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

Бабенко Александр Александрович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

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

Задание:

* Написать программу для работы со структурой данных «Кольцевой односвязный список».
* Структура данных должна быть реализована на основе статической и динамической памяти.
* Работа со структурой должна осуществляться с помощью case-меню.
* Предусмотреть наглядную визуализацию содержимого структуры.

Описание алгоритма:

Настоящий алгоритм был разработан для использования абстрактного типа данных «Кольцевой односвязный список» реализованного с помощью списка. Для визуализации списка и управлением список используется case-меню.

Для работы с кольцевым односвязным списком реализованы функции: добавления элемента в конец, взятия элемента из конца списка, выведение содержимого списка.

Каждый элемент списка хранит в себе: значение, номер элемента в списке и ссылку на следующий элемент списка.

При добавлении нового элемента в список пользователь вводит значение элемента, а номер элемента и ссылка определяются автоматически.

При взятии элемента из списка голова списка ссылается на следующий элемент от первого, то есть на второй. Номера элементов в списке пересчитываются при каждом взятии элемента.

Визуализация элементов списка происходит с помощью вывода значений списка в консоль.

Схема алгоритма с комментариями:

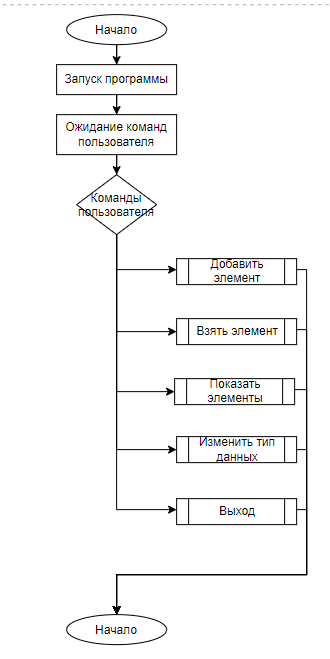


Рисунок 1 - Общие описание алгоритма

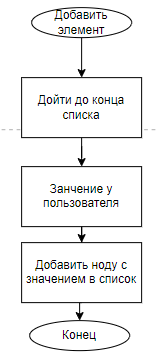


Рисунок 2 - алгоритм добавления элемента в список

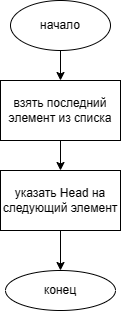


Рисунок 3- алгоритм взятия элемента из списка

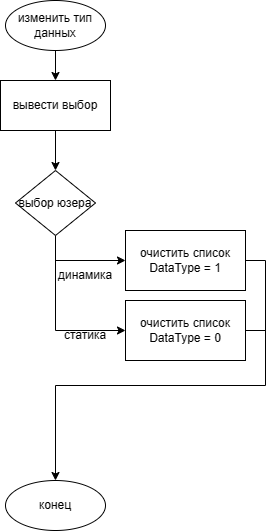


Рисунок 4 - алгоритм смены типа данных

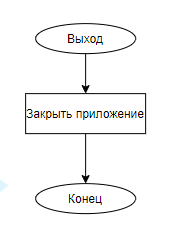


Рисунок 5 - алгоритм выхода

Результаты работы программы:

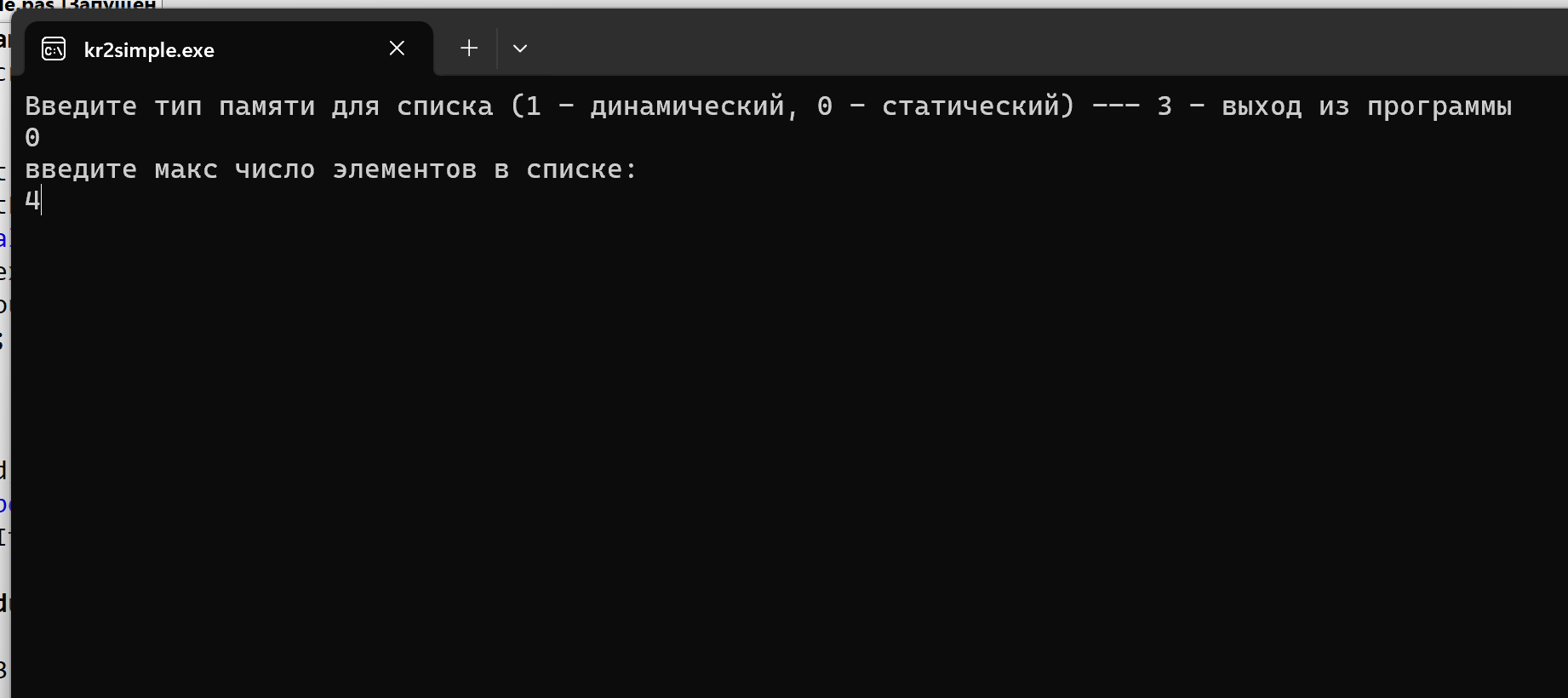


Рисунок 1 – выбор типа данных

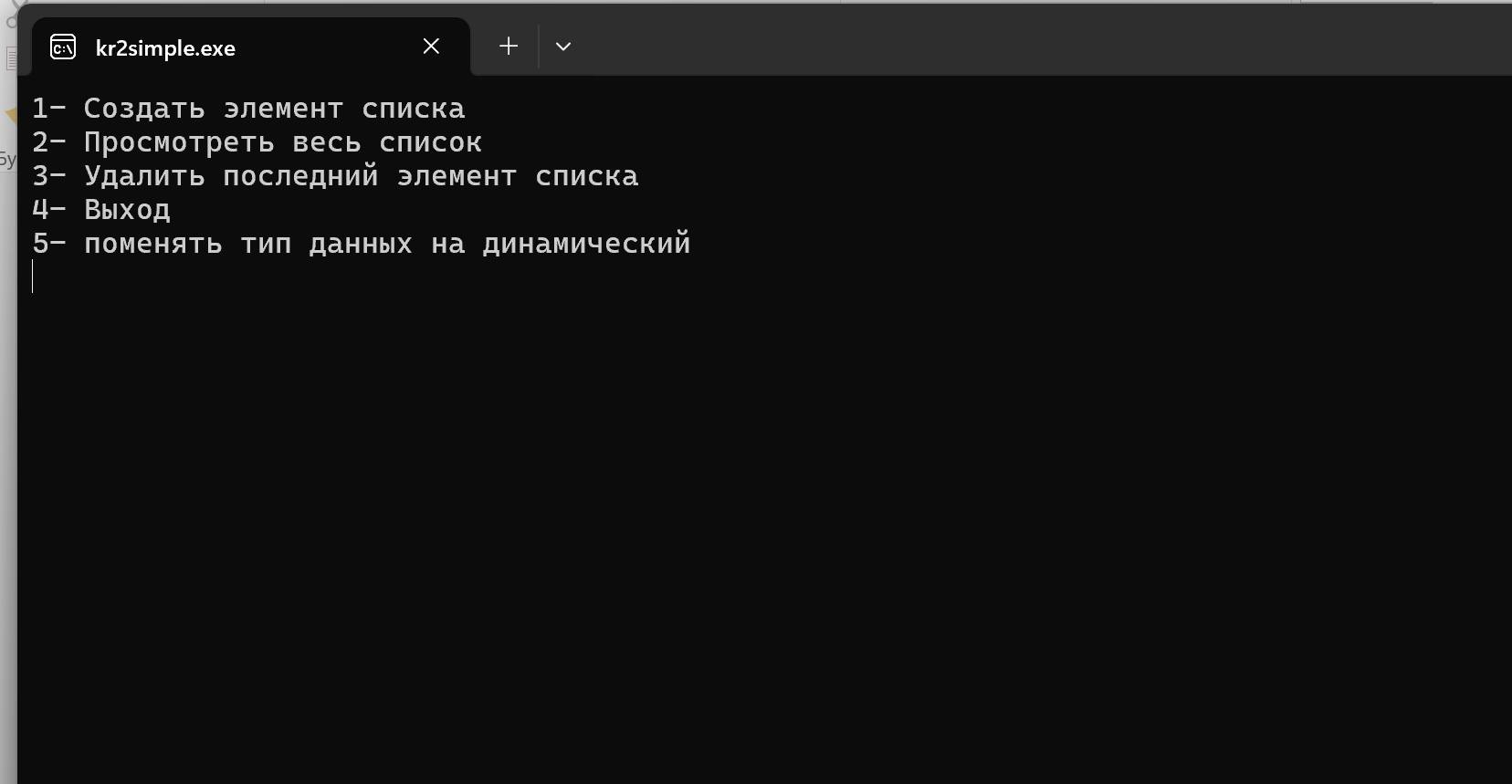


Рисунок 2 – главное меню взаимодействия с пользователем

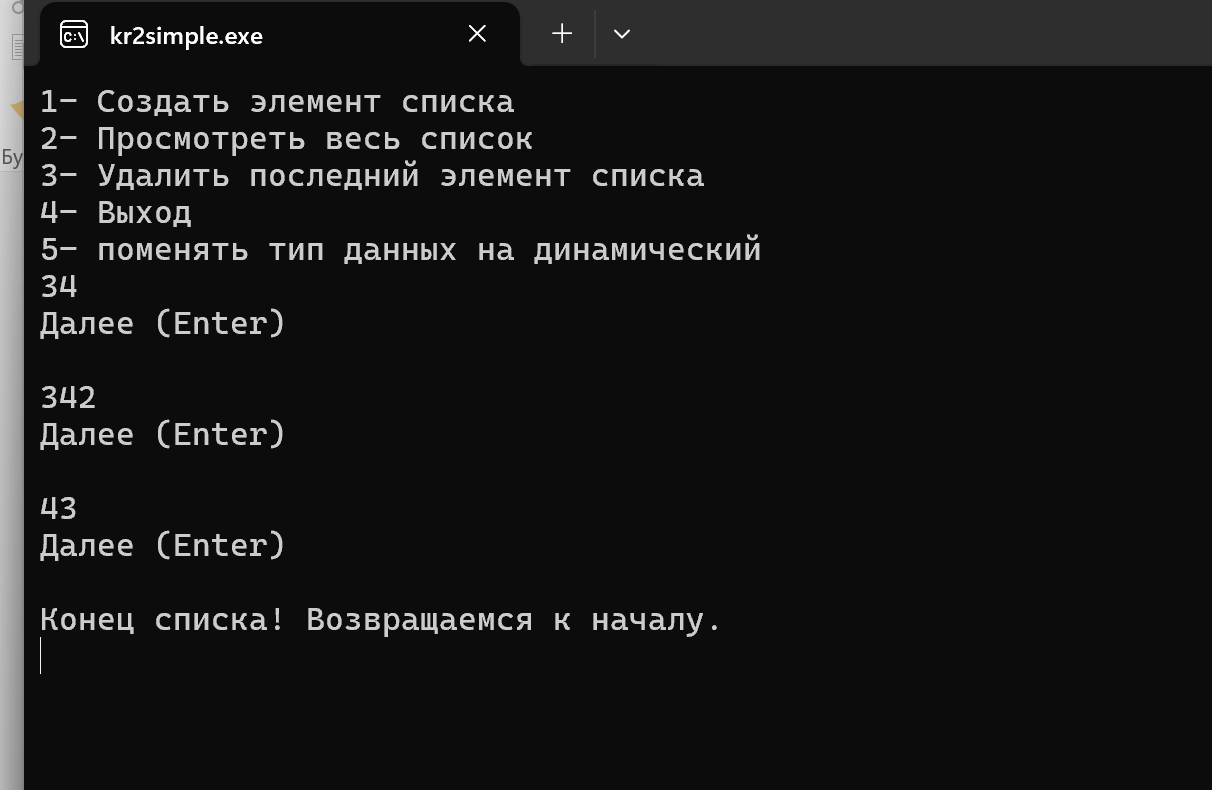


Рисунок 3 – просмотр списка

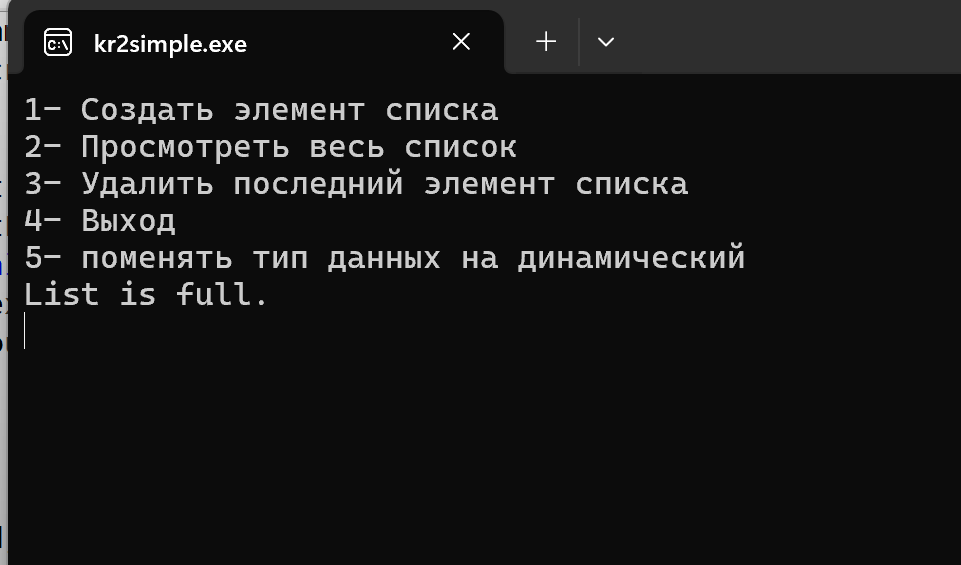


Рисунок 4 – переполнение списка

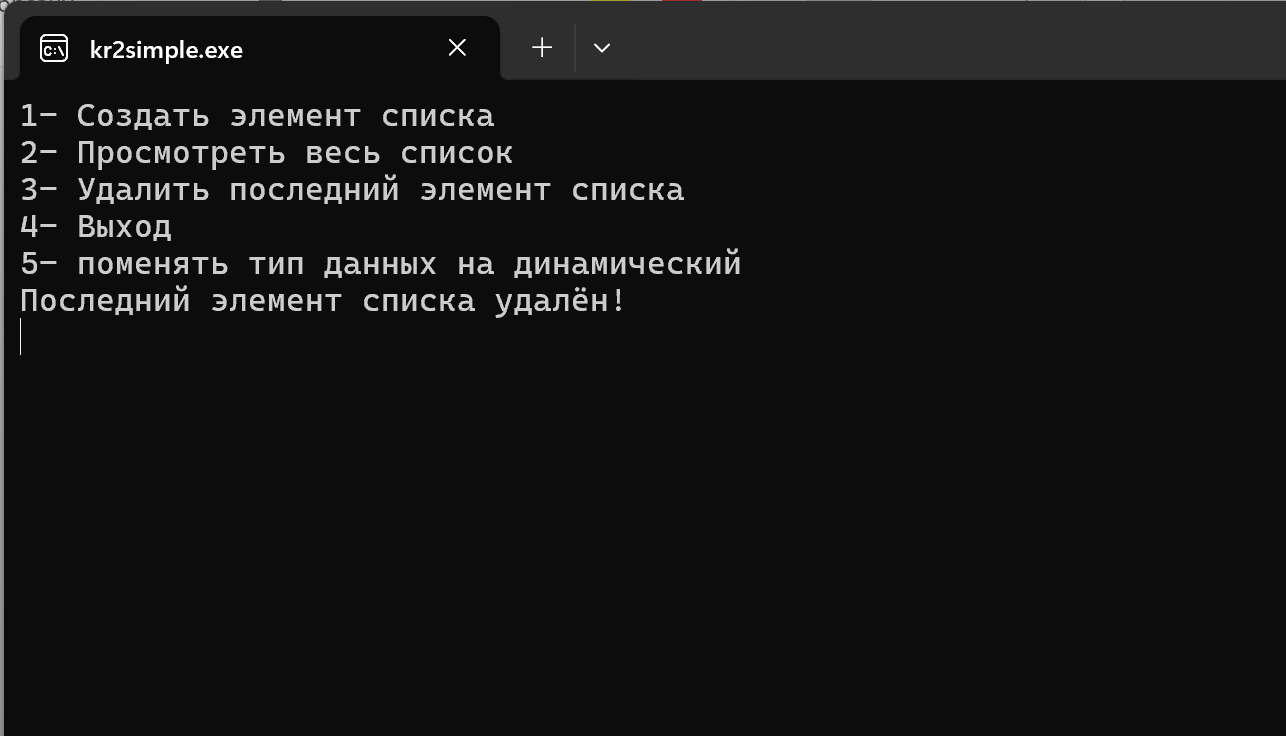


Рисунок 5 – удаление элемента

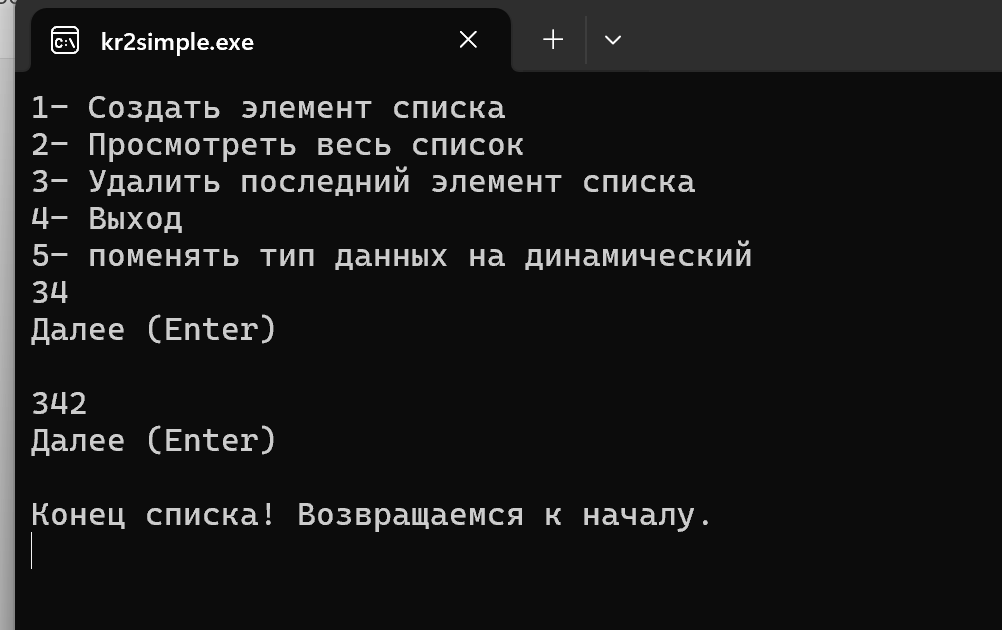


Рисунок 6 – список после удаления элемента

Вывод: По итогам работы были изучены принципы работы с базовыми структурами данных, в частности – кольцевой односвязный список, а также закреплены навыки работы с кейс-меню. При выполнении работы я столкнулся с такими сложностями, как реализация двух типов памяти для одной структуры данных, и некорректное отображение списка в кейс меню. Цель работы была полностью достигнута.

Приложение 1 – Исходный код программы

**program** Spis;

**uses** crt;

**type**

Pint = ^ListElem;

ListElem = **record**

value: integer;

next: Pint;

count:integer;

**end**;

**var**

Head,Sp,Spp: Pint;

F: boolean;

maxItems,itemCount,DataType:integer;

**procedure** NtCreat(**var** Hed: Pint);

**var**

C, B: Pint;

**begin**

**New**(C);

Write('Введите значение: ');

readln(C^.value);

**if** hed = nil **then**

**begin**

hed := c;

hed^.next := Hed;

**end**

**else**

**begin**

B := hed;

**while** B^.next <> hed **do**

B := B^.next;

C^.next := Hed;

B^.next := C;

**end**;

Inc(itemCount);

**end**;

**procedure** DeleteLastElement(**var** Hed: Pint);

**var**

prev, curr: Pint;

**begin**

**if** Hed = nil **then**

**begin**

writeln('Список пуст!');

readln;

**exit**;

**end**;

prev := nil;

curr := Hed;

**while** curr^.next <> Hed **do**

**begin**

prev := curr;

curr := curr^.next;

**end**;

**if** prev = nil **then**

**begin**

dispose(Hed);

Hed := nil;

writeln('В вашем списке удалён последний элемент');

Delay(1500);

**end**

**else**

**begin**

prev^.next := Hed;

dispose(curr);

writeln('Последний элемент списка удалён!');

Delay(1500);

**end**;

**end**;

**procedure** View(**var** Hed: Pint);

**var**

C: Pint;

**begin**

C := Hed;

**if** C = nil **then**

**begin**

writeln('Список пуст!');

readln;

**exit**;

**end**;

**repeat**

writeln(C^.value);

writeln('Далее (Enter)');

readln;

C := C^.next;

**until** C = Hed;

writeln('Конец списка! Возвращаемся к началу.');

readln;

**end**;

**function** Show(): boolean;

**var**

choice: char;

**begin**

writeln('Введите тип памяти для списка (1 - динамический, 0 - статический) --- 3 - выход из программы');

Readln(DataType);

**if** DataType = 3 **then**

**begin**

**Exit**

**end**;

**if** DataType = 0 **then**

**begin**

Writeln('введите макс число элементов в списке: ');

Readln(maxItems);

itemCount := 0;

**end**;

**while** DataType = 1 **do begin**

ClrScr;

Show := true;

writeln('1- Создать элемент списка');

writeln('2- Просмотреть весь список');

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

writeln('4- Выход');

writeln('5- Поменять тип данных на статический');

choice := readkey;

**case** choice **of**

'1': NtCreat(Sp);

'2': View(Sp);

'3': DeleteLastElement(Sp);

'4': **Exit**;

'5': DataType := 0;

**end**;

**end**;

**while** DataType = 0 **do begin**

ClrScr;

Show := true;

writeln('1- Создать элемент списка');

writeln('2- Просмотреть весь список');

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

writeln('4- Выход');

writeln('5- поменять тип данных на динамический');

choice := readkey;

**case** choice **of**

'1':

**begin**

**if** itemCount < maxItems **then**

**begin**

NtCreat(Spp);

**end**

**else**

**begin**

Writeln('List is full.');

Delay(3000);

**end**;

**end**;

'2': View(Spp);

'3': DeleteLastElement(Spp);

'4': **Exit**;

'5': DataType := 1;

**end**;

**end**;

**end**;

**begin**

ClrScr;

Head := nil;

**repeat**

F := Show();

clrscr;

**until not** F;

**end**.