Московский авиационный институт

(Национальный исследовательский университет)

Факультет №3

Кафедра 304

Отчёт по лабораторной работе

По учебной дисциплине

Структуры и алгоритмы обработки данных

На тему:  
“Работа со списками”

Выполнил:

Плотников Кирилл

Крещук Николай

Группа: М30-224БК-17

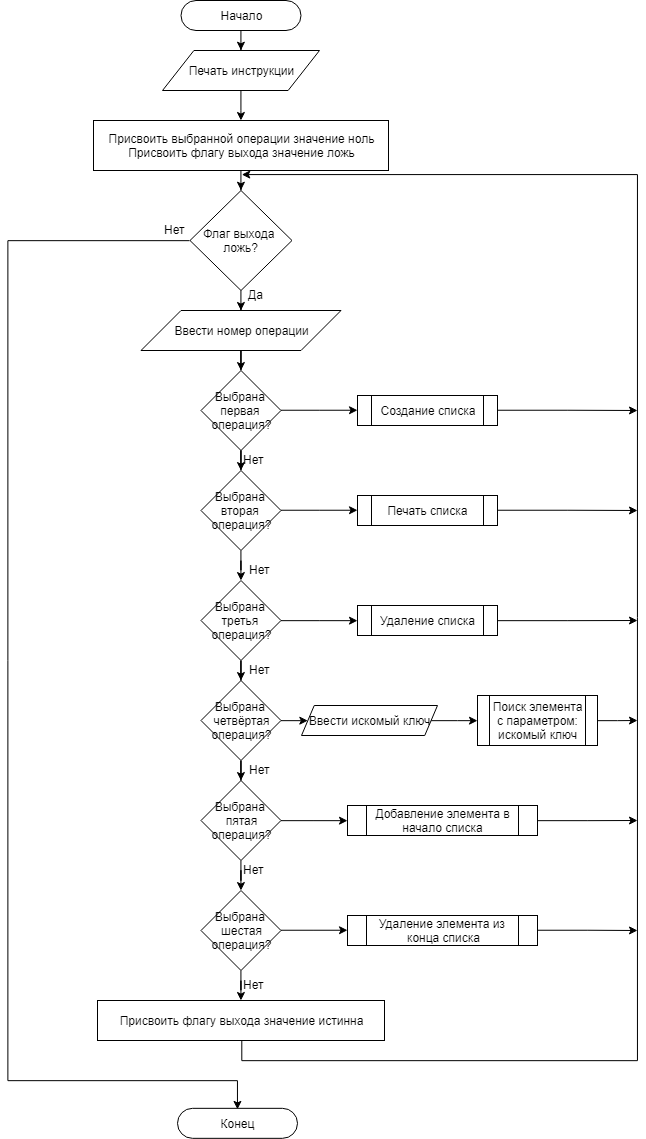
Бригада №7

Принял:

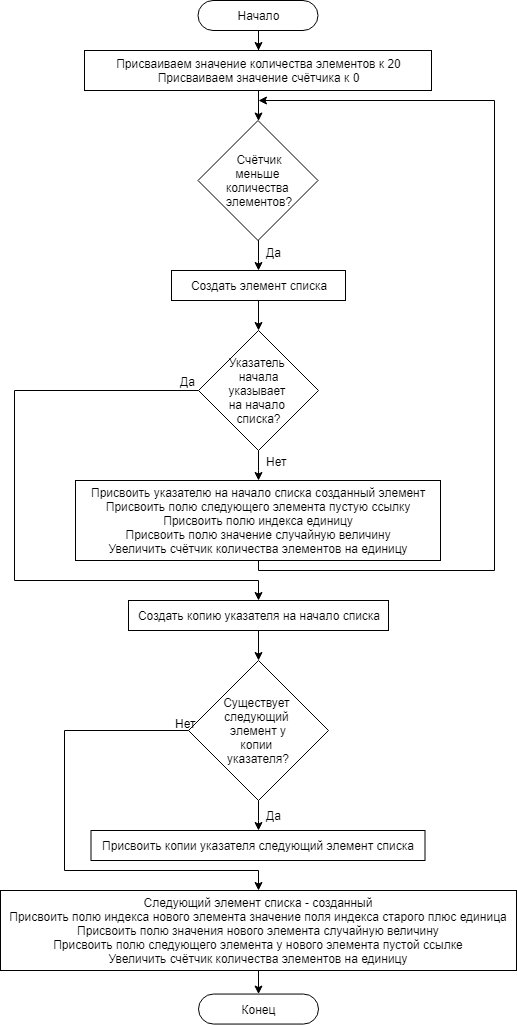
Силаев А.В.

Москва, 2018

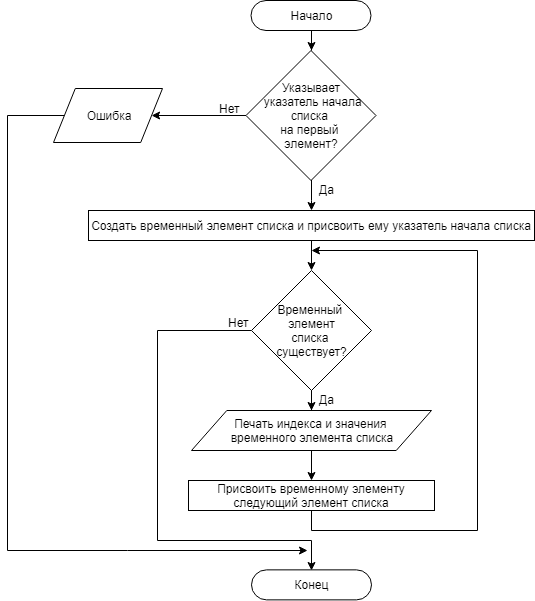
**Структурная схема**

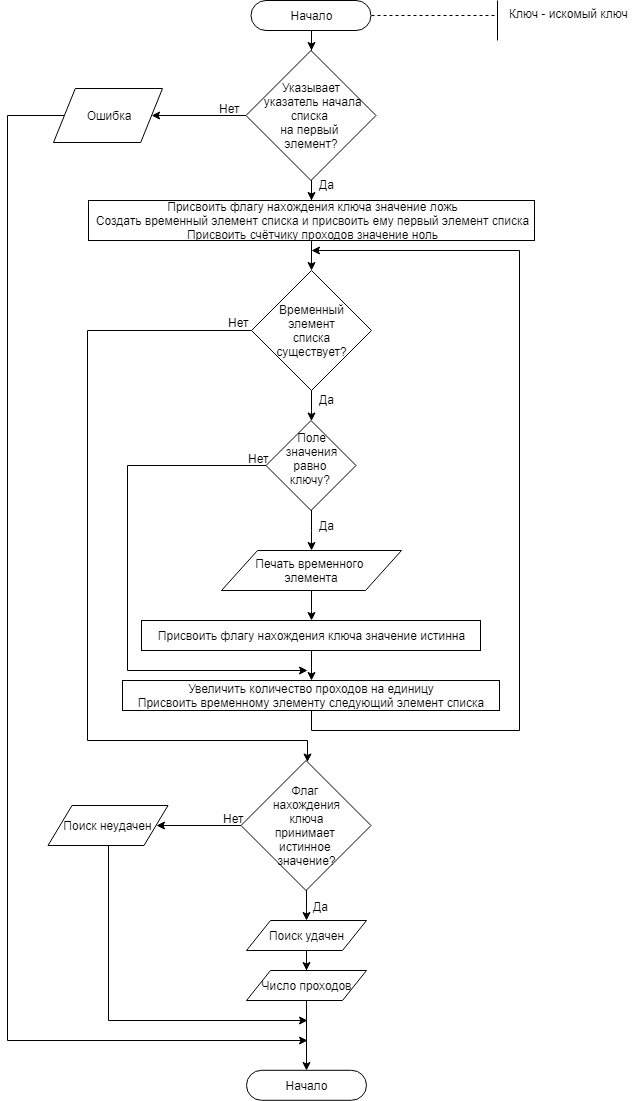
  
Функция main

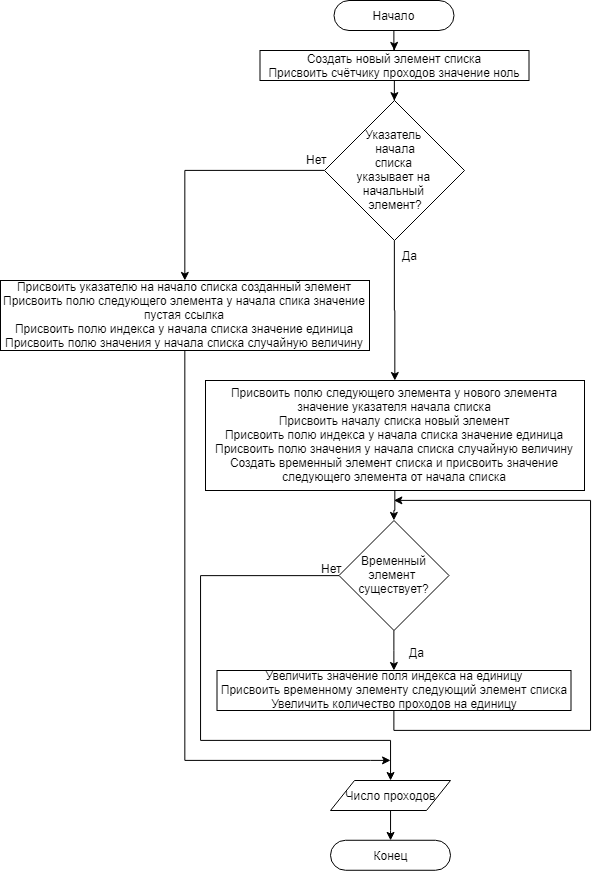
Функция создания списка

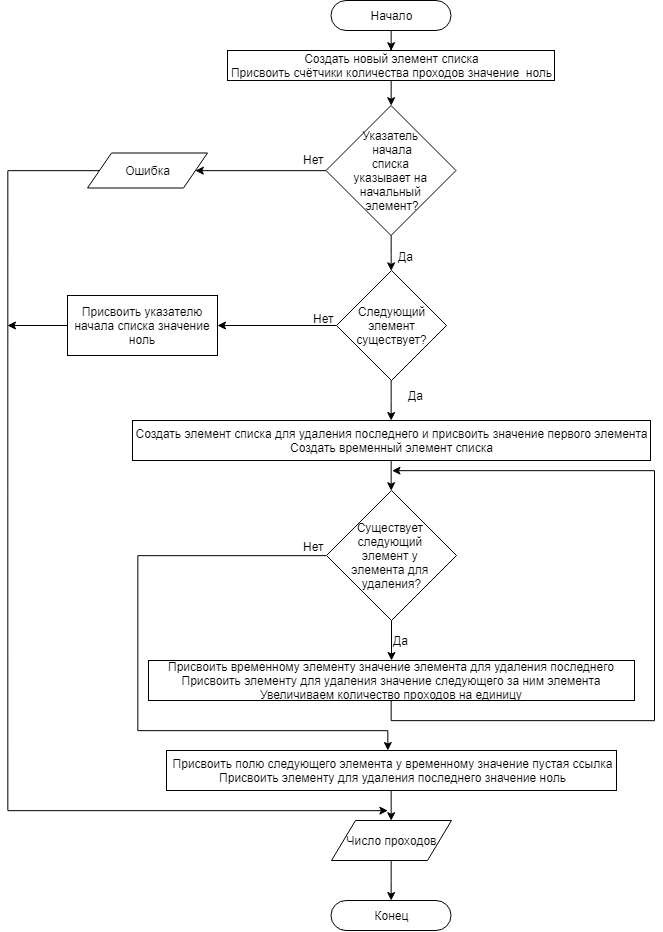


Функция печати списка



Функция поиска элемента

Функция добавления элемента в начало списка 



Функция удаления последнего

элемента

**Код программы:**

﻿/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Project Name :Lab\_2 \*

\* Project Type :Win32 Console Application \*

\* File Name :Lab\_2.cpp \*

\* Language :C++, MAVS 2010 and above \*

\* Programmer(2) :Крещук Николай, Плотников Кирилл, вариант №7 \*

\* Created :05/10/18 \*

\* Last revision :05/10/18 \*

\* Comment :Работа со списком \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "pch.h"

#include <stdlib.h>

#include <iostream>

#include <ctime>

using namespace std;

// Максимальное значение одного элемента.

const int MAX\_VALUE\_ELEMENT = 50;

struct list

{

int value;

int index;

list \*next;

};

// Указатель на первый элемент списка.

list \*head;

// Создание списка с необходимым количеством элементов (20).

void CreateList()

{

int nElements = 20;

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

{

list \*newElement = new list;

if (!head)

{

head = newElement;

head->next = NULL;

head->index = 1;

head->value = rand() % MAX\_VALUE\_ELEMENT;

continue;

}

list \*temp = head;

// Проход до последнего элемента.

while (temp->next != NULL)

temp = temp->next;

// Следующий элемент после последнего - новый.

temp->next = newElement;

// Инициализация нового элемента.

newElement->index = temp->index + 1;

newElement->value = rand() % MAX\_VALUE\_ELEMENT;

newElement->next = NULL;

}

}

// Вывод всего списка.

void PrintList()

{

if (!head)

{

cout << "List is empty. Nothing to print." << endl;

return;

}

list \*temp;

for (temp = head; temp != NULL; temp = temp->next)

cout << "Value: " << temp->value << " Index: " << temp->index << endl;

}

// Очистка всего списка.

void ClearList()

{

if (!head)

{

cout << "List is already empty." << endl;

return;

}

list \*deletePrevious = new list;

list \*temp = head;

while (temp->next != NULL)

{

deletePrevious = temp;

temp = temp->next;

delete deletePrevious;

}

head = 0;

}

// Поиск элемента в списке.

void SearchElement(int searchKey)

{

int countOfPasses = 0;

if (!head)

{

cout << "List is empty. Nothing to look for." << endl;

return;

}

bool found = false;

cout << "Elements with key: " << searchKey << endl;

for (list \*temp = head; temp != NULL; temp = temp->next)

{

if (temp->value == searchKey)

{

cout << "Index: " << temp->index << endl;

found = true;

}

countOfPasses++;

}

if (found)

{

cout << "Search successful." << endl;

cout << "Count of passes: " << countOfPasses << endl;

}

else

{

cout << "Error, no one element with this key." << endl;

cout << "Search unsuccessful." << endl;

}

}

// Добавить в список элемент на первую позицию.

void AddFirstElement()

{

int countOfPasses = 0;

list \*newFirst = new list;

if (!head)

{

head = newFirst;

head->next = NULL;

head->index = 1;

head->value = rand() % MAX\_VALUE\_ELEMENT;

cout << "Count of passes: " << countOfPasses << endl;

return;

}

newFirst->next = head;

head = newFirst;

head->index = 1;

head->value = rand() % MAX\_VALUE\_ELEMENT;

list \*temp;

for (temp = head->next; temp != NULL; temp = temp->next)

{

temp->index++;

countOfPasses++;

}

cout << "Count of passes: " << countOfPasses << endl;

}

// Удаление последнего элемент в списке.

void DeleteLastElement()

{

int countOfPasses = 0;

if (!head)

{

cout << "List is already empty." << endl;

return;

}

// Если у нас 1 элемент в списке.

if (head->next == NULL)

{

head = 0;

cout << "Count of passes: " << countOfPasses << endl;

return;

}

list \*deleteLast = head;

list \*temp = new list;

// Проход до предпоследнего элемента.

while (deleteLast->next != NULL)

{

temp = deleteLast;

deleteLast = deleteLast->next;

countOfPasses++;

}

// Удаляем ссылку на последний элемент.

temp->next = NULL;

// Переходим к последнему элементу и удаляем его.

deleteLast = 0;

cout << "Count of passes: " << countOfPasses << endl;

}

int main()

{

srand(time(NULL));

cout << "Press 1: Create a list with 20 elements." << endl;

cout << "Press 2: Print a full list." << endl;

cout << "Press 3: Clear a full list." << endl;

cout << "Press 4: Search element with the selected key." << endl;

cout << "Press 5: Add element on first position." << endl;

cout << "Press 6: Delete last element." << endl;

cout << "Press anything else to quit." << endl;

int choosedOperation = 0;

bool quit = false;

while (!quit)

{

cin >> choosedOperation;

switch (choosedOperation)

{

case 1:

CreateList();

break;

case 2:

PrintList();

break;

case 3:

ClearList();

break;

case 4:

int key;

cout << "Write a key(integer): ";

cin >> key;

SearchElement(key);

break;

case 5:

AddFirstElement();

break;

case 6:

DeleteLastElement();

break;

default:

quit = true;

break;

}

}

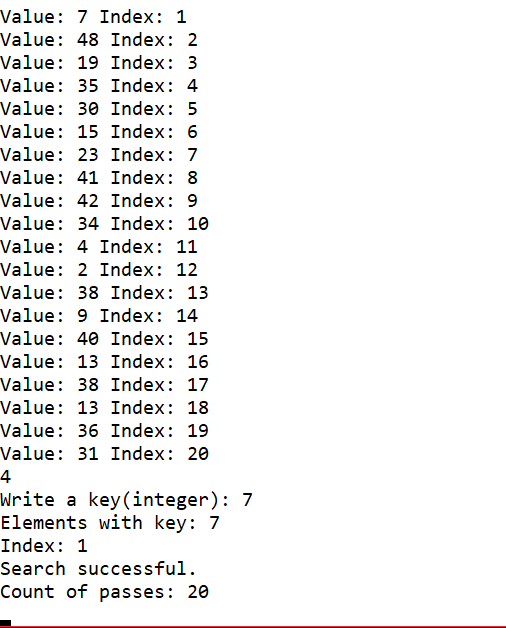
system("pause");

return 0;

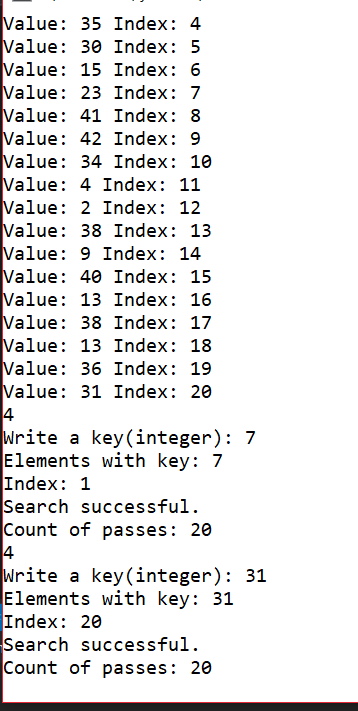
}

**Тесты**

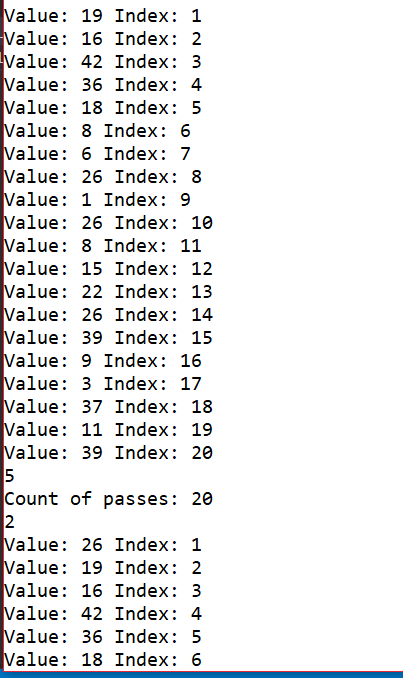
1) Поиск элемента, находящегося в начале списка:



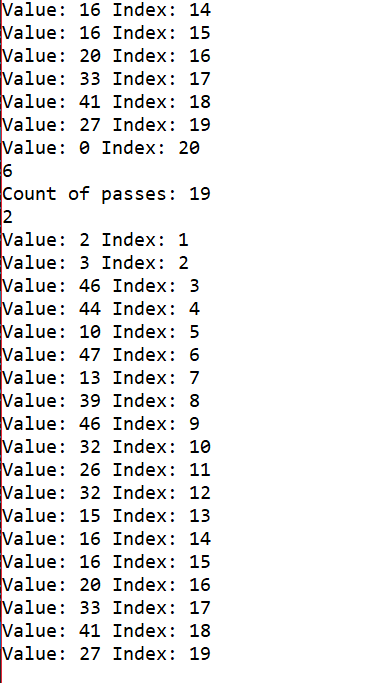
2) Поиск элемента, находящегося в конце списка:



3) Добавление элемента в начало списка:



4) Удаление последнего элемента в списке:



**Результаты сопоставительного анализа**

1) Количество проходов при поиске не зависит от местонахождение искомого элемента, так как проход осуществляется всегда по всем элеменам. Поэтому количество проходов равно N, где N – количество элементов.

2) При добавление нового элемента в начало списка, необходимо пройтись по всем элементам с целью увеличения индекса на единицу, что приводит к количествам проходов равным N, где N – количество элементов.

3) При удалении последнего элемента проход идёт до предпоследнего элемента, начиная с начала. Поэтому количество проходов равно N-1, где N – количество проходов.