new(node);

(\*phead).sled = nullptr;}// Устанавливаем указатель на следующий элемент как nullptr или 0

// Деструктор класса (для освобождения памяти, выделеной под список)

~Spisok() {delete phead;}

// Метод для построения списка пользователем

void POSTROENIE();

// Метод для вывода содержимого списка

void VYVOD();

// Метод для освобождения памяти списка

void OCHISTKA();

};

void Spisok::POSTROENIE() {

node\* t; // Указатель на действующий элемент списка

int el; // Переменная для хранения вводимого пользователем элемента

phead = new (node);// Создаение первого элемента списка

t = phead; // Устанавливаем указатель на начальный элемент

(\*t).sled = NULL; // Указатель на начало, тк следующего элемента пока нет

cout << "Вводите элементы звеньев списка: " << endl;

cin >> el;

while (el != 0) {

(\*t).sled = new(node); //Создание нового узла

t = (\*t).sled; // Переход на следующий элемент списка

(\*t).elem = el; // Присваивание элементу нового узла значение el

(\*t).sled = NULL; // Указатель на 0 (пока последний элемент)

cin >> el;

}

}

void Spisok::VYVOD()

// Вывод содержимого однонаправленнго линейного списка

// С заглавным звеном

// phead - указатель на заглавное звено списка

{

node\* t; // Объявление указателя t на структуру node

t = (\*phead).sled; //Присваивание указателю t адреса 1 элемента списка

cout << " Список :";

while (t != NULL) { // Цикл выполняется пока указаьель t не будет равен 0

cout << (\*t).elem << " "; // Вывод значения узла

t = (\*t).sled; // Перемещения указателя на следующий узел

}

cout << endl;

}

void Spisok::OCHISTKA() {

// Удаление однонаправлено списка из памяти

// \*phead - указатель на заглавное звено списка

struct node \*q, \*q1; // Рабочие указатели

q = phead;// Инициализация укаателя, указывающего на заголовочный узел

q1 = (\*q).sled; // Указатель q1 "опережает" указатель q.

while (q1 != NULL) { // Цикл выполняется пока указаьель q1 не будет равен 0

q = q1; // Передвижение указателя q на следующий узел списка(для удаления)

q1 = (\*q1).sled; // Так же перемещение указателя q1 на следующий узел списка(для продолжения цикла)

delete q; // Удаление

}

}

int main() {

setlocale(0, "");

Spisok nazva;

// Вызов функий

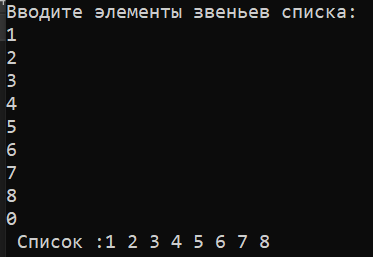
nazva.POSTROENIE();

nazva.VYVOD();

nazva.OCHISTKA();

return 0;

}



Вывод:

Списки полезны тем, что:

1. Списки позволяют хранить данные любого размера и в любом порядке.
2. Списки позволяют выделять память по мере необходимости, это полезно, когда кол-во элементов может меняться.

*И самое важное на мой взгляд:*

Возможность применения при неизвестном числе входных данных. Т.е. Списки идеально подходят для ситуаций, когда заранее неизвестно, сколько элементов будет обрабатываться. Они позволяют легко добавлять новые элементы по мере необходимости, что делает их эффективным выбором при работе с переменным или неопределенным объемом данных.