Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

по лабораторной работе № 2

«Программирование линейных списков»

Выполнил

ст. гр. И12д

Серегин А.В.

Проверил:

асс. Забаштанский А.К.

Севастополь

2015

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. Варианты заданий

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. Значение полей и количество записей в таблице студент выбирает самостоятельно. Программа должна сохранять дерево в файле и создавать его заново при её повторном запуске.

Таблица 3.2 - Расписание

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № Поезда | Станция отправки | Станция назначения | Время отправки | Время прибытия | Стоимость билета |

Вариант 20. Таблица 3.2. Процедуру, которая подсчитывает число вершин на n-ом уровне непустого дерева T.

3. Структурная схема



Рисунок 1 – Алгоритм записи дерева в файл.



Рисунок 2 – Алгоритм создания записи.

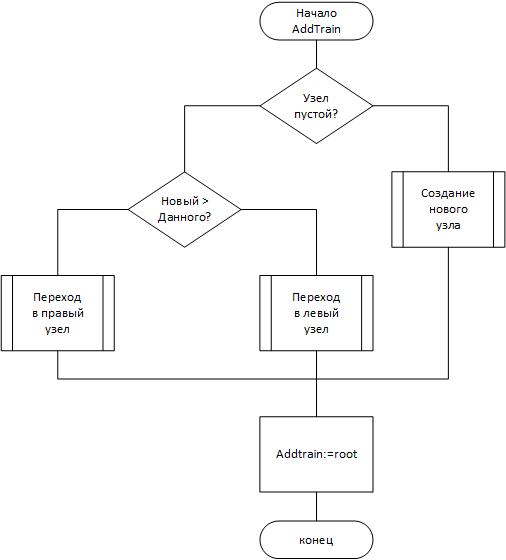


Рисунок 3 – Алгоритм добавления нового узла.

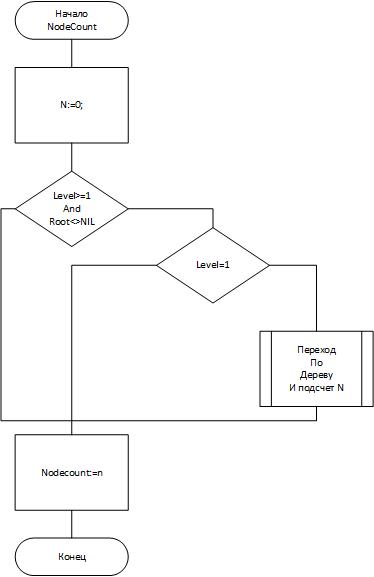


Рисунок 4 – Алгоритм подсчета узлов уровня N.



Рисунок 5 – Алгоритм вывода записи на экран.

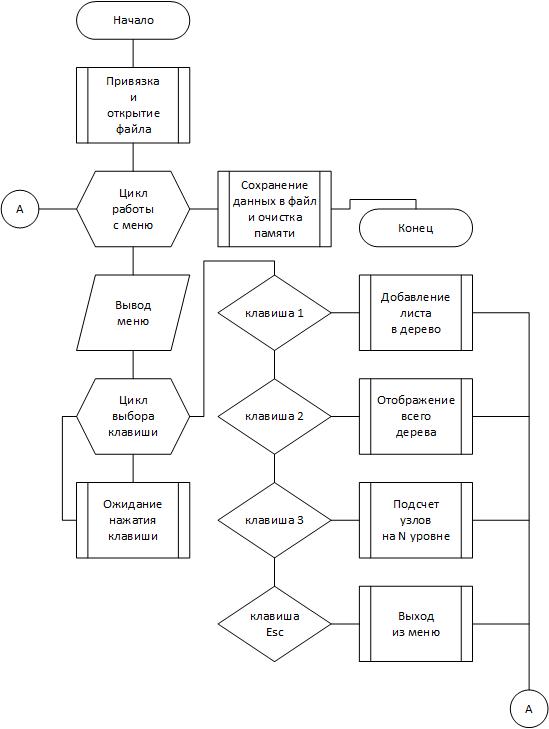


Рисунок 6 – Алгоритм программы.



Рисунок 7 – Алгоритм освобождения памяти.

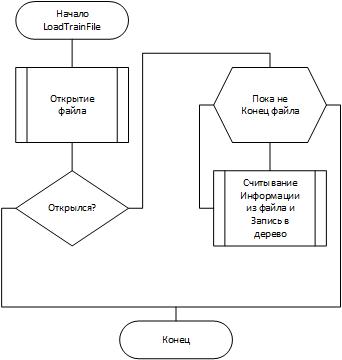


Рисунок 8 – Алгоритм загрузки файла.



Рисунок 9 – Алгоритм сохранения файла.

4. Код программы

program lab12\_20;

uses crt;

type

time = record {запись время}

hour:integer;

minute:integer;

end;

TrainInfo = record {запись о поезде}

number:integer;

StartPoint:string[20];

EndPoint:string[20];

StartTime:time;

EndTime:time;

TicketPrice:longint;

end;

ptrain = ^traintree;

TrainTree = record {описание узла дерева}

train:traininfo;

left:ptrain;

right:ptrain;

end;

trainfile = file of traininfo;

function NewTrain():TrainInfo; {ввод информации о поезде}

begin

clrscr();

writeln('Введите номер поезда.');

readln(NewTrain.number);

writeln('Введите место отправки.');

readln(NewTrain.StartPoint);

writeln('Введите место прибытия.');

readln(NewTrain.EndPoint);

writeln('Введите время отправки.');

readln(NewTrain.StartTime.hour, NewTrain.StartTime.minute);

writeln('Введите время прибытия.');

readln(NewTrain.EndTime.hour, NewTrain.EndTime.minute);

writeln('Введите стоимость билета.');

readln(NewTrain.TicketPrice);

end;

procedure PrintTrainInfo(const Train:TrainInfo); {вывод информации о поезде}

begin

write(Train.number);

write(Train.StartPoint:11,Train.EndPoint:11);

write(Train.StartTime.hour:4,':',Train.StartTime.minute);

write(Train.EndTime.hour:5,':',Train.EndTime.minute);

write(Train.TicketPrice:10);

end;

Function AddTrain(root:ptrain; const nt:traininfo):ptrain;

{добавление узла дерева}

begin

if root=NIL then

begin {создание нового узла}

new(root);

root^.train:=nt;

root^.left:=NIL;

root^.right:=NIL;

end

else

if Root^.train.number>nt.number then {прохождение по дереву}

Root^.left:=addtrain(Root^.left,nt)

else

Root^.right:=addtrain(Root^.Right,nt);

AddTrain:=root;

end;

Procedure ShowAll(Root:ptrain); {отображение дерева}

Begin

if Root<>Nil then

begin

Showall(Root^.left); {прохождение по дереву}

PrintTrainInfo(Root^.train); {вывод информации о узле}

writeln();

Showall(Root^.right); {прохождение по дереву}

end;

End;

Procedure Free(Root:ptrain); {освобождение памяти}

begin

if Root=NIL then exit();

Free(Root^.right); {прохождение по дереву}

Free(Root^.left);

Dispose(Root); {удаление узла}

end;

procedure PrintTrainFile(var f:trainfile; root:ptrain); {запись узлов в файл}

Begin

if Root=NIL then exit();

write(f,root^.train); {запись узла в файл}

printtrainfile(f,root^.left);

printtrainfile(f,root^.right);

End;

Procedure SaveTrainFile(var f:trainfile; root:ptrain); {сохранение файла}

begin

rewrite(f);

PrintTrainFile(f,root);

close(f);

end;

procedure LoadTrainFile(var f:trainfile; var root:ptrain); {загрузка файла}

var nt:TrainInfo;

begin

{$I-}

reset(f);

{$I+}

if (IOResult=0) and (FileSize(f)<>0) then

while not eof(f) do {считывание поезда из файла}

begin

read(f,nt);

root:=AddTrain(root,nt); {добавление узла в дерево}

end;

end;

Function NodeCount(root:ptrain; Level:Integer):integer;

{подсчет k узлов n уровня}

var n:integer;

Begin

n:=0;

If (Level>=1) and (root<>Nil) Then

Begin

If Level=1 Then N:=N+1;

n:=n+NodeCount(root^.Left,Level-1); {переход на след. уровень}

n:=n+NodeCount(root^.Right,Level-1);

End;

NodeCount:=n;

End;

var root:ptrain;

key:char;

f:TrainFile;

nt:traininfo;

n:integer;

Begin

assign(f,'Train.dat'); {привязка файла}

LoadTrainFile(f,root); {загрузка файла}

while true do

begin

clrscr();

writeln('1.Add 2.ShowAll 3.NodeCount Esc.Exit'); {вывод гл.меню}

key:=' ';

while not (key in ['1','2','3',#27]) do key:=readkey;

{работа с пунктами меню}

case key of

'1': begin

nt:=NewTrain();

root:=AddTrain(root,nt);

end;

'2': begin

clrscr();

writeln('№','StPoint':11,'EndPoint':11,'StTime':7,

'EndTime':8,'Price':10);

ShowAll(root);

readkey;

end;

'3': begin

clrscr();

writeln('Введите номер уровня.');

readln(n);

writeln('На уровне ',n,' ',NodeCount(root,n),' узeл(ов)');

readkey;

end;

#27: break;

end;

end;

SaveTrainFile(f,root); {сохранение данных дерева в файл}

Free(root); {очистка памяти}

End.

5.Тестовые примеры

Пример вывода меню изображен на Рисунке 10.

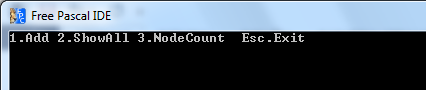


Рисунок 10 – Главное меню.

На Рисунке 11 изображен ввод информации о поезде. Сначала мы добавим поезд с номером 2, потом 1 и 3, которые пойдут в левую и правую ветвь соответственно.

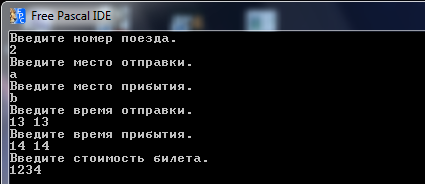


Рисунок 11 – Ввод данных.

На Рисунке 12 отображаются данные введенные нами раньше, выведенные слева направо дерева.

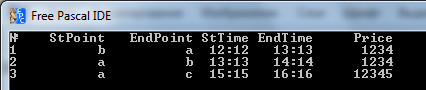


Рисунок 12 – Отображение дерева.

На Рисунке 13 показан подсчет количества узлов N уровня. Узлы 1 и 3 пошли влево и вправо от корня 2, поэтому на 2м уровне 2 узла.

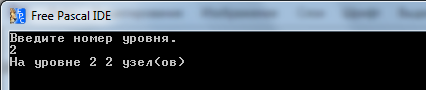


Рисунок 13 – Подсчет узлов N уровняю.

Выводы

С помощью нелинейных структур данных можно управлять различными данными. Сложностью работы с деревьями являются рекурсивные алгоритмы. Одним из минусов работы с рекурсией является быстрое переполнение стека, а также рекурсивный алгоритм работает значительно медленнее чем итерационный. Плюсом древовидных структур является быстрота нахождения нужного элемента в бинарном дереве поиска.