Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Алгоритмизация и программирование”

Лабораторная работа №8

“Программирование нелинейных структур данных”

Выполнил:

ст.гр. ИС/б-12

Долженко И.А.

Проверил:

Забаштанский А.К.

Севастополь

2017

1 ЦЕЛЬ РАБОТЫ

Исследование нелинейных структур данных и приобретение навыков разработки и отладки программ, использующих древовидные структуры. Исследование особенностей работы с поисковыми бинарными деревьями.

2 ПОСТАНОВКА ЗАДАЧИ

Представить приведенную в предыдущей работе таблицу в виде бинарного дерева. Написать процедуры создания и обхода дерева, а также процедуру, которая определяет уровень, на котором находится элемент Е в дереве Т. Значения полей и количество записей в таблице студент выбирает самостоятельно.

3 СТРУКТУРНАЯ СХЕМА

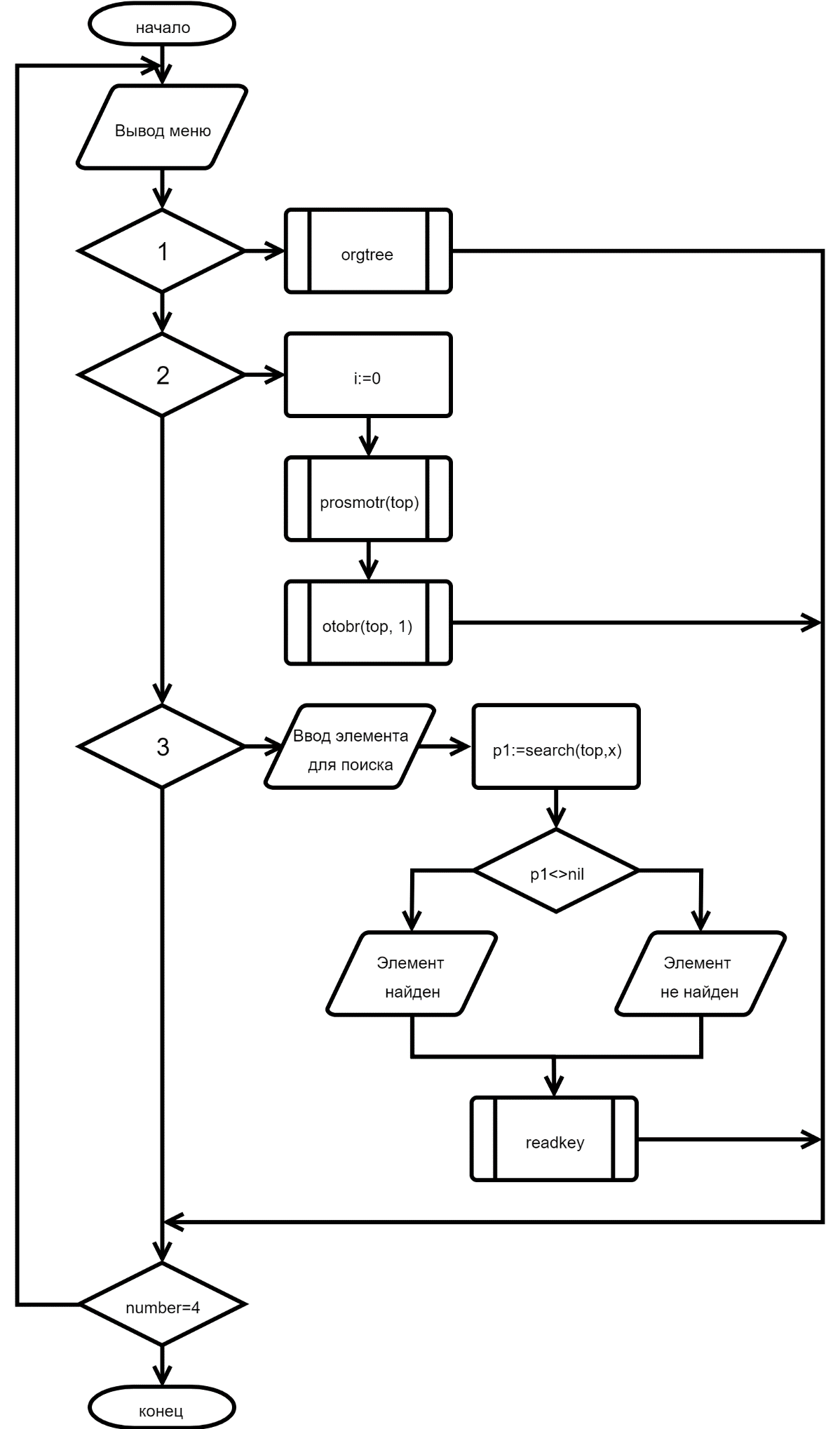


Рисунок 1 – Структурная схема основной программы

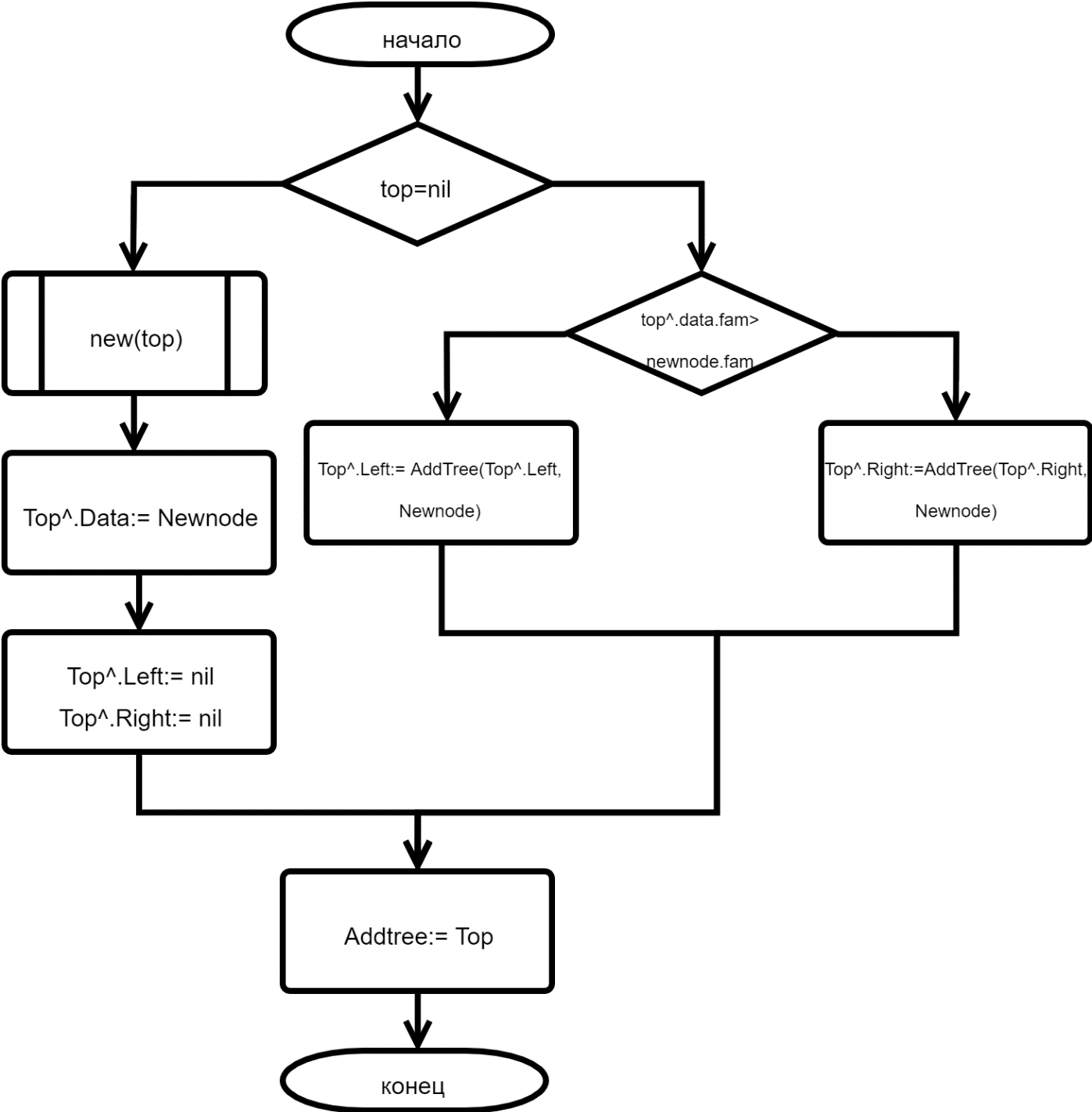


Рисунок 2 – Структурная схема функции добавления в дерево

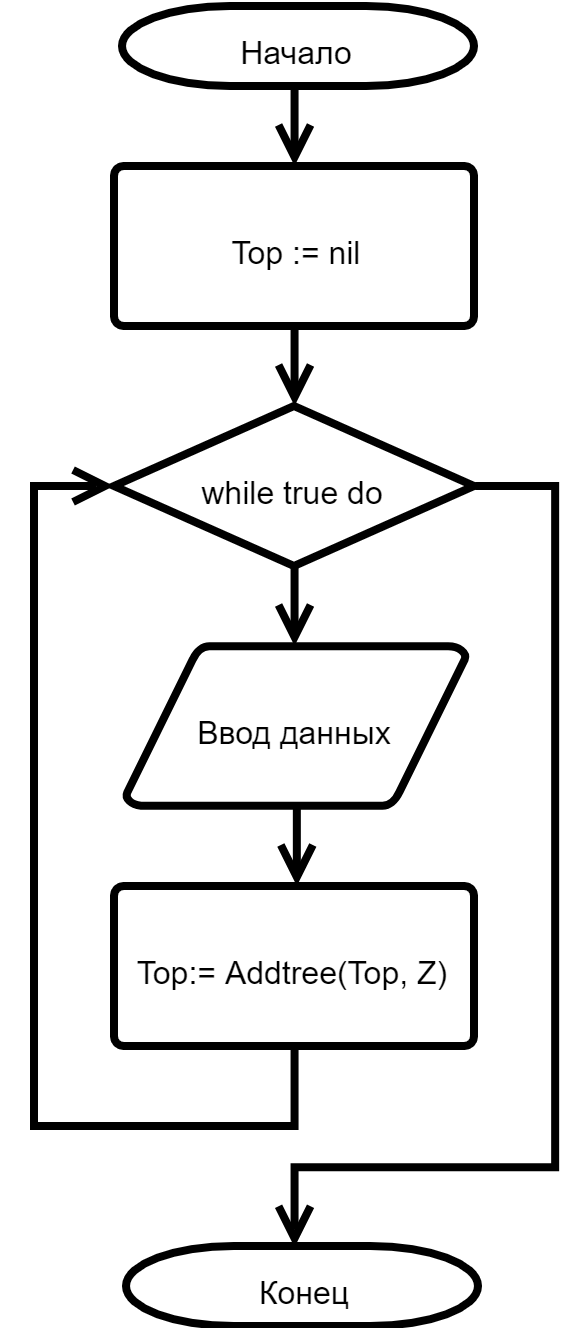
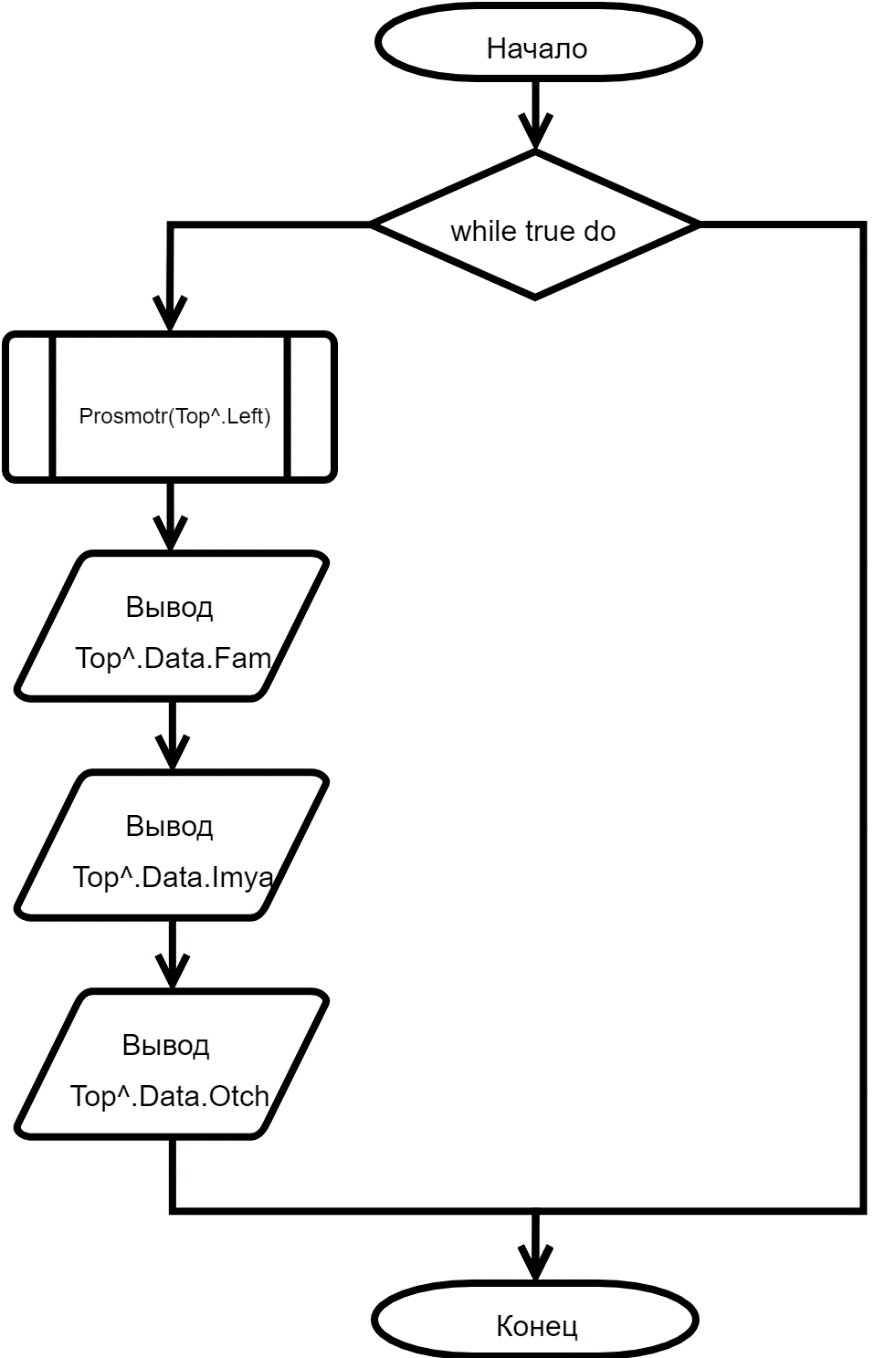


Рисунок 3 – Структурная схема

процедуры создания дерева Рисунок 4 – Структурная схема процедуры

просмотра дерева

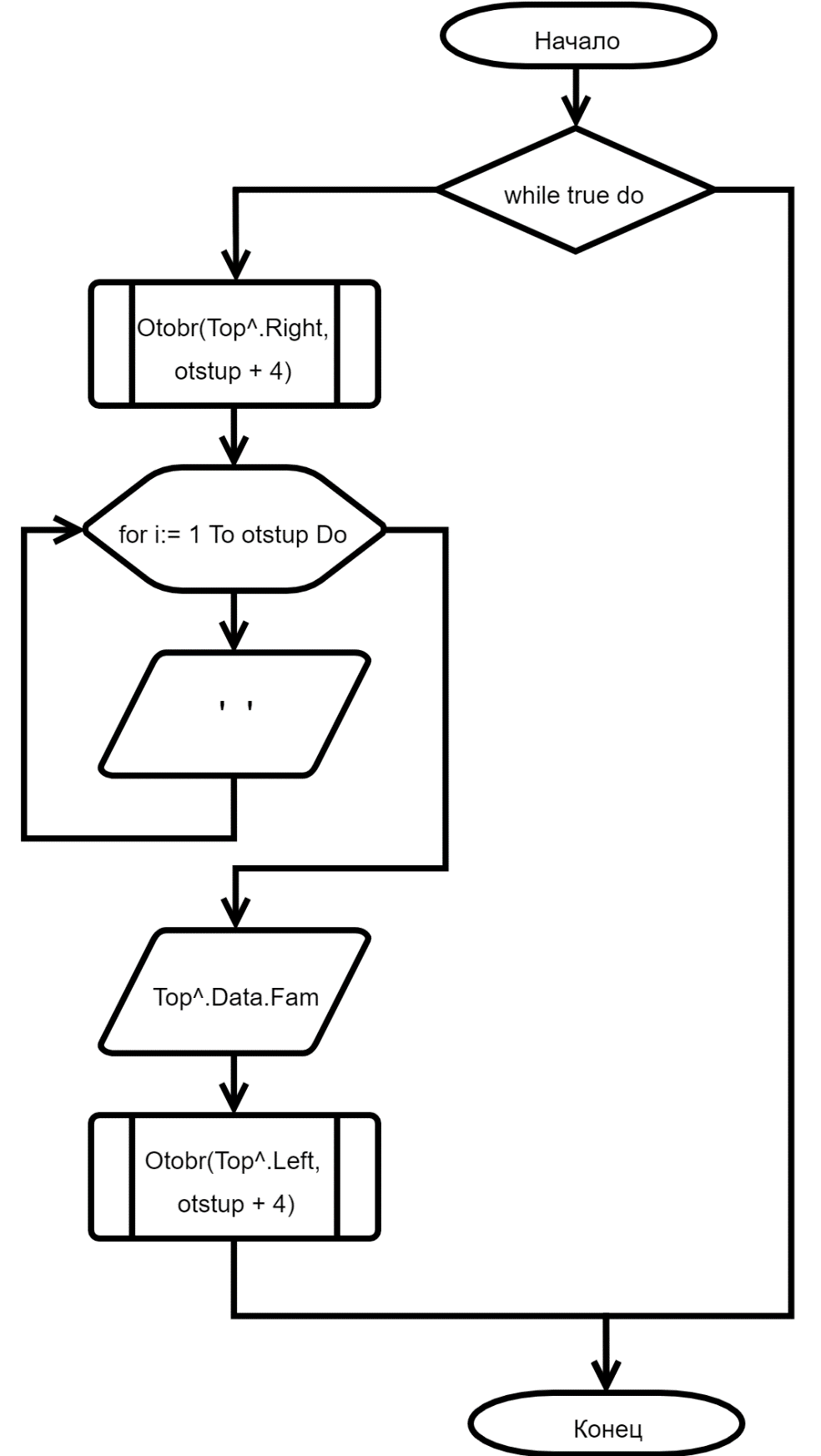


Рисунок 5 – Структурная схема процедуры вывода дерева

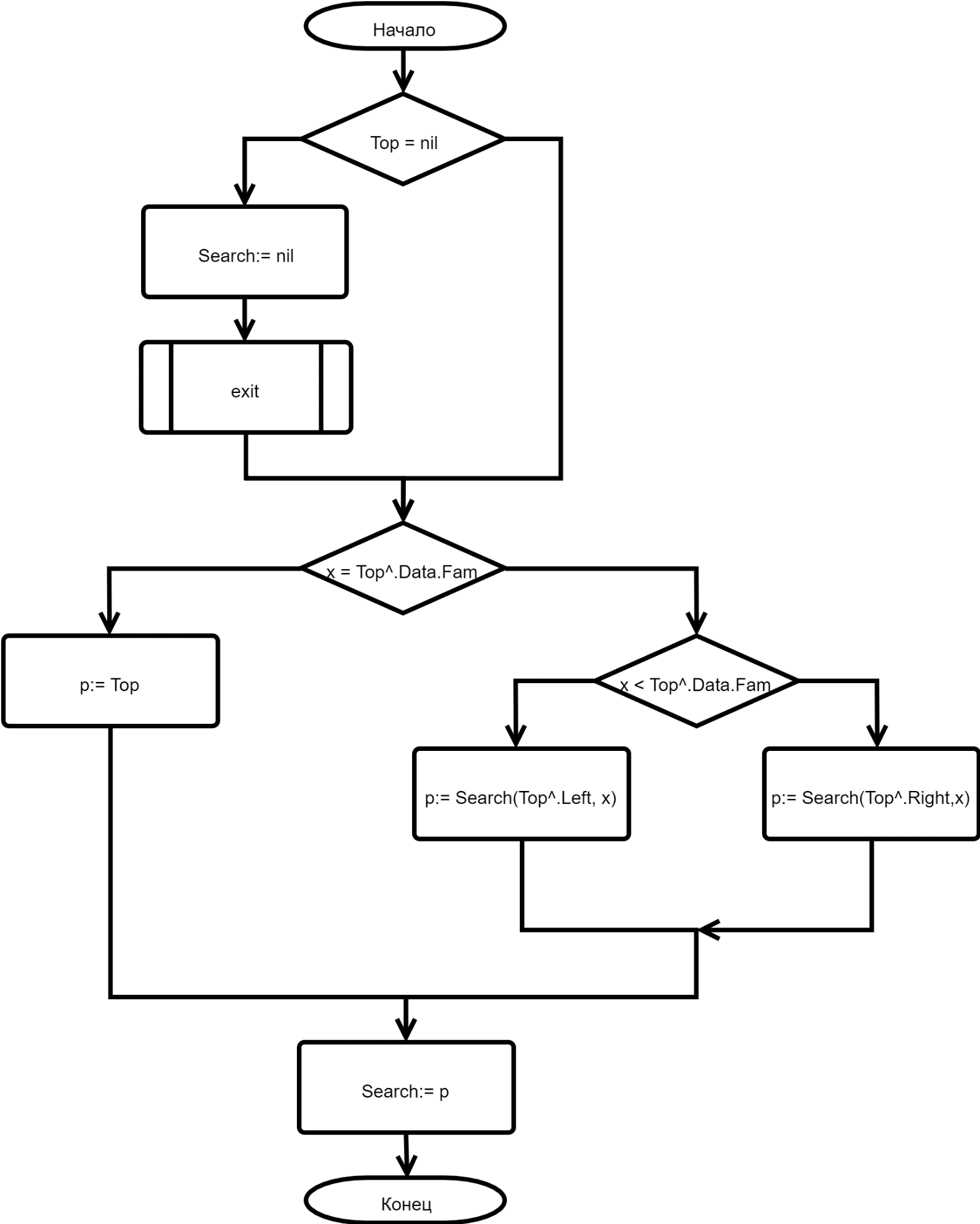


Рисунок 6 – Структурная схема функции поиска в дереве

4 ТЕКСТ ПРОГРАММЫ

**Uses** Crt;

**type**

Zap = **Record**

Fam: string[13];

Imya: string[10];

Otch: string[13];

Mat: string[3];

Ist: string[3];

Fiz: string[3];

**End**;

TreePtr = ^Tree; {Описание узла дерева}

Tree = **Record**

Data: Zap;

Left, Right: TreePtr;

**End**;

**var**

Top, p1: Treeptr;

Z: Zap;

i: Integer;

x: string;

Number: Integer;

**function** AddTree (Top: TreePtr; Newnode: Zap): TreePtr;

**begin**

**if** Top = nil **THEN** {Если дерево пустое то создаём его корень}

**begin**

**New**(Top); {Выделяем память}

Top^.Data:= Newnode; {Добавляем данные}

Top^.Left:= nil; {Зануляем указатели левого}

Top^.Right:= nil; {Зануляем указатели правого}

**end**

**ELSE**

**if** Top^.Data.Fam > Newnode.Fam **Then** Top^.Left:= AddTree(Top^.Left, Newnode) {если меньше корня то в левое поддерево}

**Else** Top^.Right:= AddTree(Top^.Right, Newnode); {если больше корня то в правое поддерево}

Addtree:= Top

**end**;

**procedure** OrgTree;

**begin**

Writeln('Выполняется процедура организации дерева');

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

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

Top := nil;

**while** True **Do**

**begin**

Writeln('Введите фамилию:'); Readln(Z.Fam);

Writeln('Введите имя:'); Readln(Z.Imya);

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 Физике:'); Readln(Z.Fiz);

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

Top:= Addtree(Top, Z);

**end**

**end**;

**procedure** Prosmotr (Top: TreePtr); {Просмотр узлом слева на право}

**begin**

**if** Top <> nil **Then**

**begin**

Prosmotr(Top^.Left);

Writeln(i, ' ', Top^.Data.Fam, ' ', Top^.Data.Imya, ' ', Top^.Data.Otch, ' ', Top^.Data.Mat, ' ', Top^.Data.Ist, ' ', Top^.Data.Fiz);

i:= i + 1;

Prosmotr(Top^.Right)

**end**;

**end**;

**procedure** Otobr(top: TreePtr; Otstup: Integer); {Вывод дерева}

**var**

i: Integer;

**begin**

**if** Top <> nil **Then**

**begin**

Otobr(Top^.Right, otstup + 4);

**for** i:= 1 **To** otstup **Do** Write(' ');

Writeln(Top^.Data.Fam);

Otobr(Top^.Left, otstup + 4);

**end**;

**end**;

**function** Search (Top: Treeptr; x: string): Treeptr;

**var**

p: Treeptr;

**begin**

**if** Top = nil **then**

**begin**

Search:= nil;

**exit**;

**end**;

**if** x = Top^.Data.Fam **then** {если искомый элемент равен корню дерева то}

p:= Top {запоминаем его адрес}

**else**

**if** x < Top^.Data.Fam **then**

p:= Search(Top^.Left, x) {то ищем его в левом поддереве}

**else**

p:= Search(Top^.Right, x); {ищем в правом поддереве}

Search:= p; {присваиваем переменной с именем функции результат работы}

**end**;

{==============основная программа================}

**begin**

**repeat**

ClrScr;

Writeln('1-Создание дерева');

Writeln('2-Вывод дерева');

Writeln('3-Поиск по Фамилии');

Writeln('4-Выход');

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

Writeln('Введите номер пункта меню');

Readln(Number);

**case** Number **Of**

1: OrgTree;

2:

**begin**

Writeln('Выполняется процедура просмотра дерева');

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

i:= 0;

Prosmotr(Top);

Otobr(Top, 1);

Writeln('Нажмите клавишу Enter');

Readln

**end**;

3:

**begin**

Writeln('Введите элемент для поиска:');

Readln(x);

p1:= Search(Top, x);

**if** p1 <> nil **then** Writeln('Найден.')

**else** Writeln('Такого элемента нет.');

Readkey;

**end**;

**End**;

**until** Number = 4;

**end**.

5 ТЕСТИРОВАНИЕ



Рисунок 7 – Добавление данных в очередь

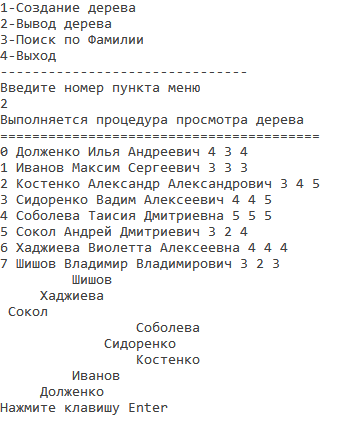


Рисунок 8 – Вывод дерева

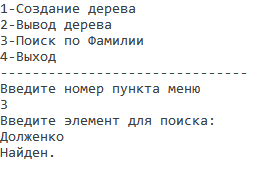


Рисунок 9 – Поиск в дереве по фамилии, элемент найден

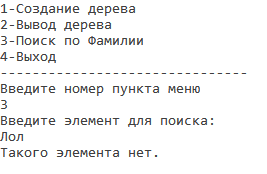


Рисунок 10 – Поиск в дереве по фамилии, элемент не найден

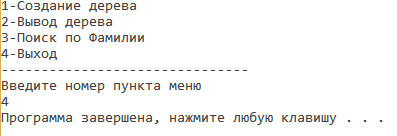


Рисунок 11 – Выход из программы

ВЫВОД

В ходе выполнения лабораторной работы были получены навыки работы с бинарными деревьями и алгоритмами их обработки. Рассмотрены функции поиска и добавления элемента в дерево. Изучены процедуры обхода дерева. Была написана программа, в которой была представлена таблица в виде бинарного дерева, написана процедура создания и обхода дерева, а также процедура которая определяет уровень, на котором находится элемент в дереве.