ФЕДЕРАЛЬННОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ УПРАВЛЕНИЯ И ПРЕДПРИНИМАТЕЛЬСТВА

КАФЕДРА ИНФОРМАЦИОННОГО МЕНЕДЖМЕНТА

Ломакин В.В.

РЕШЕНИЕ АЛГОРИТМИЧЕСКИХ ЗАДАЧ НА ЯЗЫКЕ ПАСКАЛЬ

практикум

Белгород 2008г.

УДК 681.3.068+800.92
Л74

Печатается по решению редакционно-издательского совета
Белгородского государственного университета

Рецензенты:

Кижук А.С., кандидат технических наук, доцент кафедры технической кибернетики Белгородского государственного технологического университета им. В.Г. Шухова

Михелев В.М., кандидат технических наук, доцент кафедры математического и программного обеспечения информационных систем Белгородского государственного университета

Л74 Ломакин Владимир Васильевич

Решение алгоритмических задач на языке Паскаль: Практикум/ Ломакин В.В. – Белгород: Изд-во БелГУ, 2008.-40с.

Практикум «Решение алгоритмических задач на языке Паскаль» представляет собой методические рекомендации, описывающие порядок и особенности выполнения лабораторных работ по программированию на языке Паскаль. Практикум состоит из восьми лабораторных работ, для каждой работы изложены теоретические сведения, представлена блок-схема алгоритма и приведена программа для решения задачи.

Практикум предназначен для студентов информационных специальностей высших учебных заведений.

УДК 681.3.068+800.92

Практикум (лабораторный)

Практикум состоит из восьми лабораторных работ.

В первой работе рассматриваются задачи программирования с применением множеств, определяются требования к содержанию, оформлению и порядку выполнения. Работ лабораторного практикума. Кратко рассмотрены теоретические вопросы.

Первая работа посвящена практической реализации программ с применением типа «множество». Приведен пример программы с применением типа «множество», кратко изложены теоретические вопросы.

Вторая работа рассматривает тип «запись». Приводятся примеры использования типа «запись». Кратко рассмотрены теоретические вопросы.

Третья работа рассматривает задачи программирования с использованием файлов.

В четвертой работе рассмотрены подходы к построению программ с применением рекурсии. Кратко изложены теоретические сведения, приведены блок-схемы и программный код.

В пятой работе рассмотрены подходы к решению задач обработки строковых данных и последовательностей данных. Кратко изложены теоретические сведения, приведены блок-схемы и программный код.

В шестой работе рассмотрены ссылочные типы и указатели.

В седьмой работе рассмотрен вопрос применения объектно-ориентированного программирования в решении геометрических задач.

В восьмой работе рассмотрены приемы программирования и блок-схемы алгоритмов при обработке списков с помощью указателей.

Для успешного выполнения лабораторных работ необходимо изучение соответствующих теоретических сведений по языку Паскаль.

Лабораторная работа №1. Задачи с применением множеств.

Цель: изучить особенности задач с применением множеств и основные подходы к разработке программ с применением типа множество.

1. Требования к содержанию, оформлению и порядку выполнения

Указанные в настоящей лабораторной работе требования относятся ко всем последующим лабораторным работам настоящего курса.

Лабораторная работа – небольшой отчет, обобщающий проведенную студентом работу, которую представляют для защиты преподавателю.

Отчет по лабораторной работе является учебным документом, выполненным студентом по учебному плану при изучении дисциплины.

Лабораторные работы, предусмотренные программой учебной дисциплины, выполняются обучающимися на лабораторных занятиях в соответствии с расписанием учебных занятий.

К выполнению лабораторных работ обучающиеся допускаются на основании результатов устного опроса, проводимого преподавателем в начале лабораторного занятия с целью выяснения степени освоения обучающимся рассматриваемого на лабораторных занятиях теоретического материала, цели, порядка выполнения и оформления результатов лабораторной работы.

Оформление отчета по лабораторной работе и его защита преподавателю осуществляются, как правило, в период лабораторного занятия.

Повторная защита обучающимся отчета по лабораторной работе допускается, как правило, в течение одной недели со дня ее выполнения на лабораторном занятии в соответствии с расписанием учебных занятий.

К лабораторным работам предъявляется ряд требований, основным из которых является полное, исчерпывающее описание всей проделанной работы, позволяющее судить о полученных результатах, степени выполнения заданий и профессиональной подготовке студентов.

В отчет по лабораторной работе должны быть включены следующие пункты:

- титульный лист;
- введение (содержит: цель работы; краткие теоретические сведения; условные обозначения, символы и сокращения);
- основная часть (должна содержать краткое описание используемых для проведения лабораторной работы технических средств, методики проведения работы, полученные результаты);
- заключение (с анализом результатов работы и выводами);
- библиографический список;
- приложение (если есть необходимость).

2. Теоретическая часть

Множество в математике – произвольный набор объектов, понимаемый как единое целое. На вид и число элементов множества не накладывается никаких ограничений. Понятие множества в языке Pascal существует уже, чем математическое понятие. Допускается ограниченное число типов данных. Это может быть любой тип, за исключением real и integer. Последнее ограничение связано с тем, что общее число элементов множества, в зависимости от компилятора колеблется от 128 до 512 элементов. Синтаксическая диаграмма для множественного типа приведена на рис.1.

Рис.1. Синтаксическая диаграмма множественного типа.

Пример:

```
type letter = 'A' … 'Z';
            var x:letter; ch:char;
            begin
{ нельзя вводить множество оператором ввода readln(x)}
```

…

Совокупность конкретных элементов множества задается с помощью конструктора множеств.

```
readln(ch);
x:=[ch]; – конструктор множества
[]; – пустое множество
[i+1;j] – множество состоящее из двух элементов
[0…9];
['A','B','C'];
```

3.1.Операции над множествами

Если X – переменное множество, то X:=E; допускается только в том случае, если тип E относится к базовому типу X. Над множествами допустимы следующие операции:

Если A и B множества, то A+B означает объединение множеств;

A-B – разность множеств, т.е. множество элементов A, не входящих в B;

A*B – пересечение множеств.

К множественным операндам применимы следующие операции:

Если e – ординальное выражение базового типа, то e in A = true, если e входит в A.

A=B – равенство множеств;

A<>B – неравенство множеств;

A<=B=true, если A – подмножество B;

A>=B=true, если B – подмножество A.

Множество нельзя выводить с помощью оператора вывода, но можно следующим образом:

```
ch:char;
a:set of 'A'..'Z';
{текст программы}
```

```
For ch:='A' to 'Z' do
If ch in a then writeln(ch).
```

3. Пример выполнения работы

Дан текст, за которым следует точка. В алфавитном порядке напечатать все строчные согласные буквы, входящие в этот текст, а затем глухие согласные.

Блок-схема программы приведена на рис.2.1. Текст программы приведен на рис.2.2.

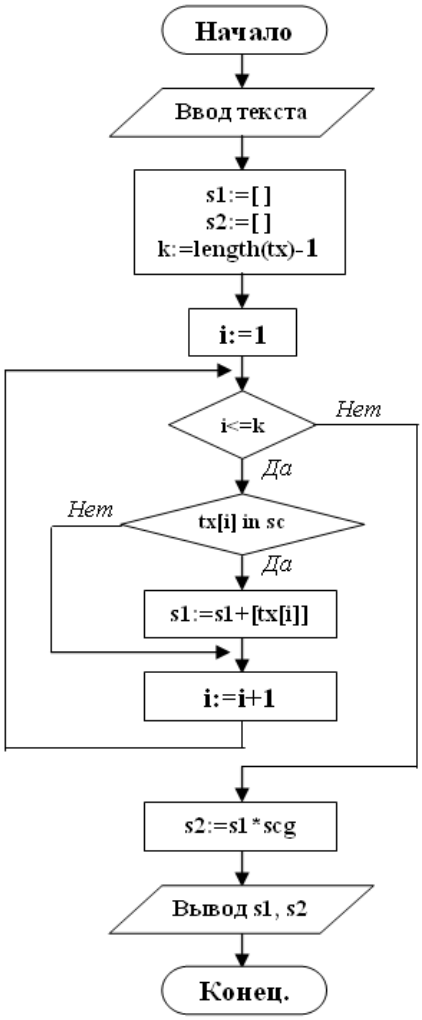


Рис.2.1. Блок-схема программы

```
program p1;
const
     sc:set of char=
['б','в','г','д','ж','з','к','л','м','н','п'
,'р','с','т','ф','х','ц','ч','ш','щ'];
     scg:set of char=
['к','п','с','т','ф','х','ц','ч','ш','щ'];
var
     s1,s2:set of char;
     k,i:integer;
     tx:string;
     c:char;
begin
     writeln('Введите текст');
     readln(tx);
     s1:=[];
     s2:=[];
     k:=length(tx)-1;
     for i:=1 to k do if tx[i] in sc then
s1:=s1+[tx[i]];
     s2:=s1*scg;
     writeln('Согласные буквы, входящие в
текст:');
     for c:='б' to 'щ' do if c in s1 then
write(c,' ');
     writeln;
     writeln('Глухие согласные буквы,
входящие в текст:');
     for c:='к' to 'щ' do if c in s2 then
write(c,' ');
     writeln;
end.
```

Рис.2.2. Листинг 1

4. Способ оценки результатов

Работа оценивается по принципу «зачёт - не зачет».

Не зачет – допущена хотя бы одна ошибка в индивидуальном задании – 0 баллов.

Зачет – отсутствуют какие либо ошибки – 100 баллов.

Лабораторная работа №2. Тип «Запись».

Цель: изучить особенности задач с применением записей и основные подходы к разработке программ с применением типа запись.

1. Требования к содержанию, оформлению и порядку выполнения

Требования к настоящей лабораторной работе соответствуют требованиям указанным в разделе 1 Лабораторной работы №1.

2. Теоретическая часть

Записи состоят из фиксированного числа компонент, называемых полями. Поля могут быть различных типов. В отличие от массива, где тип элементов один и применяется вычисляемый доступ к компонентам, в записи вычисляемый выбор не приемлем. Полям записи присваиваются имена полей, которые затем используются для выбора их значений.

Пример:

```
type   compex=record
                    re:integer;
                    im:integer;
       end;
```

или

```
type   compex=record
             re,im:integer;
       end;
```

Тип запись определяет каким образом в дальнейшем будет выделяться память под переменную типа запись, т.е. для обращения с записями в программе необходимо описывать соответствующую переменную типа запись. Для обращения к полям используется идентификатор переменных типа запись, за которым ставится точка, а затем идентификатор поля.

Пример: zapis.pole1.pole2

При обращении с полями часто используются одинаковые или длинные имена. Для сокращенной записи составных имен используют оператор присоединения with. Его синтаксическая диаграмма изображена на рис.3.

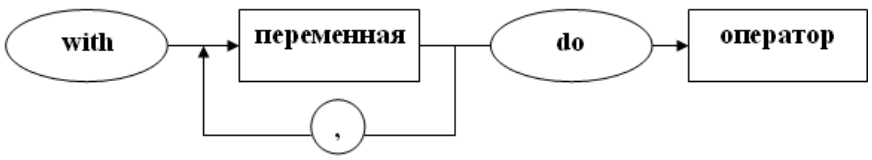


Рис.3. Синтаксическая диаграмма оператора присоединения with.

Пример:

```
with X do begin    re:=1.5;    im:=3.6     end.
```

3. Пример выполнения работы

Написать программу, выводящую сведения о самом старшем мужчине в группе (считать, что такой есть и он единственный). Структыра данных задается следующим описанием типов:

```
type
      date=record
                d:1..31;
                m:1..12;
                y:1900..1990;
      end;
      anketa=record
                  family:string[20];
                  pol:(mn,w);
                  bisday:date;
      end;
      group:array[1..10] of anketa;
```

Блок-схема программы приведена на рис.4.1.

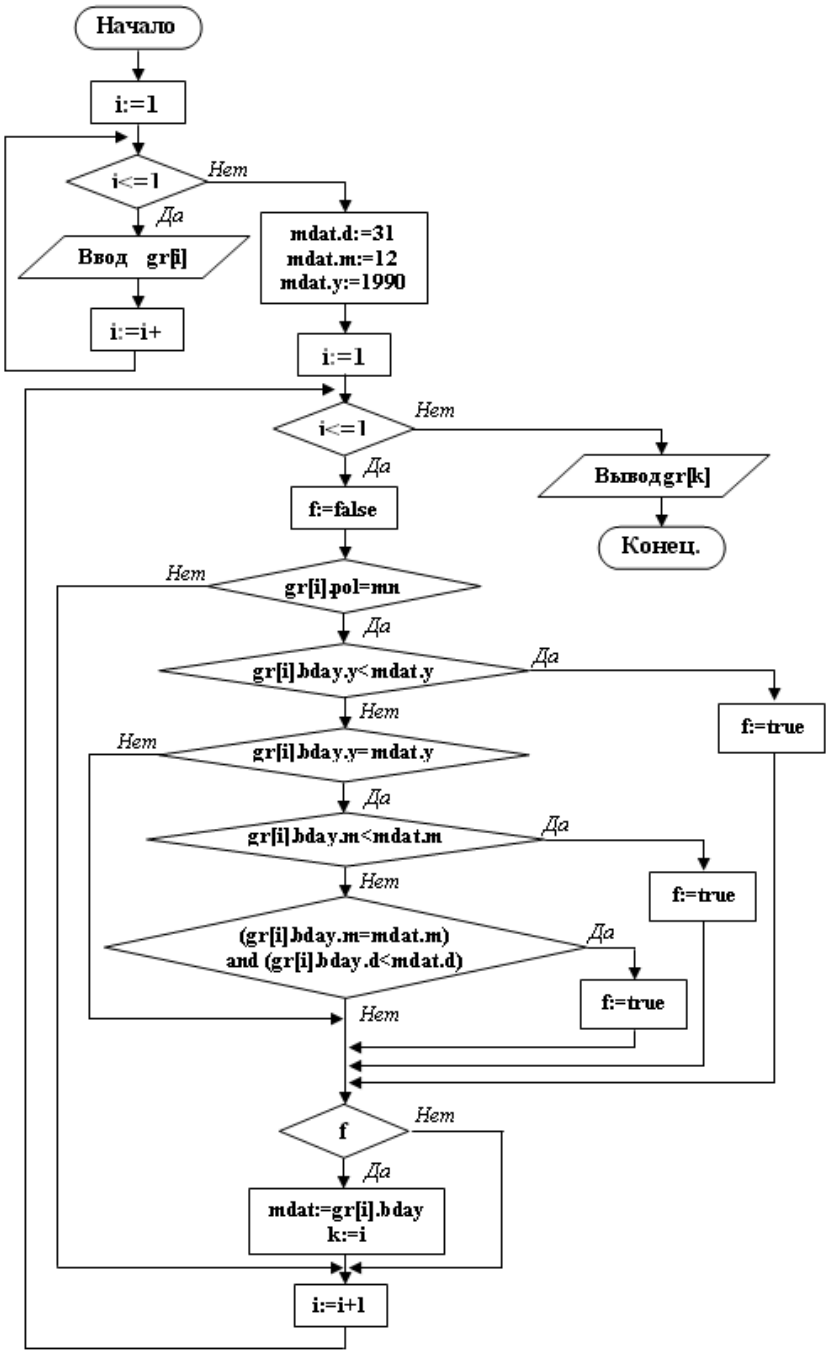


Рис.4.1. Блок-схема программы

Текст программы приведен на рис.4.2.

```
program p2;
type
     date=record
               d:1..31;
               m:1..12;
               y:1900..1990;
     end;
     anketa=record
                 family:string[20];
                 pol:(mn,w);
                 bday:date;
     end;
     group=array[1..10] of anketa;
var
     gr:group;
     i,k:integer;
     p:char;
     mdat:date;
     f:boolean;
begin
     writeln('Введите сведения о группе:');
     for i:=1 to 10 do
     with gr[i] do
     begin
          writeln(i);
          write('Фамилия: ');
          readln(family);
          write('Пол (м,ж):  ');
          readln(p);
          if p='м' then pol:=mn
          else pol:=w;
          write('Дата рождения (формат: dd m yyyy):  ');
          with bday do readln(d,m,y);
     end;
     with mdat do
     begin
          d:=31;
          m:=12;
          y:=1990;
     end;
     for i:=1 to 10 do
     begin
          f:=false;
          if gr[i].pol=mn then
          begin
               with gr[i].bday do
                    if y<mdat.y then f:=true
                    else if y=mdat.y then
                            if m<mdat.m then f:=true
                            else   if   (m=mdat.m)   and   (d<mdat.d)   then
f:=true;
               if f then
               begin
                    mdat:=gr[i].bday;
                    k:=i;
               end;
          end;
     end;
     writeln;
     writeln('Самый старший мужчина в группе:');
     with gr[k] do
     begin
          writeln('Фамилия:  ',family);
          with bday do writeln('Дата рождения:  ',d,' ',m,' ',y);
     end;
end.
```

Рис.4.2. Листинг 2.

4. Способ оценки результатов

Работа оценивается по принципу «зачёт - не зачет».

Не зачет – допущена хотя бы одна ошибка в индивидуальном задании – 0 баллов.

Зачет – отсутствуют какие либо ошибки – 100 баллов.

Лабораторная работа №3. Задачи с использованием файлов.

Цель: изучить особенности задач с применением файлов и основные подходы к разработке программ с файлового типа данных.1. Требования к содержанию, оформлению и порядку выполнения

1. Требования к содержанию, оформлению и порядку выполнения

Требования к настоящей лабораторной работе соответствуют требованиям указанным в разделе 1 Лабораторной работы №1.

2. Теоретическая часть

Понятие файла означает поименованную совокупность данных, состоящую из логических записей, относящихся к одной теме. Важным свойством файла является его целостность.

Набор данных – термин, означающий поименованную совокупность данных, объединенных общим назначением и имеющую конкретную физическую организацию.

Синтаксическая диаграмма для файлового типа приведена на рис.3.

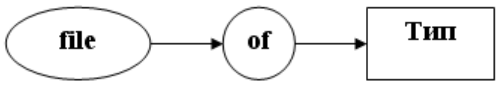


Рис.5. Синтаксическая диаграмма файлового типа данных.

Пример:

```
type  file_real=file of real;
      var   F:file_real;
```

Переменная файлового типа используется в процедурах read(F,…) и write(F,…) в качестве первого параметра, указывающего с каким файлом будут работать процедуры.

Для работы с файлами предусмотрены следующие процедуры:

eof(F) – конец файла, выдает true, если F находится в режиме формирования или до обращения был установлен в последнюю позицию;

reset(F) – подготавливает файл для чтения, помещая указатель на первую компоненту файла;

rewrite(F) – подготавливает файл для записи, старое содержимое отбрасывается и можно формировать новый файл;

assign(F,<имя файла>) – процедура использующаяся для связи файловой переменной и имени конкретного файла;

close(F) – закрытие файла, рекомендуется осуществлять по окончанию работы с файлом в программе;

seek(F,<номер компоненты>) – перемещение по файлу на указанную компоненту;

truncate(F) – используется для отсечения от файла его хвостовой части;

filesize(F) – общее число компонент файла (не размер);

filepos(F) – номер компоненты, на которой установлен текущий указатель.

3. Пример выполнения работы

Задача: 1) Сформировать текстовый файл; 2) Задана литера ch. Написать программу, которая заменяет в файле все элементы, равные ch, на "/".

Блок-схема программы приведена на рис.6.

Текст программы приведен на рис.4.2.

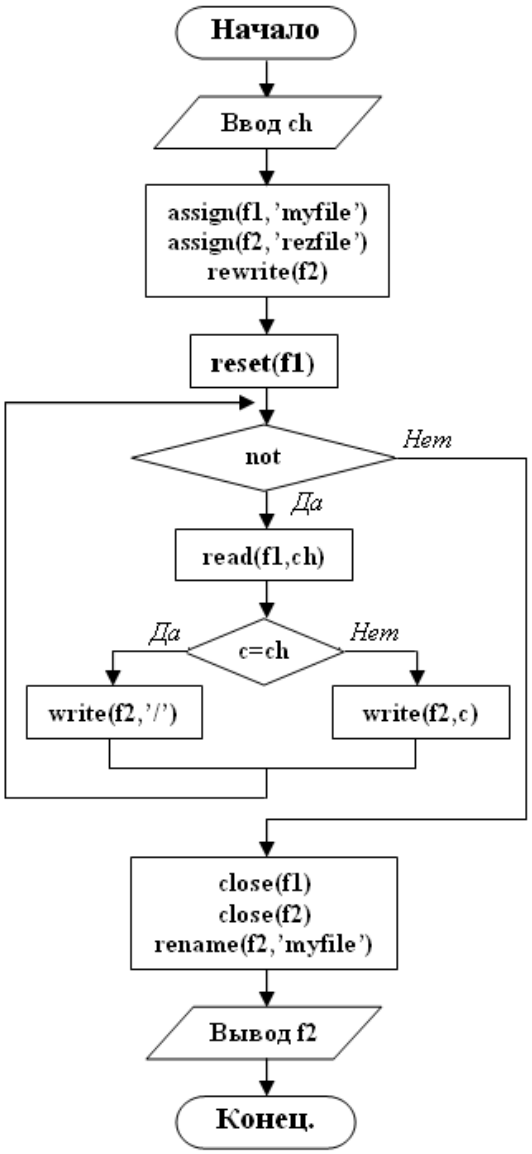


Рис.6.1. Блок-схема программы

```
program p3_1;
var
     f:text;
     c:char;
begin
     writeln('Введите текст, признак конца
текста - точка');
     assign(f,'myfile');
     rewrite(f);
     repeat
          read(c);  write(f,c);
     until c='.';
     close(f);
end.
```

Рис.6.2. Блок-схема программы

```
program p3_2;
var
     f1,f2:text;
     c,ch:char;
begin
     writeln('Введите литеру, которую
следует заменить:');
     write('ch=');
     readln(ch);
     assign(f1,'myfile');
     assign(f2,'rezfile');
     rewrite(f2);
     reset(f1);
     while not eof(f1) do
     begin
          read(f1,c);
          if c=ch then write(f2,'/')
          else write(f2,c);
     end;
     close(f1); close(f2); erase(f1);
     rename(f2,'myfile');
     writeln('Преобразованный файл:');
     reset(f2);
     while not eof(f2) do
     begin
          read(f2,c);  write(c);
     end;
     writeln;
     close(f2);
end.
```

4. Способ оценки результатов

Работа оценивается по принципу «зачёт - не зачет».

Не зачет – допущена хотя бы одна ошибка в индивидуальном задании – 0 баллов.

Зачет – отсутствуют какие либо ошибки – 100 баллов.

Лабораторная работа №4. Рекурсия.

Цель: изучить особенности задач с применением рекурсии и основные подходы к разработке программ с рекурсивными процедурами и функциями.

1. Требования к содержанию, оформлению и порядку выполнения

Требования к настоящей лабораторной работе соответствуют требованиям указанным в разделе 1 Лабораторной работы №1.

2. Теоретическая часть

Вызов процедуры или функции в тексте самой этой процедуры или функции прямо или косвенно означает рекурсивное выполнение процедуры или функции. Доказано, что во всех случаях рекурсия может быть заменена итерацией, т.е. циклом. Не следует путать рекурсивный вызов подпрограммы с определением процедуры или функции в теле другой. В этом случае процедура или функция называется вложенной.

Пример: найти f(f(x)) {n раз}, где f(x)=ex-sin(x)

```
Program     recurs;
      var   z:real;
            n:integer;
function    F(x:real; par:integer):real;
      var   y:real;
begin
      y:=exp(x)-sin(x);
      if par>1 then F:=F(y,par-1)
            else F:=y
end;
begin
      readln(z,n);
      writeln(F(z,n))
end.
```

При использовании рекурсии необходимо убедиться в том, что в рекурсивной подпрограмме присутствует условие выхода из рекурсии и что это условие обязательно достигается, в противном случае мы получим бесконечные вызовы рекурсивных подпрограмм, пока не будет исчерпана память.

3. Пример выполнения работы

Во входном файле задан текст, за которым следует точка. Проверить, является ли этот текст правильной записью "формулы":

<формула>::=<цифра>|(<формула><знак><формула>)

<знак>::=+|-|*

<цифра>::=0|1|2|3|4|5|6|7|8|9.

Блок-схема программы приведена на рис.7.

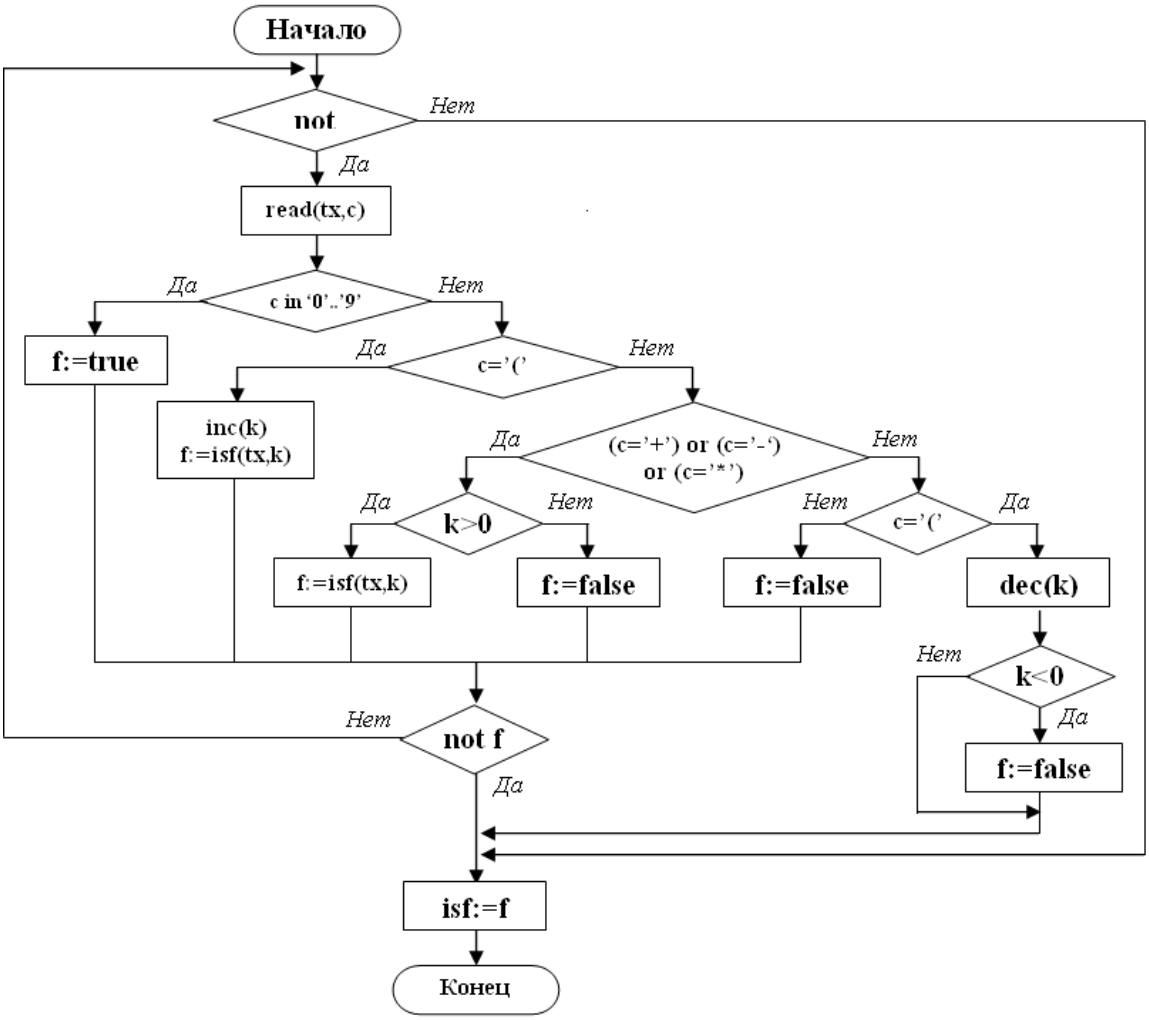


Рис.7.1. Блок-схема программы

Текст программы приведен на рис.7.2.

```
program p4;
var
     file_:text;
     ch:char;
     n:integer;
function isf(var tx:text;var k:integer):boolean;
var
     c:char;
     f:boolean;
begin
     while not eof(tx) do
     begin
          read(tx,c);
          case c of
               '0'..'9','.': f:=true;
               '(': begin
                         inc(k);
                         f:=isf(tx,k);
                    end;
               '+','-','*': if k>0 then f:=isf(tx,k)
                            else f:=false;
               ')': begin
                         dec(k);
                         if k<0 then f:=false;
                         break;
                    end;
              else f:=false;
```

```
          end;
          if not f then break;
     end;
     isf:=f;
end;
begin
     assign(file_,'myfile_');
     writeln('Введите выражение (признак конца - точка):');
     rewrite(file_);
     while ch<>'.' do
     begin
          read(ch);
          write(file_,ch);
     end;
     close(file_);
     reset(file_);
     n:=0;
     if isf(file_,n) then writeln('Данное выражение является записью
формулы')
     else writeln('Данное выражение не является записью формулы');
     close(file_);
end.
```

Рис.7.2. Листинг 4.

4. Способ оценки результатов

Работа оценивается по принципу «зачёт - не зачет».

Не зачет – допущена хотя бы одна ошибка в индивидуальном задании – 0 баллов.

Зачет – отсутствуют какие либо ошибки – 100 баллов.

Лабораторная работа №5. Строковый тип, последовательности.

Цель: изучить особенности задач с применением строкового типа и последовательностей и основные подходы к разработке программ со строковым типом.

1. Требования к содержанию, оформлению и порядку выполнения

Требования к настоящей лабораторной работе соответствуют требованиям указанным в разделе 1 Лабораторной работы №1.

2. Теоретическая часть

Строки постоянной длины рассматриваются как константы типа определенного как packed array[1..N] of char. Если строки A и B относятся к одному и тому же строковому типу, то возможно присваивание A:=B. Если в строках одно и тоже число символов, то возможно использование операторов сравнения: ‘ABCD’ < ’ABDC’. Строки сравниваются побуквенно, начиная с первой буквы. Фактически сравниваются коды (порядковые номера).

Использование строк постоянной длины приводит к неудобству, т.к. часто приходится иметь дело со строками разной длины. Поэтому в Turbo Pascal для реализации строкового типа используются строки переменной длины типа String. Этот тип определяет множество символьных цепочек произвольной длины от 0 до 255 символов.

Пример:

```
type  str=string[50];
      my_str=string;
var   s1:str;
      s2:my_str;
      s3:string;
begin
      s1:='СТРОКА';
      s2:='СИМВОЛОВ';
      s3:=s1+' '+s2;
writeln(s3)
end.
```

Операция «+» над строками означает объединение строк. Кроме «+» определены операции сравнения <, <=, >, >=, =, <>. При сравнении строк типа string действует следующее правило:

Более короткая строка – меньше;

Если длины строк равны, то производится поэлементное сравнение;

Turbo Pascal имеет следующие функции для работы со строками:

Length(s) – длина строки s;

Copy(s,index,count) – копирует из строки s, count символов, начиная с символа с номером index;

Delete(s,index,count) – удаляет count символов из строки s, начиная с символа с номером index;

Insert(sub_s,s,index) – вставляет подстроку sub_s в строку s, начиная с символа с номером index;

Pos(sub_s,s) – возвращает позицию начала заданной подстроки sub_s в строке s.

3. Пример выполнения работы

Задача: 1) Сформировать текстовый файл; 2) Задана литера ch. Написать программу, которая заменяет в файле все элементы, равные ch, на "/".

Блок-схема программы приведена на рис.8.1.

Текст программы приведен на рис.8.2.

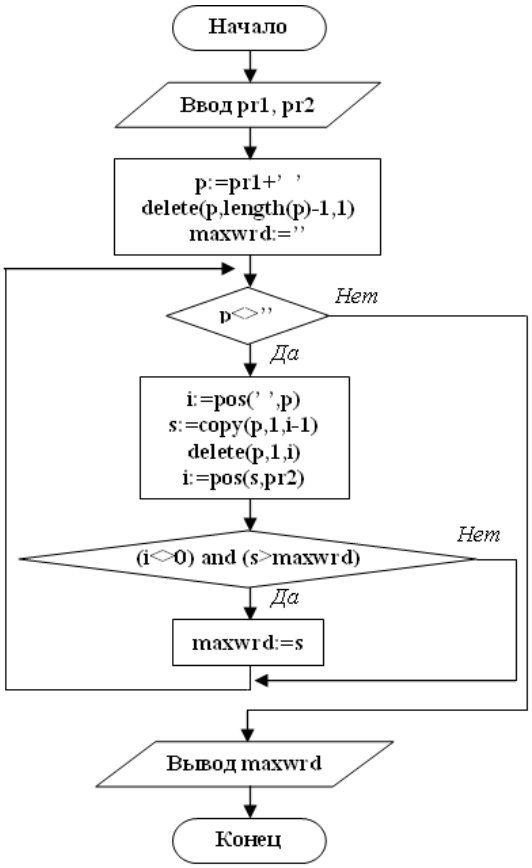


Рис.8.1. Блок-схема программы

```
program p5;
var
     pr1,pr2,p,s,maxwrd:string;
     i:integer;
begin
     writeln('Введите          первое
предложение:');
     readln(pr1);
     writeln('Введите          второе
предложение:');
     readln(pr2);
     p:=pr1+' ';
     delete(p,length(p)-1,1);
     maxwrd:='';
     while p<>'' do
     begin
          i:=pos(' ',p);
          s:=copy(p,1,i-1);
          delete(p,1,i);
          i:=pos(s,pr2);
          if (i<>0) and (s>maxwrd)
then maxwrd:=s;
     end;
     write('Самое    длинное    слово
данных предложений:  ');
     writeln(maxwrd);
end.
```

Рис.8.2. Листинг 5.

4. Способ оценки результатов

Работа оценивается по принципу «зачёт - не зачет».

Не зачет – допущена хотя бы одна ошибка в индивидуальном задании – 0 баллов.

Лабораторная работа №6. Ссылочный тип, указатели.

Цель: изучить особенности задач с применением указателей и основные подходы к разработке программ с ссылочным типом данных.

1. Требования к содержанию, оформлению и порядку выполнения

Требования к настоящей лабораторной работе соответствуют требованиям указанным в разделе 1 Лабораторной работы №1.

2. Теоретическая часть

Переменные бывают двух типов: статические и динамические.

Память для статических переменных отводится при загрузке программы в память, т.е. перед выполнением программы. Для определения идентификатора переменной и объема выделяемой памяти статические переменные необходимо описывать (var). Отведенная память сохраняется за переменной на все время выполнения модуля, в котором она локализована. Для динамических переменных память отводится и уничтожается в процессе выполнения программы. Динамические переменные не указываются в описании переменных, а порождаются и уничтожаются с помощью предопределенных процедур New и Dispose. Описание ссылочного типа (указателя) имеет вид: ссыл.тип=^идентиф. типа;

Пример:

```
type   p_integer=^integer;
var    p:p_integer;          {p имеет тип – указатель на integer}
```

p^ - динамическая переменная

Если p=nil или неопределенно, то обращение p^ - ошибка.

Если указателю присваивается nil, то он ни на что не ссылается.

3. Пример выполнения работы

```
const d=…{длина строки}
      n=…{макс. число строк}
 type file_string=string[d];
      pfile_string=^file_string;
      str_text=array[1..n] of file_string;
```

Используя данное представление текста, написать программу, использующую логическую функцию search_lit(T,c,i,j) определяющую входит ли литера с в тексте с в тексте T, и, если входит, присваивающую параметрам i и j «координаты» первого вхождения этой литеры: i – номер строки, j – номер позиции в этой строке.

Блок-схема программы приведена на рис.9.1.

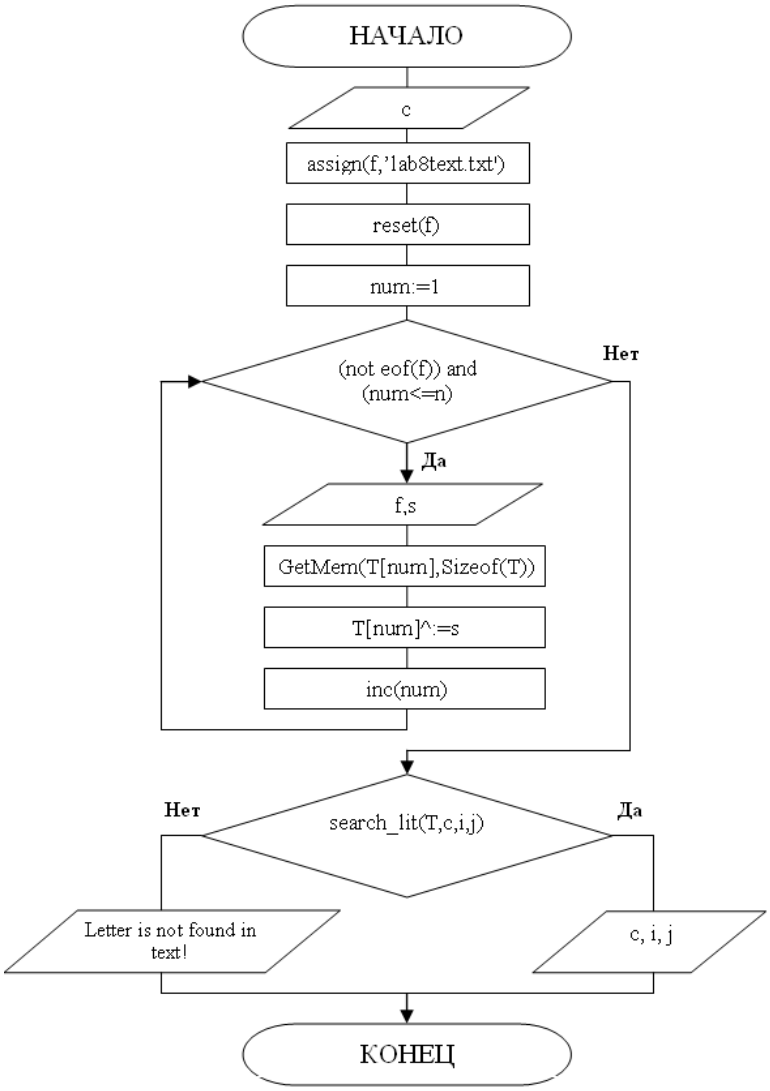


Рис.9.1. Блок-схема программы

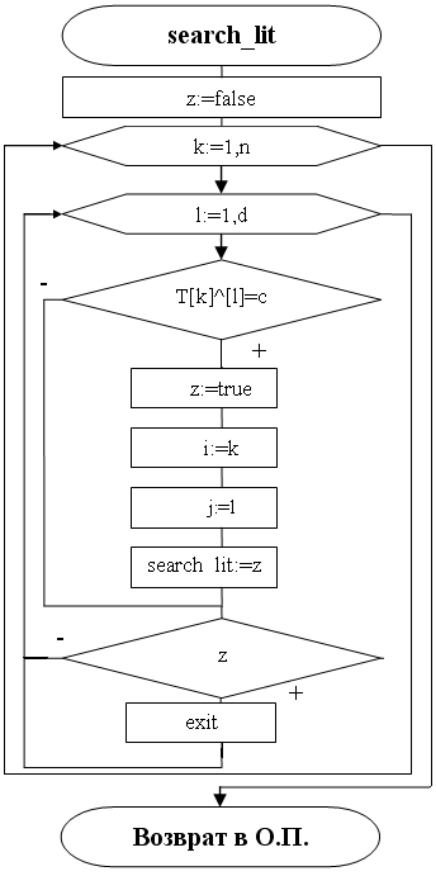


Рис.9.2. Блок-схема функции search_lit

Блок-схема функции search_lit приведена на рис.9.2.

Текст программы приведен на рис.9.3.

```
uses crt;
const d=30;
      n=3;
type
    file_string=string[d];
    pfile_string=^file_string;
    str_text=array[1..n] of pfile_string;
var
  i,j,num:integer;
  T:str_text;
  c:char;
  f:text;
  s:string;
function search_lit (T:str_text; c:char; var i,j:integer):boolean;
  var z:boolean;
      k,l:integer;
  begin
    z:=false;
    for k:=1 to n do
      for l:=1 to d do
```

```
        begin
          if T[k]^[l]=c then begin
                                z:=true;
                                i:=k;
                                j:=l;
                                search_lit:=z
                             end;
          if z then exit;
        end;
  end;
begin
clrscr;
  writeln('Input letter');
  readln(c);
  assign(f,'lab8text.txt');
  reset(f);
  num:=1;
  while (not eof(f)) and (num<=n) do
    begin
      readln(f,s);
      GetMem(T[num],Sizeof(T));
      T[num]^:=s;
      inc(num)
    end;
  if search_lit(T,c,i,j) then
    begin
      writeln('Letter ',c,' is in the text');
      writeln('at ',i,' string  ',j,' position')
    end
  else writeln('Letter is not found in text!');
readkey
end.
```

Рис.10.3. Листинг 6.

4. Способ оценки результатов

Работа оценивается по принципу «зачёт - не зачет».

Не зачет – допущена хотя бы одна ошибка в индивидуальном задании – 0 баллов.

Зачет – отсутствуют какие-либо ошибки – 100 баллов.

Лабораторная работа №7. ООП в решении геометрических задач.

Цель: изучить особенности задач с применением ООП и основные подходы к разработке программ с использованием ООП.

1. Требования к содержанию, оформлению и порядку выполнения

Требования к настоящей лабораторной работе соответствуют требованиям указанным в разделе 1 Лабораторной работы №1.

2. Теоретическая часть

Объекты моделируют свойства и «поведение» компонентов реального мира. Они являются конечной абстракцией данных. Язык ООП характеризуют 3 основных свойства:

Инкапсуляция – комбинирование записей с процедурами и функциями, которые манипулируют этой записью для получения нового типа объекта. Причем процедуры и функции описывают определенные действия, а их реализация вынесена за описание объектов;

Пример: шар
x,y,z – координаты центра
R – радиус
перемещение
вращение

Наследование - это определение объекта и затем, использование его для построения иерархии произвольных объектов, причем каждый производный объект (потомок) наследует доступ к коду и данным всех своих прародителей.

Пример: группа шаров (прародитель – шар)
{координаты центра x,y,z – не описываются, т.к. можно наследовать от прародителя}
……
перемещение
вращение

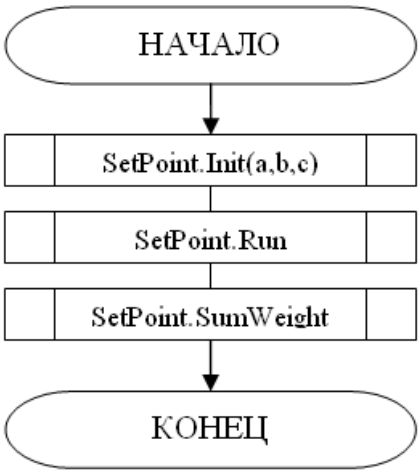
Полиморфизм – это придание одного имени, которое совместно используется объектами всей иерархии, причем каждый объект реализует это действие своим подходящим для него образом.

3. Пример выполнения работы

В трехмерном пространстве задано множество материальных точек с макс. массой исчезает, теряя десятую часть своей массы и раздавая оставшуюся массу поровну всем остальным более легким точкам. Определить суммарную массу множества материальных точек в тот момент, когда все оставшиеся в нем точки имеют одинаковую массу.

Блок-схема программы приведена на рис.11.

Блок-схема процедуры TSetPoint.Init приведена на рис.12.



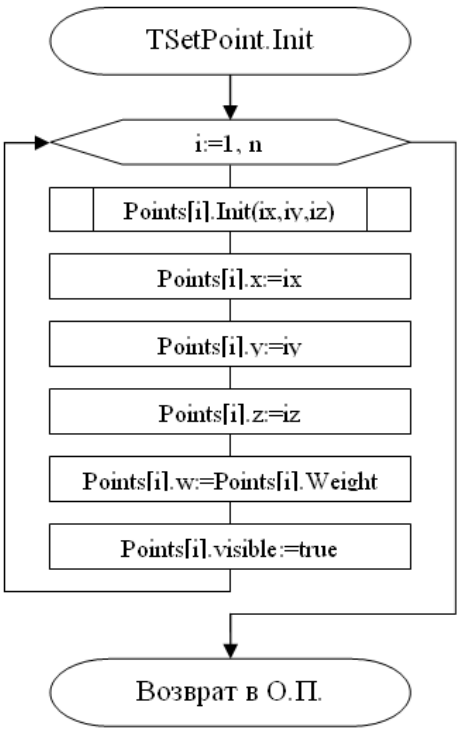


Рис.11. Блок-схема программы.

Рис.12. Блок-схема процедуры SetPoint.Init.

Блок-схема процедуры TSetPoint.Run приведена на рис.13.

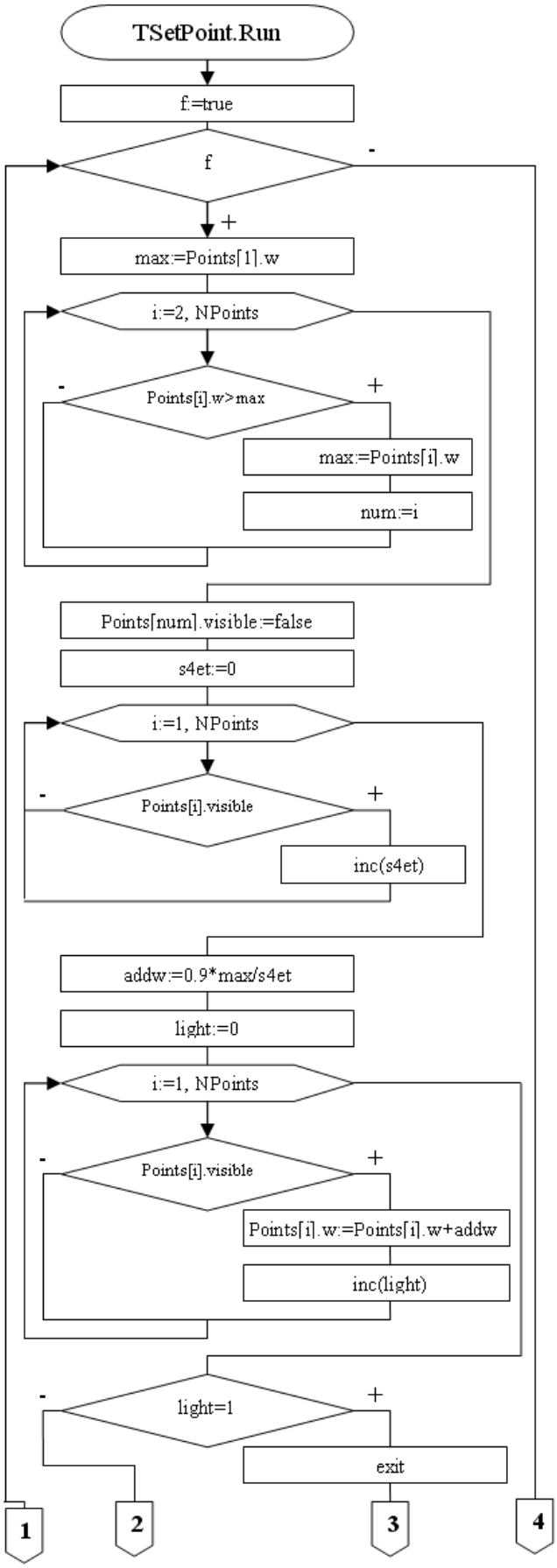


Рис.13. Блок-схема программы (начало).

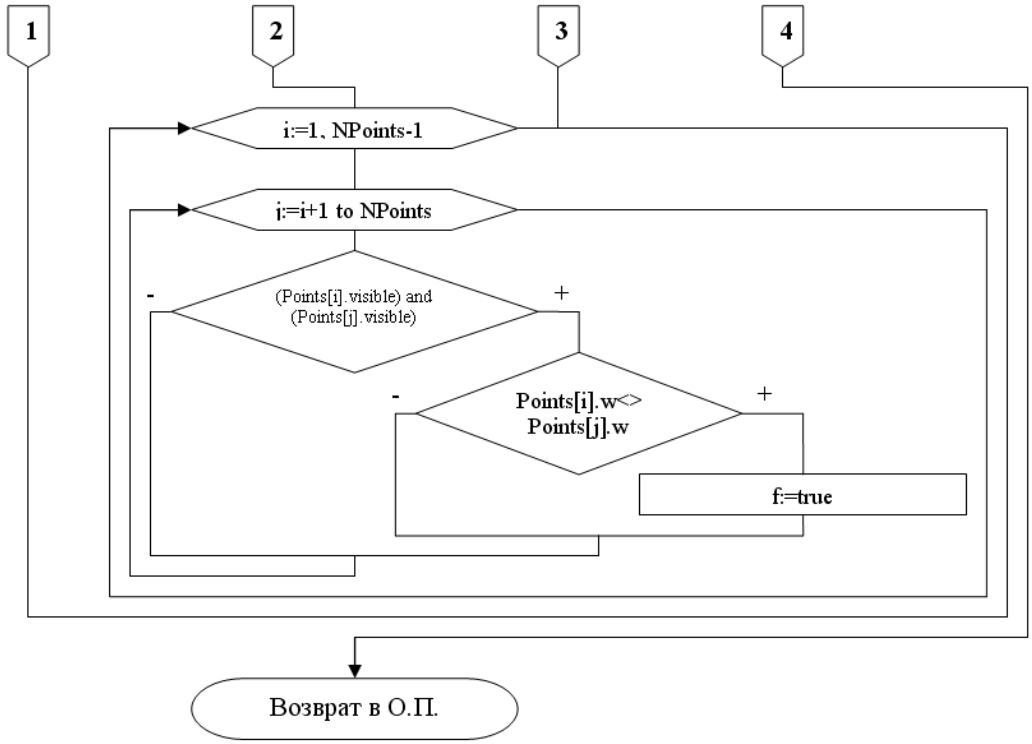


Рис.13. Блок-схема программы (окончание).

Блок-схема процедуры TSetPoint.SumWeight приведена на рис.14.

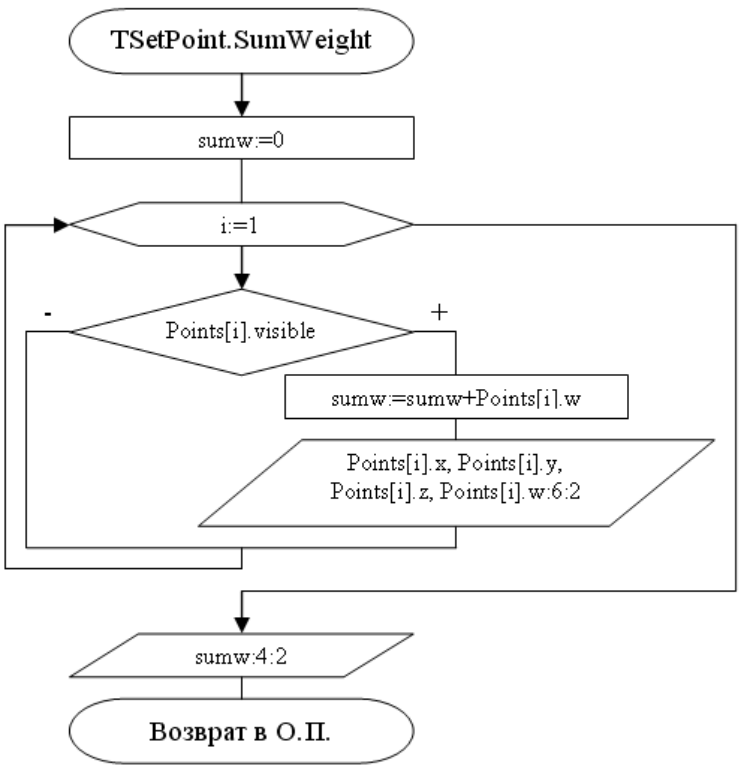


Рис.14. Блок-схема процедуры TSetPoint.SumWeight

Текст программы приведен в Листинге 7

```
 uses crt;
const NPoints=3;
type TPoint=object
       x,y,z:integer;
       w:real;
       visible:boolean;
       Procedure Init(var initx,inity,initz:integer);
       Function Weight:real;
     end;
     TSetPoint=object(TPoint)
       Points:array[1..NPoints] of TPoint;
     Constructor Init(var initx,inity,initz:integer);
     Procedure SumWeight;
     Procedure Run;
     end;
Procedure TPoint.Init(var initx,inity,initz:integer);
begin
  read(initx,inity,initz)
end;
Function TPoint.Weight:real;
var tmp_w:real;
begin
  read(tmp_w);
  Weight:=tmp_w;
end;
Constructor TSetPoint.Init(var initx,inity,initz:integer);
var i:integer;
    ix,iy,iz:integer;
begin
  writeln('Input coords & weights:');
  writeln('X  Y  Z');
  for i:=1 to NPoints do
    begin
      Points[i].Init(ix,iy,iz);
      Points[i].x:=ix;
      Points[i].y:=iy;
      Points[i].z:=iz;
      Points[i].w:=Points[i].Weight;
      Points[i].visible:=true
    end;
end;
Procedure TSetPoint.SumWeight;
var i:integer;
    sumw:real;
begin
  sumw:=0;
  writeln('It remains points with coords & weights:');
  for i:=1 to NPoints do
    if Points[i].visible then
      begin
        sumw:=sumw+Points[i].w;
        writeln('   ',Points[i].x,'   ',Points[i].y,'   ',Points[i].z,'   w
=',Points[i].w:6:2)
      end;
  Writeln('Sum weight is ',sumw:4:2)
end;
Procedure TSetPoint.Run;
var i,j,num,s4et,light:integer;
    max,addw:real;
    f:boolean;
begin
  f:=true;
  while f do
    begin
      max:=Points[1].w;
      for i:=2 to NPoints do
        if Points[i].w>max then begin
                                  max:=Points[i].w;
                                  num:=i
```

```
                                end;
      Points[num].visible:=false;
      s4et:=0;
      for i:=1 to NPoints do
        if Points[i].visible then inc(s4et);
      addw:=0.9*max/s4et;
      light:=0;
      for i:=1 to NPoints do
        if Points[i].visible then
          begin
            Points[i].w:=Points[i].w+addw;
            inc(light)
          end;
      if light=1 then exit
       else
         for i:=1 to NPoints-1 do
           for j:=i+1 to NPoints do
             if (Points[i].visible) and (Points[j].visible) then
               if Points[i].w<>Points[j].w then f:=true
    end;
end;
var SetPoint:TSetPoint;
    a,b,c:integer;
BEGIN
clrscr;
  SetPoint.Init(a,b,c);
  SetPoint.Run;
  SetPoint.SumWeight;
readkey
END.
```

Рис.14.1. Листинг 7.

4. Способ оценки результатов

Работа оценивается по принципу «зачёт - не зачет».

Не зачет – допущена хотя бы одна ошибка в индивидуальном задании – 0 баллов.

Зачет – отсутствуют какие либо ошибки – 100 баллов.

Лабораторная работа №8. Обработка списков при помощи указателей.

Цель: изучить особенности задач с применением списков и основные подходы к разработке программ связанных с обработкой списков.

1. Требования к содержанию, оформлению и порядку выполнения

Требования к настоящей лабораторной работе соответствуют требованиям указанным в разделе 1 Лабораторной работы №1.

2. Теоретическая часть

Динамические структуры данных – это те, которые при выполнении программы логически увеличиваются или уменьшаются в величине.

Рассмотрим три вида динамических структур данных:

Стек

Очередь

Список

Стек – это линейный список с одной точкой доступа, называемой вершиной. Данная структура данных работает по принципу «последний пришел, первый ушел» (см. рис.15).

Память организованная в виде стека называют ещё магазинной памятью. Работа со стеком в памяти организуется следующим образом: статически или динамически выделяется область памяти под стек и имеется указатель на «верхний» элемент. Добавление или уменьшение количества элементов происходит при изменении указателя.

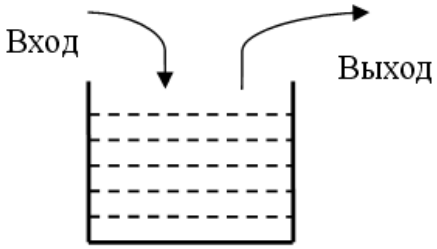


Рис.15. Схематическое изображение структуры стека.

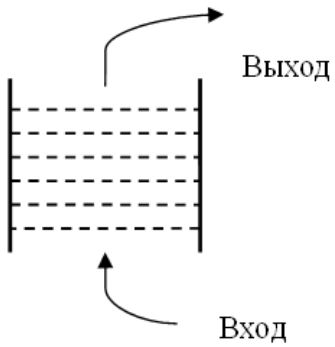


Рис.16. Схематическое изображение структуры очереди

Связанный список.

Пусть необходимо хранить список названий, упорядоченных по алфавиту (см. рис.17).

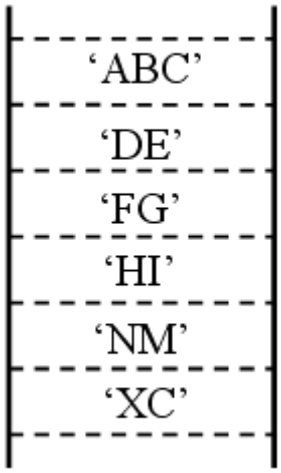


Рис. 17. Схематическое изображение структуры списка

Пусть необходимо вставить в список новое название ‘KL’.

Если мы имеем дело со статической структурой данных типа массива нужно освободить место для нового элемента, передвинув на одну позицию нижние элементы массива. Альтернативным является представление списка названий с помощью структуры данных, называемой связанным списком. Она организуется следующим образом (см. рис.18).

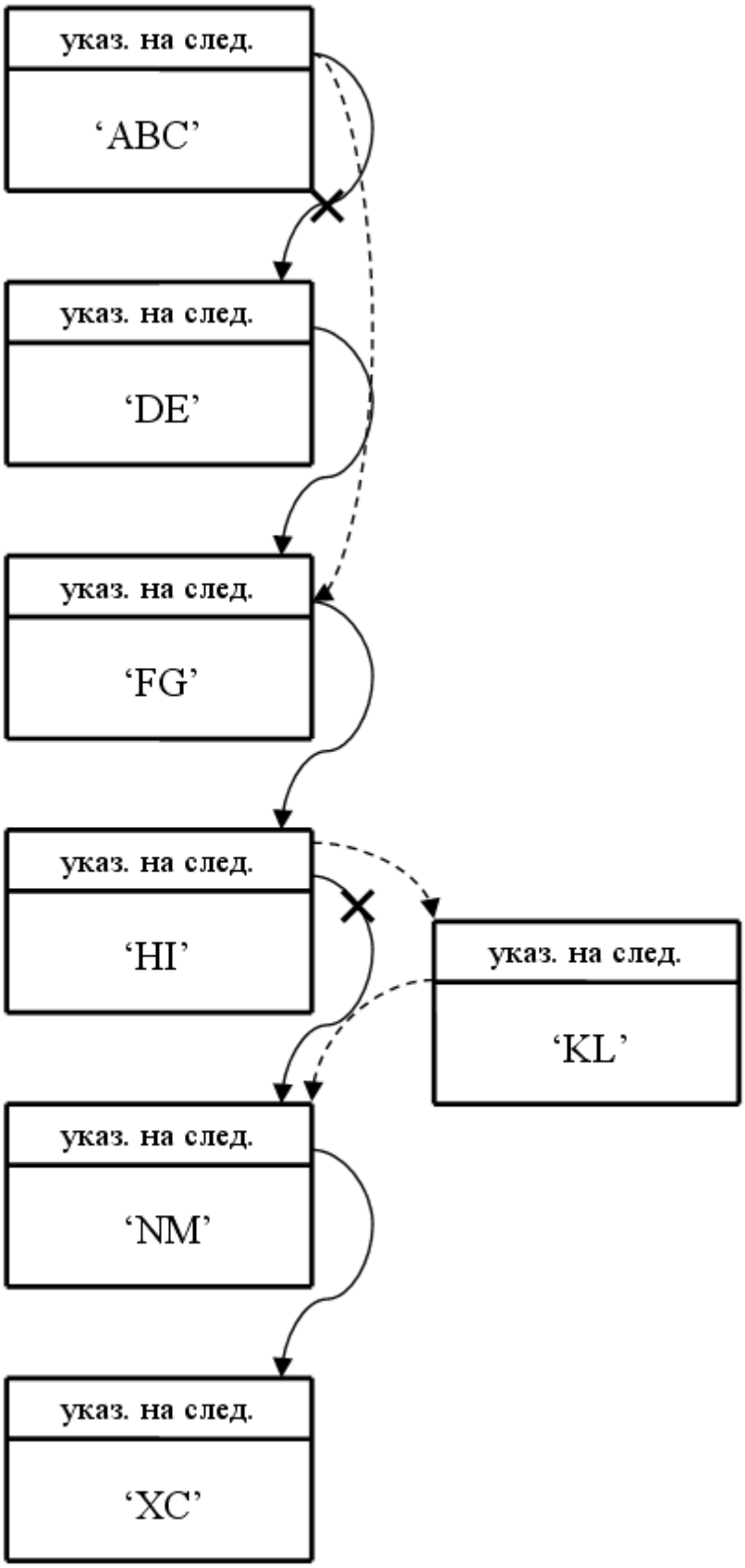


Рис.18. Схематическое изображение вставки элемента в связанный список

Вставка узла в связанный список осуществляется с помощью применения нескольких указателей.

Удаление узла также подразумевает изменение указателей. Наиболее эффективно реализуется связанный список с помощью динамических переменных и указателей.

3. Пример выполнения работы

Задан линейный однонаправленный список из элементов типа INTEGER. Написать функцию или процедуру, которая:

а) определяет, является ли список пустым;

б) удаляет положительные элементы;

в) удаляет все элементы, встречающиеся более одного раза.

Блок-схема программы приведена на рис.19.

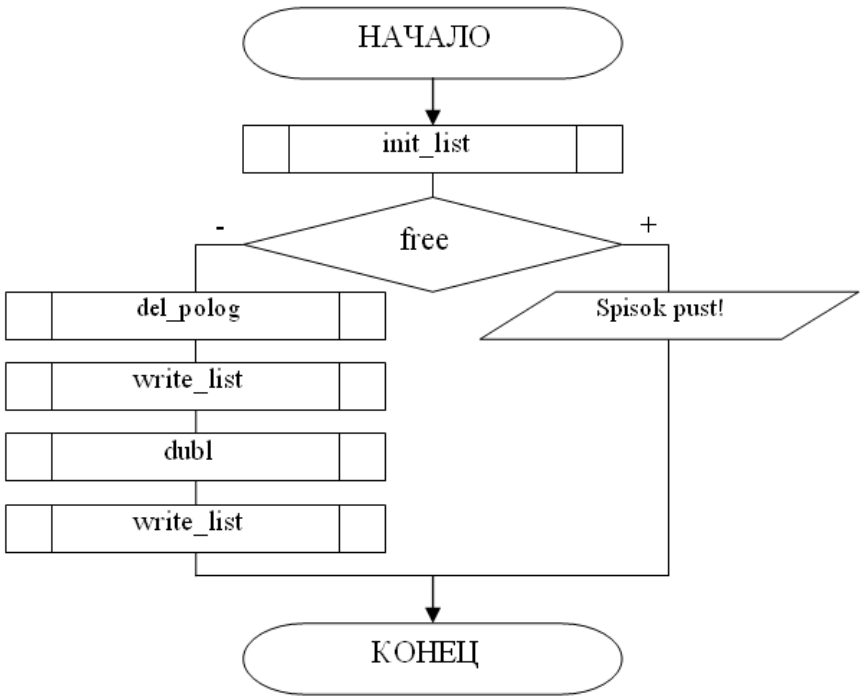


Рис.19. Блок-схема программы.

Блок-схема процедуры init_list изображена на рис.20.

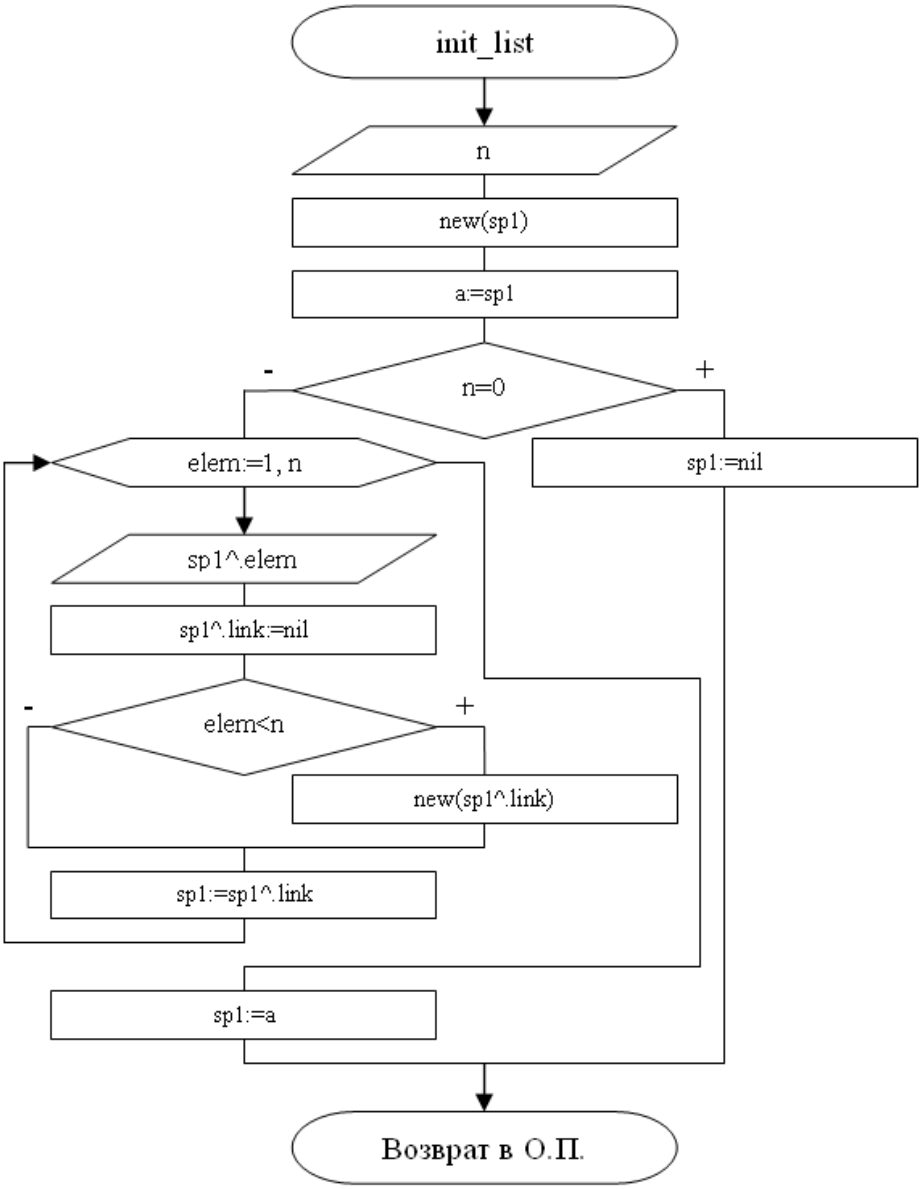


Рис.20. Блок-схема процедуры init_list.

Блок-схема процедуры write_list изображена на рис.21.

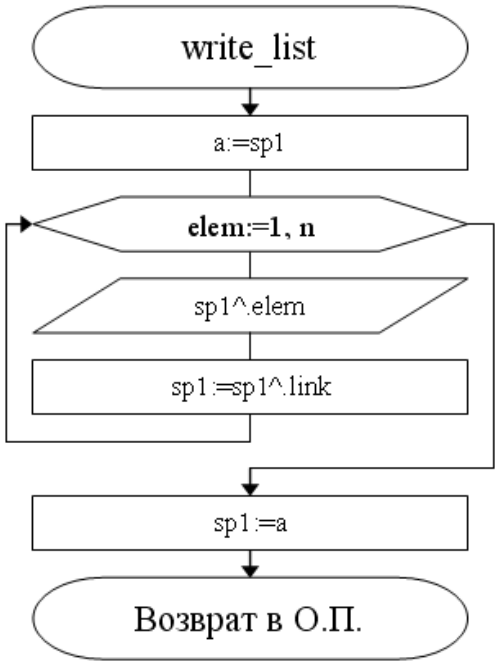


Рис.21. Блок-схема процедуры write_list.

Блок-схема процедуры del_polog изображена на рис.22.

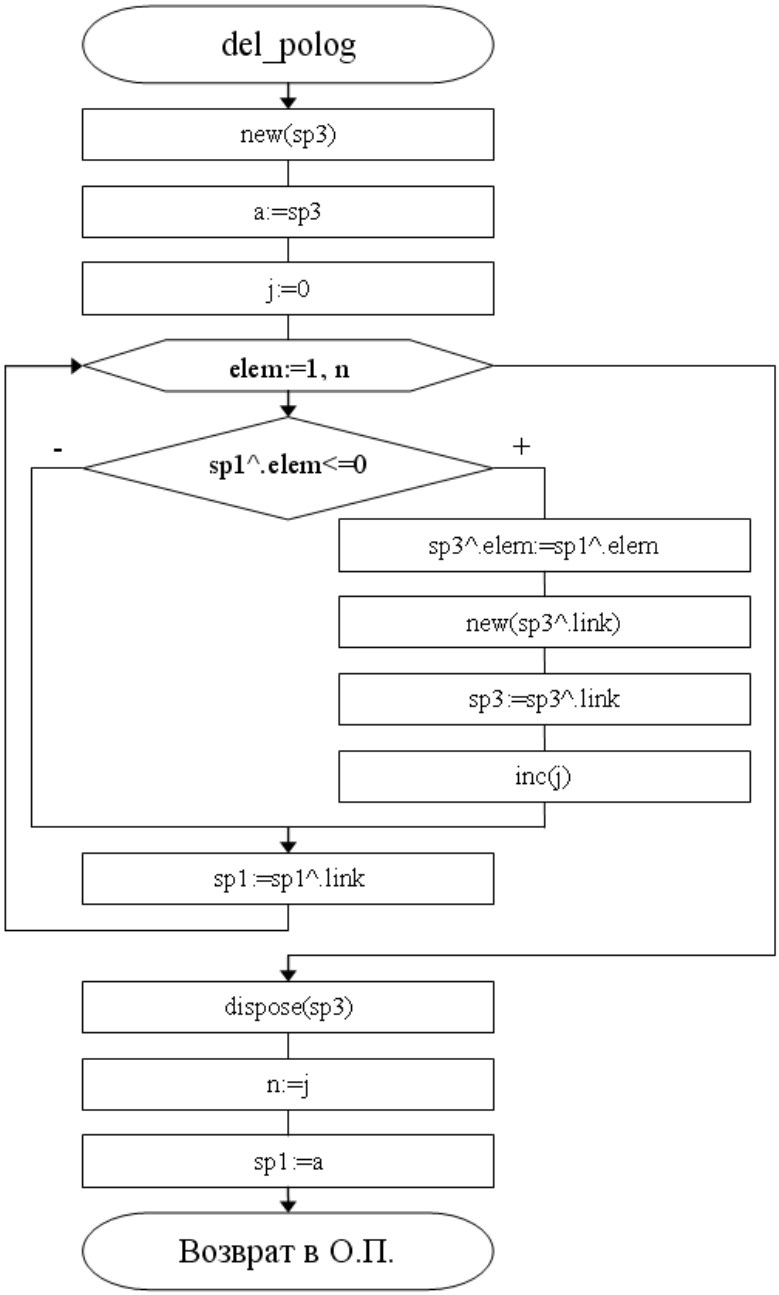


Рис.22. Блок-схема процедуры del_polog.

Блок-схема функции free изображена на рис.23.

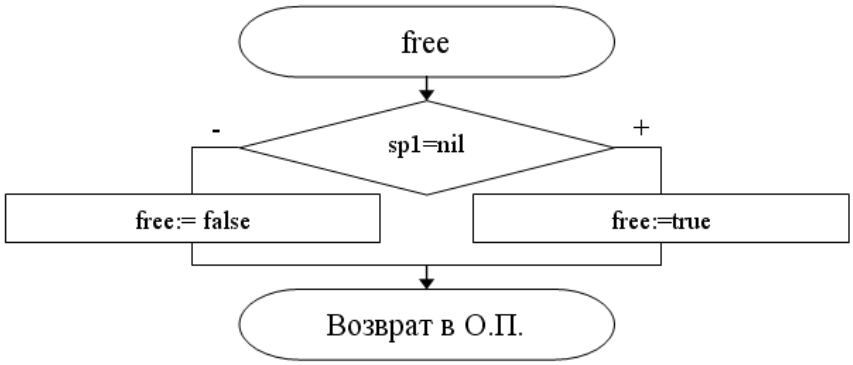


Рис.23. Блок-схема функции free

Блок-схема процедуры dubl изображена на рис.24.

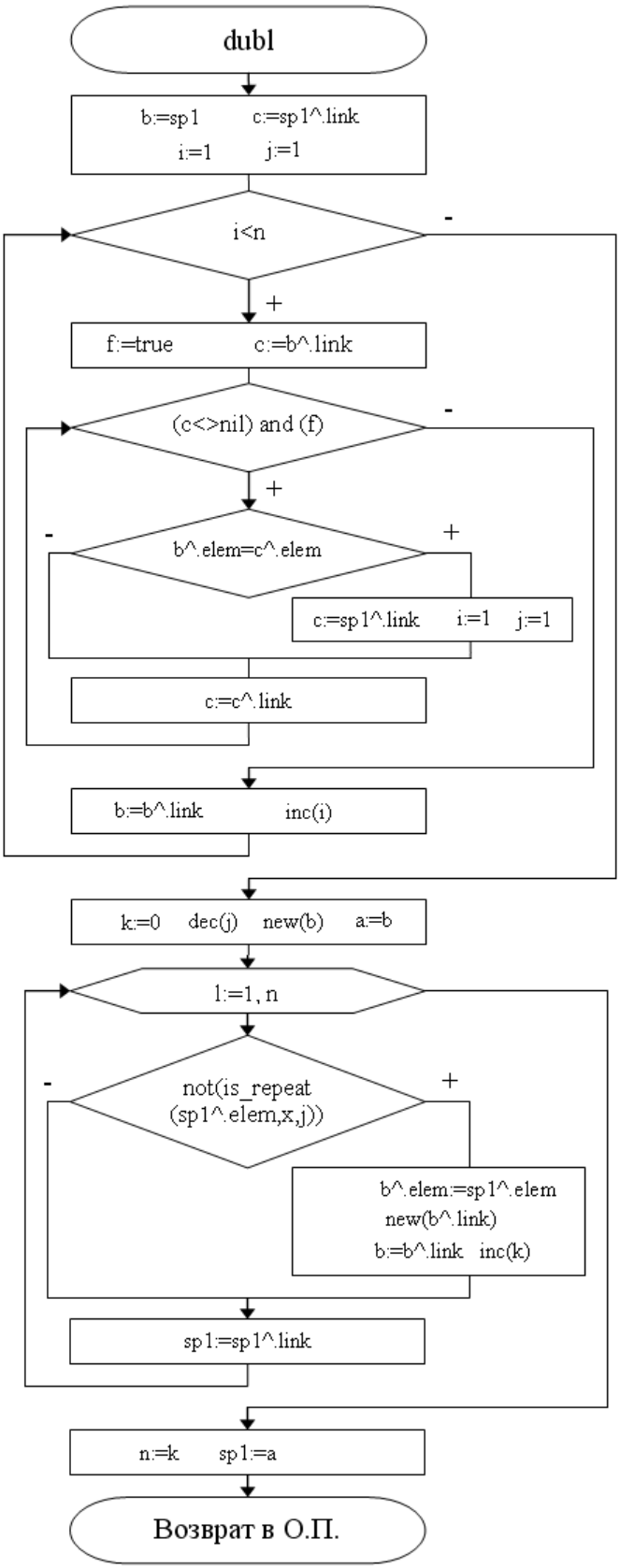


Рис.24. Блок-схема процедуры dubl

Блок-схема функции is_repeat изображена на рис.25.

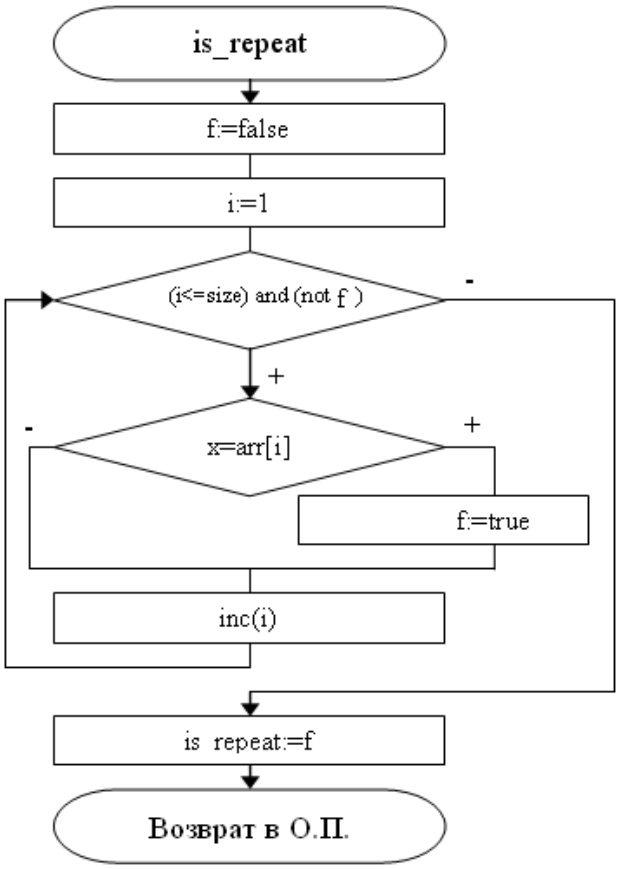


Рис.25. Блок-схема функции is_repeat.

Текст программы приведен в Листинге 8 на рис. 25.1.

```
 program ten;
uses crt;
type TArr=array[1..100] of integer;

     pointer=^spisok;
     spisok=record
              elem:integer;
              link:pointer
            end;
var
   sp1,sp3,a,b,head:pointer;
   n:integer;
procedure init_list;
var elem:integer;
begin
  write('Input n=');
  readln(n);
  new(sp1);
  a:=sp1;
  writeln('Input elements:');
  if (n=0) then sp1:=nil
  else
    begin
      for elem:=1 to n do
        begin
          readln(sp1^.elem);
          sp1^.link:=nil;
          if elem<n then new(sp1^.link);
          sp1:=sp1^.link
        end;
      sp1:=a;
    end;
end;
```

```
procedure write_list;
var elem:integer;
begin
  a:=sp1;
  for elem:=1 to n do
    begin
      writeln(sp1^.elem);
      sp1:=sp1^.link
    end;
  sp1:=a;
end;
function free:boolean;
begin
  if sp1=nil then free:=true
  else free:=false
end;
procedure del_polog;
var
   elem,j:integer;
begin
  new(sp3);
  a:=sp3;
  j:=0;
  for elem:=1 to n do
    begin
      if sp1^.elem<=0 then
        begin
          sp3^.elem:=sp1^.elem;
          new(sp3^.link);
          sp3:=sp3^.link;
          inc(j)
        end;
      sp1:=sp1^.link
    end;
  dispose(sp3);
  n:=j;
  sp1:=a;
end;

function  is_repeat(x:integer;arr:TArr;size:Integer):boolean;
var i:integer;
    f:boolean;
begin
  f:=false;
  i:=1;
  while (i<=size) and (not f) do
    begin
      if (x=arr[i]) then f:=true;
      inc(i);
    end;
  is_repeat:=f;
end;


procedure dubl;
var i,j,k,l:integer;
    h,c,tmp:pointer;
    f:boolean;
    x:TArr;
begin
  b:=sp1;
  c:=sp1^.link;
  i:=1; j:=1;
  while (i<n)  do
    begin
      f:=true;
      c:=b^.link;
      while (c<>nil) and (f)  do
        begin
          if b^.elem=c^.elem then
            begin
              f:=false;
```

```
              x[j]:=b^.elem;
              inc(j);
            end;
          c:=c^.link;
        end;
      b:=b^.link;
      inc(i);
    end;
  k:=0; dec(j);
  new(b);
  a:=b;
  for l:=1 to n do
    begin
      if not(is_repeat(sp1^.elem,x,j)) then
        begin
          b^.elem:=sp1^.elem;
          new(b^.link);
          b:=b^.link;
          inc(k);
        end;
      sp1:=sp1^.link
    end;
  n:=k;
  sp1:=a;
end;
{=======MAIN=======}
begin
clrscr;
  init_list;
  if free then writeln('Spisok pust!!')
   else begin
          del_polog;
          writeln('Spisok bez pologitelnyx elementov:');
          write_list;
          dubl;
          writeln('Spisok bez povtorjayushixsja elementov:');
          write_list;
       end;
readkey
end.
```

Рис.25.1. Листинг 7.

4. Способ оценки результатов

Работа оценивается по принципу «зачёт - не зачет».

Не зачет – допущена хотя бы одна ошибка в индивидуальном задании – 0 баллов.

Зачет – отсутствуют какие либо ошибки – 100 баллов.

Оглавление