Основы программирования

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

Тема 9.

ПОДПРОГРАММЫ

СОДЕРЖАНИЕ

Подпрограммы

- При разработке программы иногда появляются повторяемые группы действий или возникает необходимость расчленить программу на функциональные модули, сделать ее структуру иерархической.
- Для этого во всех языках программирования существуют средства организации подпрограмм.
- Для решения сложной задачи рекомендуется сначала алгоритм, а затем и программу разрабатывать «*сверху вниз*», *от более общего плана к детальному.*
- В таком виде г**лавная программа соответствует укрупненному плану решения задачи, а ее команды — вызову соответствующей подзадачи, реализованной в виде подпрограммы.**

Подпрограммы

- Идя по такому пути создания программы, можно отдельные мелкие функции реализовывать сначала в виде небольших подпрограмм, проверять их на контрольных примерах и, только убедившись в правильности работы, включать в состав основной программы.
- Это удобнее проделывать, имея в распоряжении язык, позволяющий полностью выделить подпрограмму из текста основной программы в виде отдельного модуля.
- Использование подпрограмм дает возможность разрабатывать программу по частям, поручать реализацию больших проектов группам разработчиков.
- В состав групп входят:
  - специалисты данной науки или производства, из области которой решается задача, разрабатывающие алгоритм и структурирующие данные,
  - а также программисты, объединяющие алгоритм и данные в программе для компьютера.

В Паскале подпрограмма является частью основной программы...

- В Паскале подпрограмма является частью основной программы, ее описание располагается между разделом **var** главной программы и ее программным блоком (первым **begin**).
- Подпрограмм может быть несколько, их описания располагаются в произвольном порядке одно за другим.
- Описание подпрограммы можно сравнить с записываемой в математике формулой «в общем виде», в которую при расчетах подставляются конкретные значения.
- Поскольку далеко не каждую задачу удастся свести к некоторой формуле, но всегда можно записать алгоритм ее решения, подпрограмма — это та же инструкция по решению некоторой задачи.
- Как и формула, подпрограмма используется для различных данных, передаваемых из главной программы или других подпрограмм.

Подпрограмма

- Подпрограмма — это специальным образом оформленный алгоритм, который может многократно использоваться при решении более общей задачи.
- В Паскале различают два вида подпрограмм: процедуры и функции.
- **Основное различие** между ними заключается в том, что процедура получает в результате своей работы любое количество данных, а функция — только одно значение.

Структура подпрограммы

Подпрограммы имеют структуру, аналогичную главной программе.

Они содержат **заголовок** со специальным словом — признаком подпрограммы, **имя** и, при необходимости, **списки** передаваемых на обработку и получаемых из подпрограммы **данных**.

Затем могут располагаться все имеющиеся в главной программе разделы описаний: меток, констант, типов и переменных.

В этих разделах описываются данные, используемые только внутри подпрограммы и являющиеся промежуточными при ее выполнении. Такие данные называются **локальными**.

В подпрограмме могут участвовать переменные, описанные в главной программе. Эти данные называются **глобальными**, их значения и подпрограмма, и главная программа берут из общей памяти.

Два этапа при работе с подпрограмой

- При работе с подпрограммой всегда выделяется **два этапа**:
  - *описание подпрограммы*, т. е. запись алгоритма решения задачи в специальной форме,
  - и *вызов подпрограммы* — передача ей данных на обработку из вызываемой программы, которая, в свою очередь, может быть подпрограммой, и получение обратно результатов.

Рассмотрим способы организации подпрограмм в Паскале.

- **Процедура** — **подпрограмма, имеющая любое количество входных и выходных данных**.
- Процедура может быть описана **без параметров и с параметрами.**
- **Параметры** — это данные из заголовка процедуры, как передаваемые ей на обработку, так и получаемые в виде результатов.
- *Таким образом,* с помощью параметров происходит обмен информацией между процедурой и вызывающей ее программой.

Тема 9.

ПОДПРОГРАММЫ:
ПРОЦЕДУРЫ БЕЗ ПАРАМЕТРОВ

Процедуры без параметров

Описание процедуры имеет вид:

procedure имя;

{описание локальных переменных}

begin

{операторы}

end;

Процедуры без параметров

- Процедура без параметров может реализовывать любой алгоритм.
- Все переменные, над которыми производят действия операторы процедуры, определяются в вызывающей программе, им присваиваются необходимые для выполнения процедуры значения.

Рассмотрим пример

Задача 1.

- Рассмотрим пример вычисления наименьшего общего кратного двух натуральных чисел **НОК (X,Y)**, которое можно вычислить, используя наибольший общий делитель этих чисел, по формуле:
- **НОК(X,Y) = X*У/НОД(X,У).**

Вычисление НОК (X, У),

- При составлении программы оформим как процедуру без параметров программу E7 вычисление НОД по алгоритму Евклида.
- Результат работы процедуры будет заноситься в ячейку с именем M, переменная M описана как глобальный параметр и используется и главной программой, и процедурой.

Рассмотрим пример

Задача 1.

- В программе будем вычислять НОК нескольких чисел, занеся их в массив C.
- Этот массив формируется в разделе констант главной программы.
- **Если данные определяются в разделе констант**, то они не требуют дополнительного описания в разделе переменных (**var**).
- Переменная X сначала содержит значение первого числа, а затем ей присваивается результат — НОК двух первых чисел.

Вычисление НОК (X, Y),

- Переменная Y имеет своим значением второе число из пары, для которой вычисляется наименьшее общее кратное.
- Таким образом, при каждом шаге цикла вычисляется НОК двух чисел, первое из которых X содержит результат предыдущего шага.
- Вызов процедуры **NOD** вычисления наибольшего общего делителя осуществляется только по имени: **NOD**;

В содержание

Рассмотрим пример выполнения программы вычисления НОК нескольких чисел.

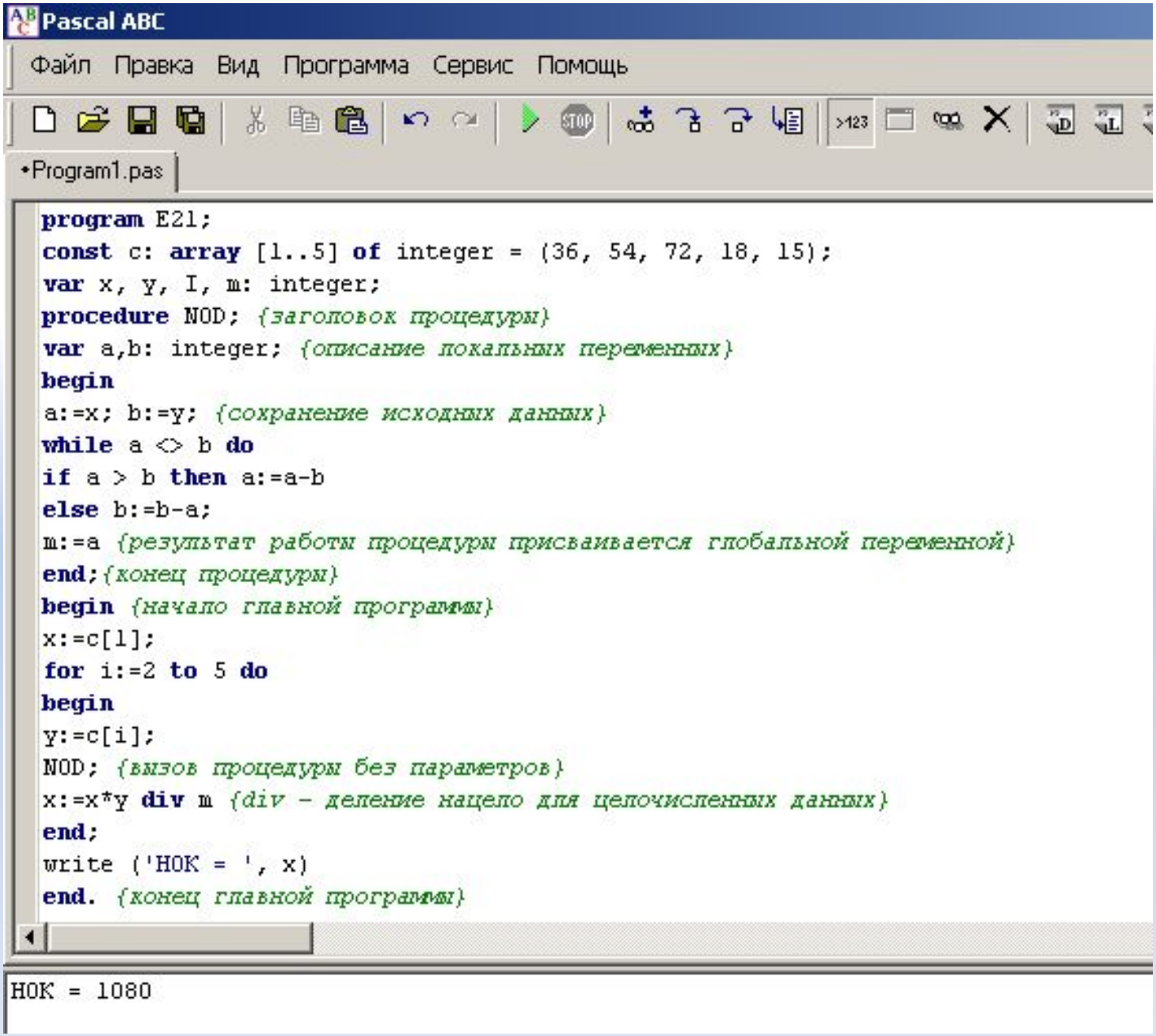
X :=НОК (X, Y)	36	108	216	216	1080
Y :=C[i]	54	72	18	15	
M :=НОД (X, Y)	18	36	18	3	

Текст программы имеет вид:

```
program E21;
const c: array [1..5] of integer = (36, 54, 72, 18,
      15);
var x, y, I, m: integer;

procedure NOD; {заголовок процедуры}
var a, b: integer; {описание локальных
      переменных}
begin
      a: = x; b: = y; {сохранение исходных
      данных}
      while a <> b do
        if a > b then a: = a-b
                 else b: = b-a;
        m: = a {результат работы процедуры
      присваивается глобальной
      переменной}
end; {конец процедуры}
```

```
begin {начало главной программы}
      x:= c [1];
  for i: = 2 to 5 do
      begin
         y:=c[i];
         NOD; {вызов процедуры без
      параметров}
         x: = x*y div m {div – деление
      нацело для целочисленных
      данных}
      end;
  write ('НОК = ', x)
end. {конец главной программы}
```


Pascal ABC
Файл Правка Вид Программа Сервис Помощь
•Program1.pas
program E21;
const c: array [1..5] of integer = (36, 54, 72, 18, 15);
var x, y, I, m: integer;
procedure NOD; {заголовок процедуры}
var a,b: integer; {описание покальных переменных}
begin
a:=x; b:=y; {сохранение исходных данных}
while a <> b do
if a > b then a:=a-b
else b:=b-a;
m:=a {результат работы процедуры присваивается глобальной переменной}
end;{конец процедуры}
begin {начало главной программы}
x:=c[1];
for i:=2 to 5 do
begin
y:=c[i];
NOD; {вызов процедуры без параметров}
x:=x*y div m {div - деление нацело для целочисленных данных}
end;
write ('НОК = ', x)
end. {конец главной программы}
НОК = 1080

Тема 9.

ПОДПРОГРАММЫ:
ПРОЦЕДУРЫ С ПАРАМЕТРАМИ

Формальные и фактические параметры

- Для удобства передачи данных в процедуру и получения из нее результата используются **формальные и фактические параметры.**
- **Формальные** — условные обозначения в описании процедуры — описываются в ее заголовке.
- **Фактические** — с которыми требуется выполнить процедуру — перечисляются при вызове процедуры.
- Формальные и фактические параметры должны соответствовать по количеству, типу и порядку следования.

Формальные и фактические параметры

- Формальные параметры описываются только в заголовке процедуры и больше нигде.
- Их описание похоже на описание данных в разделе переменных и может также содержать слово **var**.
- Слово **var** в заголовке процедуры ставится перед теми параметрами, имена которых соответствуют выходным данным.
- Фактические параметры, соответствующие формальным, перед которыми стоит слово **var**, могут быть только именами переменных.

Формальные и фактические параметры

- Перед именами формальных переменных, являющимися входными данными процедуры, слово **var** указывать не обязательно.
- Если перед формальным параметром в заголовке процедуры нет слова **var**, то ему может соответствовать формальный параметр, имеющий вид выражения соответствующего типа.
- Если для входных данных процедуры при описании формальных параметров указано слово **var**, то им также соответствуют фактические параметры — имена переменных.

Процедура NOD с параметрами

Например, процедура NOD с параметрами может иметь заголовок:

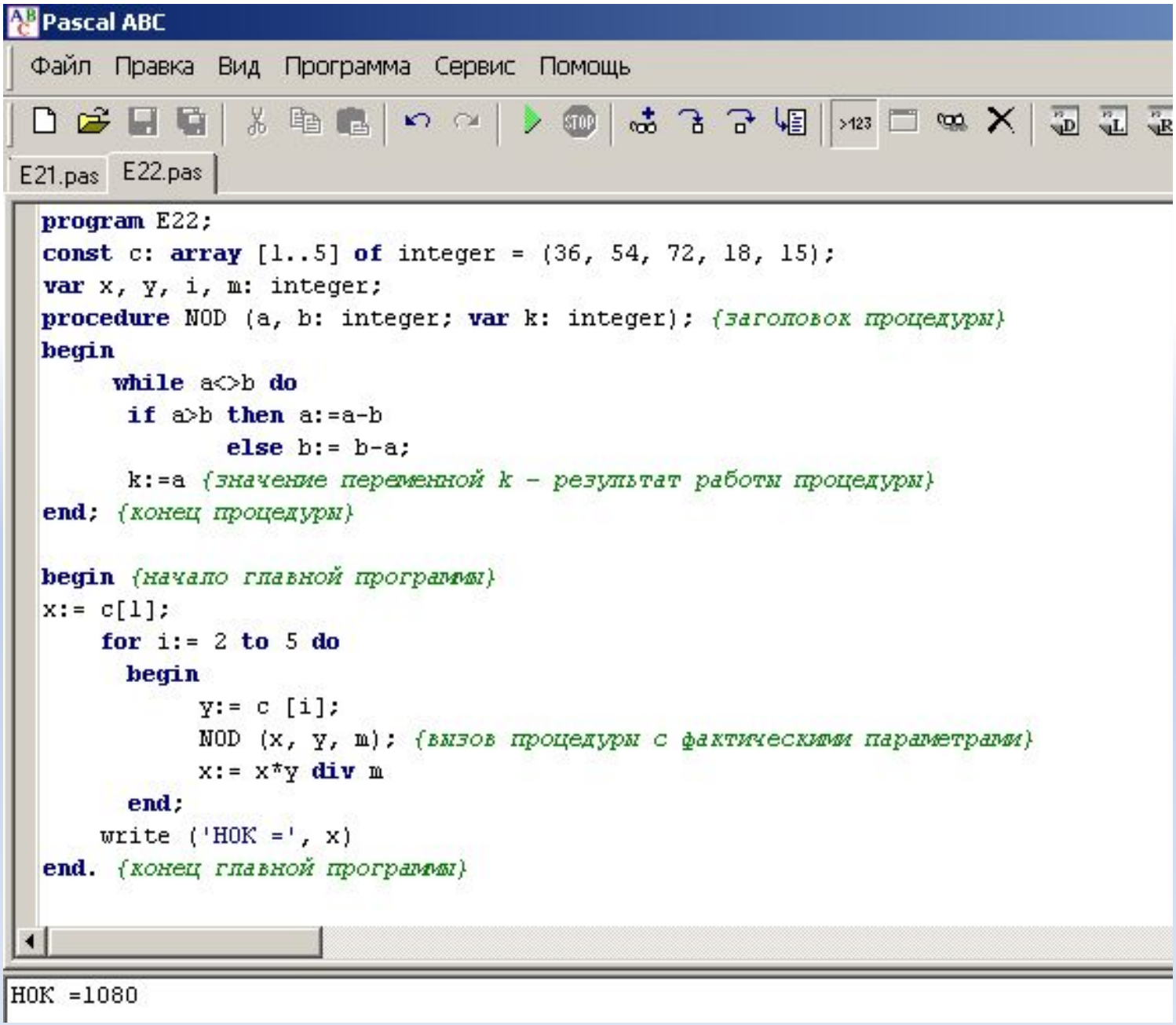
procedure NOD (a, b: integer; var k: integer);

Вызов этой процедуры: **NOD (x, y, m);** или: **NOD (36, 54, m);**

- Переменные в заголовке процедуры — ***формальные параметры***, заменяемые при выполнении процедуры на конкретные значения переменных ***x*** и ***y*** или числа ***36*** и ***54***.
- В заголовке процедуры **NOD** описаны формальные параметры:
  - ✔ ***a*** и ***b*** — входные данные, для которых находится наибольший общий делитель;
  - ✔ ***k*** — результат работы процедуры.
- При вызове процедуры переменная ***a*** примет значение ***x***, а переменная ***b*** — значение ***y***.
- Результат работы процедуры при вызове попадет в ячейку с именем ***m***, которой соответствует формальный параметр ***k***.

Программа при использовании процедуры с параметрами примет вид:

```
program E22;
const c: array [1..5] of integer = (36, 54, 72, 18, 15);
var x, y, i, m: integer;
procedure NOD (a, b: integer; var k: integer); {заголовок процедуры}
begin
    while a<>b do
     if a>b then a:=a-b
              else b:= b-a;
     k:=a {значение переменной k – результат работы процедуры}
end; {конец процедуры}
begin {начало главной программы}
x:= c[1];
   for i:= 2 to 5 do
     begin
         y:= c [i];
         NOD (x, y, m); {вызов процедуры с фактическими параметрами}
         x:= x*y div m
     end;
   write ('НОК =', x)
end. {конец главной программы}
```

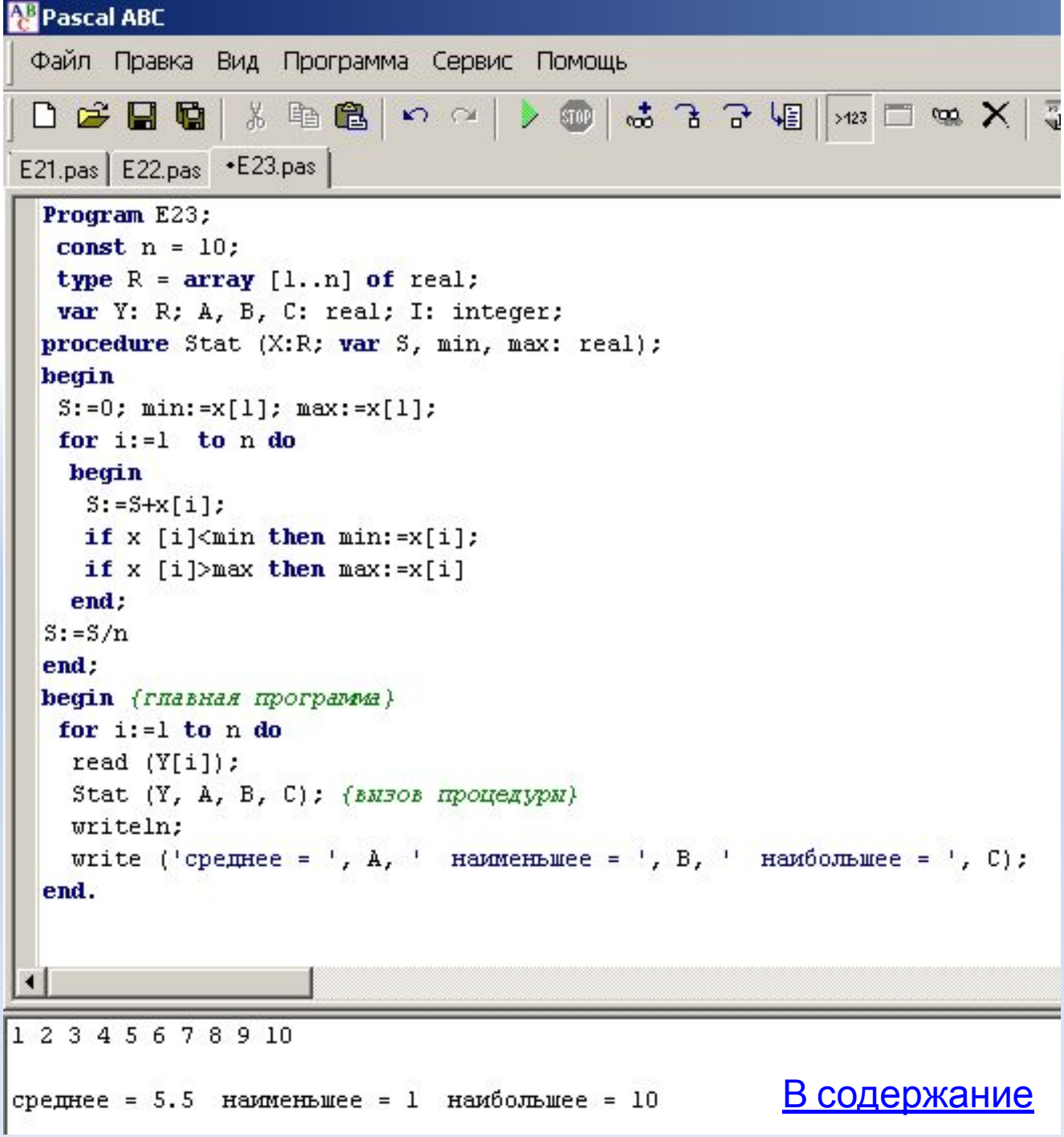

Pascal ABC
Файл Правка Вид Программа Сервис Помощь
E21.pas E22.pas
program E22;
const c: array [1..5] of integer = (36, 54, 72, 18, 15);
var x, y, i, m: integer;
procedure NOD (a, b: integer; var k: integer); {заголовок процедуры}
begin
while a<>b do
if a>b then a:=a-b
else b:= b-a;
k:=a {значение переменной k - результат работы процедуры}
end; {конец процедуры}
begin {начало главной программы}
x:= c[1];
for i:= 2 to 5 do
begin
y:= c [i];
NOD (x, y, m); {вызов процедуры с фактическими параметрами}
x:= x*y div m
end;
write ('НОК =', x)
end. {конец главной программы}
НОК =1080

Рассмотрим еще один пример использования процедуры с параметрами.

Найдем с помощью процедуры среднее арифметическое, наибольший и наименьший элементы массива.

```
Program E23;
 const n = 10;
 type R = array [1..n] of real;
 var Y: R; A, B, C: real; I: integer;
procedure Stat (X:R; var S, min, max: real);
begin
 S:=0; min:=x[1]; max:=x[1];
 for i:=  to n do
     begin
     S:=S+x[i];
     if x [i]<min then min:=x[i];
     if x [i]>max then max:=x[i]
    end;
   S:=S/n
end;
begin {главная программа}
   for i:=1 to n do
     read (Y[i]);
     Stat (Y, A, B, C); {вызов процедуры}
     writeln;
     write ('среднее = ', A, 'наименьшее = ', B, 'наибольшее = ', C);
end.
```

В содержание



В содержание

Раздел типов данных type

- В программе E23 появился новый раздел описаний — раздел типов данных **type**.
- В этом разделе можно описать новый тип данных через уже известные типы, которые могут быть так же ранее описаны в данном разделе.
- Тип данных **R** — это массивы из n вещественных чисел, **R** — имя типа.
- В дальнейшем этот тип позволяет сократить описания, он используется в главной программе при описании исходного массива **Y** и в заголовке процедуры при описании формального параметра — массива **X**.

Главная программа

- Главная программа состоит из трех основных этапов:
  1) ввода данных — массива **Y**;
  2) вызова процедуры **Stat** с фактическими параметрами — массивом **Y** и получаемыми результатами, попадающими соответственно в ячейки **A** (среднее значение),
     B (наименьшее) и **C** (наибольшее);
  3) печати результатов работы программы.

Тема 9.

ПОДПРОГРАММЫ:
ПОДПРОГРАММЫ-ФУНКЦИИ

Подпрограмма как функция

Подпрограмма, имеющая единственный результат, может быть оформлена, как функция.

function имя_функции (описание входных данных):

тип_результата;

{описания локальных переменных}

begin

{операторы}

имя_функции:= результат;

end;

После описания формальных параметров

- После описания формальных параметров, которые являются аргументами функции, в заголовке указывается тип результата, т. е. тип самой функции.
- Это описание относится к имени функции, которому необходимо присвоить значение результата работы подпрограммы.
- Как и процедура, функция может содержать все четыре раздела описаний локальных переменных.
- Имя функции нельзя использовать для промежуточных вычислений.

Функция вызывается с помощью указателя.

- Функция вызывается с помощью указателя.
- **Указатель** — это имя функции, после которого в круглых скобках перечислены фактические параметры — аргументы функции.
- Указатель имеет вид:

  имя_функции (список фактических параметров)
- Указатель может появиться в выражении соответствующею типа, в условиях операторов **If**, **while** и **repeat** после слова **until**, а также в операторе печати **write**.
- Примерами являются встроенные арифметические функции, такие, как sin(x):

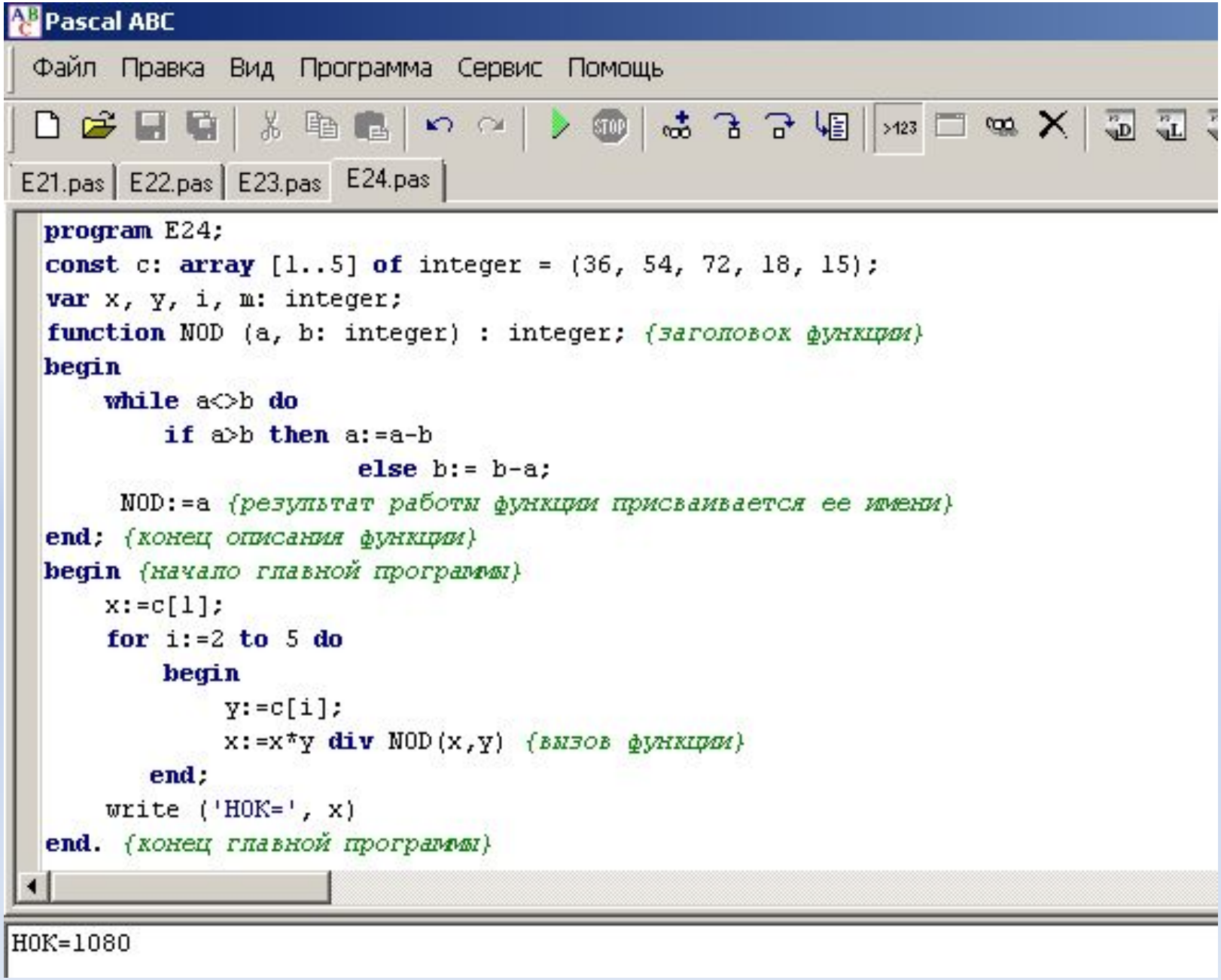
  write (sin(x));

Рассмотрим третий вариант программы вычисления наименьшего общего кратного.

- Поскольку наибольший общий делитель двух натуральных чисел — единственное число, то вычисляющую его подпрограмму можно оформить, как функцию.

Программа имеет вид:

```
program E24;
const c: array [1..5] of integer = (36, 54, 72, 18, 15);
var x, y, i, m: integer;
function NOD (a, b: integer) : integer;
begin
   while a<>b do
      if a>b then a:=a-b
                 else b:= b-a;
    NOD:=a
end;
begin
   x:=c[1];
   for i:=2 to 5 do
      begin
         y:=c[i];
         x:=x*y div NOD(x,y)
      end;
   write ('ÍÎÊ=', x)
end.
```


Pascal ABC
Файл Правка Вид Программа Сервис Помощь
E21.pas E22.pas E23.pas E24.pas
program E24;
const c: array [1..5] of integer = (36, 54, 72, 18, 15);
var x, y, i, m: integer;
function NOD (a, b: integer) : integer; {заголовок функции}
begin
while a<>b do
if a>b then a:=a-b
else b:= b-a;
NOD:=a {результат работы функции присваивается ее имени}
end; {конец описания функции}
begin {начало главной программы}
x:=c[1];
for i:=2 to 5 do
begin
y:=c[i];
x:=x*y div NOD(x,y) {вызов функции}
end;
write ('НОК=', x)
end. {конец главной программы}
НОК=1080

Вопросы и задания

1. Что такое подпрограмма и для чего она используется?
2. Объясните назначение локальных и глобальных переменных.
3. Как происходит обмен данными с процедурой без параметров?
4. Что такое формальные и фактические параметры?
5. К чему относится описание типа в конце заголовка подпрограммы-функции?
6. Чем отличается вызов функции от вызова процедуры?
7. Как задать значения элементов массива без использования оператора ввода?
8. Примеры программ предыдущих параграфов, кроме рассмотренных в данном, оформите с использованием процедур.

Литература

- **А.А.Кузнецов, Н.В.Ипатова**
  «Основы информатики», 8-9 кл.:
  - Раздел 3. ОСНОВЫ ПРОГРАММИРОВАНИЯ, С.122-129