Министерство образования и науки Российской Федерации

Севастопольский Государственный Университет

Кафедра ИС

ОТЧЕТ

по лабораторной работе №2

ПРОГРАММИРОВАНИЕ НЕЛИНЕЙНЫХ

СТРУКТУР ДАННЫХ

Выполнил: ст. гр. ИТб-11

Куркчи А.Э.

Проверил:

Забаштанский А.К.

Севастополь

2014

# 1. Цель

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

2. Постановка задач

Представить приведенную в предыдущей работе таблицу в виде бинарного дерева. Написать процедуры создания и обхода дерева, а также одну из процедур или функций, приведенных ниже. Значения полей и количество записей в таблице студент выбирает самостоятельно. Программа должна сохранять дерево в файле и создавать его заново при её повторном запуске.

Вариант №2

Таблица 3.2. Процедуру, которая присваивает параметру Е элемент из самого левого листа непустого дерева Т.

3. Структурная схема

На рисунке 1 изображена схема алгоритма работы программы.

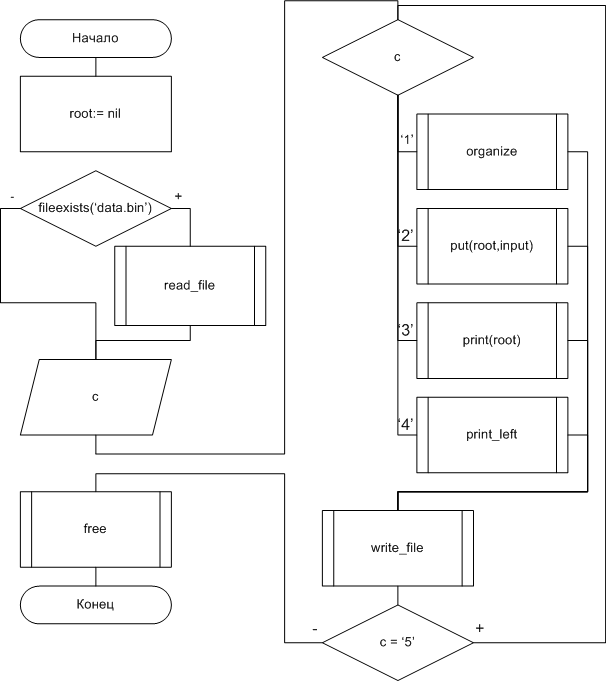
****

Рисунок 1 – структурная схема

На рисунке 2 изображена схема алгоритма работы процедуры free.

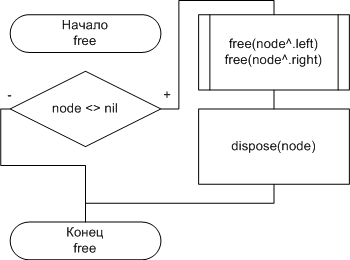
****

Рисунок 2 – структурная схема процедуры free

На рисунке 3 изображена схема алгоритма работы функции input.

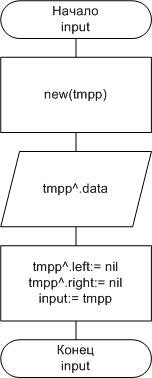
****

Рисунок 3 – структурная схема функции input

На рисунке 4 изображена схема алгоритма работы процедуры organize.

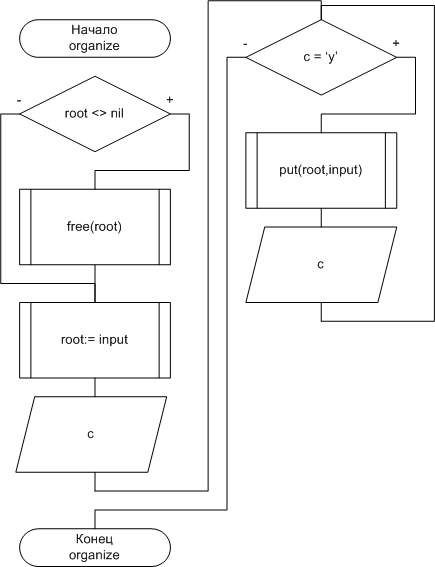
****

Рисунок 4 – структурная схема процедуры organize

На рисунке 5 изображена схема алгоритма работы процедуры print.

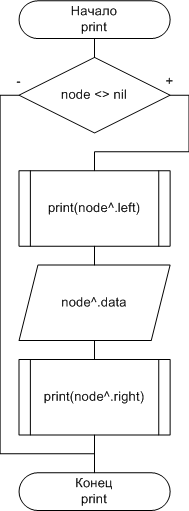
****

Рисунок 5 – структурная схема процедуры print

На рисунке 6 изображена схема алгоритма работы процедуры print\_left.

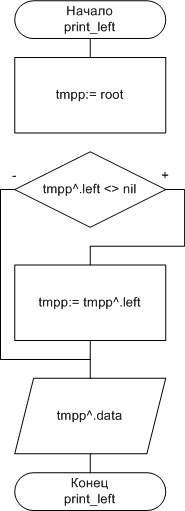
****

Рисунок 6 – структурная схема процедуры print\_left

На рисунке 7 изображена схема алгоритма работы процедуры put.

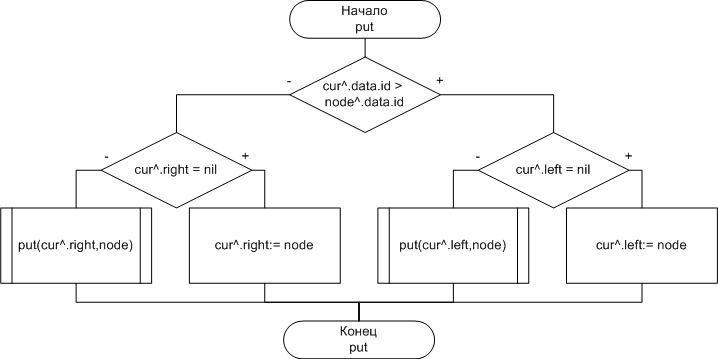
****

Рисунок 7 – структурная схема процедуры put

На рисунке 8 изображена схема алгоритма работы процедуры read\_file.

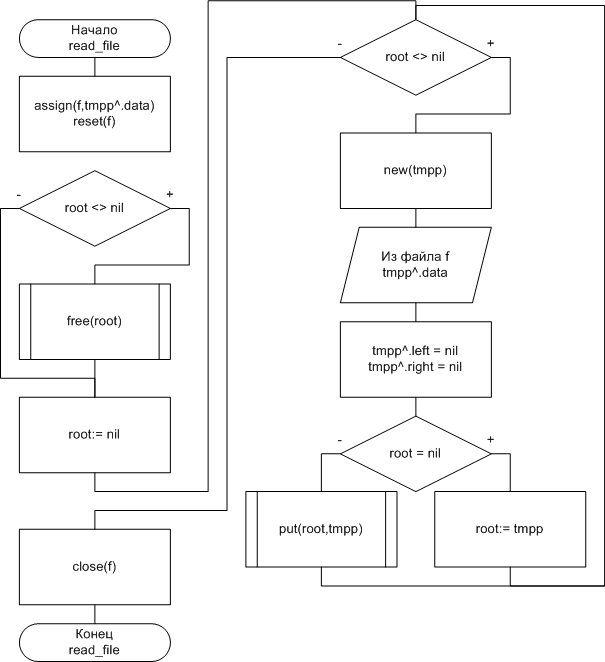
****

Рисунок 8 – структурная схема процедуры read\_file

На рисунке 9 изображена схема алгоритма работы процедуры write\_file.

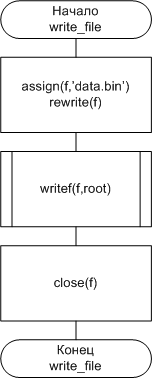
****

Рисунок 9 – структурная схема процедуры write\_file

На рисунке 10 изображена схема алгоритма работы процедуры writef.

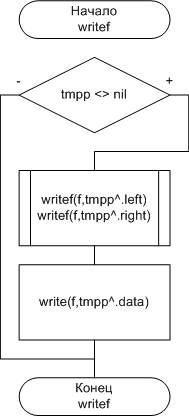
****

Рисунок 10 – структурная схема процедуры writef 4. Текст программы

**program** lab02;

**uses** crt;

**type**

TNodePointer=^TNode;

TData=**record**

id:integer;

origin,destination:string[16];

time\_departure,time\_arrival:string[5];

price:integer;

**end**;

TNode=**record**

data:TData;

left,right:TNodePointer;

**end**;

**var** root:TNodePointer;

**procedure** free(node:TNodePointer);

**begin**

**if**(node <> nil) **then begin**

free(node^.left);

free(node^.right);

dispose(node);

**end**;

**end**;

**procedure** put(cur,node:TNodePointer);

**begin**

**if**(cur^.data.id > node^.data.id) **then begin**

**if**(cur^.left = nil) **then**

cur^.left:= node

**else**

put(cur^.left,node);

**end else begin**

**if**(cur^.right = nil) **then**

cur^.right:= node

**else**

put(cur^.right,node);

**end**;

**end**;

**procedure** read\_file;

**var** f:**file of** TData;

tmpp:TNodePointer;

**begin**

assign(f,'data.bin');

reset(f);

**if**(root <> nil) **then**

free(root);

root:= nil;

**while**(**not**(EOF(f))) **do begin**

**new**(tmpp);

read(f,tmpp^.data);

tmpp^.left:= nil;

tmpp^.right:= nil;

**if**(root = nil) **then**

root:= tmpp

**else**

put(root,tmpp);

**end**;

close(f);

**end**;

**procedure** writef(f:**file of** TData;tmpp:TNodePointer);

**begin**

**if**(tmpp <> nil) **then begin**

writef(f,tmpp^.left);

writef(f,tmpp^.right);

write(f,tmpp^.data);

**end**;

**end**;

**procedure** write\_file;

**var** f:**file of** TData;

**begin**

assign(f,'data.bin');

rewrite(f);

writef(f,root);

close(f);

**end**;

**function** input:TNodePointer;

**var** tmpp:TNodePointer;

**begin**

**new**(tmpp);

write('Введите номер поезда: ');

readln(tmpp^.data.id);

write('Введите пункт отправления: ');

readln(tmpp^.data.origin);

write('Введите пункт назначения: ');

readln(tmpp^.data.destination);

write('Введите время отправления: ');

readln(tmpp^.data.time\_departure);

write('Введите время прибытия: ');

readln(tmpp^.data.time\_arrival);

write('Введите стоимость билета: ');

readln(tmpp^.data.price);

tmpp^.left:= nil;

tmpp^.right:= nil;

input:= tmpp;

**end**;

**procedure** organize;

**var** c:char;

**begin**

**if**(root <> nil) **then**

free(root);

writeln('Организация рассписания');

root:= input;

writeln('Коренной элемент добавлен');

writeln('Продолжить? (y/n)');

readln(c);

**while**(c = 'y') **do begin**

clrscr;

put(root,input);

writeln('Продолжить? (y/n)');

readln(c);

**end**;

**end**;

**procedure** print\_left;

**var** tmpp:TNodePointer;

**begin**

tmpp:= root;

**while**(tmpp^.left <> nil) **do**

tmpp:= tmpp^.left;

writeln('+-------+-------------------+-------------------+-------------+-------------+----------+');

writeln('| Номер | Пункт отправления | Пункт назначения | Отправление | Прибытие | Цена |');

writeln('+-------+-------------------+-------------------+-------------+-------------+----------+');

**with** tmpp^.data **do**

writeln('|',id:7,'|',origin:19,'|',destination:19,'|',time\_departure:13,'|',time\_arrival:13,'|',price:10,'|');

writeln('+-------+-------------------+-------------------+-------------+-------------+----------+');

writeln('Для продолжения нажмите клавишу Enter');

readln;

**end**;

**procedure** print(node:TNodePointer);

**begin**

**if**(node = root) **then begin**

writeln('+-------+-------------------+-------------------+-------------+-------------+----------+');

writeln('| Номер | Пункт отправления | Пункт назначения | Отправление | Прибытие | Цена |');

writeln('+-------+-------------------+-------------------+-------------+-------------+----------+');

**end**;

**if**(node <> nil) **then begin**

print(node^.left);

**with** node^.data **do**

writeln('|',id:7,'|',origin:19,'|',destination:19,'|',time\_departure:13,'|',time\_arrival:13,'|',price:10,'|');

writeln('+-------+-------------------+-------------------+-------------+-------------+----------+');

print(node^.right);

**end**;

**if**(node = root) **then begin**

writeln('Для продолжения нажмите клавишу Enter');

readln;

**end**;

**end**;

**var** c:char;

**begin**

root:= nil;

**if**(fileexists('data.bin')) **then**

read\_file;

**repeat**

clrscr;

writeln('Выберите действие:');

writeln('1 - Организация');

writeln('2 - Добавление');

writeln('3 - Печать рассписания');

writeln('4 - Печать самого левого листа');

writeln('5 - Выход');

readln(c);

clrscr;

**case** c **of**

'1':organize;

'2':put(root,input);

'3':print(root);

'4':print\_left;

**end**;

write\_file;

**until** c = '5';

free(root);

**end**.

5. Тестовые примеры

На рисунке 11 представлено меню программы

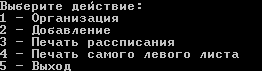


Рисунок 11 – меню программы

На рисунке 12 представлен результат работы пункта меню 3 «Печать расписания»

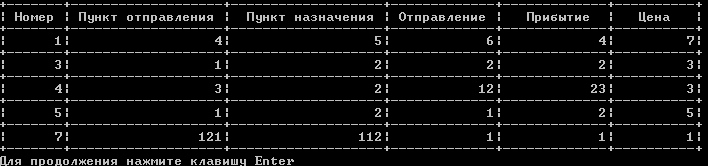


Рисунок 12 – расписание

Вывод

В ходе лабораторной работы были исследованы нелинейные структуры данных и приобретены навыки разработки и отладки программ, использующих древовидные структуры. Исследование особенностей работы с поисковыми бинарными деревьями.