Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Алгоритмизация и программирование”

Лабораторная работа №7

“Программирование линейных списков”

Выполнил:

ст.гр. ИС/б-12

Долженко И.А.

Проверил:

Забаштанский А.К.

Севастополь

2017

1 ЦЕЛЬ РАБОТЫ

Исследование списковых структур данных и приобретение навыков разработки и отладки программ, использующих динамическую память. Исследование особенностей использования переменных ссылочного типа.

2 ПОСТАНОВКА ЗАДАЧИ

Представить одну из приведенных ниже таблиц в виде линейного списка L, элементами которого являются строки таблицы. Написать процедуры организации, добавления элемента в список, исключения элемента из списка, просмотра списка, а также одну процедуру, которая вставляет в конец списка L новый элемент Е (Таблица 3.1.).

Значения и количество записей в таблице студент выбирает самостоятельно. Исходные данные после организации списка должны сохраняться в файле и при повторном запуске программы считываться из файла. Количество строк таблицы не задается.

3 СТРУКТУРНАЯ СХЕМА АЛГОРИТМА

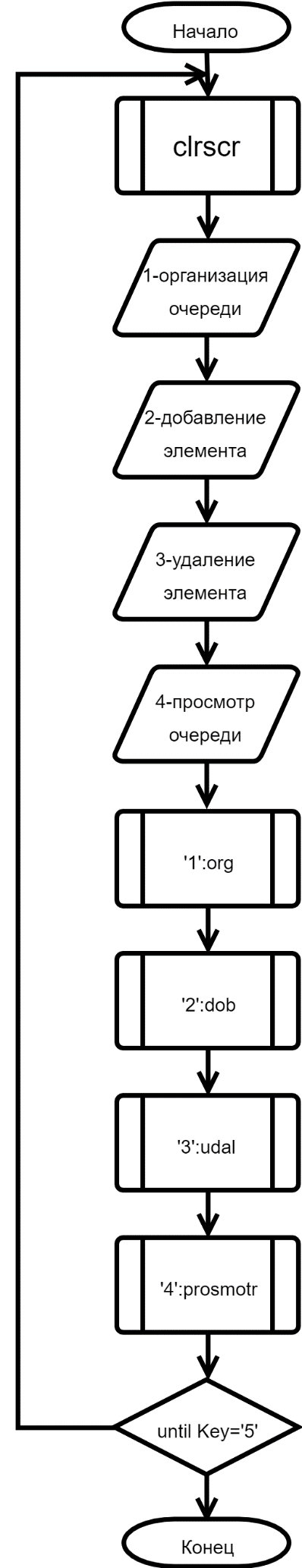


Рисунок 1 – Структурная схема основной программы

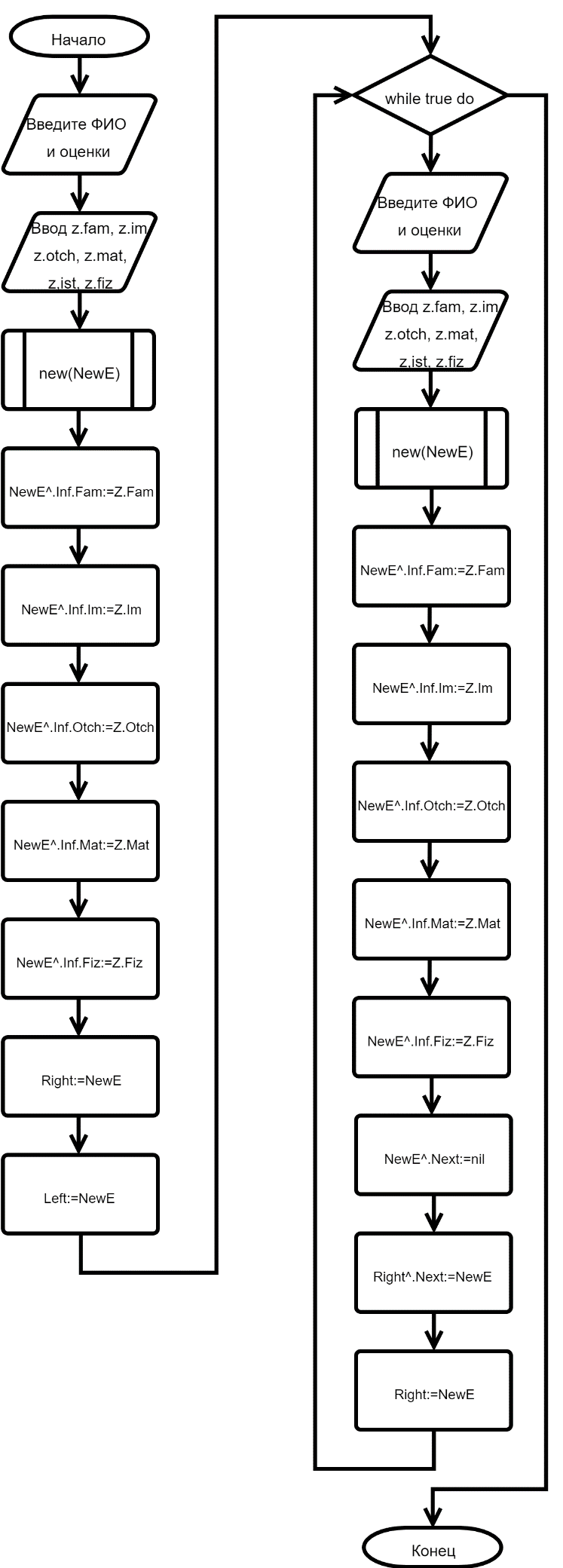


Рисунок 2 – Структурная схема процедуры создания списка

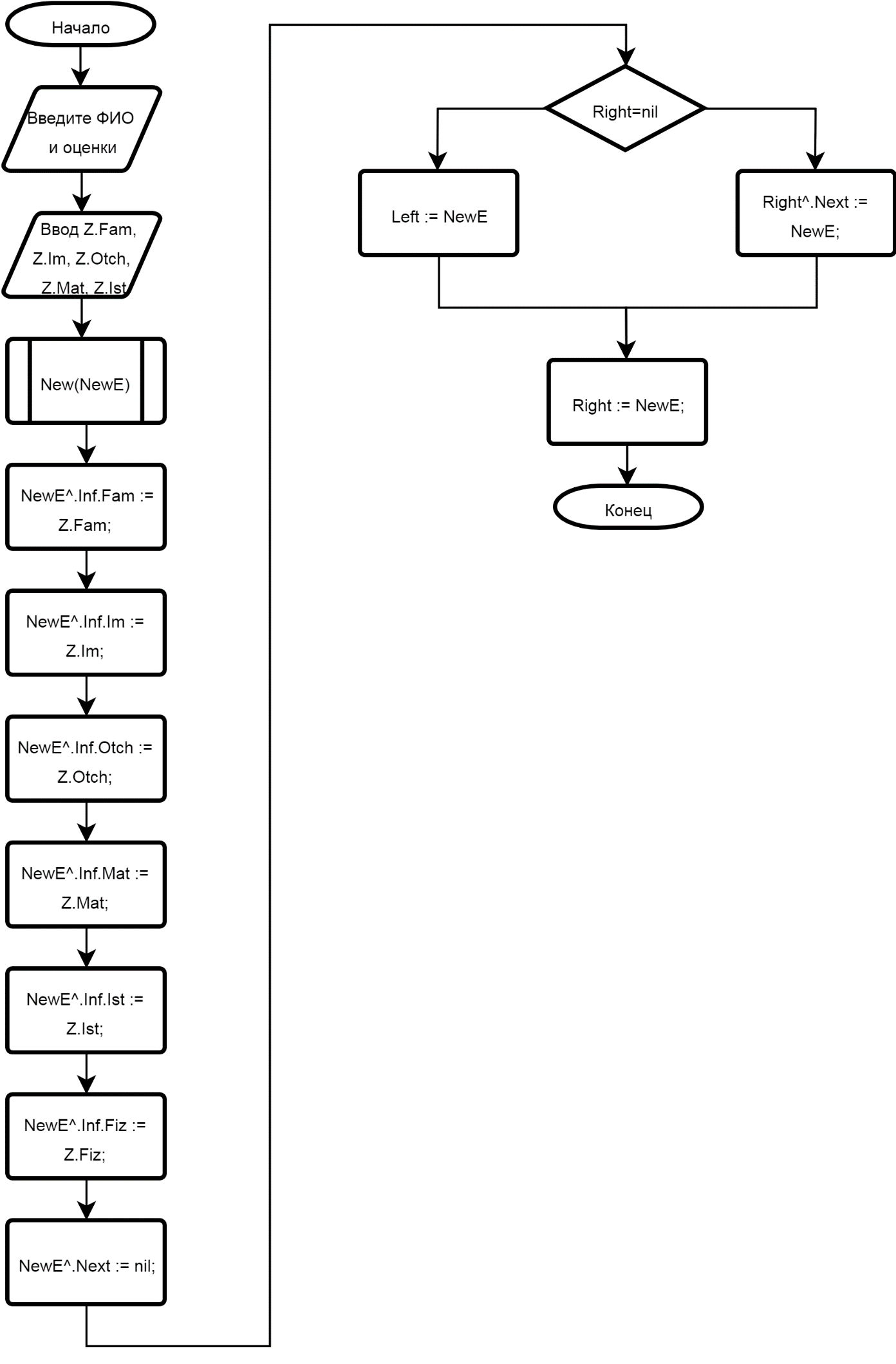


Рисунок 3 – Структурная схема процедуры добавления элемента

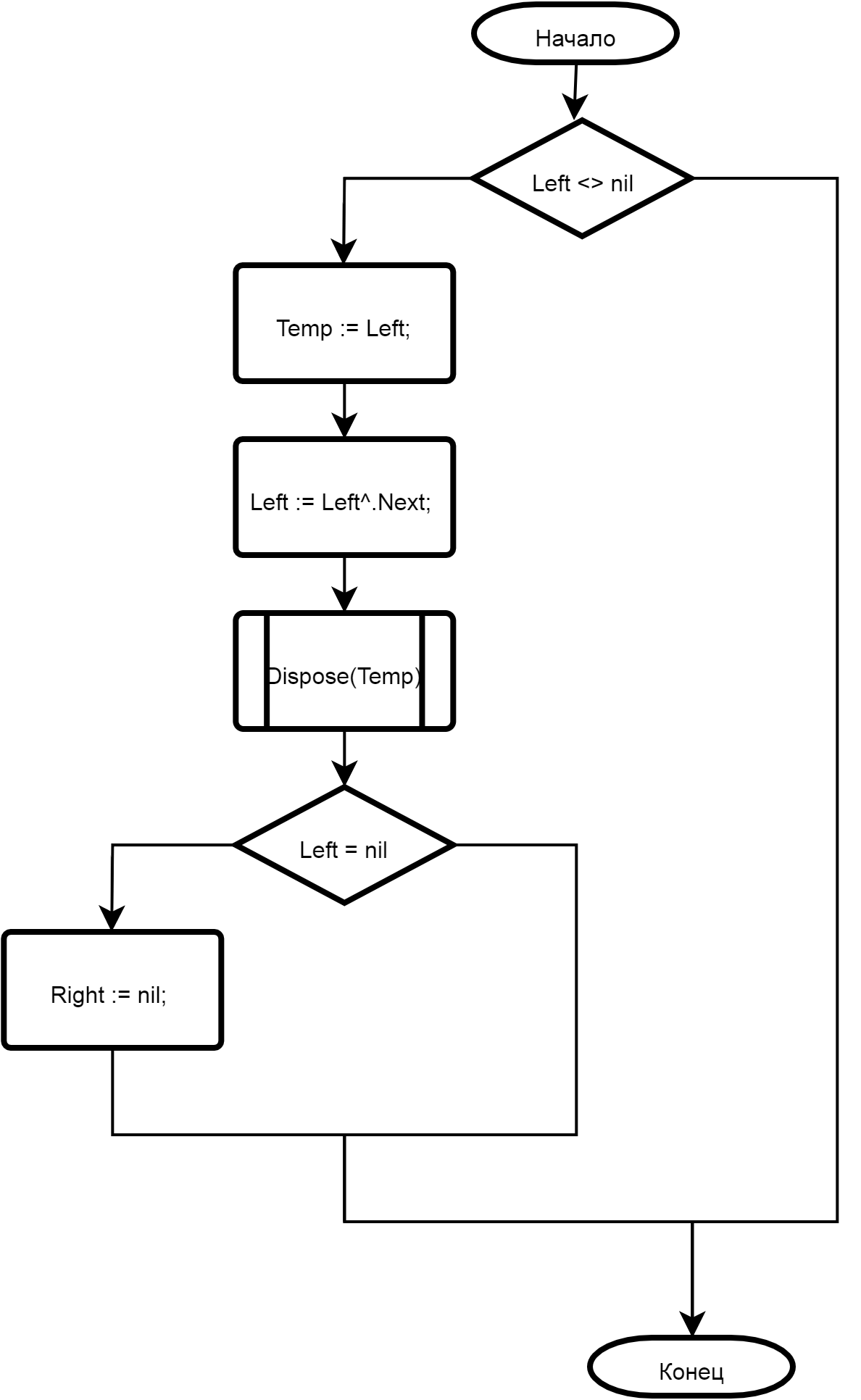


Рисунок 4 – Структурная схема процедуры удаления

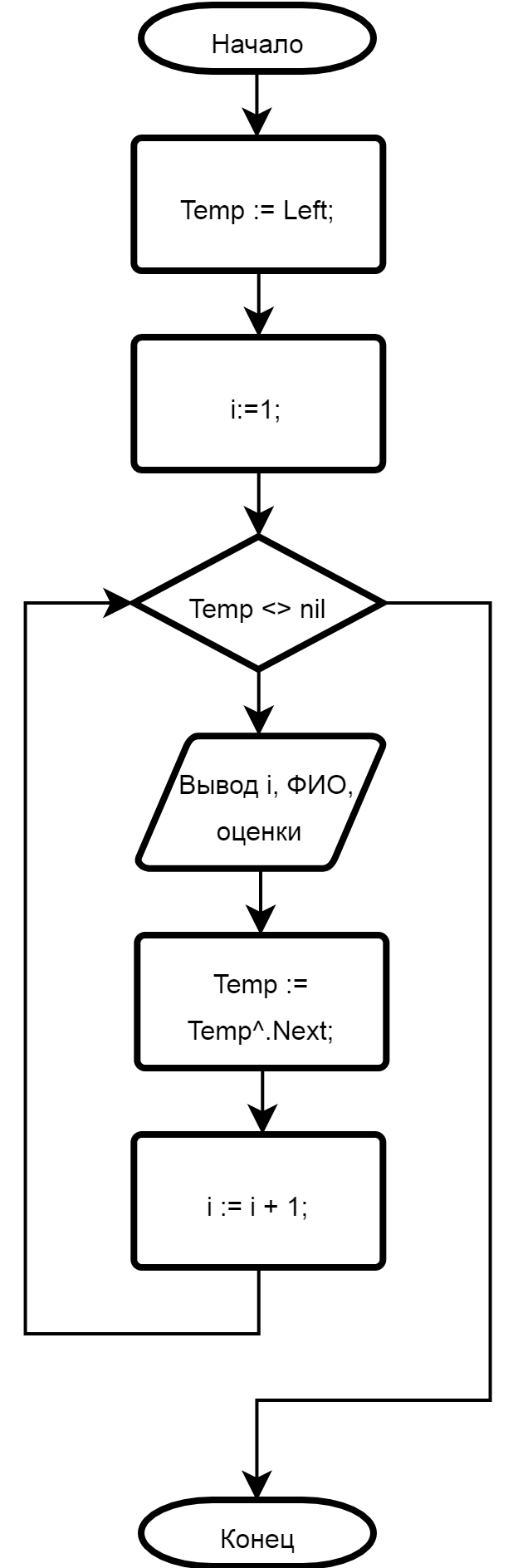


Рисунок 5 – Структурная схема процедуры просмотра списка

4 ТЕКСТ ПРОГРАММЫ

**Uses** Crt;

**type**

Data = **Record** {Описание записи о студенте}

Fam: string[15];

Im: string[10];

Otch: string[15];

Mat: string[3];

Ist: string[3];

Fiz: string[3];

**End**;

Ukaz = ^Query; {Описание указателя на элемент очереди}

Query = **Record**

Inf: Data;

Next: Ukaz

**End**;

**var**

NewE, Left, Right, Temp: Ukaz; {Указатели}

Z: Data; {Запись, добавляемая в очередь}

Key: Char; {номер пункта меню программы}

**procedure** Org; {Процедура организации очереди}

**begin**

Writeln('Выполняется процедура организации очереди');

Writeln('Для выхода из процедуры вводите символ \* ');

Writeln('======================================== ');

Writeln('Введите Фамилию: ');

Readln(Z.Fam);

Writeln('Введите Имя: ');

Readln(Z.Im);

Writeln('Введите Отчество: ');

Readln(Z.Otch);

Writeln('Введите оценку по Математике: ');

Readln(Z.Mat);

Writeln('Введите оценку по Истории: ');

Readln(Z.Ist);

Writeln('Введите оценку по Физики: ');

Readln(Z.Fiz);

**if** Z.Fam = '\*' **Then Exit**;

**New**(NewE); {Создание нового элемента}

NewE^.Inf.Fam := Z.Fam; {Заполнение его полей}

NewE^.Inf.Im := Z.Im;

NewE^.Inf.Otch := Z.Otch;

NewE^.Inf.Mat := Z.Mat;

NewE^.Inf.Ist := Z.Ist;

NewE^.Inf.Fiz := Z.Fiz;

NewE^.Next := nil;

Right := NewE; {Right - указатель хвоста очереди}

Left := NewE; {Left - указатель головы очереди}

**while** True **Do** {Повторение этих же действий}

**begin**

Writeln('Введите Фамилию: ');

Readln(Z.Fam);

Writeln('Введите Имя: ');

Readln(Z.Im);

Writeln('Введите Отчество: ');

Readln(Z.Otch);

Writeln('Введите оценку по Математике: ');

Readln(Z.Mat);

Writeln('Введите оценку по Истории: ');

Readln(Z.Ist);

Writeln('Введите оценку по Физики: ');

Readln(Z.Fiz);

**if** Z.Fam = '\*' **Then Exit**;

**New**(NewE);

NewE^.Inf.Fam := Z.Fam;

NewE^.Inf.Im := Z.Im;

NewE^.Inf.Otch := Z.Otch;

NewE^.Inf.Mat := Z.Mat;

NewE^.Inf.Ist := Z.Ist;

NewE^.Inf.Fiz := Z.Fiz;

NewE^.Next := nil;

Right^.Next := NewE; {Связь с предыдущим элементом}

Right := NewE {Перемещение указателя хвоста очереди}

**end**

**end**;

**procedure** Dob; {Добавление элемента в конец очереди}

**begin**

Writeln('Введите Фамилию: ');

Readln(Z.Fam);

Writeln('Введите Имя: ');

Readln(Z.Im);

Writeln('Введите Отчество: ');

Readln(Z.Otch);

Writeln('Введите оценку по Математике: ');

Readln(Z.Mat);

Writeln('Введите оценку по Истории: ');

Readln(Z.Ist);

Writeln('Введите оценку по Физики: ');

Readln(Z.Fiz);

**if** Z.Fam = '\*' **Then Exit**;

**New**(NewE);

NewE^.Inf.Fam := Z.Fam;

NewE^.Inf.Im := Z.Im;

NewE^.Inf.Otch := Z.Otch;

NewE^.Inf.Mat := Z.Mat;

NewE^.Inf.Ist := Z.Ist;

NewE^.Inf.Fiz := Z.Fiz;

NewE^.Next := nil;

**if** Right = nil **Then** {Если добавляется первый элемент, то}

Left := NewE {инициализировать указатель головы}

**Else** Right^.Next := NewE; {иначе добавить в конец очереди}

Right := NewE;

**end**;

**procedure** Udal;{Процедура исключения элемента}

**begin**

Writeln('Исключается головной элемент очереди');

Writeln('Нажмите клавишу Enter');

Readln;

**if** Left <> nil **Then** {Если очередь не пустая, то}

**begin**

Temp := Left; {запоминаем указатель на голову очереди}

Left := Left^.Next; {указатель головы смещаем на 2-ой элемент}

Dispose(Temp); {oсвобождаем память от головного элемента}

**if** Left = nil **Then** {Если удалили последний элемент, то}

Right := nil; {указатель на конец очереди равен Nil}

**end**

**end**;

**procedure** Prosmotr; {Процедура просмотра очереди}

**var**

i: integer; {Просмотр выполняется от головы к хвосту}

**begin**

Writeln('Очередь содержит следующие элементы:');

Temp := Left;

i := 1;

**while** Temp <> nil **Do**

**begin**

Writeln(i, ' ', Temp^.Inf.Fam, ' ', Temp^.Inf.Im, ' ', Temp^.Inf.Otch, ' ', Temp^.Inf.Mat, ' ', Temp^.Inf.Ist, ' ', Temp^.Inf.Fiz);

Temp := Temp^.Next;

i := i + 1;

**end**;

Writeln('Нажмите клавишу Enter');

Readln;

**end**;

{основная программа}

**begin**

Right := nil;

Left := nil;

**repeat**

ClrScr;

Writeln('1-Организация очереди');

Writeln('2-Добавление элемента в очередь');

Writeln('3-Удаление элемента из очереди');

Writeln('4-Просмотр очереди');

Writeln('5-Выход');

Writeln('-------------------------------');

Writeln('Нажмите клавишу от 1 до 5');

Key := ReadKey; {считывание кода нажатой клавиши}

**case** Key **Of** {вызов необходимой процедуры по номеру}

'1': Org;

'2': Dob;

'3': Udal;

'4': Prosmotr;

**End**

**until** Key = '5'

**end**.

5 ТЕСТИРОВАНИЕ

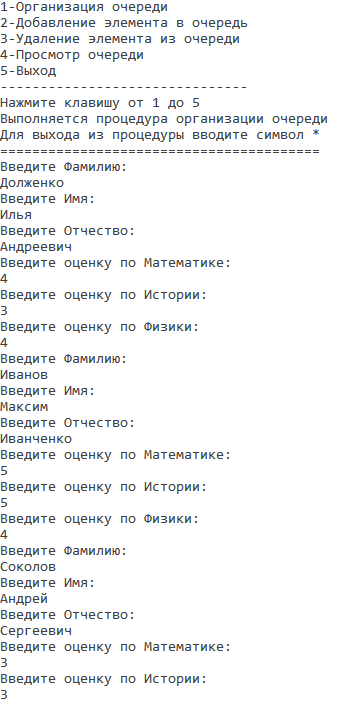


Рисунок 6 – Ввод данных

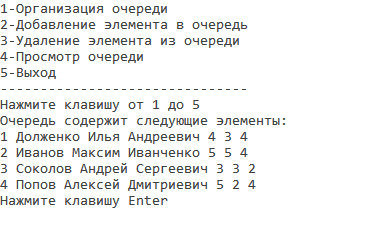


Рисунок 7 – Просмотр списка после ввода

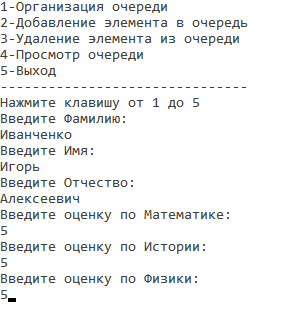


Рисунок 8 – Добавление нового элемента в очередь

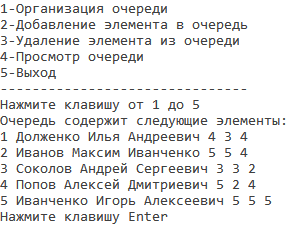


Рисунок 9 – Просмотр списка после добавления нового элемента

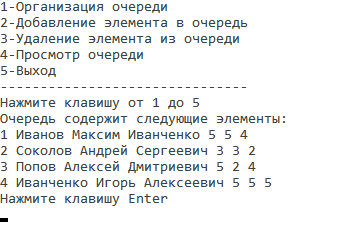


Рисунок 10 – Удаление элемента из очереди

ВЫВОД

В ходе выполнения лабораторной работы были получены навыки работы с ссылочным типом, динамической памятью и переменными ссылочного типа. Изучены типизированные и не типизированные указатели, их отличия и процедуры.

Изучены односвязные линейные списки, двусвязные линейные списки, односвязный циклический список, очередь и стек. Была написана программа, в которой была представлена таблица в виде линейного списка, написана процедура организации, добавления элемента в список, исключения элемента из списка, просмотра списка, а также процедура которая вставляет в конец списка новый элемент.