Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №6**

**«Реализация элементарных структур данных на основе статической памяти»**

**по МДК 05.02. Разработка кода информационных систем**

Выполнил: студент учебной группы

ИСПк-204-52-00

Гордеева Валерия Дмитриевна

Преподаватель:

Сергеева Елизавета Григорьевна

1. Цель работы: изучение принципов работы с базовыми структурами данных, получение навыков организации case-меню.

Формулировка задания (с вариантом)

Вариант 4

Задание:

1. Написать программу для работы со структурой данных "Кольцевой односвязный список".
2. Структура данных должна быть реализована на основе статической памяти.
3. Работа со структурой должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры.

Описание алгоритма и ответы на вопросы

Данная программа на языке Pascal реализует концепцию кольцевого односвязного списка фиксированного размера. Вместо использования динамической памяти и указателей, элементы списка хранятся в статическом массиве, а связь между ними осуществляется посредством индексов элементов. Программа предоставляет пользователю возможность создавать, просматривать и удалять элементы списка. Head - индекс головы списка, ListSize - текущее количество элементов, а FreeList - индекс первого свободного элемента. InitializeList инициализирует список, устанавливая Head и ListSize в 0 и создавая список свободных элементов. GetFreeElement возвращает индекс первого свободного элемента и обновляет FreeList. ReleaseElement освобождает элемент, добавляя его в начало списка свободных и уменьшая ListSize. NewElement добавляет новый элемент, получая индекс свободного элемента, запрашивая значение у пользователя и добавляя элемент в конец кольцевого списка. look выводит элементы списка на экран, позволяя пользователю перемещаться по списку. DeleteElement удаляет элемент по заданному значению, находит его в списке и, в зависимости от того, является ли он головой, перенаправляет указатели и освобождает элемент. show отображает меню и вызывает соответствующие процедуры. Основная программа инициализирует список и в цикле вызывает show до выбора выхода.

1. Схема алгоритма с комментариями

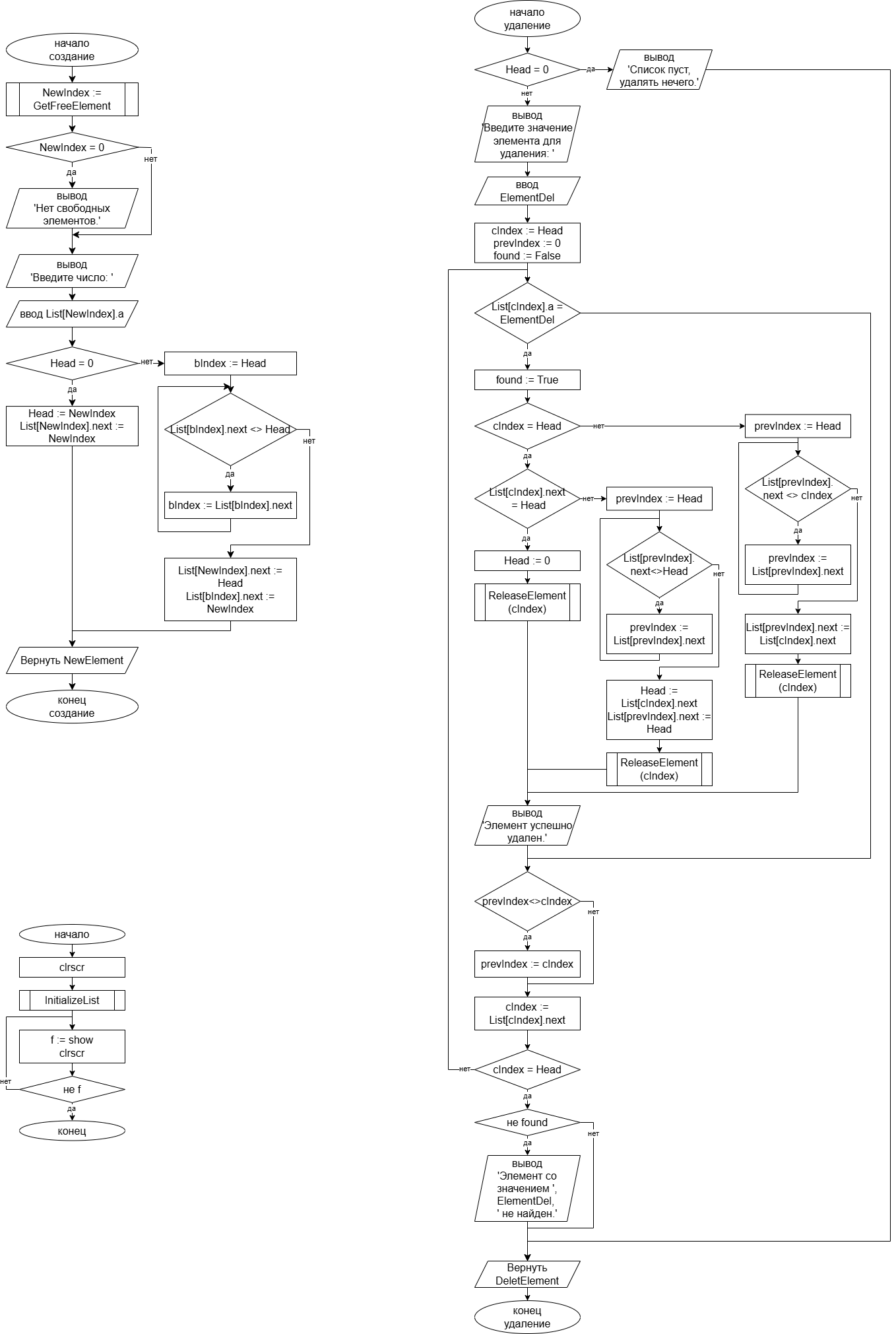
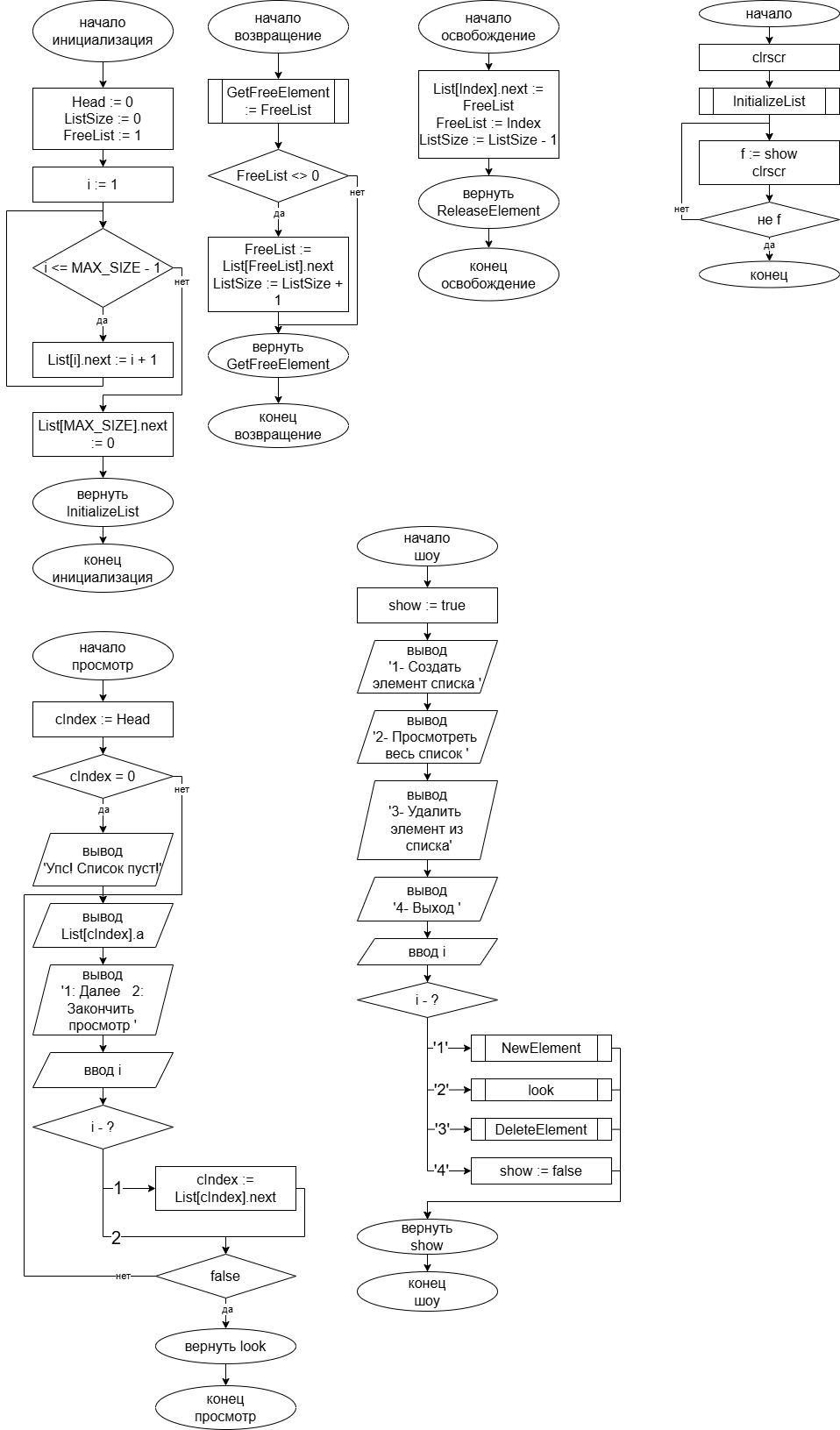


Рисунок 1 – Схема алгоритма процедуры создания узла списка

Рисунок 2 – Схема алгоритма процедуры просмотра всего списка

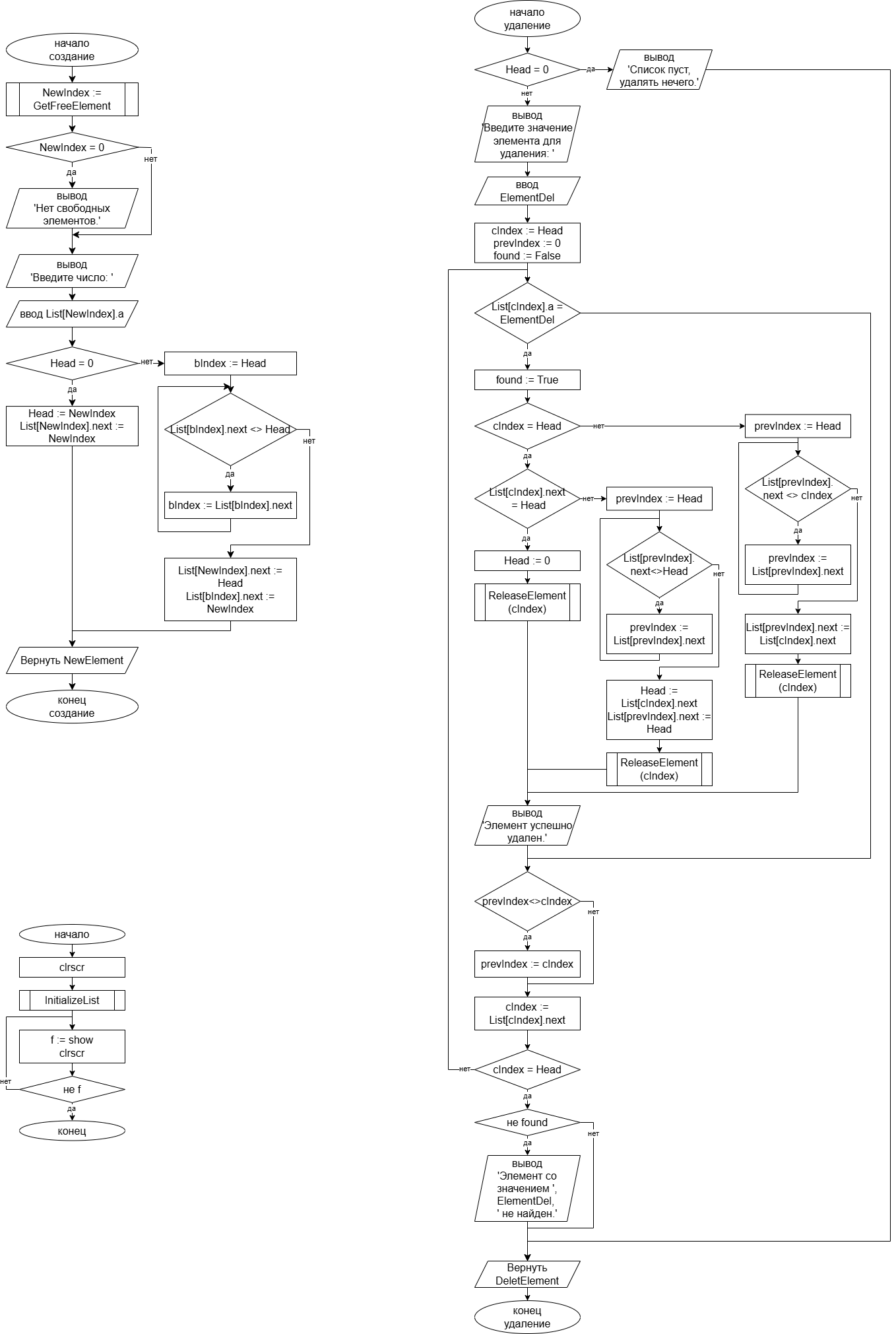


Рисунок 3 – Схема алгоритма процедуры удаления элемента списка

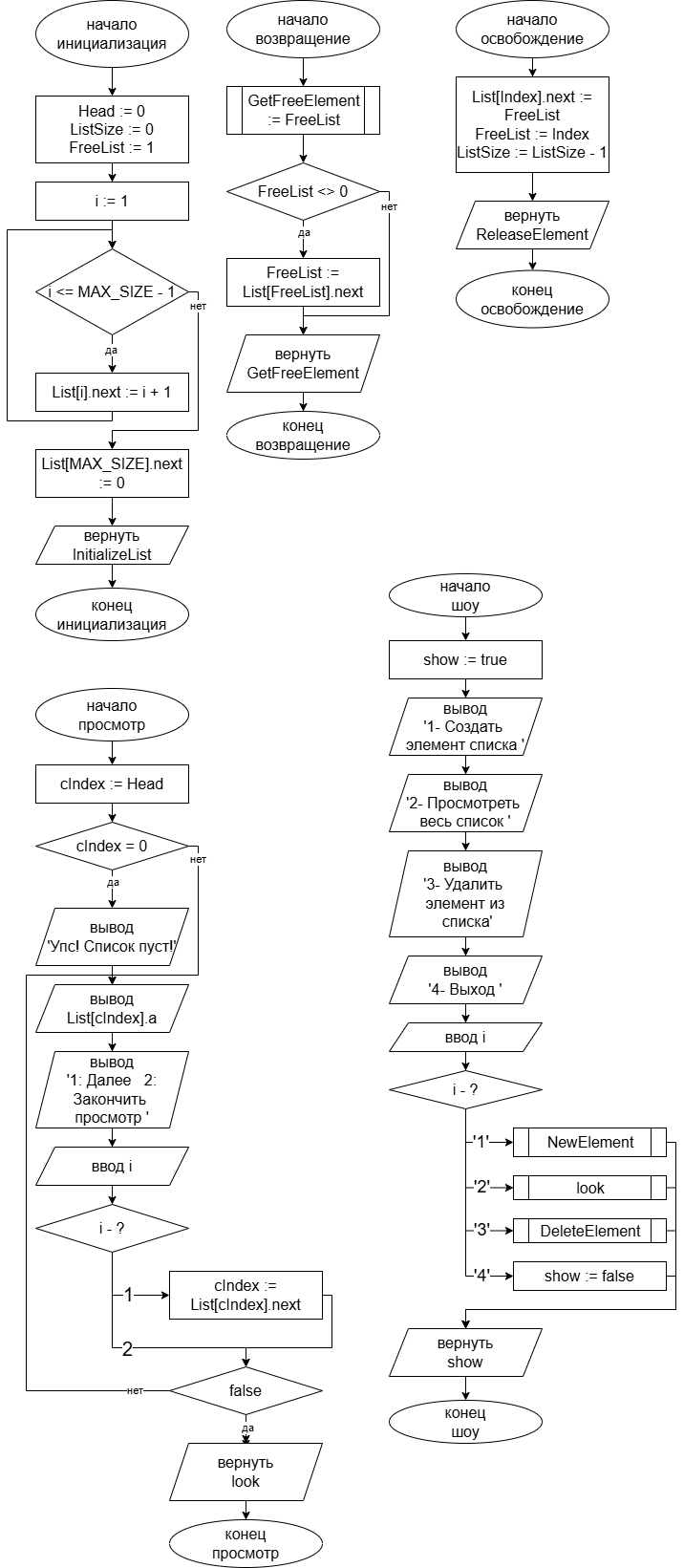
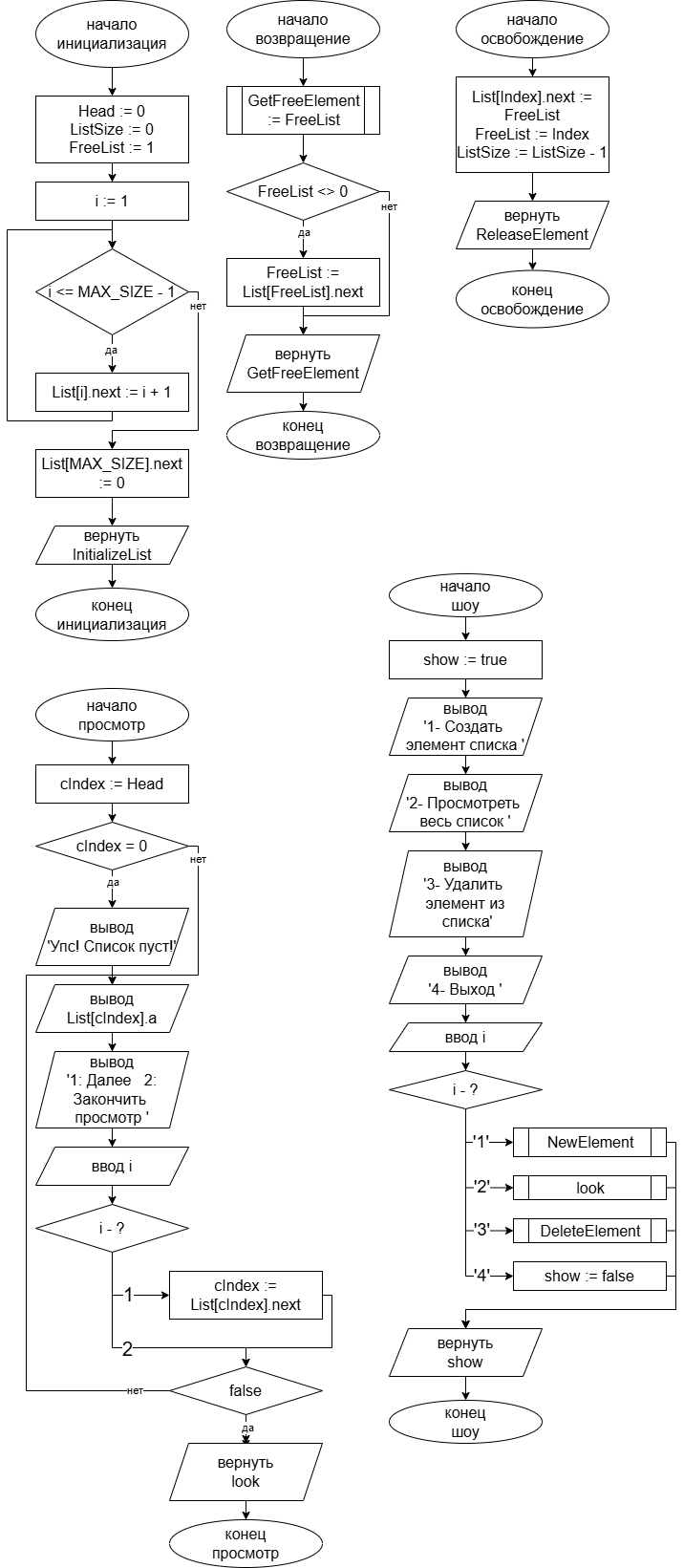


Рисунок 4 – Схема алгоритма процедуры инициализации списка

  
Рисунок 5 – Схема алгоритма процедуры возвращения

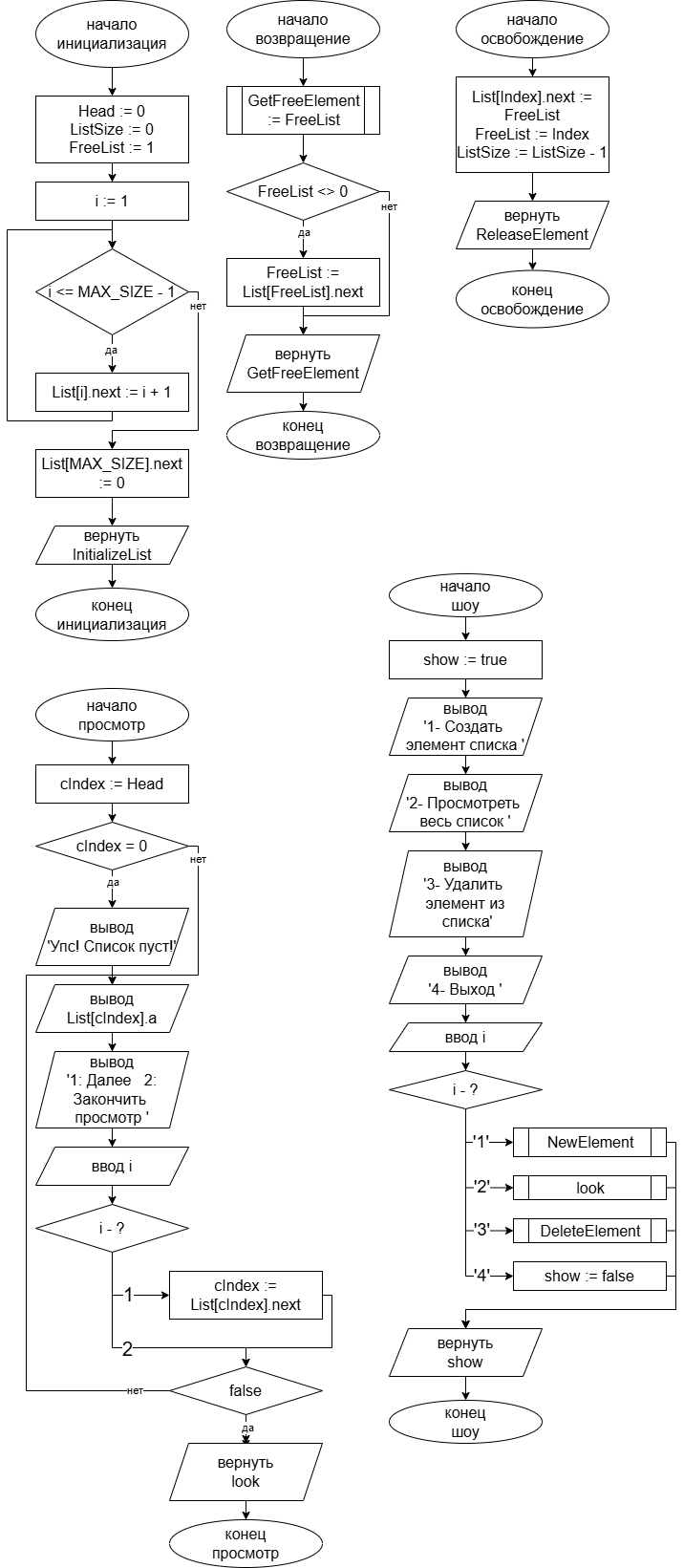


Рисунок 6 – Схема алгоритма процедуры освобождения элемента списка

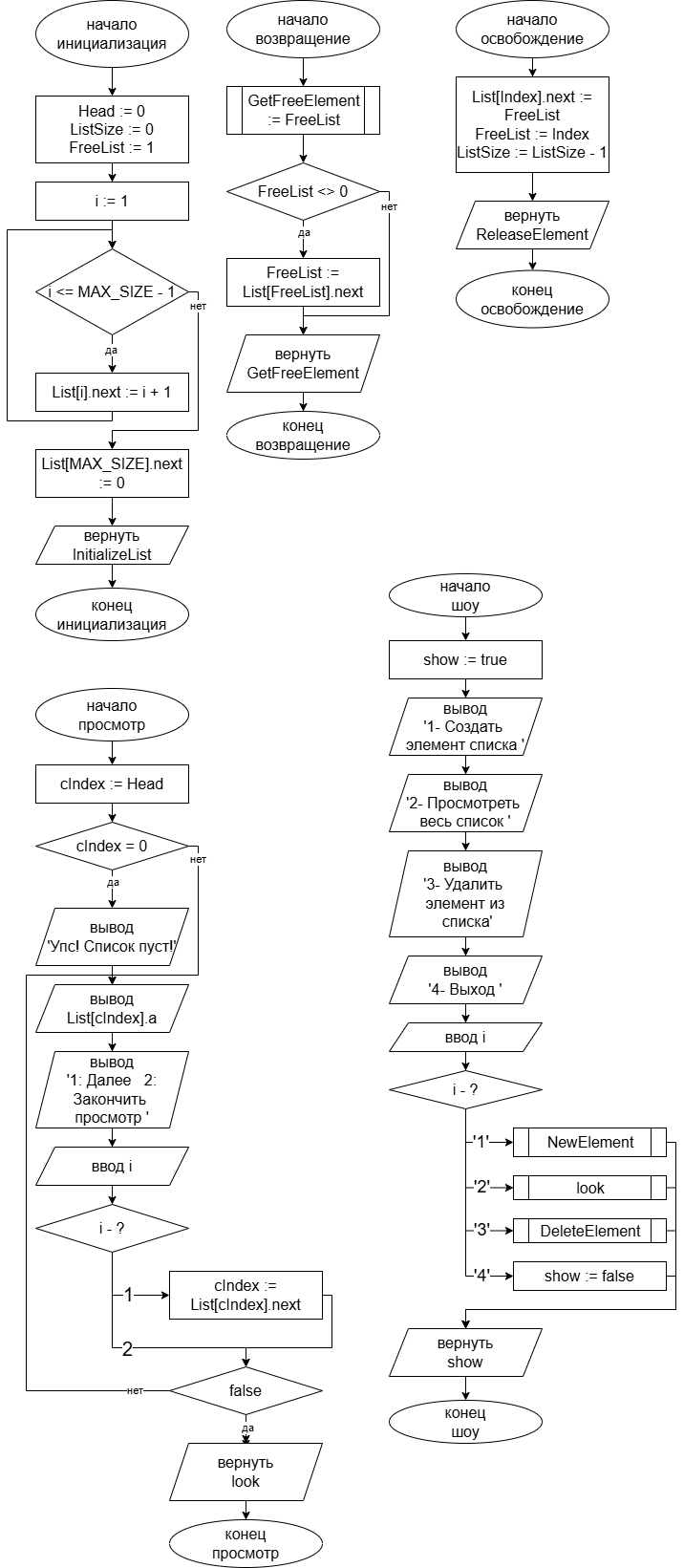
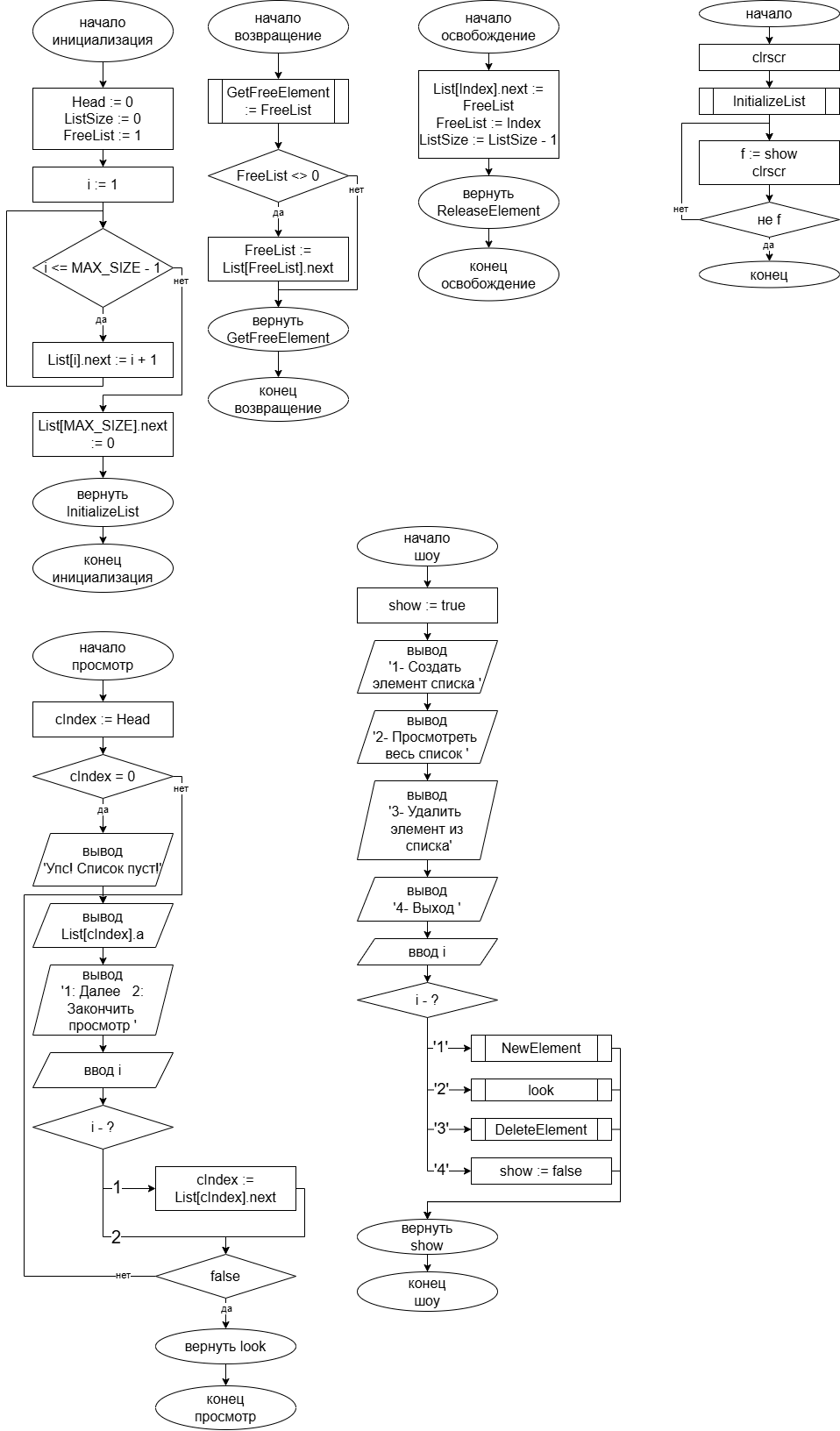


Рисунок 7 – Схема алгоритма процедуры выбора команды

Рисунок 8 – Схема алгоритма основной программы

1. Код программы

**uses** crt;

**const**

MAX\_SIZE = 5;

**type**

intr = **record**

a: integer;

next: integer;

**end**;

**var**

List: **array**[1..MAX\_SIZE] **of** intr;

Head: integer;

ListSize: integer;

FreeList: integer;

**procedure** InitializeList;

**var** i: integer;

**begin**

Head := 0;

ListSize := 0;

FreeList := 1;

**for** i := 1 **to** MAX\_SIZE - 1 **do**

List[i].next := i + 1;

List[MAX\_SIZE].next := 0;

**end**;

**function** GetFreeElement: integer;

**begin**

GetFreeElement := FreeList;

**if** FreeList <> 0 **then**

**begin**

FreeList := List[FreeList].next;

**end**;

**end**;

**procedure** ReleaseElement(Index: integer);

**begin**

List[Index].next := FreeList;

FreeList := Index;

ListSize := ListSize - 1;

**end**;

**procedure** NewElement;

**var**

NewIndex, a, bIndex: integer;

**begin**

NewIndex := GetFreeElement;

**if** NewIndex = 0 **then**

**begin**

writeln('Нет свободных элементов.');

readln;

**exit**;

**end**;

write('Введите число: ');

readln(List[NewIndex].a);

**if** Head = 0 **then**

**begin**

Head := NewIndex;

List[NewIndex].next := NewIndex;

**end**

**else**

**begin**

bIndex := Head;

**while** List[bIndex].next <> Head **do**

bIndex := List[bIndex].next;

List[NewIndex].next := Head;

List[bIndex].next := NewIndex;

**end**;

**end**;

**procedure** look;

**var**

cIndex, sIndex, i: integer;

**begin**

cIndex := Head;

**if** cIndex = 0 **then**

**begin**

writeln('Упс! Список пуст!');

readln;

**exit**;

**end**;

**repeat**

writeln(List[cIndex].a);

write('1: Далее 2: Закончить просмотр ');

readln(i);

**case** i **of**

1: cIndex := List[cIndex].next;

2: **break**;

**end**;

**until** false;

**end**;

**procedure** DeleteElement;

**var**

cIndex, prevIndex, ElementDel: integer;

found: boolean;

**begin**

**if** Head = 0 **then**

**begin**

writeln('Список пуст, удалять нечего.');

readln;

**exit**;

**end**;

write('Введите значение элемента для удаления: ');

readln(ElementDel);

cIndex := Head;

prevIndex := 0;

found := False;

**repeat**

**if** List[cIndex].a = ElementDel **then**

**begin**

found := True;

**if** cIndex = Head **then**

**begin**

**if** List[cIndex].next = Head **then**

**begin**

Head := 0;

ReleaseElement(cIndex);

**end**

**else**

**begin**

prevIndex := Head;

**while** List[prevIndex].next <> Head **do**

prevIndex := List[prevIndex].next;

Head := List[cIndex].next;

List[prevIndex].next := Head;

ReleaseElement(cIndex);

**end**;

**end**

**else**

**begin**

prevIndex := Head;

**while** List[prevIndex].next <> cIndex **do**

prevIndex := List[prevIndex].next;

List[prevIndex].next := List[cIndex].next;

ReleaseElement(cIndex);

**end**;

writeln('Элемент успешно удален.');

readln;

**exit**;

**end**;

**if** prevIndex<> cIndex **then**

prevIndex := cIndex;

cIndex := List[cIndex].next;

**until** cIndex = Head;

**if not** found **then**

**begin**

writeln('Элемент со значением ', ElementDel, ' не найден.');

readln;

**end**;

**end**;

**function** show: boolean;

**var**

i: char;

**begin**

show := true;

writeln('1- Создать элемент списка ');

writeln('2- Просмотреть весь список ');

writeln('3- Удалить элемент из списка');

writeln('4- Выход ');

readln(i);

**case** i **of**

'1': NewElement;

'2': look;

'3': DeleteElement;

'4': show := false;

**end**;

**end**;

**var**

f: boolean;

**begin**

clrscr;

InitializeList;

**repeat**

f := show;

clrscr;

**until not** f;

**end**.

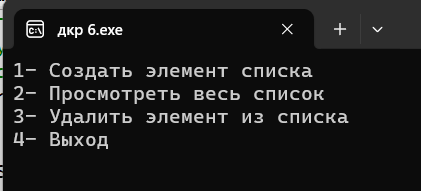
1. Результат выполнения программы

Рисунок 9 – Скриншот главного меню при запуске

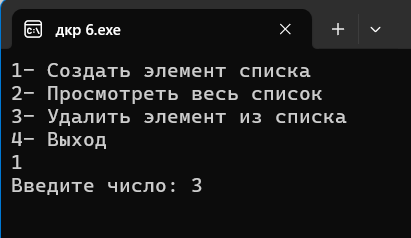
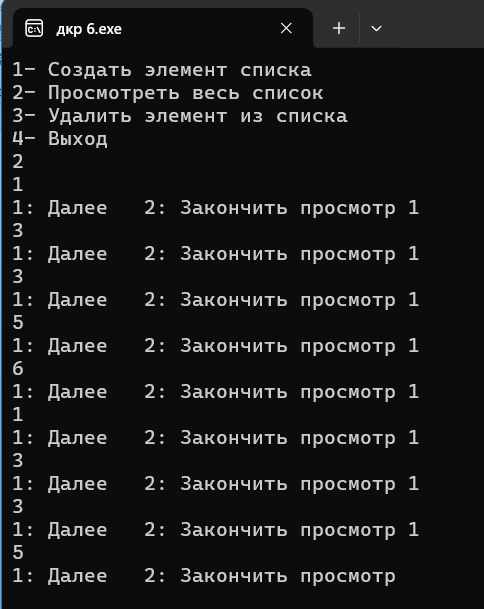
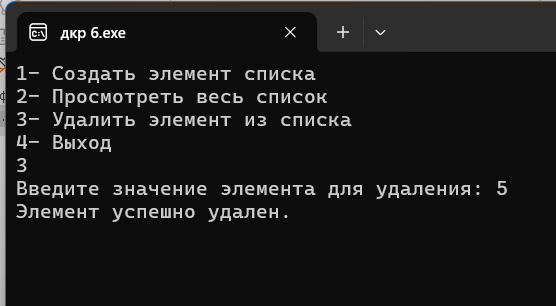
Рисунок 10 – Скриншот создание элемента списка

Рисунок 11 – Скриншот просмотреть весь список

Рисунок 12 – Скриншот удаление элемента списка

Вывод:

В ходе работы, благодаря доступной необходимой теоретической информации, не возникло особых сложностей. Я смогла глубже понять изучаемую тему и применить полученные знания на практике. Процесс работы был увлекательным и познавательным, что позволило не только улучшить навыки анализа и критического мышления, но и развить творческий подход к решению задач.

Кроме того, сотрудничество с одногруппниками и обмен идеями способствовали более глубокому осмыслению материала и расширению горизонтов. Я уверена, что полученный опыт будет полезен в будущем, и с нетерпением жду возможности применить его в новых проектах. В целом, проделанная работа стала важным этапом в моем профессиональном развитии.