Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

по лабораторной работе № 1

«Программирование линейных списков»

Выполнил

ст. гр. И12д

Серегин А.В.

Проверил:

асс. Забаштанский А.К.

Севастополь

2014

1. Цель работы

Исследование способов обработки логически связанных данных различных типов. Создание программ, использующих записи и типизированные файлы.

2. Варианты заданий

Представить одну из приведенных ниже таблиц в виде линейного списка L, элементами которого являются строки таблицы. Написать процедуры организации, добавления элемента в список, исключения элемента из списка, просмотра списка, а также одну из процедур в соответствии с вариантом, приведенным ниже.

Значения и количество записей в таблице студент выбирает самостоятельно. Исходные данные после организации списка должны сохраняться в файле и при повторном запуске программы считываться из файла. Количество строк таблицы не задается.

Таблица 3.2 - Расписание

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

Вариант 20. Таблица 3.2. Процедуру, которая удаляет из непустого списка L первый элемент.

3. Структурная схема алгоритма

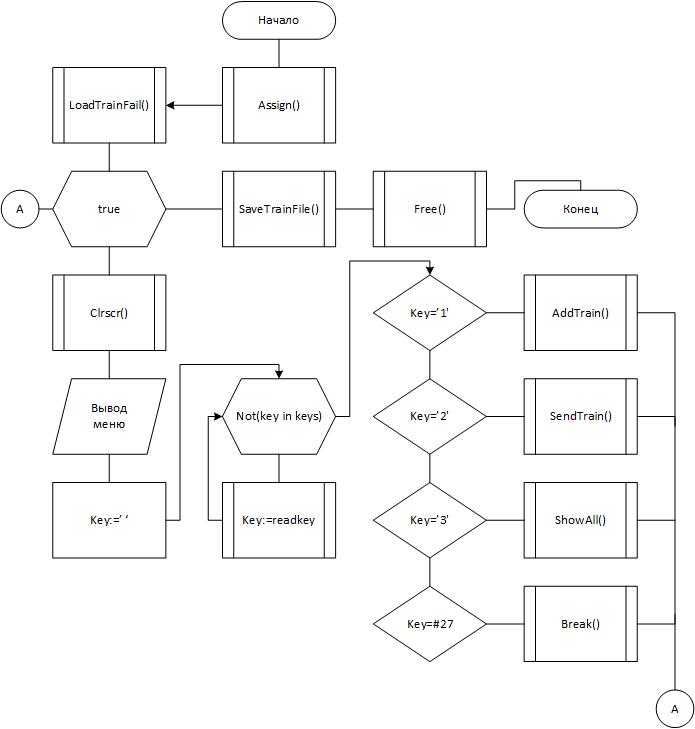


Рисунок 1 – алгоритм программы.

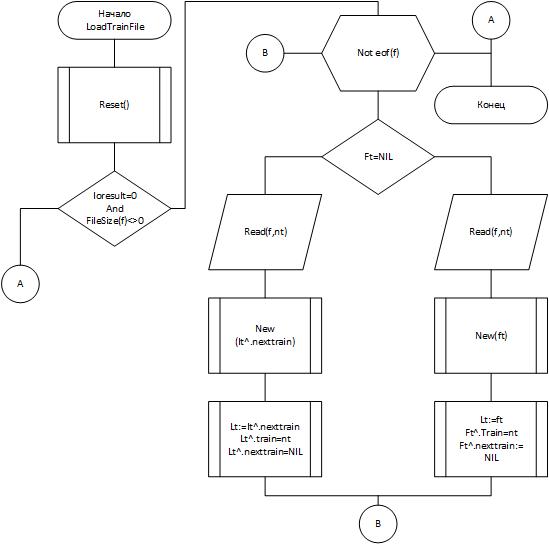


Рисунок 2 – алгоритм загрузки файла.

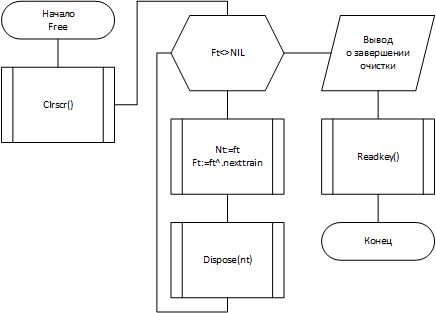


Рисунок 3 – алгоритм очистки памяти.

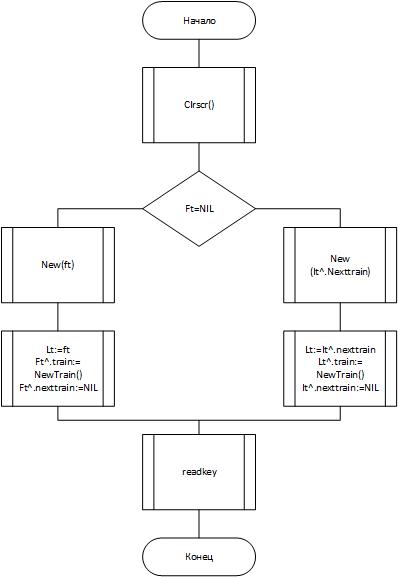


Рисунок 4 – алгоритм добавления элемента

в список.



Рисунок 5 – алгоритм создания

нового поезда.



Рисунок 6 – алгоритм вывода

информации о поезде.

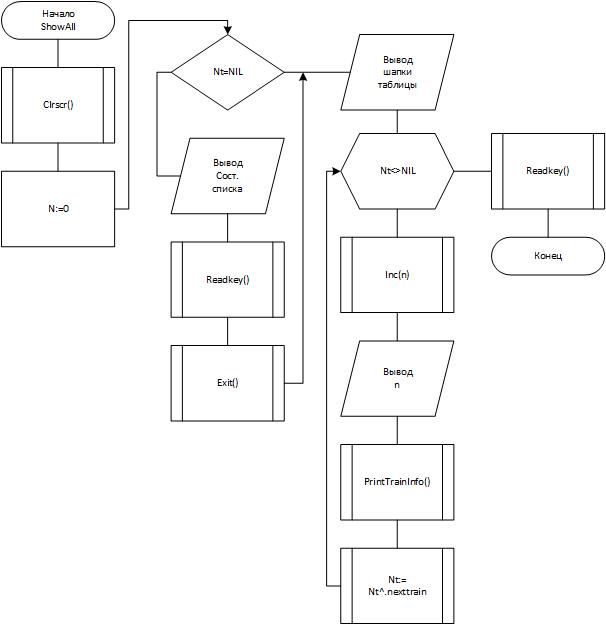


Рисунок 7 – алгоритм вывода списка

поезов.

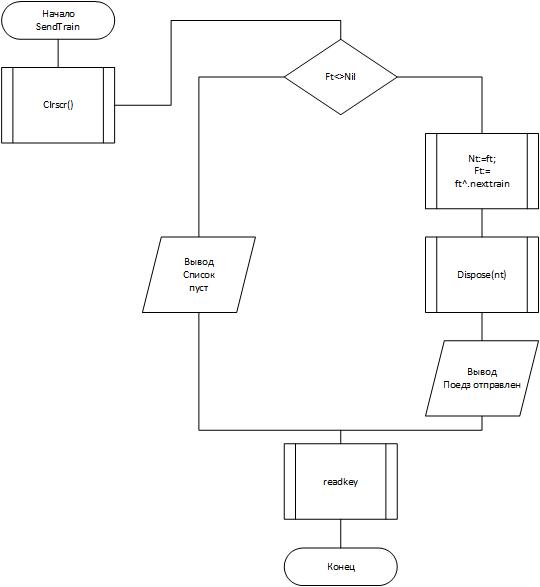


Рисунок 8 – алгоритм отправки

1-го поезда.

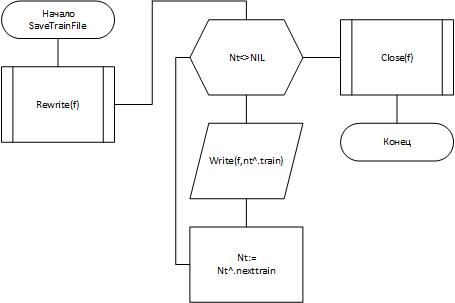


Рисунок 9 – алгоритм сохранения списка в файл.

4. Код программы

program lab11\_20;

uses crt;

type

time = record

hour:integer;

minute:integer;

end;

TrainInfo = record

StartPoint:string[20];

EndPoint:string[20];

StartTime:time;

EndTime:time;

TicketPrice:longint;

end;

ptrain = ^trainlist;

TrainList = record

train:traininfo;

nexttrain:ptrain;

end;

trainfile = file of traininfo;

function NewTrain():TrainInfo;

begin

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);

begin

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;

procedure AddTrain(var ft,lt:ptrain);

begin

clrscr();

if ft<>NIL then

begin

New(lt^.nexttrain);

lt:=lt^.nexttrain;

lt^.train:=NewTrain();

lt^.nexttrain:=NIL;

end

else

begin

New(ft);

lt:=ft;

ft^.train:=NewTrain();

ft^.nexttrain:=NIL;

end;

writeln('Новый поезд добавлен успешно.');

readkey();

end;

procedure SendTrain(var ft:ptrain);

var nt:ptrain;

begin

clrscr();

if ft<>NIl then

begin

nt:=ft;

ft:=ft^.nexttrain;

dispose(nt);

writeln('1й поезд отправлен');

end

else

writeln('Список поездов пуст.');

readkey();

end;

procedure ShowAll(nt:ptrain);

var n:integer;

begin

clrscr();

n:=0;

if nt=NIL then

begin

writeln('Список пуст');

readkey();

exit();

end;

writeln('№','StPoint':11,'EndPoint':11,'StTime':7,'EndTime':8,'Price':10);

while nt<> NIL do

begin

inc(n);

write(n);

PrintTrainInfo(nt^.train);

writeln;

nt:=nt^.nexttrain;

end;

readkey();

end;

procedure Free(var ft:ptrain);

var nt:ptrain;

begin

clrscr;

while ft<>NIL do

begin

nt:=ft;

ft:=ft^.nexttrain;

Dispose(nt);

end;

writeln('Память успешно очищена.');

readkey();

end;

procedure SaveTrainFile(var f:trainfile; nt:ptrain);

begin

rewrite(f);

while nt<>NIL do

begin

write(f,nt^.train);

nt:=nt^.nexttrain;

end;

close(f);

end;

procedure LoadTrainFile(var f:trainfile; var ft,lt:ptrain);

var nt:TrainInfo;

begin

{$I-}

reset(f);

{$I+}

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

while not eof(f) do

if ft=NIL then

begin

read(f,nt);

New(ft);

lt:=ft;

ft^.train:=nt;

ft^.nexttrain:=NIL;

end

else

begin

read(f,nt);

New(lt^.nexttrain);

lt:=lt^.nexttrain;

lt^.train:=nt;

lt^.nexttrain:=NIL;

end;

end;

var FirstTrain,LastTrain:ptrain;

key:char;

f:TrainFile;

Begin

assign(f,'Train.dat');

LoadTrainFile(f,FirstTrain,LastTrain);

while true do

begin

clrscr();

writeln('1.Add 2.Send 3.ShowAll Esc.Exit');

key:=' ';

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

case key of

'1': AddTrain(FirstTrain,LastTrain);

'2': SendTrain(FirstTrain);

'3': ShowAll(FirstTrain);

#27: break;

end;

end;

SaveTrainFile(f,FirstTrain);

Free(FirstTrain);

End.

5. Тестовые примеры

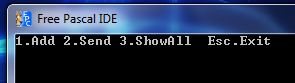


Рисунок 10 – Меню.

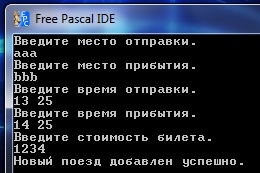


Рисунок 11 – Добавление поезда.

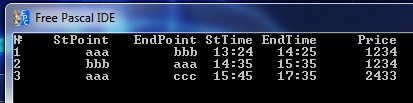


Рисунок 12 – Вывод списка поезда.

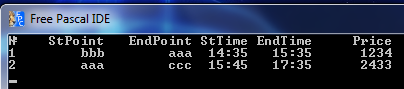


Рисунок 13 – Вывод списка поезда

после отправления.

По нажатию клавиши 1 в главном меню программы (Рисунок 10) вызывается функция добавления нового поезда в список, на рисунке 11 отображается работа этой процедуры. Добавив несколько элементов в список в главном меню нажимаем 3 и выводится список всех поездов (Рисунок 12). После в гл. меню нажимаем 2 и первый поезд из списка «отправляется». После повторно вывода на экран видим, что первый поезд был удален из списка (Рисунок 13).

Выводы

С помощью линейных списков можно управлять различными данными. Сложностью работы со списками является трудное восприятие хранения данных в памяти компьютера человеком и работа с указателями на области памяти где хранятся эти данные. Одним из минусов работы с односвязными списками является скорость обращение к какому-то конкретному элементу списка, т.к. чтобы добраться до него требуется пройти по всем предыдущим элементам. Плюсом будет простой вывод всего списка на экран или его последовательной части, на пример при прокрутке списка.