Основы программирования

Учитель информатики и ИКТ
ГОУ г.Москвы СОШ №310
«У Чистых прудов»
Цыбикова Т.Р.

Тема 5.

ОРГАНИЗАЦИЯ ЦИКЛОВ

Циклы

- В своей практической деятельности человек постоянно сталкивается с задачами, при решении которых требуется многократно повторять одни и те же действия.
- Для составления алгоритмов решения таких задач используются команды повторения (циклы).

Рассмотрим следующий пример.

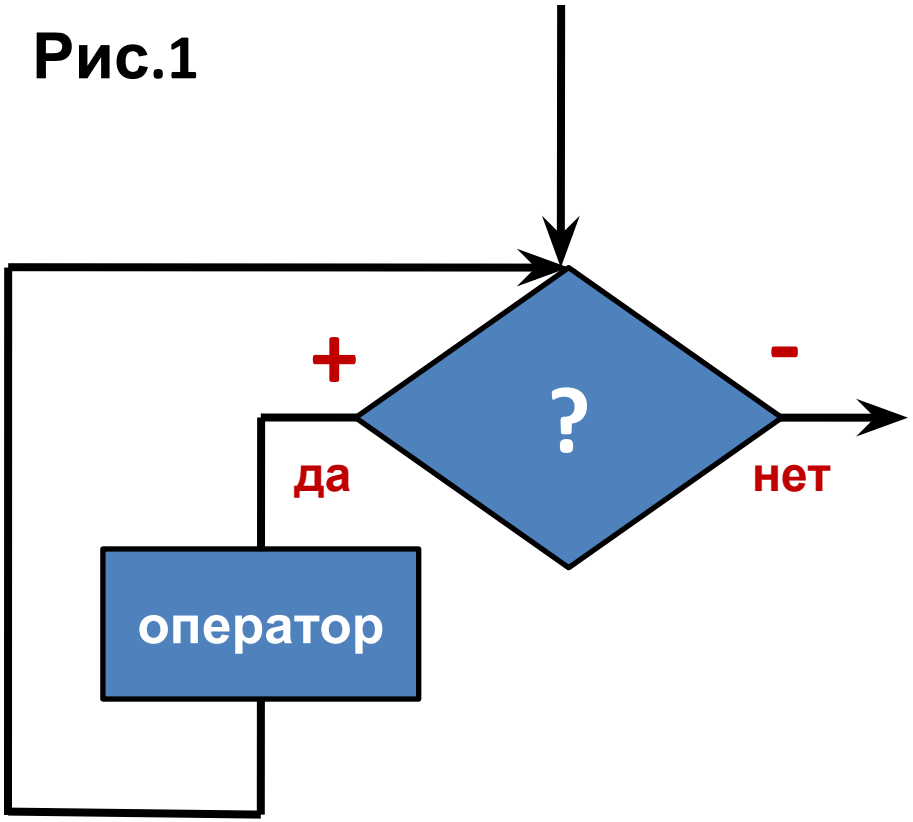
- Пусть требуется определить остаток от деления числа M на число N
  (M и N - произвольные натуральные числа).
- Самый простой способ решения этой задачи заключается в следующем:
  - проверяем, не меньше ли M, чем N
    (если M<N, то M и есть остаток от деления M на N);
  - Если M≥N, то уменьшаем значение M на значение N, если не стало, то еще раз уменьшаем значение M на величину N и т. д.
- Эти две операции (сравнения и вычитания) повторяются до тех пор, пока очередное значение M не станет меньше значения N. Значение M в этот момент и будет остатком от деления заданных вначале чисел.

Повторяющиеся действия

- Как видно, в этом примере несколько раз повторяется одна и та же последовательность действий.
- Компьютер может заданное число раз выполнить одни и те же действия с разными данными. Повторяющиеся действия в программировании называются **циклом**.
- Если изобразить в виде блок-схемы, то получатся две разные структуры (рис.1 и 2).
- Цикл не может выполняться вечно, в этом случае нарушается свойство алгоритма решить задачу за конечное число шагов.
- Цикл заканчивается по какому-либо условию.
- Проверка этого условия может производиться *в начале* каждого повторяющегося шага, в этом случае цикл называется **пока**.
- При проверке условия *в конце* каждого шага цикл называется **до**.
- Разновидностью цикла до является цикл **пересчет**.

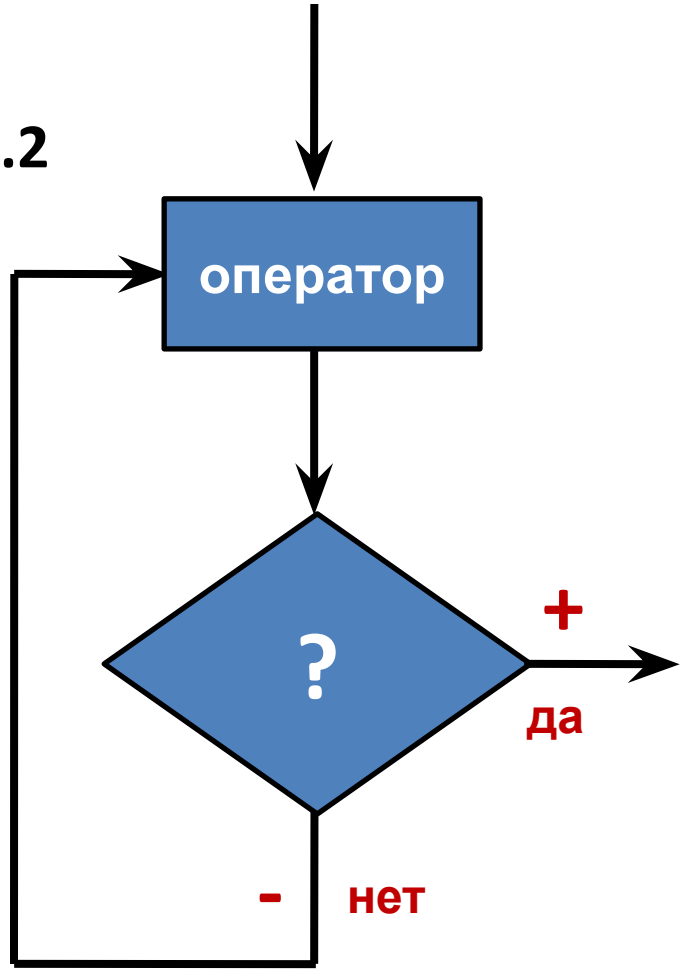
Циклические структуры

Рис.1



Цикл **пока**

Рис.2



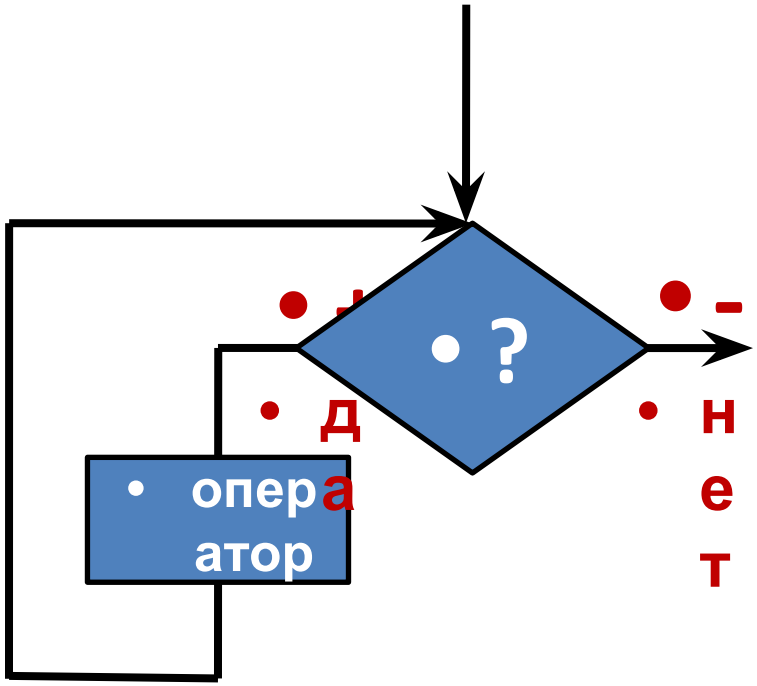
Цикл **до**

Циклические структуры

Цикл пока

- В цикле пока проверяется условие, и если оно выполняется, т.е. логическое выражение истинно, то выполняется оператор и снова проверяется условие.
- Записанное в цикле пока условие является условием продолжения цикла.
- Как только оно перестанет выполняться, цикл завершится.
- На рис.1 выход из ромба «+» (или **да**) означает выполнение условия цикла, «-» (или **нет**) – невыполнение.
- Цикл пока не выполнится ни разу, если условие при входе с структуру окажется ложным.

Рис.1



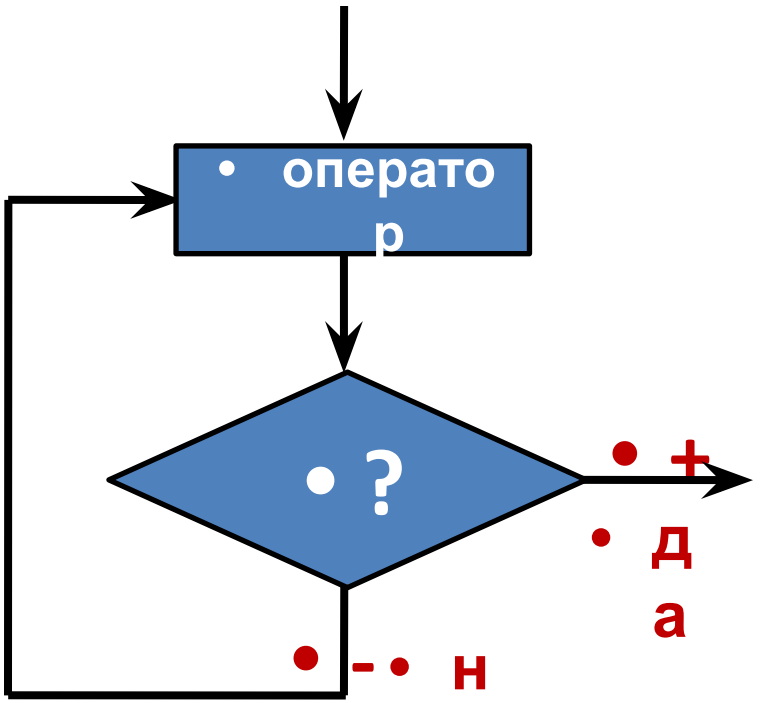
- Цикл **пока**

Циклические структуры

Цикл до

- Как правило, цикл **пока** содержит *условие повторения*, о цикл **до** условие *окончания работы цикла*.
- Обе структуры имеют один вход и один выход.
- Однако цикл **до** всегда выполняется хотя бы один раз, потому что условие проверяется после выполнения действия. Это затрудняет проверку правильности программы, поэтому лучше использовать цикл **пока**.
- Оператор в цикле может быть простым или составным, заключенным о операторные скобки.
- В этом случае в цикле могут повторяться несколько операторов, а не один.
- Повторяющиеся в цикле операторы называются **телом цикла**.

Рис.2



- Цикл до

Оператор безусловного перехода

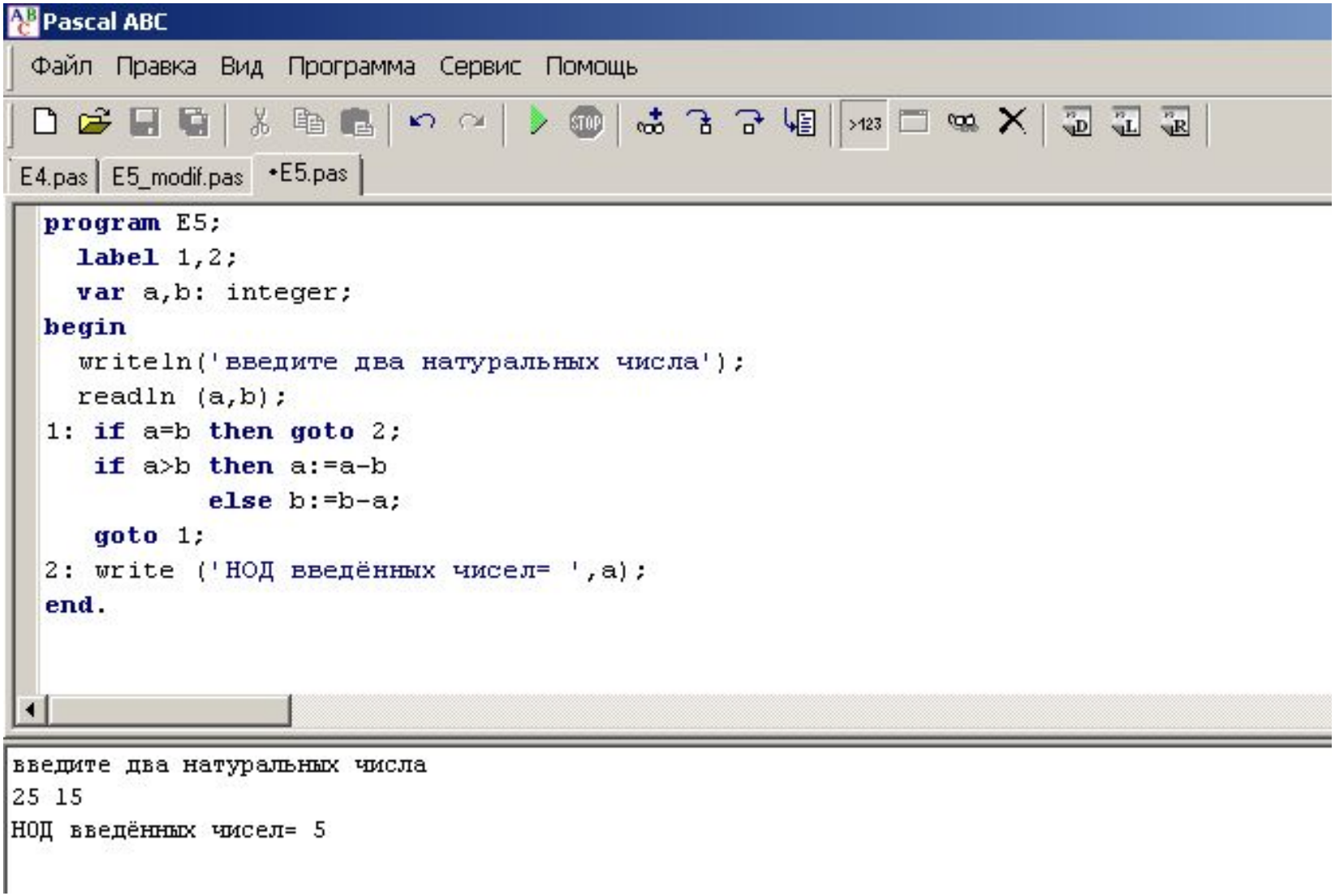
- Циклы можно организовывать, используя различные средства языка Паскаль.
- Этот оператор позволяет перейти без проверки условия либо на один из предыдущих операторов, либо на один из последующих, т.е.изменить порядок выполнения команд.
- Общий вид оператора: **goto n;**
  - где n- целое число, не более чем из 4 цифр, называемое меткой.
  - Метка появляется в программе 3 раза:

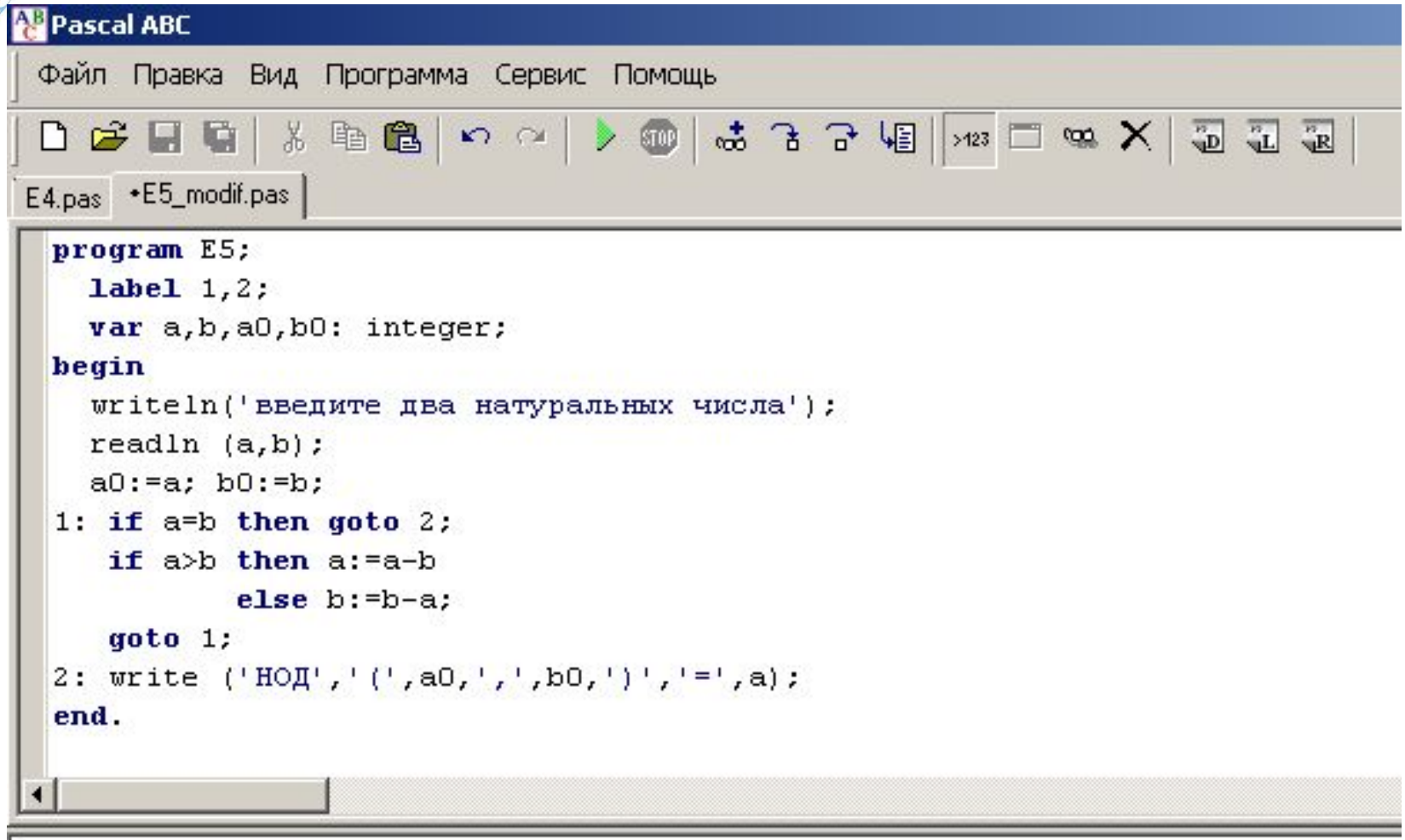
1) в описательной части в разделе **Label;**
2) в операторе **goto n;**
3) перед оператором, на который осуществляется безусловный переход, в этом случае метка от оператора отделяется двоеточием.

Организация циклов с помощью операторов условного и безусловного переходов

- Пусть требуется вычислить НОД двух натуральных чисел A и B.
- Воспользуемся алгоритмом Евклида: будем уменьшать каждый раз большее из чисел на величину меньшего до тех пор, пока оба числа не станут равны.

Исходные данные	Первый шаг	Второй шаг	Третий шаг	НОД(A,B)=5
A=25	A=10	A=10	A=5	
B=15	B=15	B=5	B=5	


Pascal ABC
Файл Правка Вид Программа Сервис Помощь
E4.pas E5_modif.pas •E5.pas
program E5;
label 1,2;
var a,b: integer;
begin
writeln('введите два натуральных числа');
readln (a,b);
1: if a=b then goto 2;
if a>b then a:=a-b
else b:=b-a;
goto 1;
2: write ('НОД введённых чисел= ',a);
end.
введите два натуральных числа
25 15
НОД введённых чисел= 5


Pascal ABC
Файл Правка Вид Программа Сервис Помощь
E4.pas
*E5_modif.pas
program E5;
label 1,2;
var a,b,a0,b0: integer;
begin
writeln('введите два натуральных числа');
readln (a,b);
a0:=a; b0:=b;
1: if a=b then goto 2;
if a>b then a:=a-b
else b:=b-a;
goto 1;
2: write ('НОД','(',a0,',',b0,')','=',a);
end.
введите два натуральных числа
25 15
НОД(25,15)=5

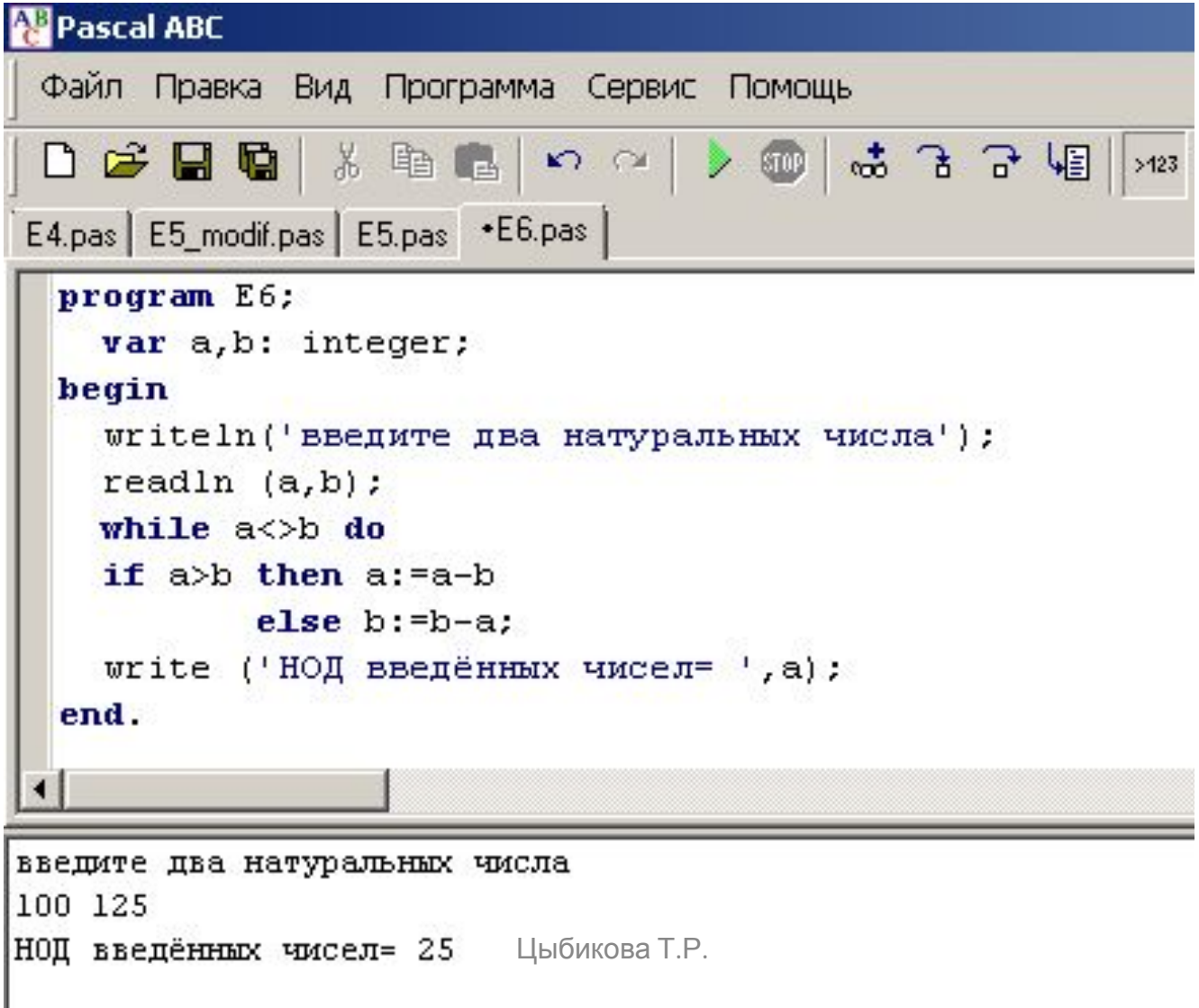
Оператор цикла **пока**

- Как видно из предыдущего примера, циклический процесс можно организовать без использования специальных операторов.
- Однако при составлении достаточно серьезных программ использовать оператор безусловного перехода не рекомендуется, так как можно быстро запутаться при проверке программы.
- Оператор цикла **пока** имеет вид:

while условие do оператор;

- И выполняется следующим образом: оператор(тело цикла) повторяется до тех пор, пока выполняется условие (истинно логическое выражение).
- Оператор может быть простым или составным, заключенным в операторные скобки **begin … end**.

Для алгоритма Евклида программа примет вид:



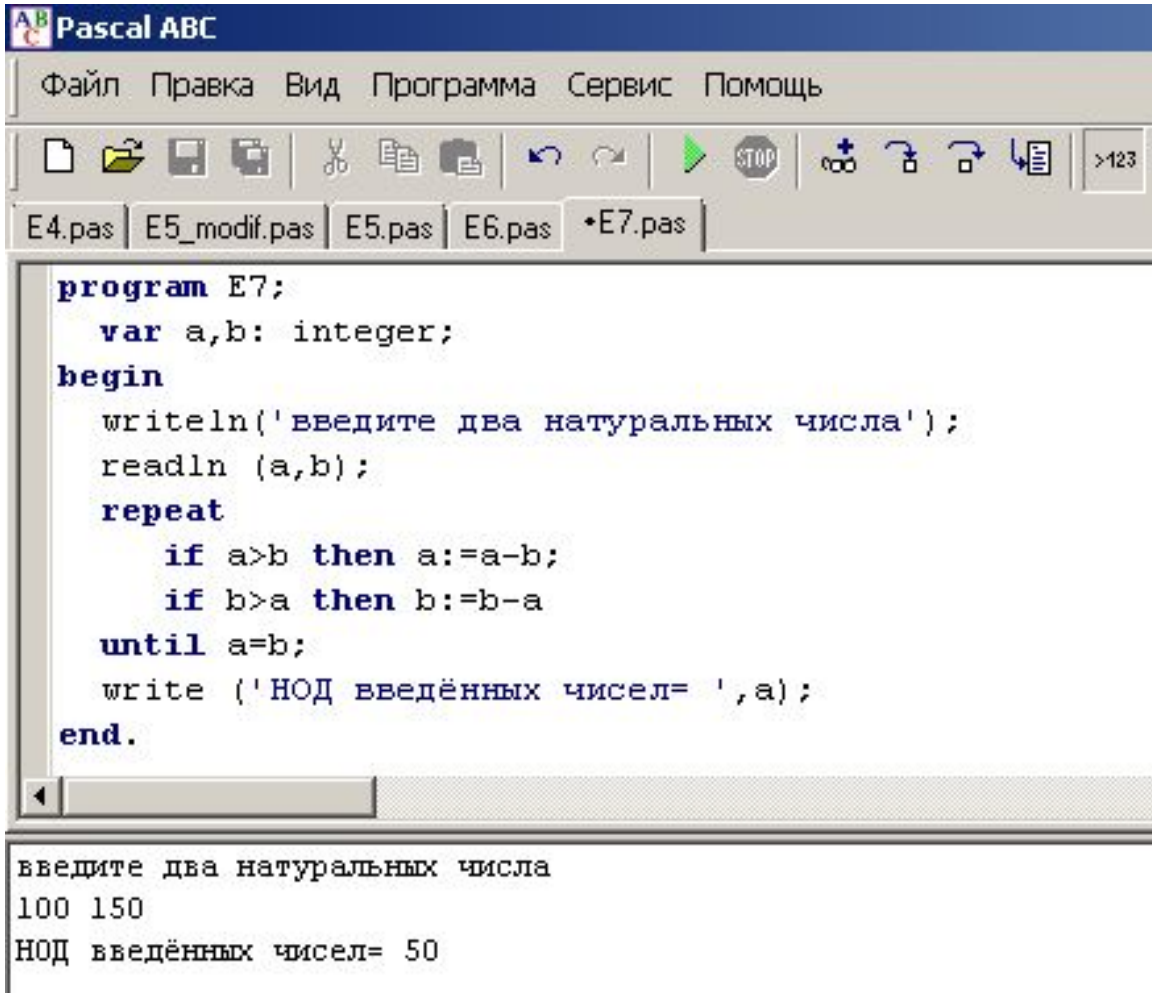
Оператор цикла **до**

- Проверка условия в цикле **до** осуществляется после выполнения оператора.
- Если условие в цикле **пока** является условием продолжения повторений, то условие **до** – условием выхода из цикла, его завершением.
- Поэтому для одной и той же задачи эти условия противоположны.
- Общий вид оператора цикла **до**:

repeat оператор until условие;

- Между словами **repeat** (повторить) и **until** (до тех пор пока) можно записать любое количество операторов без использования операторных скобок.
- Перед словом **until** не ставится точка с запятой.

Программа нахождения НОД чисел примет вид:



Оператор циклов **пересчет**

- При выполнении программ нахождения НОД число повторений различно для разных чисел.
- Когда известно число повторений, удобно использовать цикл **пересчет.**
- В Паскале имеется два оператора для организации циклов **пересчет**: прямой и обратный.
- Прямой пересчет идет от известного меньшего числа до известного большего, на каждом шаге прибавляется единица (например, от 120 до 140: 121, 122, 123, …, 139, 140).

Оператор прямого пересчета:

for i:=n1 to n2 do оператор;

читается как «для i начиная **n1**с до **n2** выполнить оператор».

- Переменная i называется переменной цикла, она при прямом пересчете всегда меняется от меньшего значения до большего.
- При **i:=n1** цикл выполняется первый раз.
- Затем к значению **i** добавляется единица и осуществляется проверка, не превысило ли полученное значение величину **n2**.
- Если **i+1≤n2,** то оператор выполняется, если нет, то происходит выход из цикла и выполнение следующего по порядку оператора программы.

Оператор прямого пересчета:

- Поскольку оператор цикла **for** сам изменяет значение переменной цикла, ее нельзя менять другими способами, например присваиванием ей какого-либо значения в теле цикла (она не должна появиться слева от знака «:=»).
- Оператор в цикле может быть простым или составным, заключенным в операторные скобки.
- Оператор **пересчет** работает как цикл **до**, поэтому надо быть внимательным, оператор в теле цикла выполнится всегда хотя бы один раз.
- Рассмотрим примеры использования операторов цикла.

Операторы циклов **пересчет**

Пример 1.

- Известно, что для получения целой степени ***n*** числа ***a*** его надо умножить само на себя ***n*** раз.
- Это произведение при выполнении программы будет храниться в ячейке с именем ***p***.
- Каждый раз, при очередном выполнении цикла, из этой ячейки будет считываться предыдущий результат, домножаться на основание степени ***a*** и снова записываться в ячейку ***p***.
- Основной оператор в теле цикла повторяется ***n*** раз и имеет вид:

p:=p*a;

Пусть надо вычислить a^n.

- При первом выполнении цикла в ячейке p должно находиться число, не влияющее на умножение, т.е. до цикла туда надо записать единицу.

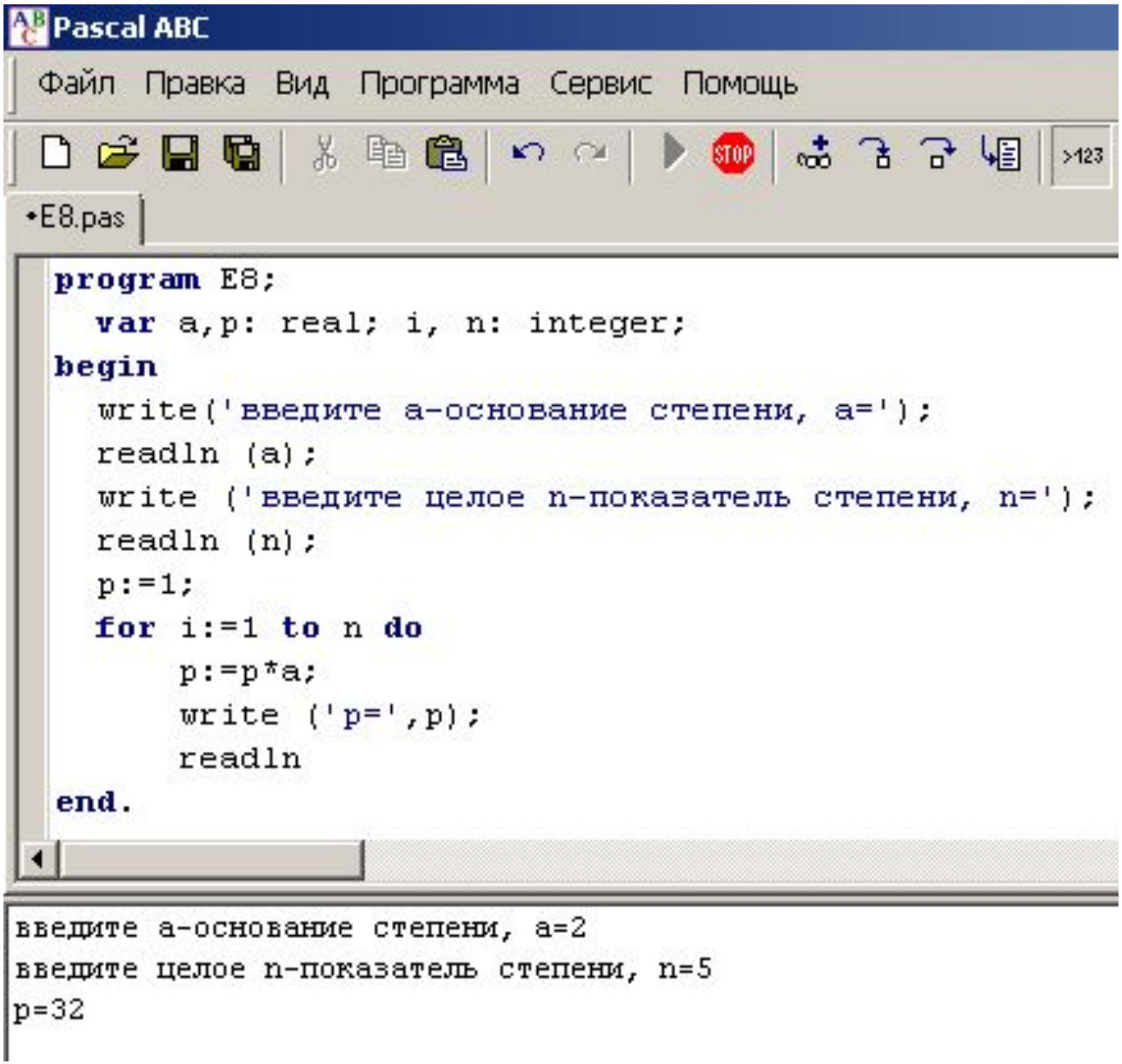
Выполнение программы	
a=2	*n=5*
i	*p*
	1
1	2
2	4
3	8
4	16
5	32

Программа имеет вид:

```
program E8;
  var a,p: real; i, n: integer;
begin
  write(‘введите a – основание  степени, a =');
  readln (a);
  write (‘введите целое n – показатель степени, n=');
  readln (n);
  p:=1;
  for i:=1 to n do
     p:=p*a;
     write ('p=',p);
     readln
end.
```

Отладка программы

- Перед текстом программы представлен **протокол** её выполнения при возведении числа 2 в пятую степень.
- Таблица заполнена вручную, процесс её заполнения называется отладкой программы.
- **Отладка** - это проверка всех этапов работы программы.
- Для сложных задач сначала составляется **контрольный пример** (**тест**) и программа выполняется человеком, который выполняет каждый оператор так, как его выполняет компьютер.
- Затем программу выполняет компьютер и сверяет все промежуточные, полученные при счете данные и конечные результаты.
- Только после полного совпадения программа выполняется с реальными данными.
- Для понимания работы программы, выполнения отдельных операторов полезно заполнять подобные протоколы для всех учебных задач.


Pascal ABC
Файл Правка Вид Программа Сервис Помощь
STOP
>123
•E8.pas
program E8;
var a,p: real; i, n: integer;
begin
write('введите a-основание степени, a=');
readln (a);
write ('введите целое n-показатель степени, n=');
readln (n);
p:=1;
for i:=1 to n do
p:=p*a;
write ('p=',p);
readln
end.
введите a-основание степени, a=2
введите целое n-показатель степени, n=5
p=32

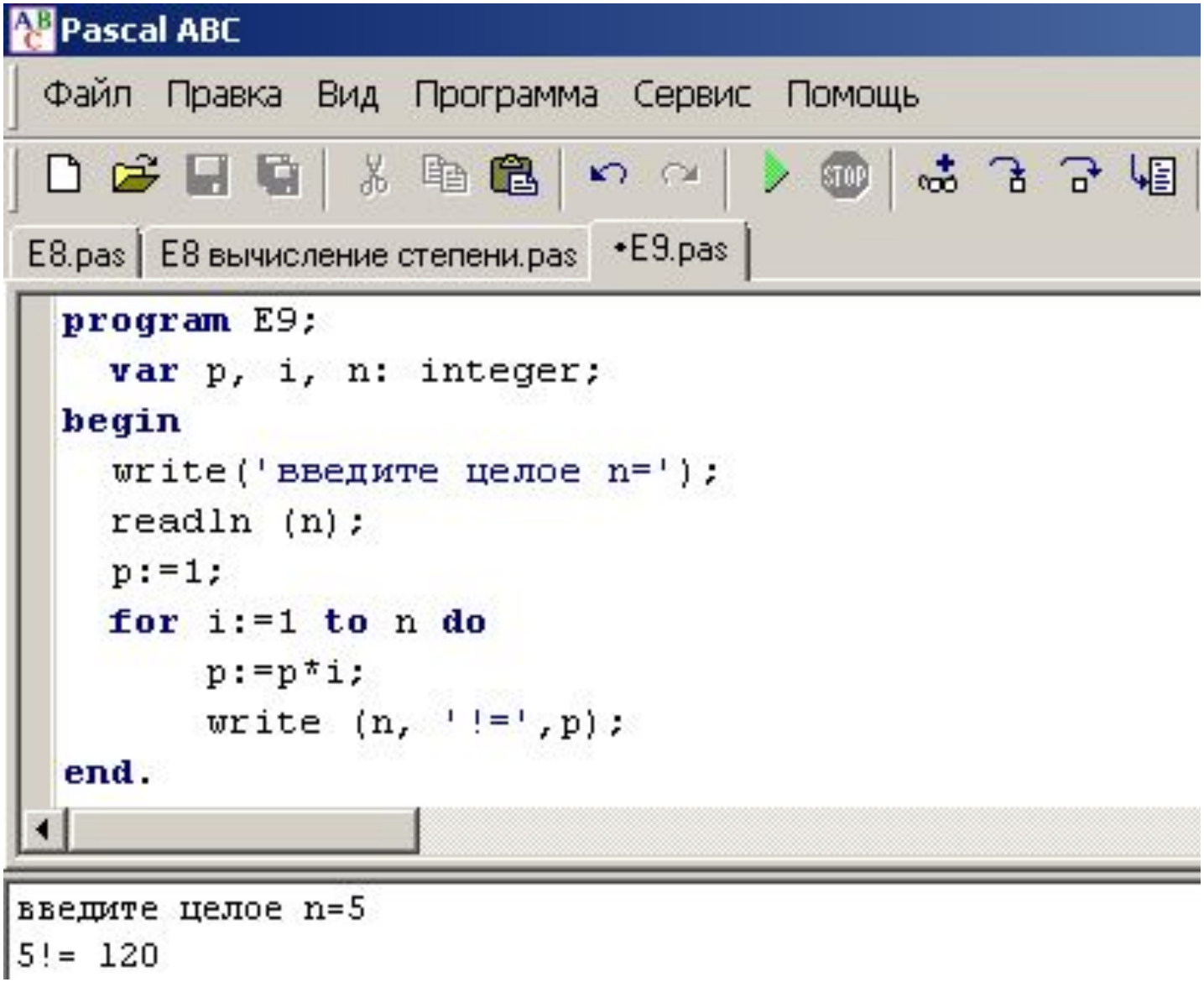
Оператор циклов **пересчет**

Пример 2.

- По определению n!=1*2*3*…*n.
- Используя предыдущую программу, вычислим ***p*** как произведение чисел от ***1*** до ***n***, т.е. ***p*** каждый раз умножается не на одно и то же число, а на значение переменной цикла.

Вычисление p=n! (n факториал)

```
program E9;
  var p, i, n: integer;
begin
  write(‘введите целое n=');
  readln (n);
  p:=1;
  for i:=1 to n do
     p:=p*i;
     write (n, '!=',p);
end.
```


Pascal ABC
Файл Правка Вид Программа Сервис Помощь
E8.pas
E8 вычисление степени.pas
*E9.pas
program E9;
var p, i, n: integer;
begin
write('введите целое n=');
readln (n);
p:=1;
for i:=1 to n do
p:=p*i;
write (n, '!=',p);
end.
введите целое n=5
5!= 120

Оператор циклов **пересчет**

Пример 3.

- Пусть требуется составить таблицу значений функции на отрезке [0;3,14] с шагом 0,1.
- Чтобы не определять количество повторений вычислений, можно воспользоваться циклом **пока**.
- Используя вывод вещественных чисел с фиксированной точкой, определим, что количество цифр после запятой в значении функции будет равно 5.
- Тогда все число, учитывая область значений синуса, займет 7 позиций (числа положительные, значит, добавится позиция для десятичной точки и целой части числа).

Составление таблицы значений функции y=sin x.

```
program E10;
 var x,y: real;
begin
 x:=0;
 write('x':100, 'sin x':100);
 while x<=3.14 do
  begin
   y:=sin(x);
   writeln (x:100,' ',y:7:5);
   x:=x+0.1
  end;
  readln
end.
```

```
program E10;
  var x,y: real;
begin
  x:=0;
  write('x':10, 'sin x':10);
  while x<=3.14 do
    begin
      y:=sin(x);
      writeln (x:10,' ',y:7:5);
      x:=x+0.1
    end;
end.
```

```
1.00E+000 0.84147
1.10E+000 0.89121
 0.93204
1.30E+000 0.96356
1.40E+000 0.98545
1.50E+000 0.99749

1.70E+000 0.99166
1.80E+000 0.97385
1.90E+000 0.94630
2.00E+000 0.90930
2.10E+000 0.86321
2.20E+000 0.80850
2.30E+000 0.74571
2.40E+000 0.67546
2.50E+000 0.59847
2.60E+000 0.51550
2.70E+000 0.42738
2.80E+000 0.33499
2.90E+000 0.23925
3.00E+000 0.14112
3.10E+000 0.04158
```

Оператор циклов **пересчет**

Пример 4.

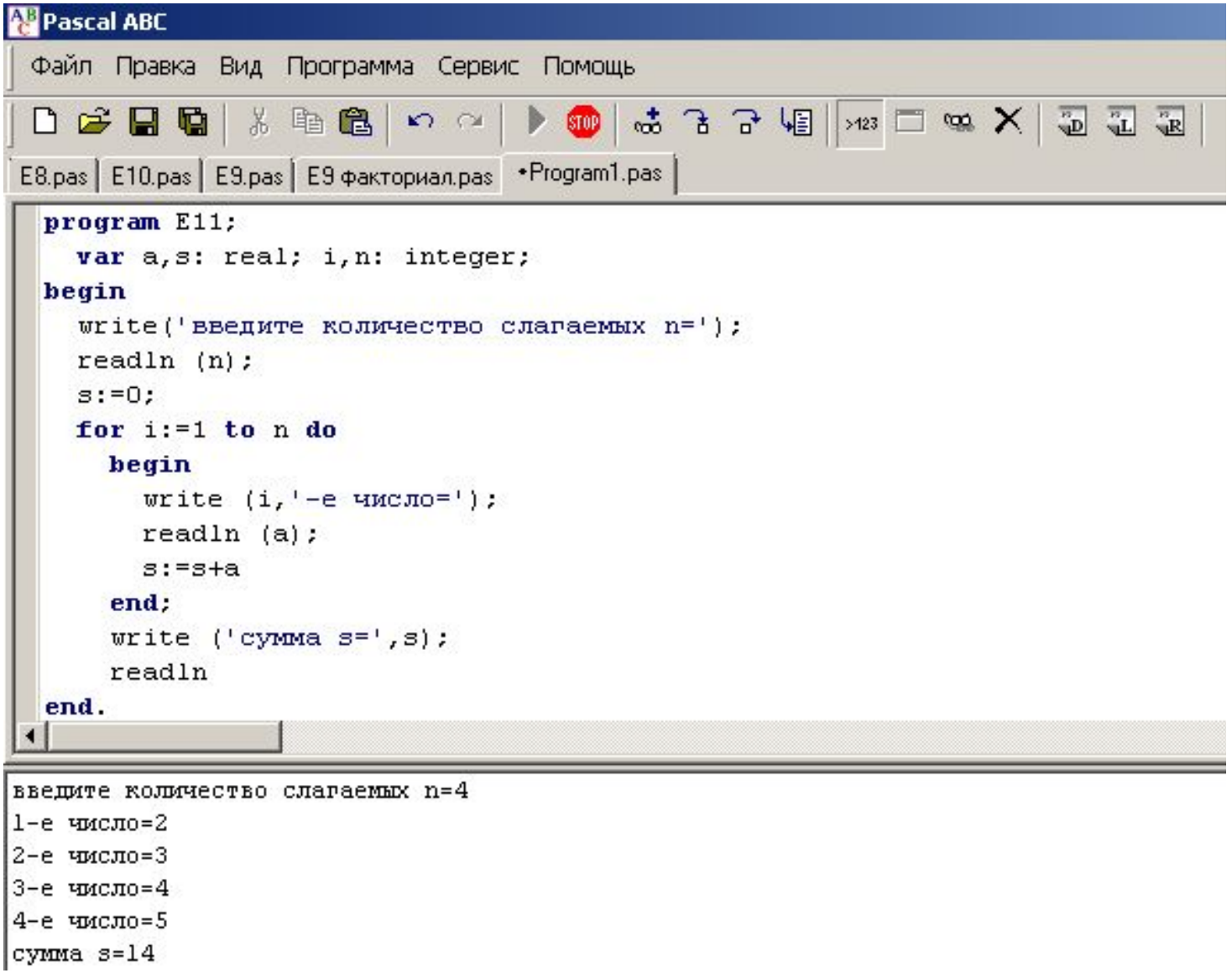
- При суммировании, как и при умножении нескольких чисел, необходимо накапливать результат в некоторой ячейке памяти, каждый раз считывая из этой ячейки предыдущее значение суммы и увеличивая его на очередное слагаемое.
- Пусть известно, что будет складываться ***n*** слагаемое.
- Путь известно, что будет складываться ***n*** чисел.

Суммирование чисел.

- В этом случае надо n раз выполнить действие ***s:=s+a;*** здесь ***a*** – очередное число, вводимое с клавиатуры.
- Для первого выполнения этого оператора присваивания надо из ячейки с именем ***s*** взять такое число, которое не повлияло бы на результат сложения.
- Следовательно, прежде чем начать выполнять цикл, надо поместить в эту ячейку (или, что то же самое, присвоить переменной ***s***) число нуль.

Программа имеет вид:

```
program E11;
 var a,s: real; i,n: integer;
begin
 write(‘введите количество слагаемых n=');
 readln (n);
 s:=0;
 for i:=1 to n do
  begin
   write (i,'-ое число=');
   readln (a);
   s:=s+a
  end;
  write (‘сумма s=',s);
  readln
end.
```


Pascal ABC
Файл Правка Вид Программа Сервис Помощь
E8.pas E10.pas E9.pas E9 факториал.pas •Program1.pas
program E11;
var a,s: real; i,n: integer;
begin
write('введите количество слагаемых n=');
readln (n);
s:=0;
for i:=1 to n do
begin
write (i,'-е число=');
readln (a);
s:=s+a
end;
write ('сумма s=',s);
readln
end.
введите количество слагаемых n=4
1-е число=2
2-е число=3
3-е число=4
4-е число=5
сумма s=14

Если количество чисел неизвестно,
то можно задать число-ограничитель, например 0.
В таком случае используется цикл **while** или **repeat**.

```
s:=0;
readln (a);
while a<>0 do
begin s:=s+a;
    readln (a)
end;
```

```
s:=0;
repeat
    readln (a);
    s:=s+a;
until a=0;
```

Оператор цикла обратный пересчет

- Оператор цикла обратный пересчет работает аналогично оператору цикла прямого пересчета, только переменная цикла не возрастает с каждым шагом на единицу, а на единицу убывает.
- Оператор имеет вид:

```
for i:=n2 downto n1 do оператор;
```

Для этого оператора должно также выполняться n2≥n1.

Правила

- **При использовании в программе операторов цикла необходимо соблюдать следующие правила:**
  - внутри цикла может находиться другой цикл, но необходимо, чтобы циклы имели разные переменные и внутренний цикл полностью находился в теле внешнего цикла;
  - нельзя передавать управление в тело цикла, минуя заголовок (это значит, что метка и оператор **goto** с этой меткой должны находиться в теле цикла);
  - если требуется обойти группу операторов в теле цикла и продолжить цикл, т.е.выполнить его следующий шаг, то надо передать управление на замыкающий цикл **end**;
  - Можно досрочно выйти из цикла, или используя оператор **goto,** или изменив параметр условия в операторах **while** и **repeat** так, чтобы цикл больше не выполнялся.

Вопросы и задания

1. Пусть тело цикла в программе E7 такое же, как в программе E6. Как будет работать программа E7, если ввести два одинаковых числа ***a*** и ***b***?
2. Сколько раз выполнится оператор цикла **repeat**, если условие после слова **until** истинно при входе в цикл?
3. Объясните, какая разница между условиями, записанными после слов **while** и **repeat** для одной и той же задачи.

Вопросы и задания

4. Напишите программы вычисления сумм:
   a) Сорока слагаемых вида n-i, где i= 1, 2, 3,…, 40, а n – данное число;
   b) n слагаемых вида x + i, где x – данное число, а i меняется от 1 до n;
   c) Ста слагаемых, имеющих вид дроби (i+1)/(i+2);
   d) N слагаемых вида $(i+1)^2$, где i = 1, 2, …, n;
   e) N слагаемых $\sin x + \sin x^2 + \sin x^3 + \ldots + \sin x^n$;
   f) N слагаемых $\sin x + \sin^2 x + \sin^3 x + \ldots + \sin^n x$;
   g) Кубов n первых натуральных чисел.

Вопросы и задания

5. Для различных вводимых с клавиатуры целых чисел найдите сумму положительных нечетных.
6. Напишите программы вычисления произведений:
   - a) $a*(a+1)*(a+2)*...*(a+n-1);$
   - b) $a*(a-n)*(a-2n)*...*(a-n^2);$
   - c) $(x-1)(x-2)(x-3)...(x-n);$
   - d) $2*4*6*...*(2n);$
   - e) $(1+\sin 0.1)(1+\sin 0.2)...(1+\sin 10);$
   - f) *Всех чисел от 1 до 100 кратных 3, но не кратных 6;*
   - g) *n сомножителей вида* $(x+i)^2$.

Вопросы и задания

7. Дано положительное число A. Найдите среди чисел 1, 1+1/2, 1+1/2+1/3,… первое, большее A.
8. Вводя числа с клавиатуры без ограничения их количество (конец ввода – число нуль), найдите сумму положительных и произведение отрицательных.
9. $$\frac{\cos 1}{\sin 1} \cdot \frac{\cos 1 + \cos 2}{\sin 1 + \sin 2} \cdot \ldots \cdot \frac{\cos 1 + \cos 2 + \ldots + \cos n}{\sin 1 + \sin 2 + \ldots + \sin n}$$

Литература

- **А.А.Кузнецов, Н.В.Ипатова**
  «Основы информатики», 8-9 кл.:
  - Раздел 3. ОСНОВЫ ПРОГРАММИРОВАНИЯ, С.99-107