Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Алгоритмизация и программирование”

Лабораторная работа №8

“ПРОГРАММИРОВАНИЕ НЕЛИНЕЙНЫХ СТРУКТУР ДАННЫХ”

Вариант 3

Выполнил:

ст.гр. ИС/б-12

Волобуев Ю.С.

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

Процедуру, которая присваивает параметру Е элемент из самого левого листа непустого дерева Т (Таблица 1).

 (Таблица 1)

3 СТРУКТУРНАЯ СХЕМА АЛГОРИТМА

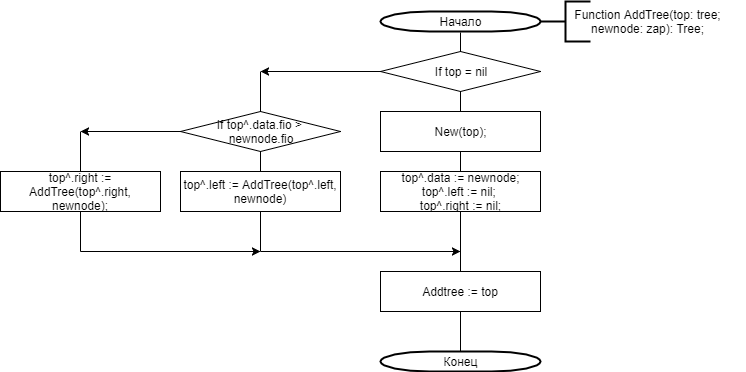


Рисунок 1 – Структурная схема функции добавления листа к дереву.

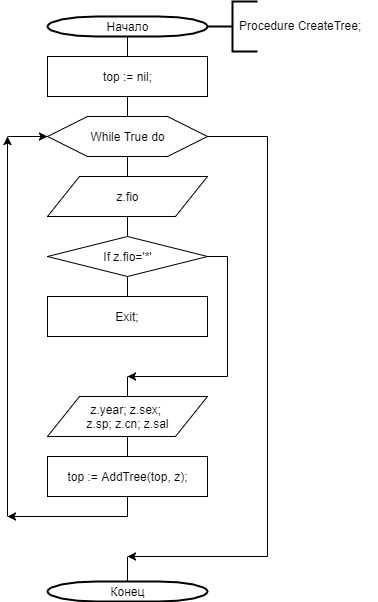
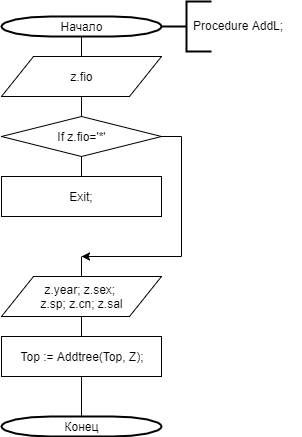


Рисунок 2 – Структурная схема Рисунок 3 – Структурная схема процедуры

создания дерева добавления листа в дерево

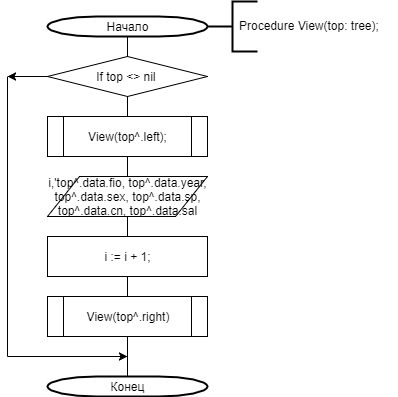
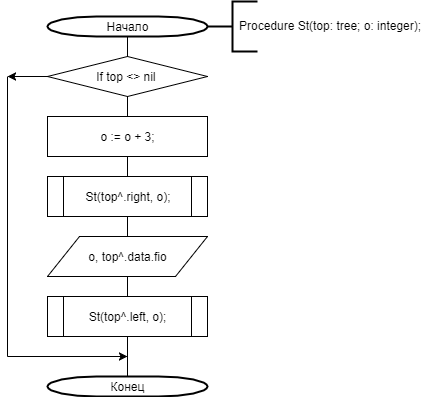


Рисунок 4 – Структурная схема Рисунок 5 – Структурная схема процедуры

процедуры просмотра узлов просмотра структуры дерева

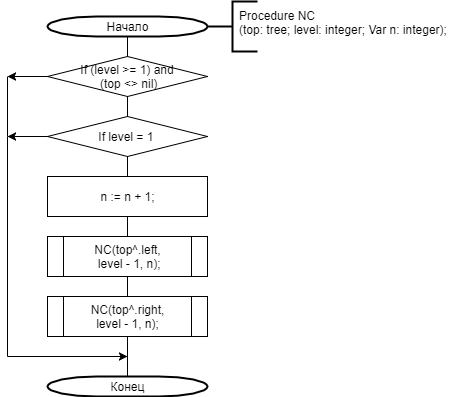


Рисунок 6 – Структурная схема процедуры подсчета количества вершин на определенном уровне

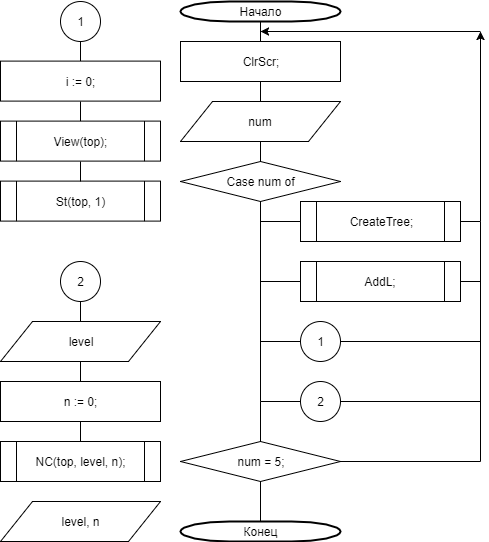


Рисунок 7 – Структурная схема алгоритма программы

4 ТЕКСТ ПРОГРАММЫ

**Program** LAB8;

**Uses** Crt;

**Type**

zap = **Record**

fio:string[20];

year:string[12];

sex:char;

sp:string[15];

cn:byte;

sal:integer;

**End**;

tree = ^tr;

tr = **Record**

data: Zap;

left, right: tree

**End**;

**Var**

top: Tree;

z: zap;

level, n, i: integer;

num: integer;

**Function** AddTree(top: tree; newnode: zap): Tree;

**Begin**

**If** top = nil **then**

**Begin**

**New**(top);

top^.data := newnode;

top^.left := nil;

top^.right := nil;

**End**

**else**

**If** top^.data.fio > newnode.fio **then**

top^.left := AddTree(top^.left, newnode)

**else**

top^.right := AddTree(top^.right, newnode);

Addtree := top

**End**;

{---------------------------------------------------}

**Procedure** CreateTree;

**Begin**

Writeln('Выполняется процедура организации дерева');

Writeln('Для выхода из процедуры нажмите "\*" ');

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

top := nil;

**While** True **do**

**Begin**

Writeln('Введите ФИО сотрудника:');

Readln(z.fio);

**If** z.fio='\*' **then Exit**;

Writeln('Введите дату рождения и пол сотрудника:');

Readln(z.year);

Readln(z.sex);

Writeln('Введите семейное состояние и количество детей сотрудника сотрудника:');

Readln(z.sp);

Readln(z.cn);

Writeln('Введите оклад сотрудника:');

Readln(z.sal);

top := AddTree(top, z);

**End**

**End**;

{--------------------------------------------------------}

**Procedure** AddL;

**Begin**

Writeln('Выполняется процедура добавления листа');

Writeln('Для выхода из процедуры нажмите "\*" ');

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

Writeln('Введите ФИО сотрудника:');

Readln(z.fio);

**If** z.fio='\*' **then Exit**;

Writeln('Введите дату рождения и пол сотрудника:');

Readln(z.year);

Readln(z.sex);

Writeln('Введите семейное состояние и количество детей сотрудника сотрудника:');

Readln(z.sp);

Readln(z.cn);

Writeln('Введите оклад сотрудника:');

Readln(z.sal);

Top := Addtree(Top, Z);

**End**;

{----------------------------------------------------------}

**Procedure** View(top: tree);

**Begin**

**If** top <> nil **then**

**Begin**

View(top^.left);

Writeln(i,' ',top^.data.fio,' ',top^.data.year,' ',top^.data.sex,' ',top^.data.sp,' ',top^.data.cn,' ',top^.data.sal);

i := i + 1;

View(top^.right)

**End**;

**End**;

{---------------------------------------------------------------}

**Procedure** St(top: tree; o: integer);

**Begin**

**If** top <> nil **then**

**Begin**

o := o + 3;

St(top^.right, o);

Writeln(' ':o, top^.data.fio);

St(top^.left, o);

**End**

**End**;

{---------------------------------------------------------------}

**Procedure** NC(top: tree; level: integer; **Var** n: integer);

**Begin**

**If** (level >= 1) **and** (top <> nil) **then**

**Begin**

**If** level = 1 **then** n := n + 1;

NC(top^.left, level - 1, n);

NC(top^.right, level - 1, n);

**End**

**End**;

{---------------------------------------------------------------}

**Begin**

**Repeat**

ClrScr;

Writeln('1-Организация двоичного дерева');

Writeln('2-Добавление листа к дереву');

Writeln('3-Просмотр дерева');

Writeln('4-Подсчет количества вершин на n-oм уровне');

Writeln('5-Выход');

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

Writeln('Выберите необходимый пункт меню');

Readln(num);

**Case** num **of**

1: CreateTree;

2: AddL;

3:**Begin**

Writeln('Выполняется процедура просмотра дерева');

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

i := 0;

View(top);

St(top, 1);

Writeln('Нажмите Enter');

Readln

**End**;

4:**Begin**

Writeln('Подсчет количества вершин на n-ном уровне');

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

Write('Введите значение уровня:');

Read(level);

n := 0;

NC(top, level, n);

Writeln;

Writeln('На уровне ', level, ' находится ', n, ' вершин');

Writeln('Нажмите клавишу Enter');

ReadKey

**End**;

**End**;

**Until** num = 5;

**End**.

5 ТЕСТОВЫЕ ПРИМЕРЫ

Вводим данные о сотрудниках (ФИО, дата рождения, пол, семейное положение, количество детей и оклад). Проверяем процедуру вывода данных на экран (Рисунок 6). Проверяем процедуры добавления нового сотрудника (Рисунок 7) и удаления записи (Рисунок 8).

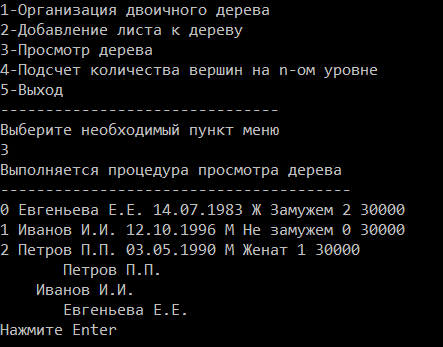


Рисунок 6 – Вывод введенных данных

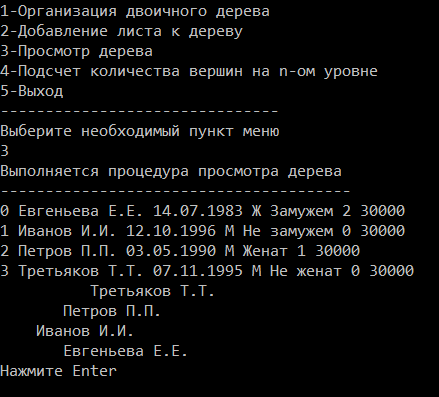


Рисунок 7 – Добавление нового листа

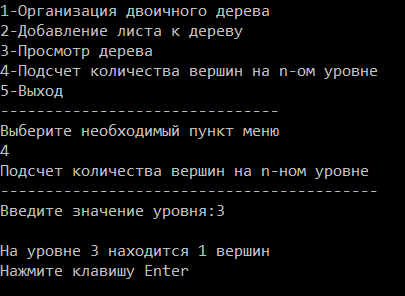


Рисунок 8 – Процедура подсчета количества вершин на n-ном уровне

ВЫВОД

В ходе лабораторной работы были получены навыки работы с бинарными деревьями. Были получены навыки работы с поисковыми деревьями. Была написана программа и построена структурная схема алгоритма. Были проведены тесты каждой процедуры программы.