МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Программирование

Отчет по лабораторной работе № 4

«Линейные списки»

Выполнили студенты группы М3О-210Б-20  
Гудынин Д. Д.

Артёмов И. Е.

Проверили Дмитриева Е.А.  
Офицеров В.П.

**Оглавление**

[**Цель работы** 3](#_Toc88255073)

[**Задание (9)** 3](#_Toc88255074)

[**Структурная схема алгоритма.** 4](#_Toc88255075)

[**Основная функция (main)** 4](#_Toc88255076)

[**Функция меню (Menu)** 5](#_Toc88255077)

[**Листинг программы** 7](#_Toc88255078)

[**Файл основной программы** 7](#_Toc88255079)

[**Файл заголовков (Header.h)** 8](#_Toc88255080)

[**Файл реализации функций (Functions.cpp)** 9](#_Toc88255081)

[**Работа программы** 11](#_Toc88255082)

[**Вывод** 13](#_Toc88255083)

# **Цель работы**

Изучить принципы программной реализации динамической структуры данных «линейный список».

# **Задание (9)**

Реализовать заданный согласно варианту задания линейный список, состоящий из 20 элементов целочисленного типа данных.

Интерфейс программного обеспечения должен включать для всех вариантов следующие операции:

- создание списка;

- вывод на экран и/или в файл значений элементов списка с их индексами (номерами);

- удаление списка,

а также некоторые из дополнительных операций (согласно варианту задания):

2. Поиск в списке элемента с максимальным значением с получением его номера в списке (повторное вхождение одного и того же значения запрещено.

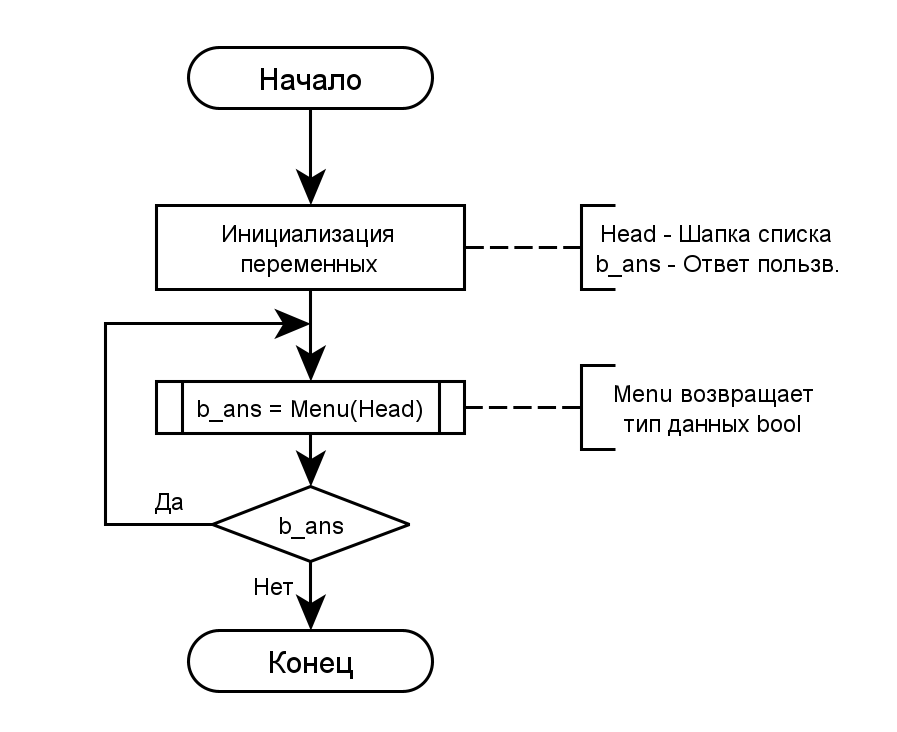
4. Включение нового элемента в конец списка.

8. Удаление элемента из позиции списка с заданным в программе номером.

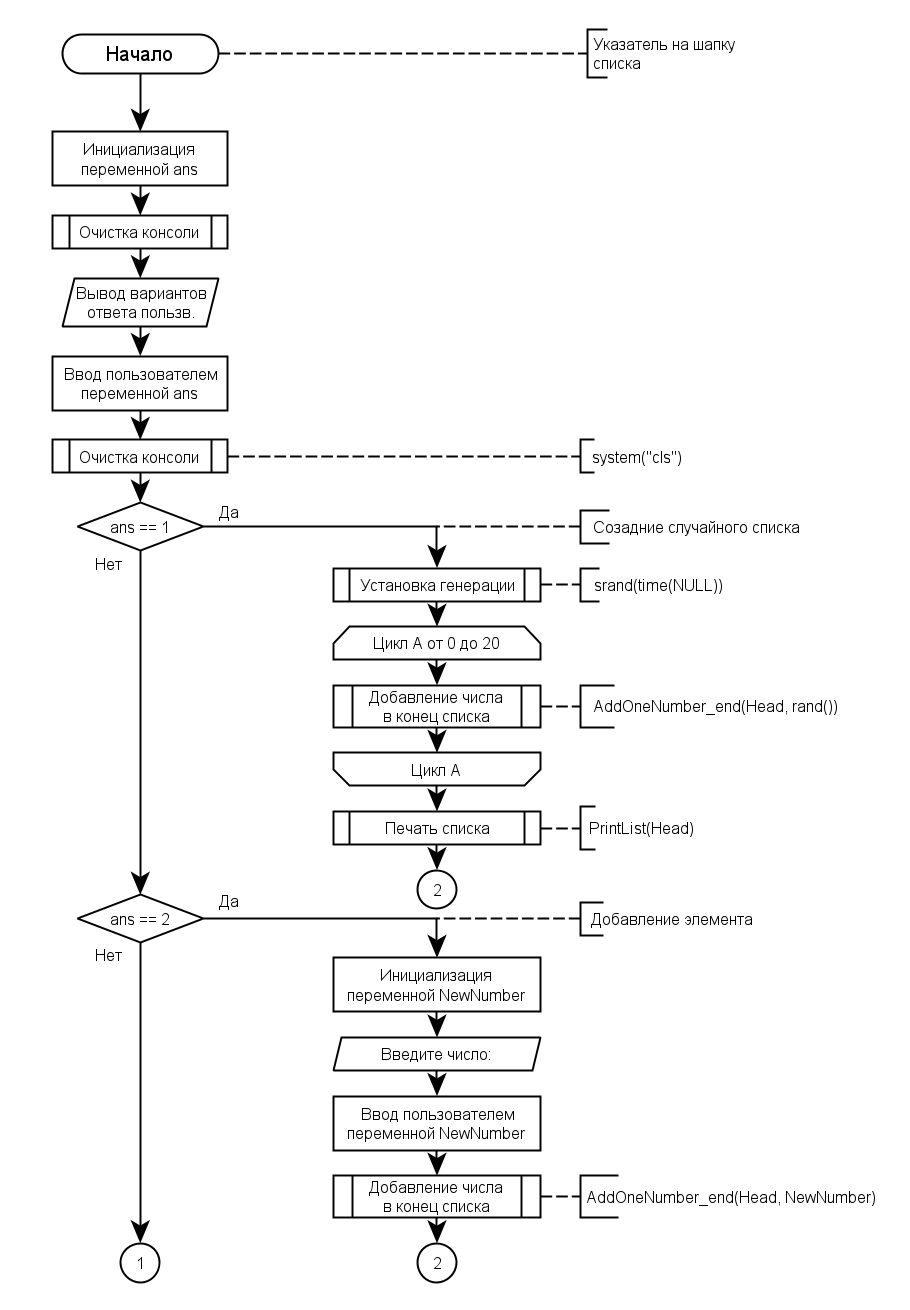
После выполнения операций включения или удаления вывести содержимое списка. Выполнение операций организовать с помощью меню.

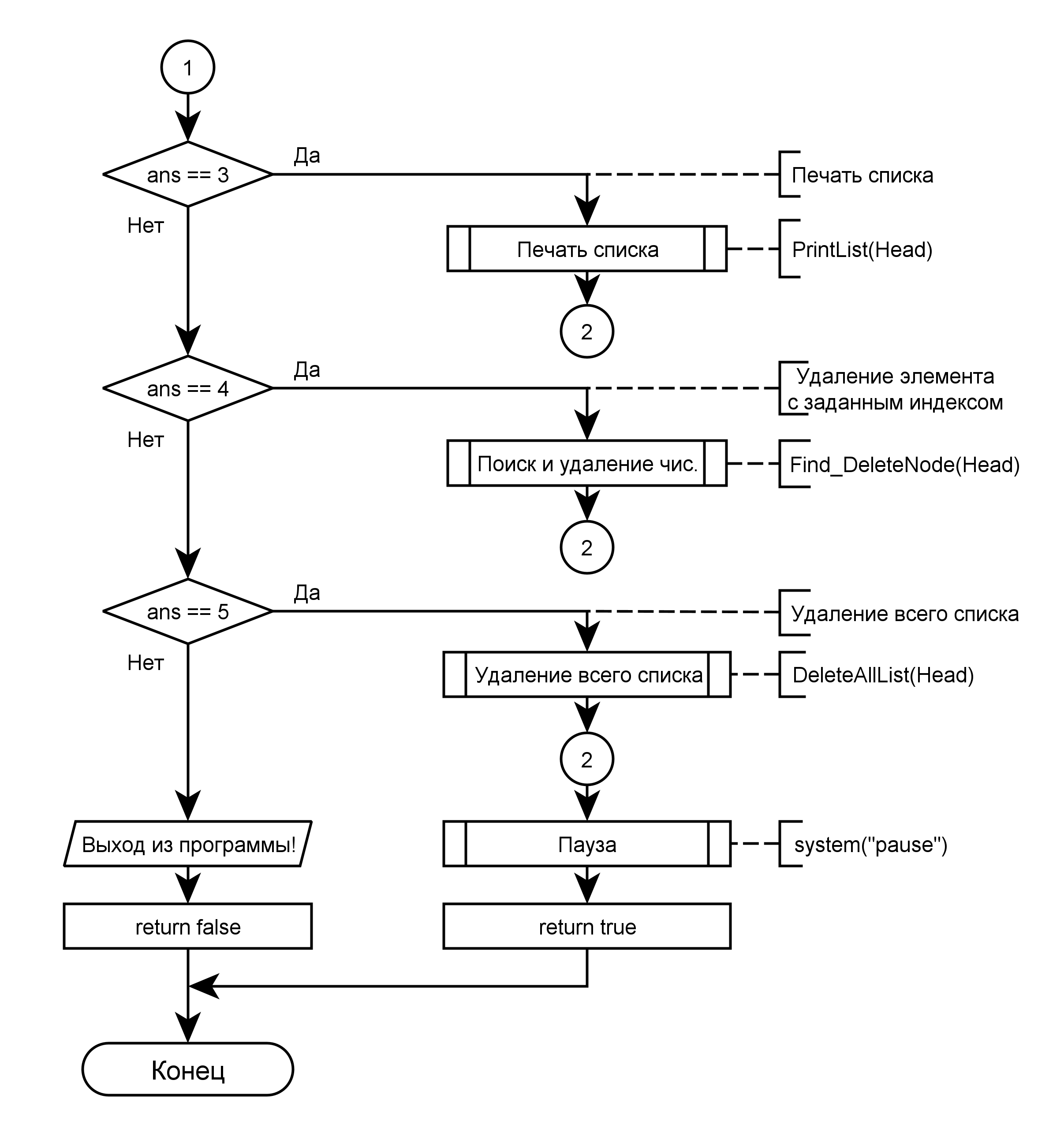
# **Структурная схема алгоритма.**

## **Основная функция (main)**

****

## **Функция меню (Menu)**

****

****

# **Листинг программы**

## **Файл основной программы**

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

\* Курс ПРОГРАММИРОВАНИЕ \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Project type : Win64 Console Application \*

\*Project name : 2k-3c-Линейные списки \*

\*File name : 2k-3c-Линейные списки \*

\*Language : CPP, MSVS 2022 \*

\*Programmers : Артемов Игорь Евгеньевич, Гудынин Данила Денисович М3О-110Б-20 \*

\*Modified By : \*

\*Created : 10.11.2021 \*

\*Last revision: 11.09.2021 \*

\*Comment : Линейные списки \*

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

#include "Header.h"

int main(){

setlocale(LC\_ALL, "ru");

PNode Head = NULL; // Шапка списка

bool b\_ans = true; // Ответ пользователя

// Цикл бесконечного меню, завершается, когда ответ выйдет из заданного

// диапазона возможных ответов

do {

b\_ans = Menu(Head);

} while (b\_ans);

return 0;

}

## **Файл заголовков (Header.h)**

#pragma once

#include <iostream>

using namespace std;

struct Node {

int number; // Число

int count; // Индекс в списке

Node\* next; // Указатель на следющий элемент

};

typedef Node\* PNode;

PNode CreateNode(int NewNumber); // Создание элемента списка

void AddLast(PNode& Head, PNode NewNode); // Добавление в список

void PrintList(PNode Head); // Печать списка

void Find\_Max\_Number(PNode Head); // Поиск максимального числа

void DeleteOneNode(PNode& Head, PNode OldNode); // Удаление одной ноды

// Удаление ноды с определённым индексом

// (при добавлении новых элементов индексирование произойдёт заново)

void Find\_DeleteNode(PNode& Head);

// Добавление числа в конец списка (автоматизация)

void AddOneNumber\_end(PNode& Head, int NewNumber);

void DeleteAllList(PNode& Head); // Удаление всего списка

bool Menu(PNode& Head); // Меню программы

## **Файл реализации функций (Functions.cpp)**

#include "Header.h"

PNode CreateNode(int NewNumber) { // Создание элемента списка

PNode NewNode = new Node; // Выделение паряти под элемент списка

NewNode->number = NewNumber; // Задание числа

NewNode->next = NULL; // Обозначение указателя на следующий элемент

return NewNode;

}

void AddLast(PNode& Head, PNode NewNode) { // Добавление в список

PNode q = Head;

if (Head == NULL) { // Проверка, не пустой ли список

NewNode->next = Head; // Указатель на NULL

Head = NewNode; // Задание новой шапки (последнего элемента)

NewNode->count = 0; // Обозначение индекса

return;

}

else {

int i;

for (i = 0; q->next != NULL; i++) { // До того, как не будет указателя на NULL

q = q->next; // Изменяем текущую ноду

q->count = i + 1; // Зада1м следующий индекс

}

NewNode->count = i + 1;

q->next = NewNode; // Задание новой шапки

}

}

void PrintList(PNode Head) { // Печать списка

PNode p = Head;

cout << "\n\nПечать списка:\n";

if (Head != NULL) { // Если список не пуст

while (p) { // пока не дошли до конца

cout << "p[" << p->count << "] = " << p->number << endl;

p = p->next; // переходим к следующему узлу

}

}

else

cout << "Список пуст!";

cout << "\n\n";

}

void Find\_Max\_Number(PNode Head) { // Поиск максимального числа

PNode q = Head, p = Head; // Реплицируем переменную Head

if (Head != NULL) { // Если сиписок не пуст

while (q != NULL) { // Пока не наткнёмся на NULL

if (q->number > p->number) { // Сравниваем числа в нодах

p = q; // Если число больше p = новой ноде

}

q = q->next; // Задаём новую ноду q

}

cout << "Max number - p[" << p->count << "] = " << p->number << endl;

}

PrintList(Head);

}

void DeleteOneNode(PNode& Head, PNode OldNode) { // Удаление одной ноды

PNode q = Head;

if (Head == OldNode)

Head = OldNode->next; // Удаляем первый элемент

else {

while (q && q->next != OldNode) // Ищем элемент

q = q->next;

if (q == NULL) return; // Если не нашли, выход

q->next = OldNode->next;

}

delete OldNode; // Освобождаем память

}

// Удаление ноды с определённым индексом

// (при добавлении новых элементов индексирование произойдёт заново)

void Find\_DeleteNode(PNode& Head) {

PNode p = Head;

if (p != NULL) { // Список не пуст

int index;

cout << "Введите индекс элемента, который хотите удалить: ";

cin >> index;

while (p->next != NULL) { // пока не дошли до конца

if (p->count == index) {

DeleteOneNode(Head, p); // Циклическое удаление нод

return;

}

p = p->next;

}

cout << "Данного индекса не существует!";

}

PrintList(Head); // Печать изменённого списка

}

// Добавление числа в конец списка (автоматизация)

void AddOneNumber\_end(PNode& Head, int NewNumber) {

PNode p = CreateNode(NewNumber); // Создаем новый узел

AddLast(Head, p); // Добавляем узел

}

void DeleteAllList(PNode& Head) { // Удаление всего списка

cout << "Удаление всего списка!";

PNode q = Head;

if (Head != NULL) { // Если список не пуст

while (q->next != NULL) { // поочерёдно удаляем элементы

DeleteOneNode(Head, q->next);

}

Head = NULL; // удаляем шапку

}

PrintList(Head);

}

bool Menu(PNode& Head) { // Меню программы

int ans = 0;

system("cls"); // Очистка консоли

cout << "\n\n1) Создание односвязного списка со случайными значениями (последующие вызовы расширяют список);\n"

<< "2) Добавление элемента в конец списка;\n"

<< "3) Печать списка;\n"

<< "4) Нахождение максимального элемента в списке;\n"

<< "5) Удаление элемента с заданным индексом;\n"

<< "6) Удаление всего списка;\n\n"

<< "Ваш выбор: ";

cin >> ans; // Ввод ответа

system("cls");

switch (ans){

case 1:

srand(time(NULL)); // Установка генерации

for (int i = 0; i < 20; i++) {

AddOneNumber\_end(Head, rand());

}

PrintList(Head);

break;

case 2:

int NewNumber;

cout << "\n\nВведите число: ";

cin >> NewNumber;

AddOneNumber\_end(Head, NewNumber);

break;

case 3:

PrintList(Head);

break;

case 4:

Find\_Max\_Number(Head);

break;

case 5:

Find\_DeleteNode(Head);

break;

case 6:

DeleteAllList(Head);

break;

default:

cout << "\n\nВыход из программы!\n\n";

return false;

break;

}

system("pause"); // Пауза

return true;

}

# **Работа программы**

Для описания и тестирования работы программы требуется воспроизвести некоторую ситуацию, описывающую её использование.

Пусть пользователь выполнил некоторый алгоритм, а именно:

1. Запустит программу;
2. Выберет в меню пункт 1 (создание нового списка);
3. Выберет в меню пункт 5 (удаление элемента с заданный индексом);
4. Выберет в меню пункт 2 (добавление элемента в конец списка);
5. Выберет в меню пункт 4 (нахождение максимального элемента в списке);
6. Выберет в меню пункт 6 (удаление всего списка);
7. Введёт число, выходящее за границы предложенного выбора.

Реализуем данную последовательность действий:

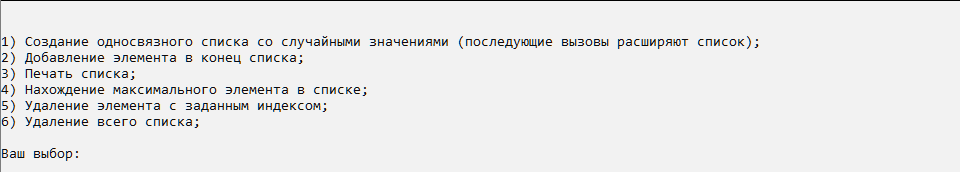


Рисунок 1

После запуска пользователя встречает основное меню программы (рис. 1). Где он может выбрать один из предложенных пунктов. В соответствии с выбором пользователя программа сначала зачистит экран, а после распечатает диалоговую информацию выбранного пункта основного меню. В случае некорректного ввода в основном меню (должны вводится только цифры в заданном диапазоне) последует завершение работы программы.

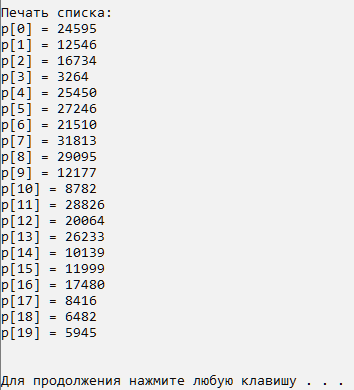


Рисунок 2

После ввода в основном меню программы цифры 1 последует выполнение алгоритма создания нового списка размерностью 20 элементов. Далее на экран будет распечатан сам список, сформированный в результате работы алгоритма. Как показано на рисунке, пользователю предлагается нажать любую клавишу для продолжения работы программы. Подобное предложение будет появляться каждый раз после отработки того или иного пункта программы (кроме ввода неверного ответа в основном меню). После того, как пользователь нажмёт любую клавишу на клавиатуре, программа зачистит экран и заново распечатает основное меню с предложение пользователю выбрать нужный пункт.

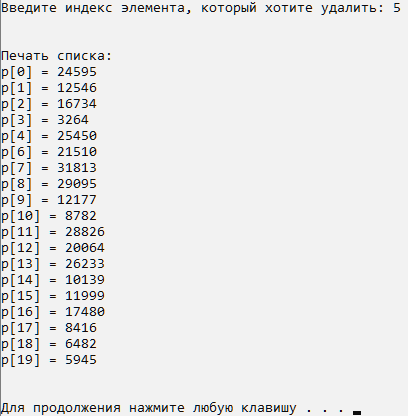
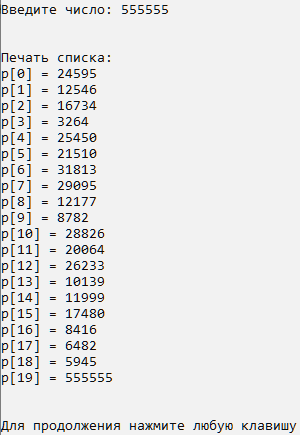
После того, как пользователь выбрал в основном меню пункт 5, будет выведен диалог, предлагающий ввод индекса элемента, который пользователь желает удалить. После того, как пользователь введёт индекс элемента, алгоритм проверит, действительно ли данный элемент есть в списке и удалит его. Как мы видим на рисунке 3 – выбранный элемент с индексом 5 был успешно удалён, что нам демонстрирует вызванный алгоритм печати списка.

Рисунок 3

В пункте 2 заложен алгоритм, который осуществляет добавление нового числа в список. Как показано на рисунке 4, алгоритм, обеспечивающий добавление нового элемента в список, заново индексирует весь список, который ему передали в качестве параметра. Мы видим, что в списке появилось введённое число (данное число является самым большим в списке, что наглядно демонстрируется на рисунке 5, где алгоритм поиска максимального числа вывел элемент с индексом 19.

После того, как пользователь в главном меню выберет пункт 6, то на экран будет выведено сообщение о том, что список успешно был удалён, по заданному алгоритму, за сообщением об успешном удалении списка сразу же будет вызван алгоритм по его печати, что подтвердит корректное выполнения алгоритма 6 (рис. 6).

Рисунок 4

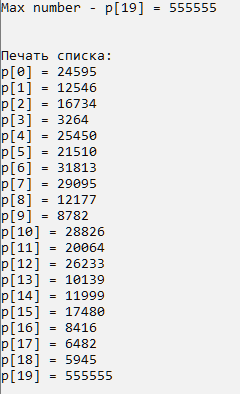
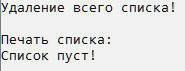
После неверного ввода пользователем ответа в главном меню программы последует незамедлительное завершение программы, что показано на рисунке 7.

Рисунок 5



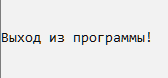


Рисунок 6

Рисунок 7

# **Вывод**

В данной лабораторной работе была изучена реализация линейных списков. Преимущество списков в отличие от массивов в том, что вставка и удаление производится за константное время. Поиск в линейных списках выполняется хуже, так как необходимо рассмотреть весь список из-за того, что элементы разбросаны в оперативной памяти в хаотичном порядке.

**Работа завершена так как:**

1) Полученные результаты совпадают с ожидаемыми;

2) Набор тестов считается полным.