**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»  
 РУТ (МИИТ)**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Кафедра: «Вычислительные системы и сети»  
  
  
  
  
  
  
  
  
  
  
**О Т Ч Ё Т  
П О Л А Б О Р А Т О Р Н О Й Р А Б О Т Е № 3**  
по дисциплине «Структуры и алгоритмы обработки данных»  
на тему: «Линейные списки при связанном размещении памяти»  
  
  
  
  
  
  
  
  
  
 Выполнил: студент группы УИБ-311   
 Владыка П.А.

Москва 2018

**Задание**

Разработать и отладить программу, выполняющую обработку нескольких линейных списков при связанном распределении памяти. Предусмотреть возможность вставки, удаления и просмотра элементов, используя меню выбора.

|  |  |  |
| --- | --- | --- |
| № | Списки | Дополнительные операции |
| 4 | стек, очередь, дек | - |

**Таблица имён**

|  |  |
| --- | --- |
| Исходные данные | |
| Knot | Указатель на 1 элемент списка |
| Промежуточные данные | |
| inf | Информационное поле элемента списка |
| pLastElement | Указатель на прошлый элемент списка |
| Element | Указатель на текущий элемент списка |
| Value | Значение, которое будет записываться в поле inf структуры |

**Блок-схема**

**Главная программа**



**Исходный код программы**

/\* Copyright (c) PeterTheMighty. All rights reserved \*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<windows.h>

#include <iomanip>

using namespace std;

/\* Класс создан для работы с односвязным списком \*/

class LIST{

public:

template<typename ClassName, typename Type>

inline static void AddRightElement(ClassName\*\* Element, Type value) {

ClassName \*LastElement = \*Element;

\*Element = new ClassName;

(\*Element)->inf = value;

(\*Element)->pLastElement = LastElement;}

template<typename ClassName, typename Type>

static void AddLeftElement(ClassName\*\* Element, Type value) {

ClassName \*LastElement = \*Element;

if (\*Element != NULL){

for (;;){

if ((\*Element)->pLastElement == NULL) break;

\*Element = (\*Element)->pLastElement;}

(\*Element)->pLastElement = new ClassName;

(\*Element)->pLastElement->inf = value;

(\*Element)->pLastElement->pLastElement = NULL;

\*Element = LastElement;}

else{

\*Element = new ClassName;

(\*Element)->inf = value;

(\*Element)->pLastElement = NULL;}}

template<typename ClassName>

inline static bool DelRightElement(ClassName\*\* Element){

if (\*Element != NULL){

ClassName \*LastElement = \*Element;

if ((\*Element)->pLastElement != NULL)

\*Element = (\*Element)->pLastElement; // Отсылка к левому элементу списка

else \*Element = NULL; // Если нет отсылки, обнуляем указатель

delete LastElement;

return true;}

return false;}

template<typename ClassName>

static bool DelLeftElement(ClassName\*\* Element){

if (\*Element != NULL){

ClassName \*LastElement = \*Element;

for (;;){

if ((\*Element)->pLastElement == NULL){

delete \*Element;

\*Element = NULL;

goto m;}

if ((\*Element)->pLastElement->pLastElement == NULL) break;

\*Element = (\*Element)->pLastElement;}

delete (\*Element)->pLastElement;

(\*Element)->pLastElement = NULL;

\*Element = LastElement;

m: return true;}

return false;}};

/\* Структура элемента списка \*/

template<typename T>

struct ELEMENT{

T inf;

ELEMENT\* pLastElement;};

/\* Класс для работы с очередью, деком и стеком \*/

template<typename T>

class USELIST : public LIST{

public:

USELIST() : Knot(NULL) {}

~USELIST() { while (DelRightElement(&Knot));}

void menu(){

char choice;

system("cls");

cout<< " {{>================================================<}}\n"

<< " || Список ||\n"

<< " {{>================================================<}}\n"

<< " || 1. Добавить элемент справа ||\n"

<< " || 2. Добавить элемент слева ||\n"

<< " || 3. Удалить элемент справа ||\n"

<< " || 4. Удалить элемент слева ||\n"

<< " || 5. Посмотреть связанный список ||\n"

<< " || 6. Выход ||\n"

<< " {{>================================================<}}\n"

<< "\n Выбранная опция: ";

(cin >> choice).get();

switch (choice){

case 49:{

system("cls");

cout << "Введите новый элемент: ";

(cin >> Value).get();

AddRightElement(&Knot, Value);

cout << "Добавлено. ";

system("pause");

menu();

break;}

case 50:{

system("cls");

cout << "Введите новый элемент: ";

(cin >> Value).get();

AddLeftElement(&Knot, Value);

cout << "Добавлено. ";

system("pause");

menu();

break;}

case 51:{

system("cls");

if (DelRightElement(&Knot))

cout << "Удалено. ";

else cout << "Нет списка. ";

system("pause");

menu();

break;}

case 52:{

system("cls");

if (DelLeftElement(&Knot))

cout << "Удалено. ";

else cout << "Нет списка. ";

system("pause");

menu();

break;}

case 53:{

system("cls");

if (!PrintTheStucture(Knot))

cerr << "No structure";

cout << endl;

system("pause");

menu();

break;}

case 54:{break;}

default: menu();}}

private:

T Value;

ELEMENT<T>\* Knot;

template<typename ClassName>

static bool PrintTheStucture(ClassName\* Element){

if (Element != NULL){

cout << "Elements:\n";

for (;;){

cout << setw(4) << Element->inf;

if (Element->pLastElement == NULL) break;

Element = Element->pLastElement;}

return true;}

return false;}};

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

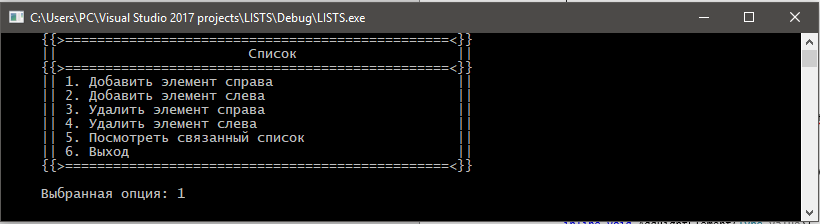
USELIST<int> list;

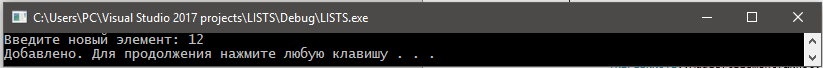
list.menu();

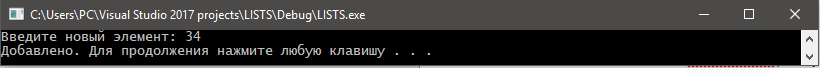
return 0;}

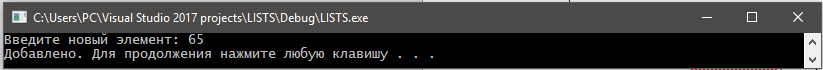
**Отладка программы**

Добавление 3 элементов справа в список через меню выбора:

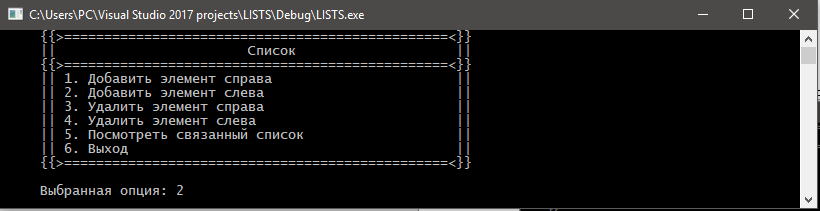


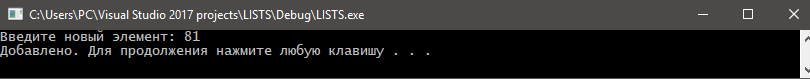


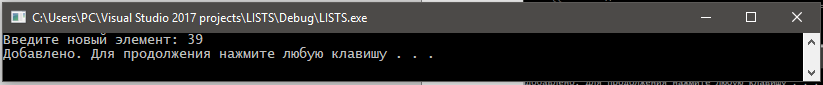




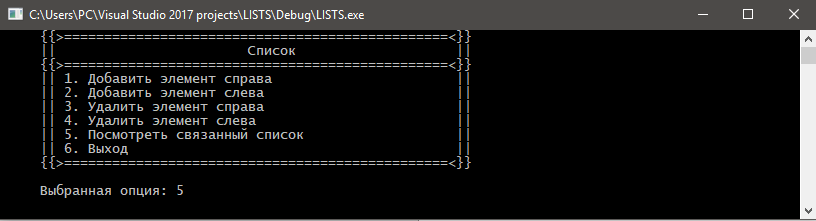
Добавление 2 элементов слева в список через меню выбора:

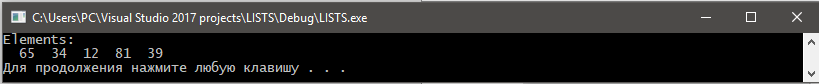




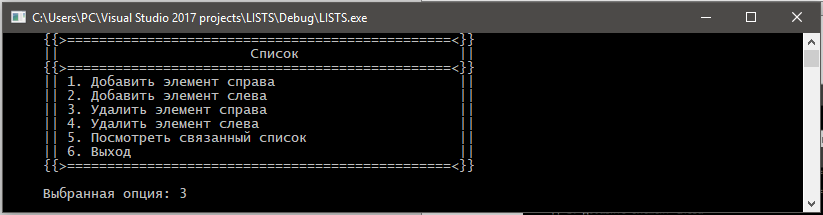


Просмотр элементов в списке (в односвязном списке в обратном порядке):





Удаление справа всех элементов списка по очереди:












Просмотр списка:

