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»**

**«МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент учебной группы

ИСПк-202-52-00

Долгополов Ярослав Максимович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

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

**Формулировка задания. Вариант 7**

**Постановка задачи**

1. Написать программу для работы со структурой данных "Кольцевой двусвязный 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**

* 1. Устанавливаем типы **elem**, являющаяся записью, содержащая строку и индексы следующего и предыдущего элементов списка, и **list**, являющаяся записью, содержащая массив элементов типа **elem**, индексы первого и последнего элементов списка и номер последнего записанного элемента.
  2. Устанавливаем цикл с постусловием. Внутри цикла устанавливаем выбор: 1 — добавить элемент к списку, 2 — удалить элемент из списка, 3 — выход из цикла.
  3. При выборе 1 производится инициализация строки, а затем будет предложен выбор: 1 — добавить в начало списка, 2 — добавить в конец списка, 3 — добавить после выбранного элемента, 4 — добавить до выбранного элемента. При выборе 3 и 4 необходимо выбрать номер элемента, после или до которого надо добавить новый элемент.
  4. При выборе 2 необходимо выбрать номер элемента, который необходимо удалить.

**Схема алгоритма**

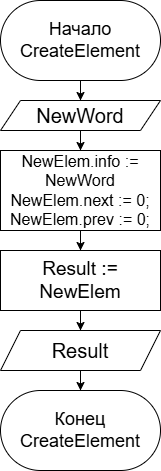
****

Рисунок 1 — функция CreateElement

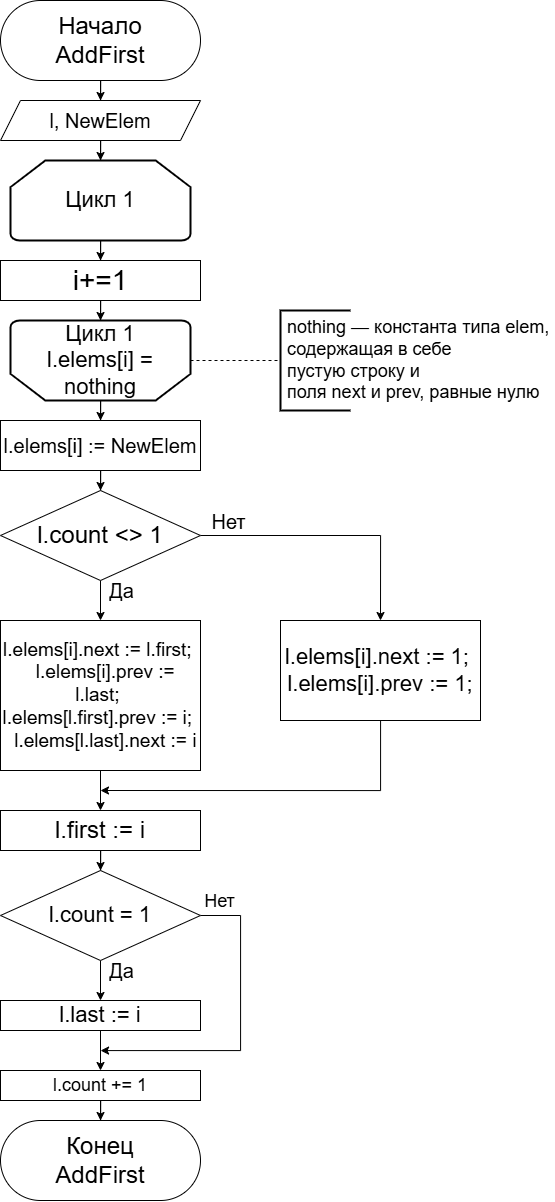


Рисунок 2 — процедура AddFirst

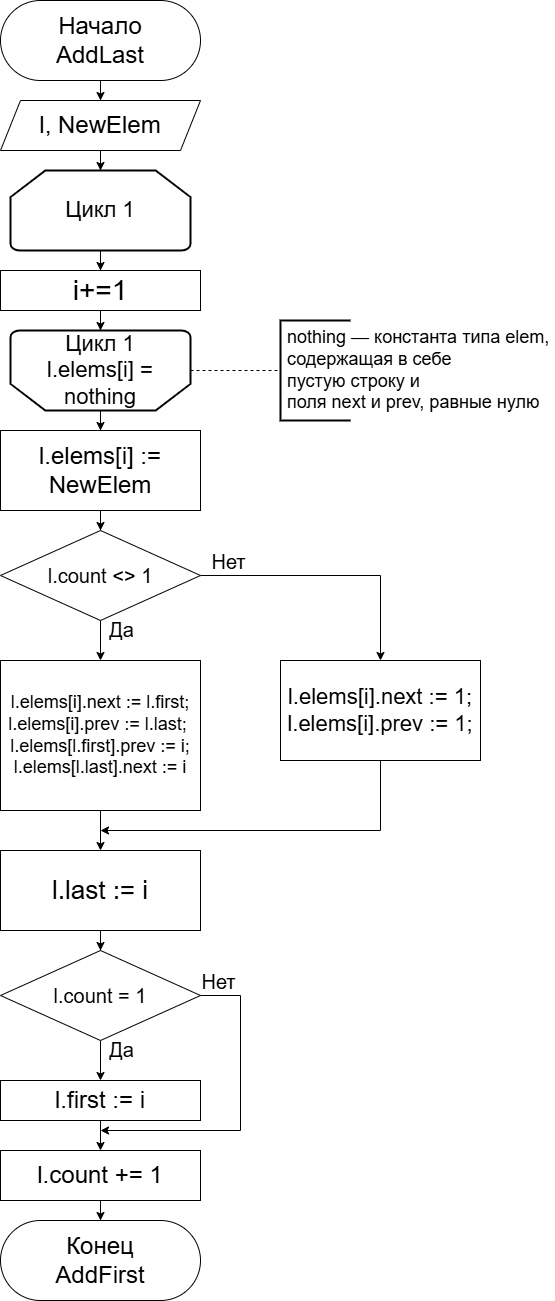
****

Рисунок 3 — процедура AddLast

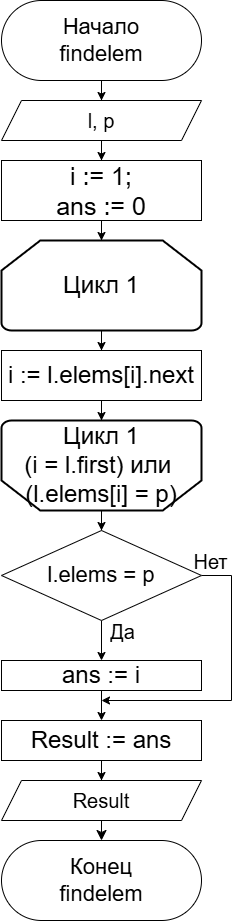
****

Рисунок 4 — функция Findelem

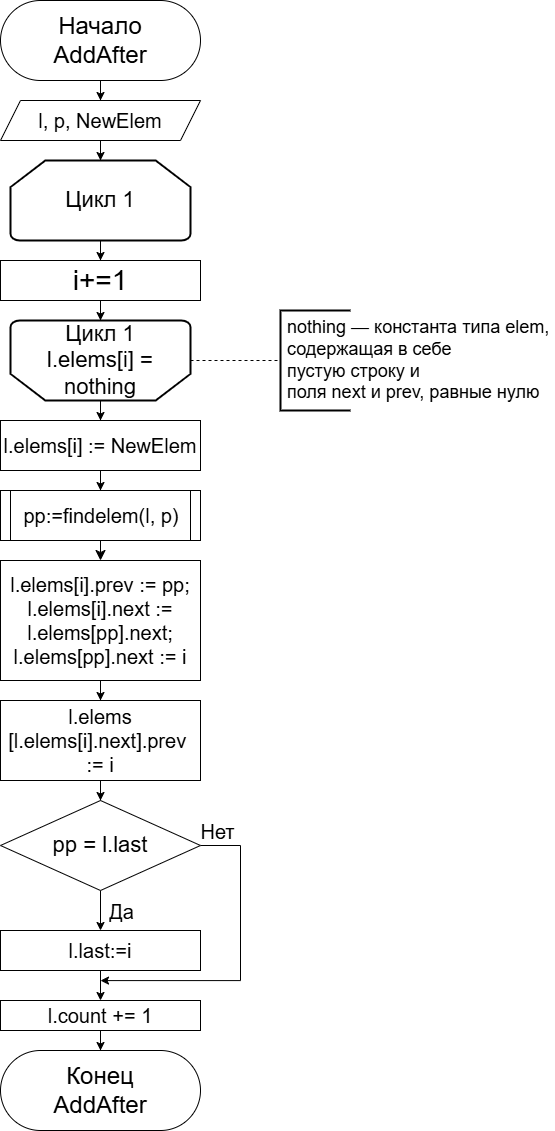


Рисунок 5 — процедура AddAfter

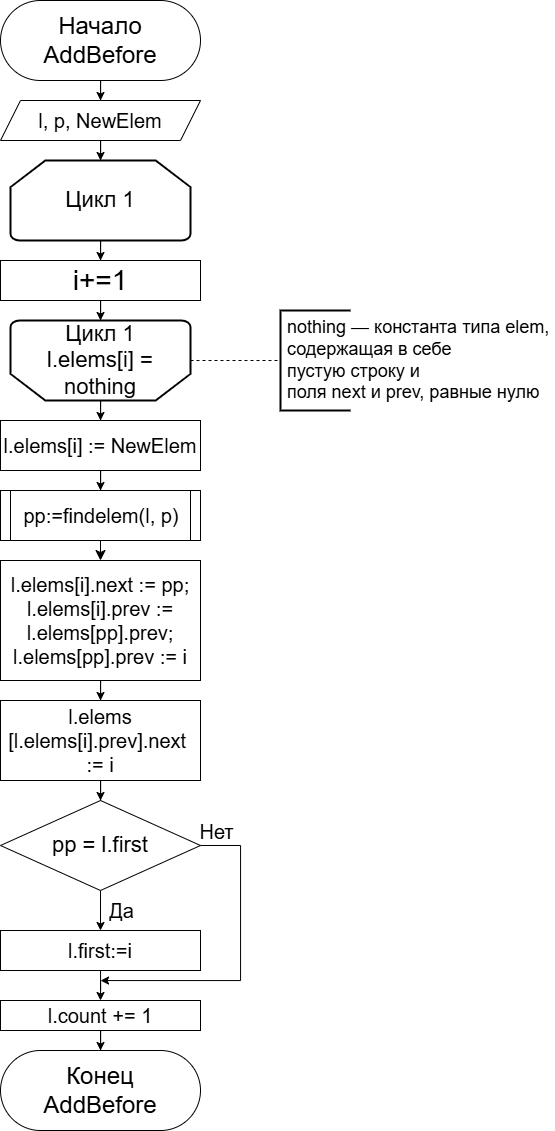


Рисунок 6 — процедура AddBefore

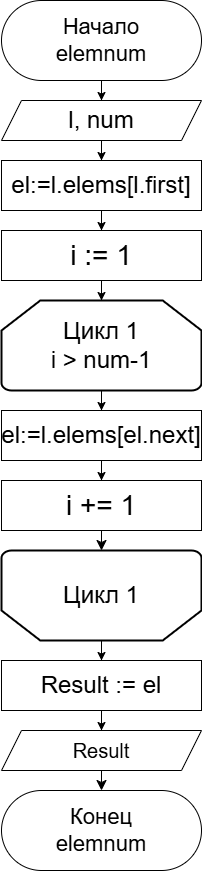


Рисунок 7 — функция elemnum

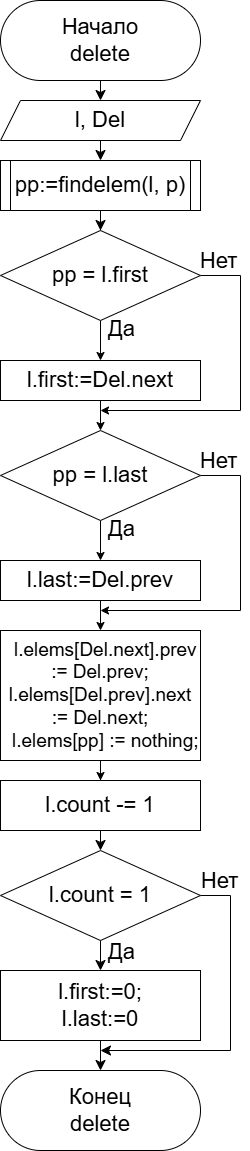


Рисунок 8 — процедура delete

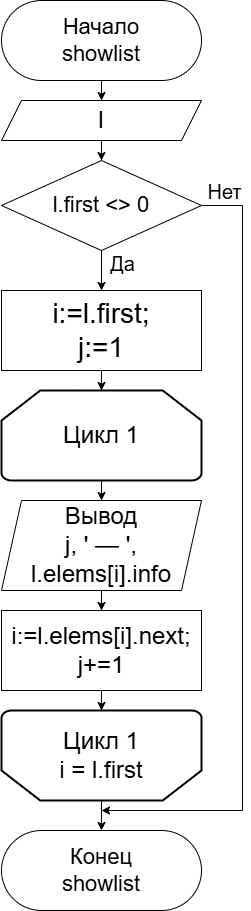


Рисунок 9 — процедура showlist

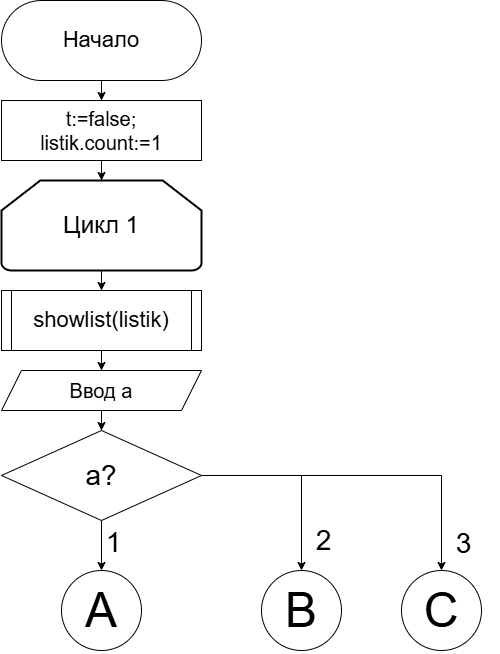


Рисунок 10 — алгоритм программы. Часть 1

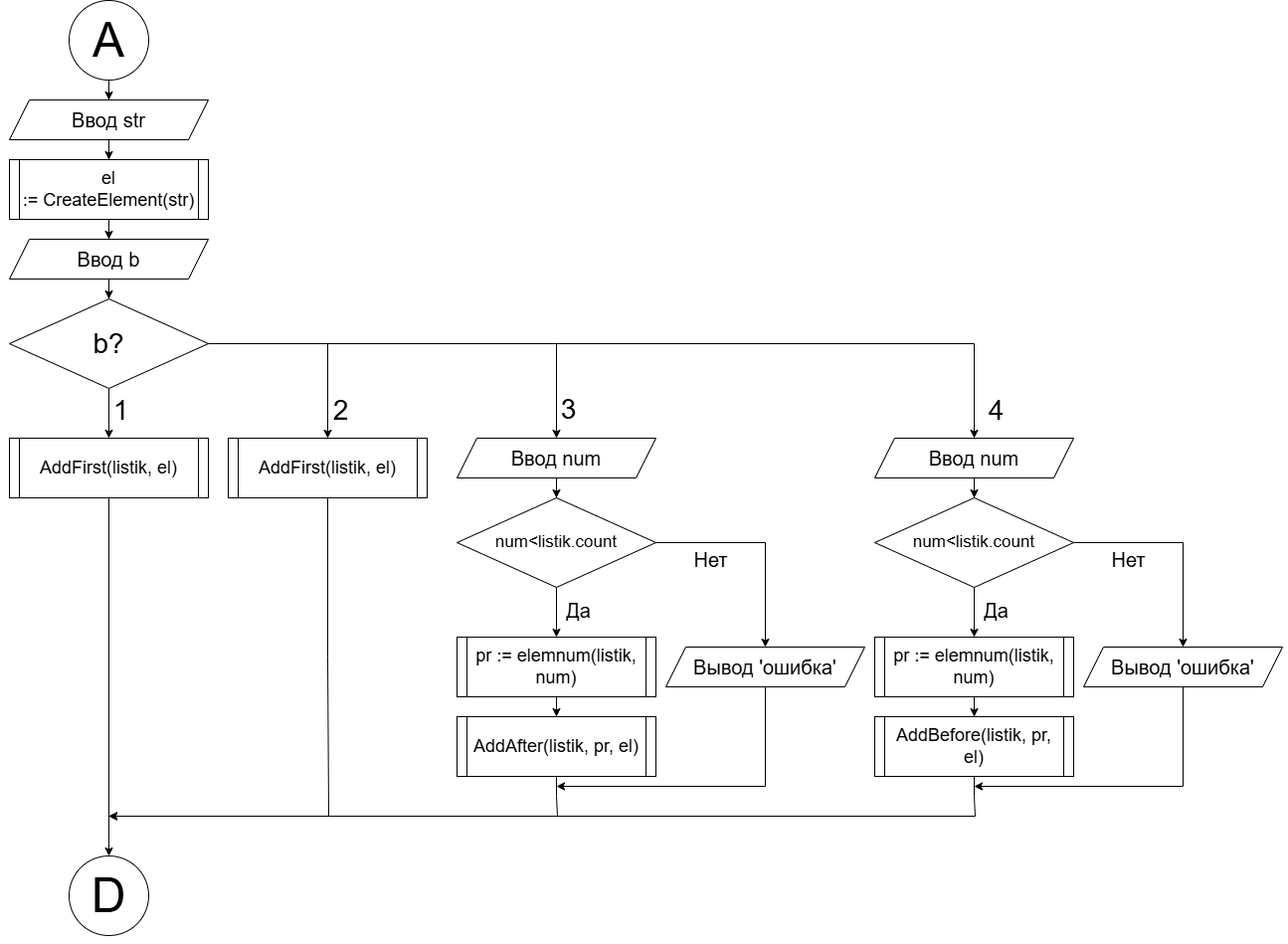


Рисунок 11 — алгоритм программы. Часть 2

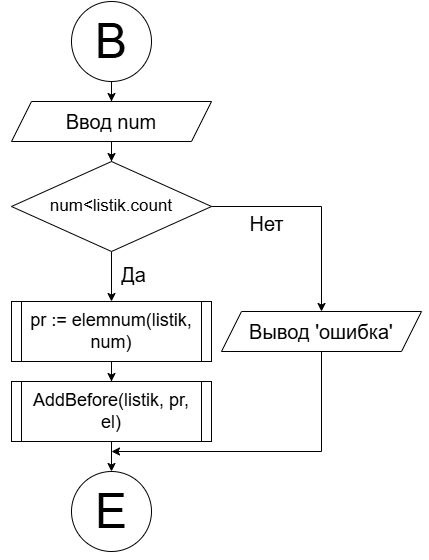


Рисунок 12 — алгоритм программы. Часть 3

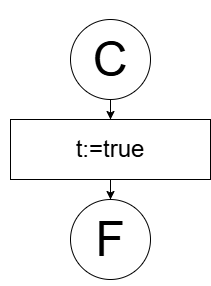


Рисунок 13 — алгоритм программы. Часть 4

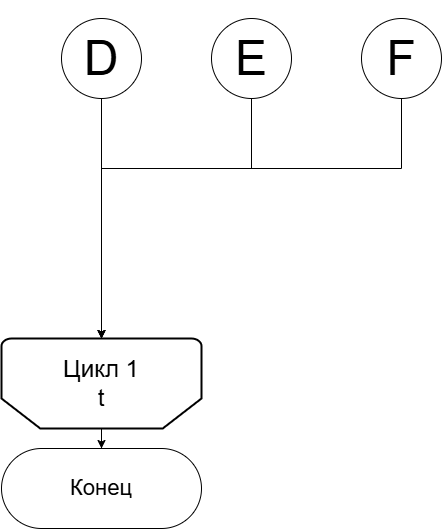


Рисунок 14 — алгоритм программы. Часть 5

**Код программы**

**type**

elem = **record**

info: string[40];

next, prev: byte;

**end**;

list = **record**

elems: **array**[1..255] **of** elem;

first, last: byte;

count: byte;

**end**;

**const**

nothing: elem = (info: ''; next: 0; prev: 0);

**function** CreateElement(NewWord: string[40]): elem;

**var** NewElem: elem;

**begin**

NewElem.info := NewWord;

NewElem.next := 0;

NewElem.prev := 0;

Result := NewElem

**end**;

**procedure** AddFirst(**var** l: list; NewElem: elem);

**var** i: byte;

**begin**

**repeat**

i+=1

**until** l.elems[i] = nothing;

l.elems[i]:=NewElem;

**if** l.count <> 1 **then**

**begin**

l.elems[i].next:=l.first;

l.elems[i].prev:=l.last;

l.elems[l.first].prev:=i;

l.elems[l.last].next:=i

**end**

**else**

**begin**

l.elems[i].next:=1;

l.elems[i].prev:=1;

**end**;

l.first:=i;

**if** l.count = 1 **then** l.last:=i;

l.count+=1

**end**;

**procedure** AddLast(**var** l: list; NewElem: elem);

**var** i: byte;

**begin**

**repeat**

i+=1

**until** l.elems[i] = nothing;

l.elems[i]:=NewElem;

**if** l.count <> 1 **then**

**begin**

l.elems[i].next:=l.first;

l.elems[i].prev:=l.last;

l.elems[l.first].prev:=i;

l.elems[l.last].next:=i

**end**

**else**

**begin**

l.elems[i].next:=1;

l.elems[i].prev:=1;

**end**;

l.last:=i;

**if** l.count = 1 **then** l.first:=i;

l.count+=1

**end**;

**function** findelem(**var** l: list; p: elem): byte;

**var** i, ans: integer;

**begin**

i:=l.first;

ans:=0;

**repeat**

**begin**

i:=l.elems[i].next

**end**

**until** (i = l.first) **or** (l.elems[i] = p);

**if** l.elems[i] = p **then** ans:=i;

Result:=ans

**end**;

**procedure** AddAfter(**var** l: list; p, NewElem: elem);

**var** pp, i: byte;

**begin**

**repeat**

i+=1

**until** l.elems[i] = nothing;

l.elems[i]:=NewElem;

pp:=findelem(l, p);

l.elems[i].prev := pp;

l.elems[i].next := l.elems[pp].next;

l.elems[pp].next := i;

l.elems[l.elems[i].next].prev := i;

**if** pp = l.last **then** l.last:=i;

l.count+=1

**end**;

**procedure** AddBefore(**var** l: list; p, NewElem: elem);

**var** pp, i: byte;

**begin**

**repeat**

i+=1

**until** l.elems[i] = nothing;

l.elems[i]:=NewElem;

pp:=findelem(l, p);

l.elems[i].next := pp;

l.elems[i].prev := l.elems[pp].prev;

l.elems[pp].prev := i;

l.elems[l.elems[i].prev].next := i;

**if** pp = l.first **then** l.first:=i;

l.count+=1

**end**;

**function** elemnum(**var** l: list; num: byte): elem;

**var**

el: elem;

i: byte;

**begin**

el:=l.elems[l.first];

**for** i:=1 **to** num-1 **do**

**begin**

el:=l.elems[el.next]

**end**;

Result:=el

**end**;

**procedure** delete(**var** l: list; Del: elem);

**var**

pp, i: byte;

**begin**

pp:=findelem(l, Del);

**if** pp = l.first **then** l.first:=Del.next;

**if** pp = l.last **then** l.last:=Del.prev;

l.elems[Del.next].prev:=Del.prev;

l.elems[Del.prev].next:=Del.next;

l.elems[pp]:=nothing;

l.count-=1;

**if** l.count = 1 **then**

**begin**

l.first:=0;

l.last:=0

**end**

**end**;

**procedure** showlist(l: list);

**var** i, j: byte;

**begin**

**if** l.first <> 0 **then**

**begin**

i:=l.first;

j:=1;

**repeat**

**begin**

writeln(j, ' — ', l.elems[i].info);

i:=l.elems[i].next;

j+=1

**end**

**until** i = l.first

**end**

**end**;

**var**

listik: list;

el, pr, del: elem;

a, b: integer;

str: string;

t: boolean;

num: byte;

**begin**

t:=false;

listik.count:=1;

**repeat**

**begin**

writeln('Список');

showlist(listik);

writeln();

writeln('Выберите вариант');

writeln('1 - добавить элемент');

writeln('2 - удалить элемент');

writeln('3 - завершить работу программы');

readln(a);

**case** a **of**

1:

**begin**

writeln('Введите слово');

readln(str);

el:=CreateElement(str);

writeln('Выберите способ вставки');

writeln('1 - в начало');

writeln('2 - в конец');

writeln('3 - после определённого элемента');

writeln('4 - до определённого элемента');

readln(b);

**case** b **of**

1: AddFirst(listik, el);

2: AddLast(listik, el);

3:

**begin**

writeln('Введите номер элемента, после которого надо вставить слово');

readln(num);

**if** num<listik.count **then**

**begin**

pr:=elemnum(listik, num);

AddAfter(listik, pr, el)

**end**

**else** writeln('Ошибка. Введёное число больше количества элементов списка')

**end**;

4:

**begin**

writeln('Введите номер элемента, перед которым надо вставить слово');

readln(num);

**if** num<listik.count **then**

**begin**

pr:=elemnum(listik, num);

AddBefore(listik, pr, el)

**end**

**else** writeln('Ошибка. Введёное число больше количества элементов списка')

**end**;

**end**;

**end**;

2:

**begin**

writeln('Введите номер элемента, который надо удалить');

readln(num);

**if** num<listik.count **then**

**begin**

del:=elemnum(listik, num);

delete(listik, del)

**end**

**else** writeln('Ошибка. Введёное число больше количества элементов списка')

**end**;

3: t:=true;

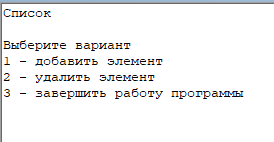
**end**

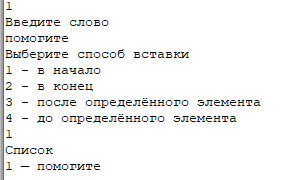
**end**

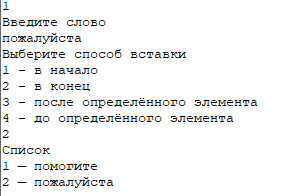
**until** t

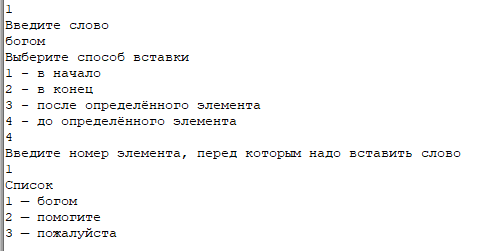
**end**.

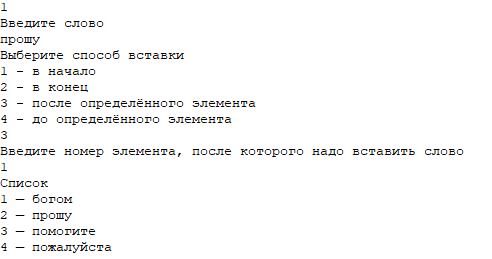
**Результат выполнения программы**











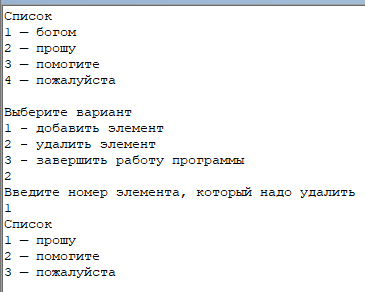


Рисунок 15 — результат выполнения программы

**Вывод**

Кольцевой двунаправленный список — структура, каждый элемент которой «знает» адрес следующего и предыдущего элементов, и конечный и начальный элементы соединены.

Основные операции с кольцевым двунаправленным списком:

* Добавление узла в список в начало, конец, после или до определённого элемента (AddBefore, AddAfter, AddFirst, AddLast);
* Удаление узла из списка (Delete);
* Создание нового узла из введённого слова (CreateElement).

Использование языка программирования Паскаль позволяет эффективно работать с такой структурой данных, как кольцевой двунаправленный список.