МИНОБРНАУКИ РОССИИ

ФГБОУ ВО «Чувашский государственный университет имени И.Н. Ульянова»

Факультет прикладной математики,

физики и информационных технологий

Кафедра дискретной математики и информатики

КУРСОВАЯ РАБОТА

По теме: структура и алгоритм компьютерной обработки данных

«Железнодорожный вокзал»

Выполнил: студент

группы ФМ-12-16

Еремеев Александр Алексеевич

Проверила:

ст. преподаватель Сидорова Е.Б.

ЧЕБОКСАРЫ 2018

##### СОДЕРЖАНИЕ

##### ВВЕДЕНИЕ…………………………..………………………………….…..3

1.РАЗРАБОТКА………………………………………………………….….4

1.1 Используемые типы данных..………………………………………......4

1.2 Процедура «выбор действий (меню)»………………………….……...5

1.3 Процедура «заполнение списка (добавление)»………………….........6

1.4 Процедура «Вывод списка на экран»………………………………….7

1.5 Процедура «Получение времени»……………………………………...8

1.6 Процедура «Добавление рейса в конец списка»……………………....8

1.7 Процедура «Удаление i-ого элемента »……………….………….........9

1.8 Процедура «Удаление всех элементов»…………………….…..……..9

1.9 Процедура «Поиск заявок»……………………………………………11

1.10 Процедура «Длина списка»………………………………………….11

1.11 Процедура «Проверка списка на пустоту»…………………….........13

1.12 Процедура «Переворот списка»……………………………………..13

1.13 Процедура «Деление списка на 2 части»…………………………...14

1.14 Процедура «Меню для разделённого списка»……………….……..15

1.15 Процедура «Соединение разделённого списка»……………………15

1.16 Процедура «Проверка двух списков на равенство»…….…….…....15

2.ПОЛНЫЙ ТЕКСТ ПРОГРАММЫ………………..……………………20

ЗАКЛЮЧЕНИЕ…………………………………………………………...34

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ……………………...34

**ВВЕДЕНИЕ**

Статическими величинами называются такие, память под которые выделяется во время компиляции и сохраняется в течение всей работы программы. В языках программирования (Pascal, C, др.) существует способ выделения памяти под данные. В этом случае память под величины отводится во время выполнения программы. Такие величины будем называть динамическими. Раздел оперативной памяти, распределяемый статически, называется статической памятью; динамически распределяемый раздел памяти называется динамической памятью (динамически распределяемой памятью). Использование динамических величин предоставляет программисту ряд дополнительных возможностей.

* Во-первых, подключение динамической памяти позволяет увеличить объем обрабатываемых данных;
* Во-вторых, если потребность в каких-то данных отпала до окончания программы, то занятую ими память можно освободить для другой информации;
* В-третьих, использование динамической памяти позволяет создавать структуры данных переменного размера;

Работа с динамическими величинами связана с использованием еще одного типа данных — ссылочного типа. Величины, имеющие ссылочный тип, называют указателями. Указатель содержит адрес поля в динамической памяти, хранящего величину определенного типа. Сам указатель располагается в статической памяти.

Величина ссылочного типа (указатель) описывается в разделе описания переменных следующим образом:

Var<идентификатор> : ^<имя типа>;

Память под динамическую величину, связанную с указателем, выделяется в результате выполнения стандартной процедуры NEW:

new(<указатель>);

Например, обозначение P1^ можно расшифровать так: динамическая переменная, на которую ссылается указатель P1.

Дальнейшая работа с динамическими переменными происходит точно так же, как со статическими переменными соответствующих типов. Им можно присваивать значения, их можно использовать в качестве операндов в выражениях, параметров подпрограмм.

Задача курсовой работы

В файловой системе каталог файлов организован как линейный список. Для каждого файла в каталоге содержатся следующие сведения:

1. номер поезда;
2. станция назначения;
3. время отправления;

Разработать программу, которая обеспечивает:

* начальное формирование каталога файлов;
* вывод каталога рейсов.

**Разработка**

**«Используемые типы данных»**

В приведённом выше коде используются:

* Type - Раздел описания типов данных ;
* Record - (Запись) структурированный комбинированный тип данных, состоящий из фиксированного числа компонент (полей) разного типа;
  + Byte – целое число, в диапазоне 0..255;
* Integer – целое число, в диапазоне -32768..32767;
* Ukazat – переменная ,созданная вручную, называемая указателем;
* String – строковая переменная;
* Deystvie, Deystvie1 – переменные для «выбора в меню»;
* dlina\_spiska – переменная, хранящая длину списка

type

ukazat = ^item; //выделение памяти для переменной где хранится ее адрес

item =record //объявляем переменную item записью

punkt:string; //пункт назначения

nom\_reys: integer; //номер рейса

fio: string; //ФИО

date: record //Дата(время)

h: byte; //часы;

m: byte; //минуты;

end;

next: ukazat; //указатель на следующий элемент

pred: ukazat; //указатель на предыдущий элемент

end;

var

deystvie, dlina\_spiska, deystvie1,i: integer; //переменныецелоготипа;

perv, konec, perv1, konec1: ukazat; //объявляем глобальные переменные типа указатели;

!!! В приведённых ниже текстах программы, в передаваемых параметрах процедуры, можно будет часто встретить такие переменные как head (head1), tail (tail1), dlina\_spiska:

* + perv (perv1) – указатель на первый элемент списка (perv1 используется в делении списка на 2);
  + konec (konec1) – указатель на последний элемент списка (konec1 используется в делении списка на 2);
  + dlina\_spiska – параметр, передаваемый в процедуру (или выходящий из неё), который содержит информацию о длине списка;

**Процедура «Меню»**

Для удобства данная процедура в программе вызывается автоматически. В ходе пользователю будет предложен список действий, которые он хотел бы выполнить. После чего он должен будет ввести число желаемой команды, и оно осуществится.

procedure menu(vardeystvie: integer);

begin

writeln();

writeln('Что вы хотите сделать:');

writeln('1. Заполнить список');

writeln('2. Показать список');

writeln('3. Добавить заявку');

writeln('4. Удаление заявки');

writeln('5. Вывод заявок по заданному номеру рейса и времени выезда;');

writeln('6. Длина списка ');

writeln('7. Проверка списка на пустоту ');

writeln('8. Переворот списка');

writeln('9. Деление на 2 списка');

writeln('10. Удаление всех элементов списка');

writeln('0. Завершить программу ');

writeln;

write('Вашвыбор: ');

readln(deystvie); // ввод нашего выбора

end;

**Процедура «заполнение списка (добавление)»**

procedurezapolnenie(varperv, konec: ukazat);

var q: ukazat;

simv: array[1..3] of integer;

punkt\_naznach: string;

h, m, n, i: integer;

avtor,name: string;

begin

n:=10;

whilen> 0 do // пока n больше 0, заполняем списки

begin

//заполняем автора

avtor := '';

simv[1] :=random(90 - 65 + 1) + 65 ; //первая заглавная буква

avtor := avtor + chr(simv[1]);

fori := 1 to (random(10) + 3) do //кол-во символов (до точки) в имени файла

begin

simv[2] := random(122 - 97 + 1) + 97; //нижний регистр

avtor := avtor + chr(simv[2]); //добавляем рандомный символ

end;

simv[1] :=random(90 - 65 + 1) + 65 ;

simv[2] :=random(90 - 65 + 1) + 65 ;

avtor :=avtor +' '+chr(simv[1])+ '.'+chr(simv[2])+'.';

punkt\_naznach := '';

punkt\_naznach := punkt\_naznach + chr(random(90 - 65 + 1) + 65);

fori := 1 to (random(10) + 3) do //кол-во символов (до точки) в имени файла

punkt\_naznach := punkt\_naznach + chr(random(122 - 97 + 1) + 97); //добавляем рандомный символ

h := random(23);

m := random(59);

// создаём указатель

New(q);

q^.next := nil;

q^.pred := nil;

q^.fio :=avtor;

q^.nom\_reys := random(5)+100; // рандомныйрейс

q^.date.h := h;

q^.date.m := m;

q^.punkt:=punkt\_naznach; // рандомныйпунктназначения

ifperv = nil then begin // еслисписокпуст

perv := q;

konec := q;

end

elsebegin // если имеются элементы в списке

konec^.next := q;

konec^.next^.pred := konec;

konec := q;

end;

n := n - 1;

end;

end;

Алгоритм заполнения: В начале в меню выбора действий вибираем пункт 1.Заполнить список, затем создаётся пустой указатель, где последовательно идёт рандомное заполнение всех данных (имени рейса, пункта назначения, ФИО и времени). После чего он заносится в список. И так повторяется, пока не создастся нужное количество элементов.

**Процедура «Вывод списка на экран»**

Вызов процедуры осуществляется командой *Print(perv)* , где perv – указатель на первый элемент списка. После вызова процедуры на экран выводятся последовательно все элементы списка, если они есть. Иначе на экране появиться сообщение о том, что «Список пуст»

Алгоритм вывода: пока указатель (начиная с 1ого) не пуст, выводим значения и переходим на следующий указатель.

{ Выводнаэкрансписка }

procedure Print(perv: ukazat);

var

R: ukazat;

i: integer;

begin

ifperv<> nil then begin // еслисписокнепустой

i := 1;

R :=perv;

while R <> nil do

begin

Writeln(i:2, '. ',

R^.punkt:18,' ' ,

R^.nom\_reys:4,' ',

R^.fio:20,' ',

R^.date.h:2, ':',

R^.date.m:2);

R :=R^.next;

Inc(i);

end;

end

else

writeln('Списокпуст');

end;

**Процедура «Получение времени»**

Процедура вызывается командой get\_date(var hrs1, min1: integer), где параметры это глобальные переменные, хранящие в себе день, месяц и год создания файла в формате целых чисел.

Алгоритм: пользователь сначала вводит время в формате чч:мм, а программа определяет где часы и минуты, и переводит из входной строки в числовой формат.

procedureget\_date(var hrs1, min1: integer);

var

data: string;

i, j, ost: integer;

begin

readln(data);

val(Copy(data,1,2),hrs1);

val(Copy(data,4,2),min1);

end;

**Процедура «Добавление рейса в конец списка»**

Вызов процедуры осуществляется командой *AddReys(varperv,konec:ukazat);,* где в параметрах передаётся сначала длина списка, затем идет добавление.

Алгоритм: После вызова процедуры, сначала проверяется список на пустоту, затем после этого пользователь вводит данные. Создаётся новый указатель, название которого придумываем мы сами, время отправления ставится в коде программы, а кол-во обращений автоматически равняется 1. После создания указателя соединяем список.

{ Добавление рейса }

procedureAddReys(varperv,konec:ukazat);

var

k,hrs1,min1:integer;

begin

ifperv=nil then begin

new(perv);

perv^.next:=nil;

perv^.pred:=nil;

konec:=perv;

write('Введитепунктназначения: '); readln(konec^.punkt);

write('Введитеномеррейса: '); readln(konec^.nom\_reys);

write('Введите FIO: '); readln(konec^.fio);

write('Введите время желаемого выезда: ');

get\_date(hrs1,min1);

konec^.date.h:=hrs1;

konec^.date.m:=min1;

end

else begin

new(konec^.next);

konec^.next^.pred:=konec;

konec:=konec^.next;

konec^.next:=nil;

write('Введитепунктназначения: '); readln(konec^.punkt);

write('Введитеномеррейса: '); readln(konec^.nom\_reys);

write('Введите FIO: '); readln(konec^.fio);

write('Введите время желаемого выезда: ');

get\_date(hrs1,min1);

konec^.date.h:=hrs1;

konec^.date.m:=min1;

end;

end;

**«Удаление i-того элемента»**

Алгоритм: Пользователь передаёт значение i-ого элемента и, если он не выходит за пределы списка, то мы идём по списку в i-ую точку, находим i-ый элемент, и удаляем его, с последующим соединением предыдущего и последующего.

{ Удалениезаписи }

Procedure DelElem(var spis1,spis2:ukazat;tmp:ukazat);

var

tmpi:ukazat;

begin

if (spis1=nil) or (tmp=nil) then

exit;

iftmp=spis1 then begin

spis1:=tmp^.next;

ifspis1<>nilthen {если список оказался не из одного элемента, то}

spis1^.pred:=nil {"зануляем" указатель}

else {в случае, если элемент был один, то}

spis2:=nil; {"зануляем" указатель конца списка, а указатель начала уже "занулён"}

dispose(tmp);

end

else

iftmp=spis2 thenbegin{если удаляемый элемент оказался последним элементом списка}

spis2:=spis2^.pred; {указатель конца списка переставляем на предыдущий элемент}

ifspis2<>nilthen {если предыдущий элемент существует,то}

spis2^.next:=nil {"зануляем" указатель}

else {в случае, если элемент был один в списке, то}

spis1:=nil; {"зануляем" указатель на начало списка}

dispose(tmp);

end

elsebegin{если же удаляется список не из начали и не из конца, то}

tmpi:=spis1;

whiletmpi^.next<>tmpdo {ставим указатель tmpi на элемент перед удаляемым}

tmpi:=tmpi^.next;

tmpi^.next:=tmp^.next; {меняемсвязи}

iftmp^.next<>nil then

tmp^.next^.pred:=tmpi; {уэлементадоудаляемогоипосленего}

dispose(tmp);

end;

end;

Procedure DelElemPos(var spis1,spis2:ukazat; posi:integer);

var i:integer;

tmp:ukazat;

begin

ifposi<1 then

exit;

if spis1=nil then begin

Write('Списокпуст');

exit

end;

i:=1;

tmp:=spis1;

while (tmp<>nil) and (i<>posi) do begin

tmp:=tmp^.next;

inc(i)

end;

iftmp=nil then begin

Writeln('Записи с порядковым номером ' ,posi, ' нет в списке.');

writeln('В списке всего ' ,i-1, ' элементов.');

exit

end;

DelElem(spis1,spis2,tmp);

Writeln('Элемент удалён.');

end;

**Процедура «Удаление всех элементов»**

Алгоритм: присваиваем локальную переменную к первому элементу списка, сам же 1ый элемент переходит на следующую, а локальную уничтожаем, и так продолжается до тех пор, пока всё не станет nil.

proceduredelete\_vseh\_el(varperv,konec:ukazat);

varq,R:ukazat;

begin

whileperv<>nil do begin

q:=perv;

perv:=perv^.next;

ifperv<>nil then

dispose(q)

else begin

konec:=nil;

dispose(q);

end;

end;

writeln('Списокочищен!' );

end;

**Процедура «Поиск заявок»**

Алгоритм поиска с заданным значением: Начиная с 1ого элемента пробегаем весь список. Если же встретим элемент сзаданным значение, мы его записываем, и идём дальше до тех пор, пока список не закончится. В результате выводим значения переменой, куда мы записывали номера нужных нам элементов. (Алгоритм аналогичен у 1.8. и 1.9.)

procedurepoisk\_zayavok(varperv,konec: ukazat);

var

R: ukazat;

hrs1,min1, prints, j,reys: integer;

s: string;

date,date\_fly,h,m: string;

begin

prints:=0;

s := '';

j := 1;

R :=perv;

if R <> nil then begin

write('Введитеномеррейса: ');readln(reys);

write('Введите время выезда: ');

get\_date(hrs1,min1);

while R <> nil do begin

if (hrs1 = R^.date.h) and

(min1 = R^.date.m) then begin

s:=s+'1';

Writeln(j:2, '. ',

R^.punkt:18,' ' ,

R^.nom\_reys:4,' ',

R^.fio:20,' ',

R^.date.h:2, ':',

R^.date.m:2);

j := j + 1;

R :=R^.next;

end

else begin

R :=R^.next;

j := j + 1;

end;

end;

if s='' then writeln('нетсоответствующихзаявок');

end

else

writeln('Списокпуст');

end;

**Процедура «Длина списка»**

Алгоритм: проходим все элементы списка, тем временем увеличиваем её каждый раз на один. В итоге получем длину списка.

proceduredlina\_sp(perv: ukazat; vardlina\_spiska: integer);

var k: integer;

R: ukazat;

begin

R :=perv;

k := 0;

while R <> nil do begin

R :=R^.next;

k := k + 1;

end;

dlina\_spiska := k;

end;

**Процедура «Проверка списка на пустоту»**

Алгоритм: Если первый элемент не пустой (head<>nil), значит список НЕ пуст.

procedureproverka\_spiska\_na\_pustotu(perv: ukazat);

begin

ifperv = nil then

writeln('списокпуст')

else

writeln('список не пуст');

end;

**Процедура «Переворот списка»**

Алгоритм состоит в том, чтобы поменять значения первого и последнего элемента местами (prevandkonec), дополнительно также значения у каждого элемента списка, а именно значения на следующий и предыдущий файл (^nextand ^prev).

procedureperevorot\_spiska(varperv,konec:ukazat);

varq,zn,w:ukazat;

begin

ifperv<>nil then begin

q:=perv;

w:=konec;

while q<>nil do begin

zn:=q^.next;

q^.next:=q^.pred;

q^.pred:=zn;

q:=zn;

end;

konec:=perv;

perv:=w;

end

else

writeln('списокпуст');

end;

**Процедура «Деление списка на 2 части»**

Алгоритм: Сначала передаётся длина списка. После чего идёт целочисленное деление на 2. Затем разрываем список в том месте списка, какое число получилось, присваивая 1ому концу konec1 и 2ому началу prev1.

procedure delenie\_na\_2\_spiska(var perv,konec,perv1,konec1:ukazat);

var

q,i,dlina\_spiska:integer;

R:ukazat;

begin

R:=perv;

dlina\_sp(R,dlina\_spiska);

ifperv<>nilthen

begin

q:=dlina\_spiskadiv 2; // целочисленно делим основной список на 2

i:=1;

whilei<>q do

begin

R:=R^.next;

i:=i+1;

end;

konec1:=R;

R:=R^.next;

perv1:=R;

konec1^.next:=nil;

perv1^.pred:=nil;

end

else

writeln('Списокпуст, надозаполнить!!!');

end;

**Процедура «Меню для разделённого списка»**

Действия: Пользователю предоставляется выбор, и он вводит число, что ему требуется

procedure menu1(var deystvie1:integer);

begin

writeln();

writeln('Что вы хотите сделать?:');

writeln('1. Показать 2 списка');

writeln('2. Проверка 2ух списков на равенство');

writeln('3. Соединение 2ух списков в один (основные действия только с одним списком)');

writeln();

write('Вашвыбор: ');

readln(deystvie1);

end;

**Процедура «Соединение разделённого списка»**

Алгоритм: 1ый конец списка соединяем с началом 2ого в обе стороны

procedure connect(konec1,perv1:ukazat);

begin

konec1^.next:=perv1; // конец 1ого на начало 2ого

perv1^.pred:=konec1; // конец 2ого на начало 1ого

end;

**Процедура «Проверка двух списков на равенство»**

Алгоритм: начиная с 1ых элементов сравниваем названия двух списков. Если же встречаются разные названия файлов, то счётчик обнуляется. В конце программы проверяется как раз этот счётчик, если 0-списки не одинаковы, иначе одинаковы.

procedureproverka(perv,perv1:ukazat);

var p:integer;

R,R1:ukazat;

namefile:string;

begin

R:=perv;

R1:=perv1;

p:=1; // счётчик

while (R^.next<>nil) and (p=1) do

begin

if (R^.nom\_reys<>R1^.nom\_reys) then // сравниваем рейсы по очереди

begin

p:=0; // если элементыне не одинаковые, зануляется на 0

end

else

begin

R:=R^.next;

R1:=R1^.next;

end;

end;

if p=1 then writeln('Спискиодинаковы')

else writeln('Спискиразличны');

end;

**Полный текст программы**

program Project1;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}{$IFDEF UseCThreads}

cthreads,

{$ENDIF}{$ENDIF}

Classes

{ you can add units after this };

type

ukazat = ^item; // указатель

item = record // указатель типа запись

punkt:string; // Пункт назначения

nom\_reys: integer; // номер рейса

fio: string; //ФИО

date: record

h: byte;

m: byte;

end;

next: ukazat; // указатель на следующий элемент

pred: ukazat; // указатель на предыдущий элемент

end;

{ Меню }

procedure menu(vardeystvie: integer);

begin

writeln();

writeln('Что вы хотите сделать:');

writeln('1. Заполнить список');

writeln('2. Показать список');

writeln('3. Добавить заявку');

writeln('4. Удаление заявки');

writeln('5. Вывод заявок по заданному номеру рейса и времени выезда;');

writeln('6. Длина списка ');

writeln('7. Проверка списка на пустоту ');

writeln('8. Переворот списка');

writeln('9. Деление на 2 списка');

writeln('10. Удаление всех элементов списка');

writeln('0. Завершить программу ');

writeln;

write('Ваш выбор: ');

readln(deystvie); // ввод нашего выбора

end;

{ заполнение списка (рандомно заполняет 10 элементов) }

procedurezapolnenie(varperv, konec: ukazat);

var q: ukazat;

simv: array[1..3] of integer;

punkt\_naznach: string;

h, m, n, i: integer;

avtor,name: string;

begin

n:=10;

whilen> 0 do // пока n больше 0, заполняем списки

begin

//заполняем автора

avtor := '';

simv[1] :=random(90 - 65 + 1) + 65 ; //первая заглавная буква

avtor := avtor + chr(simv[1]);

fori := 1 to (random(10) + 3) do //кол-во символов (до точки) в имени файла

begin

simv[2] := random(122 - 97 + 1) + 97; //нижний регистр

avtor := avtor + chr(simv[2]); //добавляем рандомный символ

end;

simv[1] :=random(90 - 65 + 1) + 65 ;

simv[2] :=random(90 - 65 + 1) + 65 ;

avtor :=avtor +' '+chr(simv[1])+ '.'+chr(simv[2])+'.';

punkt\_naznach := '';

punkt\_naznach := punkt\_naznach + chr(random(90 - 65 + 1) + 65);

fori := 1 to (random(10) + 3) do //кол-во символов (до точки) в имени файла

punkt\_naznach := punkt\_naznach + chr(random(122 - 97 + 1) + 97); //добавляем рандомный символ

h := random(23);

m := random(59);

// создаём указатель

New(q);

q^.next := nil;

q^.pred := nil;

q^.fio :=avtor;

q^.nom\_reys := random(5)+100; // рандомныйрейс

q^.date.h := h;

q^.date.m := m;

q^.punkt:=punkt\_naznach; // рандомныйпунктназначения

ifperv = nil then begin // еслисписокпуст

perv := q;

konec := q;

end

elsebegin // если имеются элементы в списке

konec^.next := q;

konec^.next^.pred := konec;

konec := q;

end;

n := n - 1;

end;

end;

{ Вывод на экран списка }

procedure Print(perv: ukazat);

var

R: ukazat;

i: integer;

begin

ifperv<> nil then begin // еслисписокнепустой

i := 1;

R :=perv;

while R <> nil do

begin

Writeln(i:2, '. ',

R^.punkt:18,' ' ,

R^.nom\_reys:4,' ',

R^.fio:20,' ',

R^.date.h:2, ':',

R^.date.m:2);

R :=R^.next;

Inc(i);

end;

end

else

writeln('Списокпуст');

end;

{ Получениедаты(времени) }

procedureget\_date(var hrs1, min1: integer);

var

data: string;

i, j, ost: integer;

begin

readln(data);

val(Copy(data,1,2),hrs1);

val(Copy(data,4,2),min1);

end;

{ Добавлениерейса }

procedureAddReys(varperv,konec:ukazat);

var

k,hrs1,min1:integer;

begin

ifperv=nil then begin

new(perv);

perv^.next:=nil;

perv^.pred:=nil;

konec:=perv;

write('Введитепунктназначения: '); readln(konec^.punkt);

write('Введитеномеррейса: '); readln(konec^.nom\_reys);

write('Введите FIO: '); readln(konec^.fio);

write('Введите время желаемого выезда: ');

get\_date(hrs1,min1);

konec^.date.h:=hrs1;

konec^.date.m:=min1;

end

else begin

new(konec^.next);

konec^.next^.pred:=konec;

konec:=konec^.next;

konec^.next:=nil;

write('Введитепунктназначения: '); readln(konec^.punkt);

write('Введитеномеррейса: '); readln(konec^.nom\_reys);

write('Введите FIO: '); readln(konec^.fio);

write('Введите время желаемого выезда: ');

get\_date(hrs1,min1);

konec^.date.h:=hrs1;

konec^.date.m:=min1;

end;

end;

{ Удалениезаписи }

Procedure DelElem(var spis1,spis2:ukazat;tmp:ukazat);

var

tmpi:ukazat;

begin

if (spis1=nil) or (tmp=nil) then

exit;

iftmp=spis1 then begin

spis1:=tmp^.next;

ifspis1<>nilthen {если список оказался не из одного элемента, то}

spis1^.pred:=nil {"зануляем" указатель}

else {в случае, если элемент был один, то}

spis2:=nil; {"зануляем" указатель конца списка, а указатель начала уже "занулён"}

dispose(tmp);

end

else

iftmp=spis2 thenbegin{если удаляемый элемент оказался последним элементом списка}

spis2:=spis2^.pred; {указатель конца списка переставляем на предыдущий элемент}

ifspis2<>nilthen {если предыдущий элемент существует,то}

spis2^.next:=nil {"зануляем" указатель}

else {в случае, если элемент был один в списке, то}

spis1:=nil; {"зануляем" указатель на начало списка}

dispose(tmp);

end

elsebegin{если же удаляется список не из начали и не из конца, то}

tmpi:=spis1;

whiletmpi^.next<>tmpdo {ставим указатель tmpi на элемент перед удаляемым}

tmpi:=tmpi^.next;

tmpi^.next:=tmp^.next; {меняемсвязи}

iftmp^.next<>nil then

tmp^.next^.pred:=tmpi; {у элементадоудаляемого и посленего}

dispose(tmp);

end;

end;

Procedure DelElemPos(var spis1,spis2:ukazat; posi:integer);

var i:integer;

tmp:ukazat;

begin

ifposi<1 then

exit;

if spis1=nil then begin

Write('Списокпуст');

exit

end;

i:=1;

tmp:=spis1;

while (tmp<>nil) and (i<>posi) do begin

tmp:=tmp^.next;

inc(i)

end;

iftmp=nil then begin

Writeln('Записи с порядковым номером ' ,posi, ' нет в списке.');

writeln('В списке всего ' ,i-1, ' элементов.');

exit

end;

DelElem(spis1,spis2,tmp);

Writeln('Элемент удалён.');

end;

{ вывод заявок по заданному номеру рейса и дате вылета; }

procedurepoisk\_zayavok(varperv,konec: ukazat);

var

R: ukazat;

hrs1,min1, prints, j,reys: integer;

s: string;

date,date\_fly,h,m: string;

begin

prints:=0;

s := '';

j := 1;

R :=perv;

if R <> nil then begin

write('Введитеномеррейса: ');readln(reys);

write('Введите время выезда: ');

get\_date(hrs1,min1);

while R <> nil do begin

if (hrs1 = R^.date.h) and

(min1 = R^.date.m) then begin

s:=s+'1';

Writeln(j:2, '. ',

R^.punkt:18,' ' ,

R^.nom\_reys:4,' ',

R^.fio:20,' ',

R^.date.h:2, ':',

R^.date.m:2);

j := j + 1;

R :=R^.next;

end

else begin

R :=R^.next;

j := j + 1;

end;

end;

if s='' then writeln('нетсоответствующихзаявок');

end

else

writeln('Списокпуст');

end;

{ Длинасписка }

proceduredlina\_sp(perv: ukazat; vardlina\_spiska: integer);

var k: integer;

R: ukazat;

begin

R :=perv;

k := 0;

while R <> nil do begin

R :=R^.next;

k := k + 1;

end;

dlina\_spiska := k;

end;

{ Проверка списка на пустоту }

procedureproverka\_spiska\_na\_pustotu(perv: ukazat);

begin

ifperv = nil then

writeln('списокпуст')

else

writeln('список не пуст');

end;

{ Переворотсписка }

procedureperevorot\_spiska(varperv,konec:ukazat);

varq,zn,w:ukazat;

begin

ifperv<>nil then begin

q:=perv;

w:=konec;

while q<>nil do begin

zn:=q^.next;

q^.next:=q^.pred;

q^.pred:=zn;

q:=zn;

end;

konec:=perv;

perv:=w;

end

else

writeln('списокпуст');

end;

{ Удалениевсехэлементов }

proceduredelete\_vseh\_el(varperv,konec:ukazat);

varq,R:ukazat;

begin

whileperv<>nil do begin

q:=perv;

perv:=perv^.next;

ifperv<>nil then

dispose(q)

else begin

konec:=nil;

dispose(q);

end;

end;

writeln('Списокочищен!' );

end;

{ Делениена 2 списка }

procedure delenie\_na\_2\_spiska(var perv,konec,perv1,konec1:ukazat);

var

q,i,dlina\_spiska:integer;

R:ukazat;

begin

R:=perv;

dlina\_sp(R,dlina\_spiska);

ifperv<>nilthen

begin

q:=dlina\_spiskadiv 2; // целочисленно делим основной список на 2

i:=1;

whilei<>q do

begin

R:=R^.next;

i:=i+1;

end;

konec1:=R;

R:=R^.next;

perv1:=R;

konec1^.next:=nil;

perv1^.pred:=nil;

end

else

writeln('Списокпуст, надозаполнить!!!');

end;

{ Меню при 2ух списках }

procedure menu1(var deystvie1:integer);

begin

writeln();

writeln('Что вы хотите сделать?:');

writeln('1. Показать 2 списка');

writeln('2. Проверка 2ух списков на равенство');

writeln('3. Соединение 2ух списков в один (основные действия только с одним списком)');

writeln();

write('Вашвыбор: ');

readln(deystvie1);

end;

{ Соединение 2ух списков в один }

procedure connect(konec1,perv1:ukazat);

begin

konec1^.next:=perv1; // конец 1ого на начало 2ого

perv1^.pred:=konec1; // конец 2ого на начало 1ого

end;

{ Проверка 2ух списков на равенство}

procedureproverka(perv,perv1:ukazat);

var p:integer;

R,R1:ukazat;

namefile:string;

begin

R:=perv;

R1:=perv1;

p:=1; // счётчик

while (R^.next<>nil) and (p=1) do

begin

if (R^.nom\_reys<>R1^.nom\_reys) then // сравниваем рейсы по очереди

begin

p:=0; // если элементыне не одинаковые, зануляется на 0

end

else

begin

R:=R^.next;

R1:=R1^.next;

end;

end;

if p=1 then writeln('Спискиодинаковы')

else writeln('Спискиразличны');

end;

var

deystvie, dlina\_spiska, deystvie1,i: integer;

perv, konec, perv1, konec1: ukazat;

begin

menu(deystvie);

writeln();

whiledeystvie<> 0 do

begin

casedeystvie of

1: begin

zapolnenie(perv, konec);

Print(perv);

end;

2: Print(perv);

3: begin

AddReys(perv,konec);

Print(perv);

end;

4:begin

Write('Введите порядковый номер удаляемой записи: ');

readln(i);

DelElemPos(perv,konec,i);

Print(perv);

end;

5: poisk\_zayavok(perv,konec);

6: begin

dlina\_sp(perv,dlina\_spiska);

writeln('Длинасписка: ',dlina\_spiska);

end;

7: proverka\_spiska\_na\_pustotu(perv);

8: begin

perevorot\_spiska(perv,konec);

print(perv);

end;

9: begin

dlina\_sp(perv,dlina\_spiska);

ifdlina\_spiska>=2 then begin

delenie\_na\_2\_spiska(perv,konec,perv1,konec1);

writeln('1ый список');

Print(perv);

writeln('2ой список');

Print(perv1);

menu1(deystvie1);

writeln();

while deystvie1<>3 do begin

case deystvie1 of

1: begin

writeln('1ый список');

Print(perv);

writeln();

writeln('2ой список');

Print(perv1);

end;

2: proverka(perv,perv1);

end;

menu1(deystvie1);

end;

connect(konec1,perv1);

end

else

writeln('Невозможно сделать это действие');

end;

10: delete\_vseh\_el(perv,konec);

end;

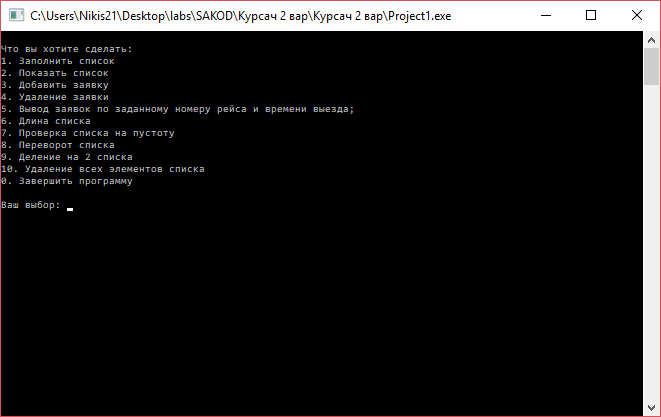
menu(deystvie);

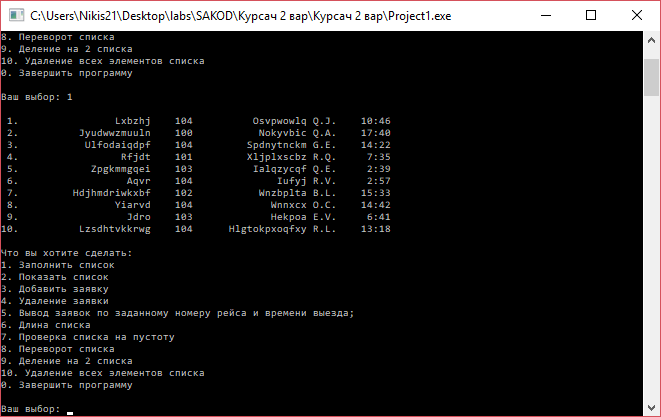
end;

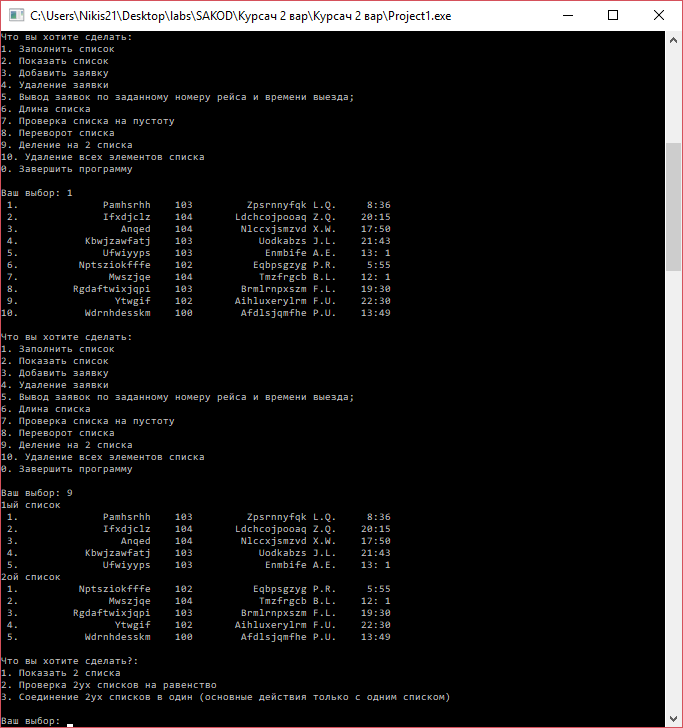
delete\_vseh\_el(perv,konec);

end.

**Результаты работы**







ЗАКЛЮЧЕНИЕ

Использование динамических структур данных эффективно применять при решении задач, так как каждому значению переменной выделяется какая-то область памяти, в ходе чего происходит учет ресурсов компьютера. При необходимости эту ячейку можно ликвидировать, если информация находящаяся в этой ячейке нам больше не понадобится.

Конечно, при решении задач с помощью статистических переменных обработка и доступ к информации облегчается. При использовании статистических структур данных происходит нерациональное использование оперативной памяти, потому что для статистических переменных выделяется фиксированный размер памяти.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Информатика и образование, №5 – 1999.
2. Бабушкина И.А., Бушмелева Н.А., Окулов С.М., Черных С.Ю. Конспекты по информатике. – Киров, 1997.
3. Грэхем Р., Кнут Д., Паташник О. Конкретная информатика. – М.: Мир, 1988.
4. Вирт Н., Алгоритм + структура данных = программа.
5. Райнтли, Абстракция и структура данных.
6. Зубов В