Министерство образования и науки Российской Федерации

Севастопольский Государственный Университет

Кафедра ИС

ОТЧЕТ

по лабораторной работе №3

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕ

ОБЪЕКТОВ И МОДУЛЕЙ

Выполнил: ст. гр. ИТб-11

Куркчи А.Э.

Проверил:

Забаштанский А.К.

Севастополь

2014

# 1. Цель

Изучение базовых понятий объектно-ориентированного программирования, приобретение навыков разработки и отладки программ, использующих модули и объекты. Исследование особенностей поисковых бинарных деревьев, представленных в виде объекта.

2. Постановка задач

Создать модуль, содержащий описание объекта, который представляет бинарное дерево в соответствии с вариантом из предыдущей лабораторной работы. Объект должен обладать возможностью добавления новых элементов, удаления существующих, поиска элемента по ключу, обхода дерева, а также выполнять дополнительные операции в соответствии с вариантом задания.

Написать программу для представления таблицы, заданной в предыдущей лабораторной работе, в виде бинарного дерева. Программа должна содержать меню, позволяющее проверить все методы объекта. Предусмотреть возможность ввода данных из файла и с клавиатуры.

3. Структурная схема

На рисунке 1 изображена схема алгоритма работы программы.

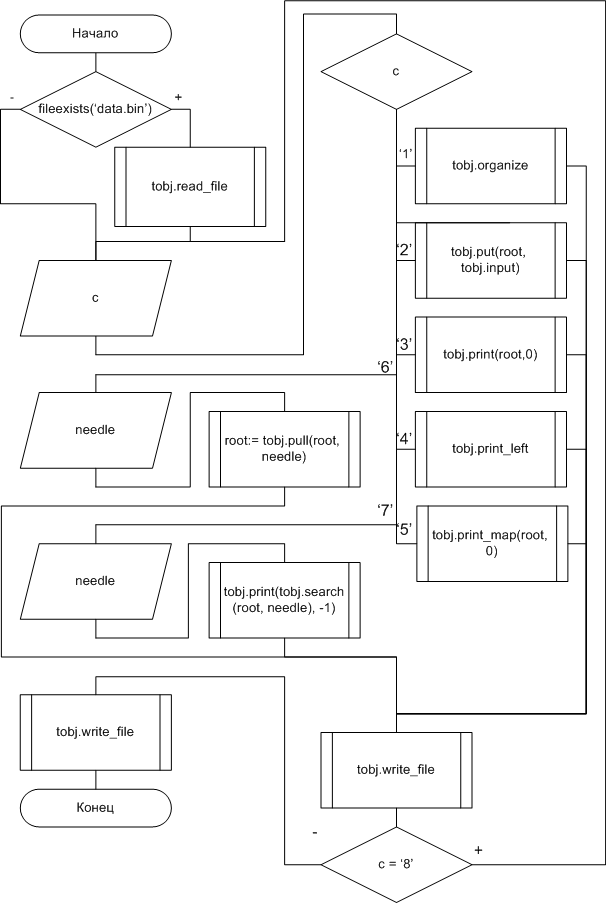
****

Рисунок 1 – структурная схема

На рисунке 2 изображена схема алгоритма работы процедуры tree.free.

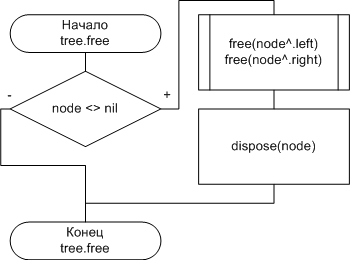
****

Рисунок 2 – структурная схема процедуры tree.free

На рисунке 3 изображена схема алгоритма работы функции tree.input.

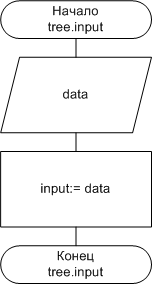
****

Рисунок 3 – структурная схема функции tree.input

На рисунке 4 изображена схема алгоритма работы процедуры tree.organize.

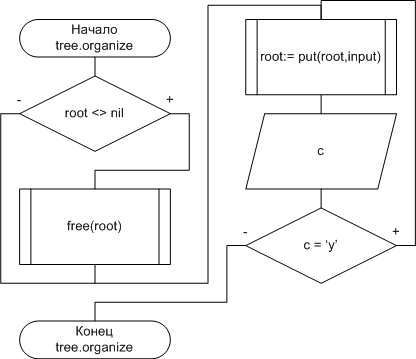
****

Рисунок 4 – структурная схема процедуры tree.organize

На рисунке 5 изображена схема алгоритма работы процедуры tree.print.

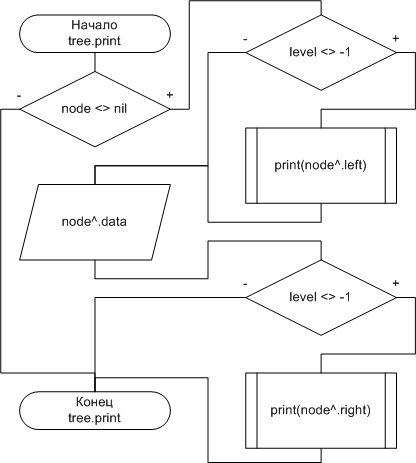
****

Рисунок 5 – структурная схема процедуры tree.print

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

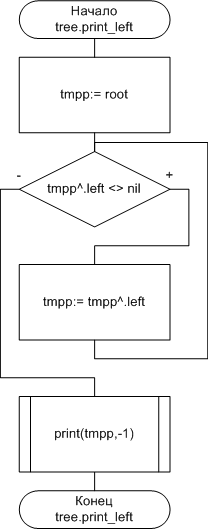
****

Рисунок 6 – структурная схема процедуры tree.print\_left

На рисунке 7 изображена схема алгоритма работы процедуры tree.put.

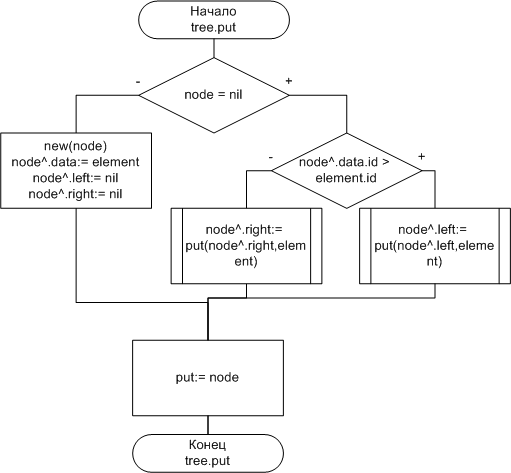
****

Рисунок 7 – структурная схема процедуры tree.put

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

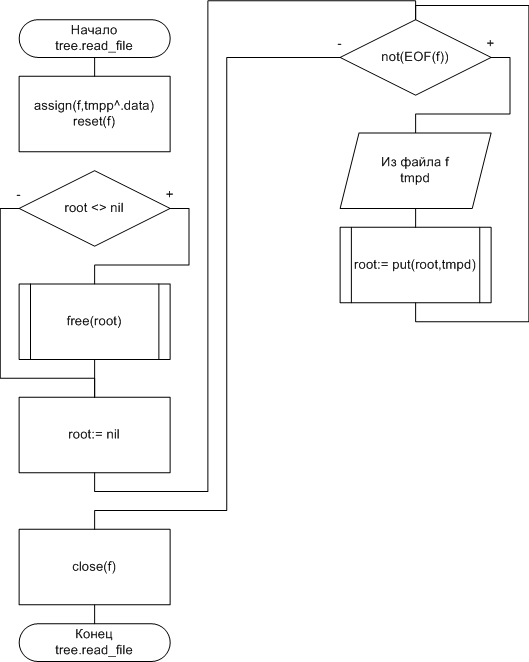
****

Рисунок 8 – структурная схема процедуры tree.read\_file

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

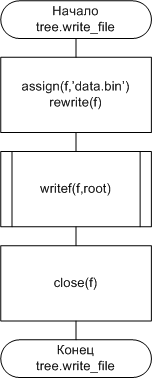
****

Рисунок 9 – структурная схема процедуры tree.write\_file

На рисунке 10 изображена схема алгоритма работы процедуры tree.writef.

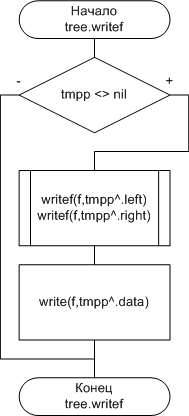
****

Рисунок 10 – структурная схема процедуры tree.writef

На рисунке 11 изображена схема алгоритма работы процедуры tree.print\_map.

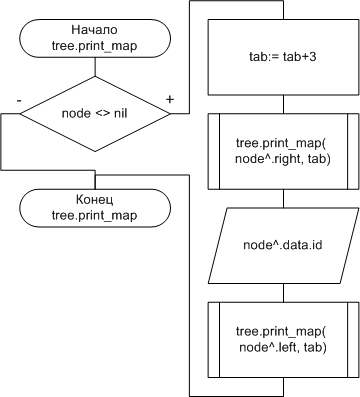
****

Рисунок 11 – структурная схема процедуры tree.print\_map

На рисунке 12 изображена схема алгоритма работы процедуры tree.search.

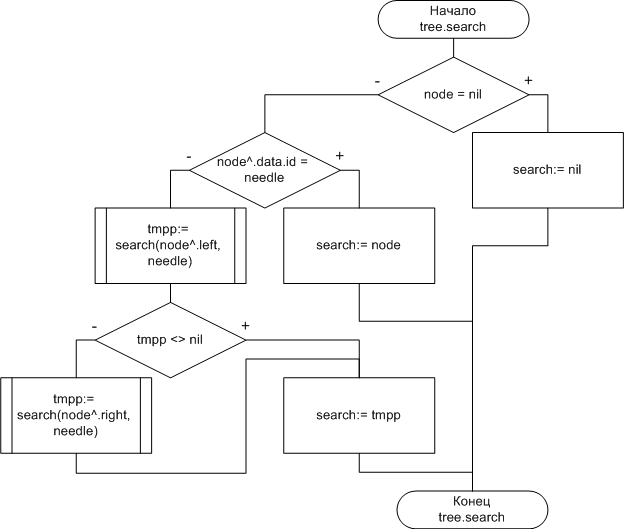
****

Рисунок 12 – структурная схема процедуры tree.search

На рисунке 13 изображена схема алгоритма работы процедуры tree.pull.

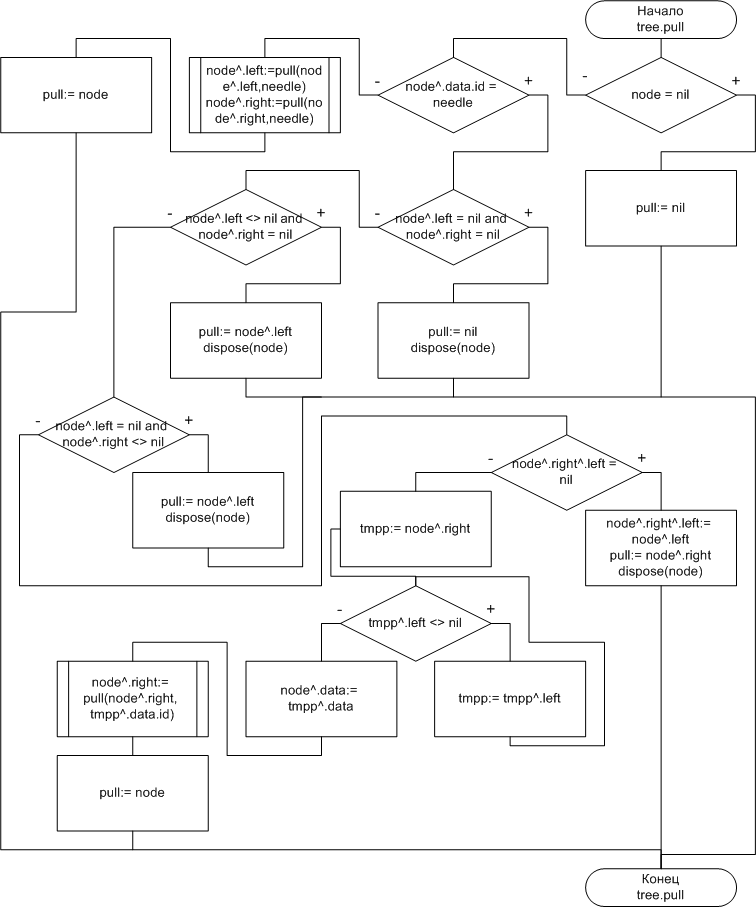
****

Рисунок 13 – структурная схема процедуры tree.pull 4. Текст программы

**program** lab03;

**uses** sysutils,crt,Lab03mod;

**var** needle:integer;

c:char;

tobj:tree;

**begin**

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

tobj.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 - Печать структуры');

writeln('6 - Удаление элемента');

writeln('7 - Поиск элемента');

writeln('8 - Выход');

readln(c);

clrscr;

**case** c **of**

'1':tobj.organize;

'2':tobj.put(root,tobj.input);

'3':tobj.print(root,0);

'4':tobj.print\_left;

'5':tobj.print\_map(root,0);

'6':**begin**

write('Введите id для удаления: ');

readln(needle);

root:= tobj.pull(root,needle);

**end**;

'7':**begin**

write('Введите id для поиска: ');

readln(needle);

tobj.print(tobj.search(root,needle),-1);

**end**;

**end**;

tobj.write\_file;

**until** c = '8';

tobj.free(root);

**end**.

5. Текст модуля

**unit** Lab03mod;

**interface**

**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**;

TDataFile=**file of** TData;

tree=object

**function** search(node:TNodePointer;needle:integer):TNodePointer;

**function** pull(node:TNodePointer;needle:integer):TNodePointer;

**function** put(node:TNodePointer;element:TData):TNodePointer;

**function** input:TData;

**procedure** organize;

**procedure** read\_file;

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

**procedure** write\_file;

**procedure** writef(**var** f:TDataFile;tmpp:TNodePointer);

**procedure** print\_left;

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

**procedure** print\_map(node:TNodePointer;tab:integer);

**end**;

**var** root:TNodePointer;

**implementation**

**function** tree.search(node:TNodePointer;needle:integer):TNodePointer;

**var** tmpp:TNodePointer;

**begin**

**if**(node = nil) **then**

search:= nil

**else if**(node^.data.id = needle) **then**

search:= node

**else begin**

tmpp:= search(node^.left,needle);

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

search:= tmpp

**else begin**

tmpp:= search(node^.right,needle);

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

search:= tmpp

**else**

search:= nil;

**end**;

**end**;

**end**;

**function** tree.pull(node:TNodePointer;needle:integer):TNodePointer;

**var** tmpp:TNodePointer;

**begin**

**if**(node = nil) **then**

pull:= nil

**else if**(node^.data.id = needle) **then begin**

**if**((node^.left = nil) **and** (node^.right = nil)) **then begin**

pull:= nil;

dispose(node);

**end else if**((node^.left <> nil) **and** (node^.right = nil)) **then begin**

pull:= node^.left;

dispose(node);

**end else if**((node^.left = nil) **and** (node^.right <> nil)) **then begin**

pull:= node^.right;

dispose(node);

**end else begin**

**if**(node^.right^.left = nil) **then begin**

node^.right^.left:= node^.left;

pull:= node^.right;

dispose(node);

**end else begin**

tmpp:= node^.right;

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

tmpp:= tmpp^.left;

node^.data:= tmpp^.data;

node^.right:= pull(node^.right,tmpp^.data.id);

pull:= node;

**end**;

**end**;

**end else begin**

node^.left:= pull(node^.left,needle);

node^.right:= pull(node^.right,needle);

pull:= node;

**end**;

**end**;

**function** tree.put(node:TNodePointer;element:TData):TNodePointer;

**begin**

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

**new**(node);

node^.data:= element;

node^.left:= nil;

node^.right:= nil;

**end else**

**if**(node^.data.id > element.id) **then**

node^.left:= tree.put(node^.left,element)

**else if** (node^.data.id < element.id) **then**

node^.right:= tree.put(node^.right,element);

put:= node;

**end**;

**function** tree.input:TData;

**var** data:TData;

**begin**

write('Введите номер поезда: ');

readln(data.id);

write('Введите пункт отправления: ');

readln(data.origin);

write('Введите пункт назначения: ');

readln(data.destination);

write('Введите время отправления: ');

readln(data.time\_departure);

write('Введите время прибытия: ');

readln(data.time\_arrival);

write('Введите стоимость билета: ');

readln(data.price);

input:= data;

**end**;

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

**begin**

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

free(node^.left);

free(node^.right);

dispose(node);

**end**;

**end**;

**procedure** tree.organize;

**var** c:char;

**begin**

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

free(root);

writeln('Организация рассписания');

**repeat**

root:= put(root,input);

writeln('Продолжить? (y/n)');

readln(c);

**until**(c <> 'y');

**end**;

**procedure** tree.read\_file;

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

tmpd:TData;

**begin**

assign(f,'data.bin');

reset(f);

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

free(root);

root:= nil;

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

read(f,tmpd);

root:= put(root,tmpd);

**end**;

close(f);

**end**;

**procedure** tree.writef(**var** f:TDataFile;tmpp:TNodePointer);

**begin**

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

writef(f,tmpp^.left);

writef(f,tmpp^.right);

write(f,tmpp^.data);

**end**;

**end**;

**procedure** tree.write\_file;

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

**begin**

assign(f,'data.bin');

rewrite(f);

writef(f,root);

close(f);

**end**;

**procedure** tree.print\_left;

**var** tmpp:TNodePointer;

**begin**

tmpp:= root;

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

tmpp:= tmpp^.left;

print(tmpp,-1);

**end**;

**procedure** tree.print(node:TNodePointer;level:integer);

**begin**

**if**(level < 1) **then begin**

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

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

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

**end**;

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

**if**(level <> -1) **then**

print(node^.left,1);

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

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

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

**if**(level <> -1) **then**

print(node^.right,1);

**end**;

**if**(level < 1) **then begin**

writeln('Для продолжения нажмите клавишу Enter');

readln;

**end**;

**end**;

**procedure** tree.print\_map(node:TNodePointer;tab:integer);

**begin**

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

tab:= tab+3;

tree.print\_map(node^.right,tab);

writeln('':tab,node^.data.id);

tree.print\_map(node^.left,tab);

**if**(tab = 3) **then begin**

writeln('Для продолжения нажмите клавишу Enter');

readln;

**end**;

**end**;

**end**;

**begin**

root:= nil;

**end**.

6. Тестовые примеры

На рисунке 14 представлено меню программы

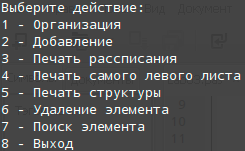


Рисунок 14 – меню программы

На рисунке 15 представлен результат работы пункта меню 3 «Печать расписания»

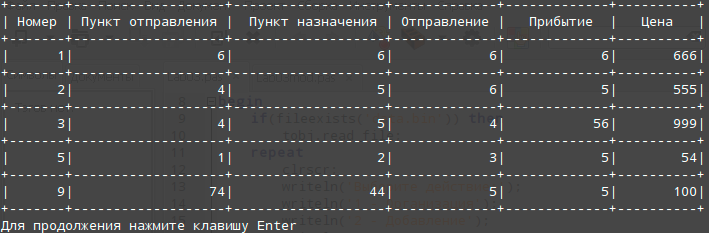


Рисунок 15 – расписание

На рисунке 15 представлен результат работы пункта меню 5 «Печать структуры»

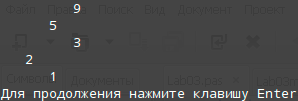


Рисунок 15 – структура

Вывод

В ходе лабораторной работы были изучены базовые понятия объектно-ориентированного программирования, приобретены навыки разработки и отладки программ, использующих модули и объекты. Исследованы особенности поисковых бинарных деревьев, представленных в виде объекта.