Программирование на языке Паскаль

Программирование на языке Паскаль

§ 59. Процедуры

Зачем нужны процедуры?

```
writeln('Ошибка программы');
```

много раз!

```
program withProc;
var n: integer;
procedure Error;
begin
  writeln('Ошибка программы')
end;
begin
  read(n);
  if n < 0 then Error;
  ...
end.
```

вызов процедуры

Что такое процедура?

Процедура – вспомогательный алгоритм, который выполняет некоторые действия.

- текст (расшифровка) процедуры записывается **до основной программы**
- в программе может быть **много процедур**
- чтобы процедура заработала, нужно **вызвать** её по имени из основной программы или из другой процедуры

Процедура с параметрами

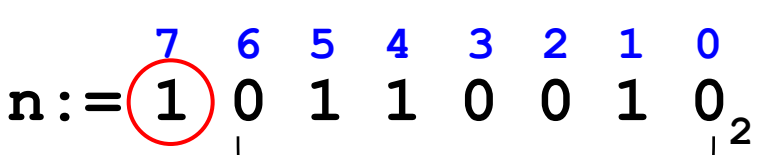
Задача. Вывести на экран запись целого числа (0..255) в 8-битном двоичном коде.

много раз!

Алгоритм:

$178 \Rightarrow 10110010_2$

? Как вывести первую цифру?



`n div 128`

`n mod 128`

? Как вывести вторую цифру?

`n1 div 64`

Процедура с параметрами

Задача. Вывести на экран запись целого числа (0..255) в 8-битном двоичном коде.

Алгоритм:

```
k:=128;
while k>0 do begin
  write(n div k);
  n:=n mod k;
  k:=k div 2
end;
```

n	k	вывод
178	128	1

178 ⇒ 10110010

Результат зависит от **n**!

Процедура с параметрами

```
program binCode;
 procedure printBin(n: integer);
 var k: integer;
 begin
   k:=128;
   while k>0 do begin
     write(n div k);
     n:=n mod k;
     k:=k div 2
   end
 end;
begin
  printBin(99)
end.
```

локальная переменная

значение параметра (**аргумент**)

Параметры – данные, изменяющие работу процедуры.

Несколько параметров

```
procedure printSred(a: integer;
                    b: integer);
begin
  write((a+b)/2);
end.
```

```
procedure printSred(a, b: integer);
begin
  write((a+b)/2);
end.
```

Задачи

№1: Напишите процедуру, которая принимает параметр – натуральное число N – и выводит на экран линию из N символов '–'.

Пример:

```
Введите N:
10
----------
```

№2: Напишите процедуру, которая выводит на экран в столбик все цифры переданного ей числа, начиная с первой.

Пример:

```
Введите натуральное число:
1234
1
2
3
4
```

Задачи

№3: Напишите процедуру, которая выводит на экран запись переданного ей числа в римской системе счисления.

Пример:

```
Введите натуральное число:
2013
MMXIII
```

Изменяемые параметры

Задача. Написать процедуру, которая меняет местами значения двух переменных.

передача по значению

```
program Exchange;
var x, y: integer;
procedure Swap(a, b: integer);
var c: integer;
begin
  c:=a; a:=b; b:=c;
end;
begin
 x:=2; y:=3;
 Swap(x, y);
 write(x, ' ', y)
end.
```

! Процедура работает с копиями переданных значений параметров!

```
2 3
```

? Почему не работает?

Изменяемые параметры

переменные могут изменяться

```
procedure Swap( var a, b: integer);
var c: integer;
begin
  c:=a; a:=b; b:=c;
end;
```

передача по **ссылке**

Вызов:

```
var a, b: integer;
...
Swap(a, b); { правильно }
Swap(2, 3); { неправильно }
Swap(a, b+3); { неправильно }
```

Задачи

№4: Напишите процедуру, которая переставляет три переданные ей числа в порядке возрастания.

Пример:

```
Введите три натуральных числа:
10 15 5
5 10 15
```

№5: Напишите процедуру, которая сокращает дробь вида M/N. Числитель и знаменатель дроби передаются как изменяемые параметры.

Пример:

```
Введите числитель и знаменатель дроби:
25 15
После сокращения: 5/3
```

Задачи

№6: Напишите процедуру, которая вычисляет наибольший общий делитель и наименьшее общее кратное двух натуральных чисел и возвращает их через изменяемые параметры.

Пример:

```
Введите два натуральных числа:
10 15
НОД(10,15)=5
НОК(10,15)=30
```

Программирование на языке Паскаль

§ 60. Функции

Что такое функция?

Функция – это вспомогательный алгоритм, который возвращает *значение-результат* (число, символ или объект другого типа).

Задача. Написать функцию, которая вычисляет сумму цифр числа.

Алгоритм:

```
сумма := 0;
while n <> 0 do begin
  сумма := сумма + n mod 10;
  n := n div 10
end;
```

Сумма цифр числа

```
program Sum;
function sumDigits(n: integer): integer;
var sum: integer;
begin
  sum:=0;
  while n<>0 do begin
    sum:=sum+n mod 10;
    n:= n div 10;
  end;
  sumDigits:=sum
end;
begin
  writeln(sumDigits(12345))
end.
```

тип результата

передача результата

Использование функций

```
x:=2*sumDigits(n+5);

z:= sumDigits(k) + sumDigits(m);

if sumDigits(n) mod 2 =0 then begin
  writeln('Сумма цифр чётная');
  writeln('Она равна ', sumDigits(n))
end;
```

Функция, возвращающая целое число, может использоваться везде, где и целая величина!

Задачи

№7: Напишите функцию, которая находит наибольший общий делитель двух натуральных чисел.

Пример:

```
Введите два натуральных числа:
7006652 112307574
НОД(7006652,112307574) = 1234.
```

№8: Напишите функцию, которая определяет сумму цифр переданного ей числа.

Пример:

```
Введите натуральное число:
123
Сумма цифр числа 123 равна 6.
```

Задачи

№9: Напишите функцию, которая «переворачивает» число, то есть возвращает число, в котором цифры стоят в обратном порядке.

Пример:

```
Введите натуральное число:
1234
После переворота: 4321.
```

Логические функции

Задача. Найти все простые числа в диапазоне от 2 до 100.

```
program PrimeNum;
var i: integer;
begin
  for i:=2 to 100 do
    if isPrime(i) then
      writeln(i)
end.
```

функция, возвращающая логическое значение (True/False)

Функция: простое число или нет?

? Какой алгоритм?

логическое значение (True/False)

```
function isPrime(n: integer): boolean;
var count, k: integer;
begin
  count:=0;
  k:=2;
  while (k*k<=n) and (count=0) do begin
    if n mod k=0 then
       count:=count+1;
    k:=k+1
  end;
  isPrime:=(count = 0)
end;
```

```
if count=0 then
    isPrime:=True
else isPrime:=False
```

Логические функции: использование

Функция, возвращающая логическое значение, может использоваться везде, где и логическая величина!

```
read(n);
while isPrime(n) do begin
  writeln('простое число');
  read(n)
end;
```

Задачи

№10: Напишите логическую функцию, которая определяет, является ли переданное ей число совершенным, то есть, равно ли оно сумме своих делителей, меньших его самого.

Пример:

```
Введите натуральное число:
28
Число 28 совершенное.
```

Пример:

```
Введите натуральное число:
29
Число 29 не совершенное.
```

Задачи

№11: Напишите логическую функцию, которая определяет, являются ли два переданные ей числа взаимно простыми, то есть, не имеющими общих делителей, кроме 1.

Пример:

```
Введите два натуральных числа:
28 15
Числа 28 и 15 взаимно простые.
```

Пример:

```
Введите два натуральных числа:
28 16
Числа 28 и 16 не взаимно простые.
```

Задачи

№12: Простое число называется гиперпростым, если любое число, получающееся из него откидыванием нескольких цифр, тоже является простым. Например, число 733 – гиперпростое, так как и оно само, и числа 73 и 7 – простые. Напишите логическую функцию, которая определяет, верно ли, что переданное ей число – гиперпростое. Используйте уже готовую функцию **`isPrime`**, которая приведена в учебнике.

Пример:

```
Введите натуральное число:
733
Число 733 гиперпростое.
```

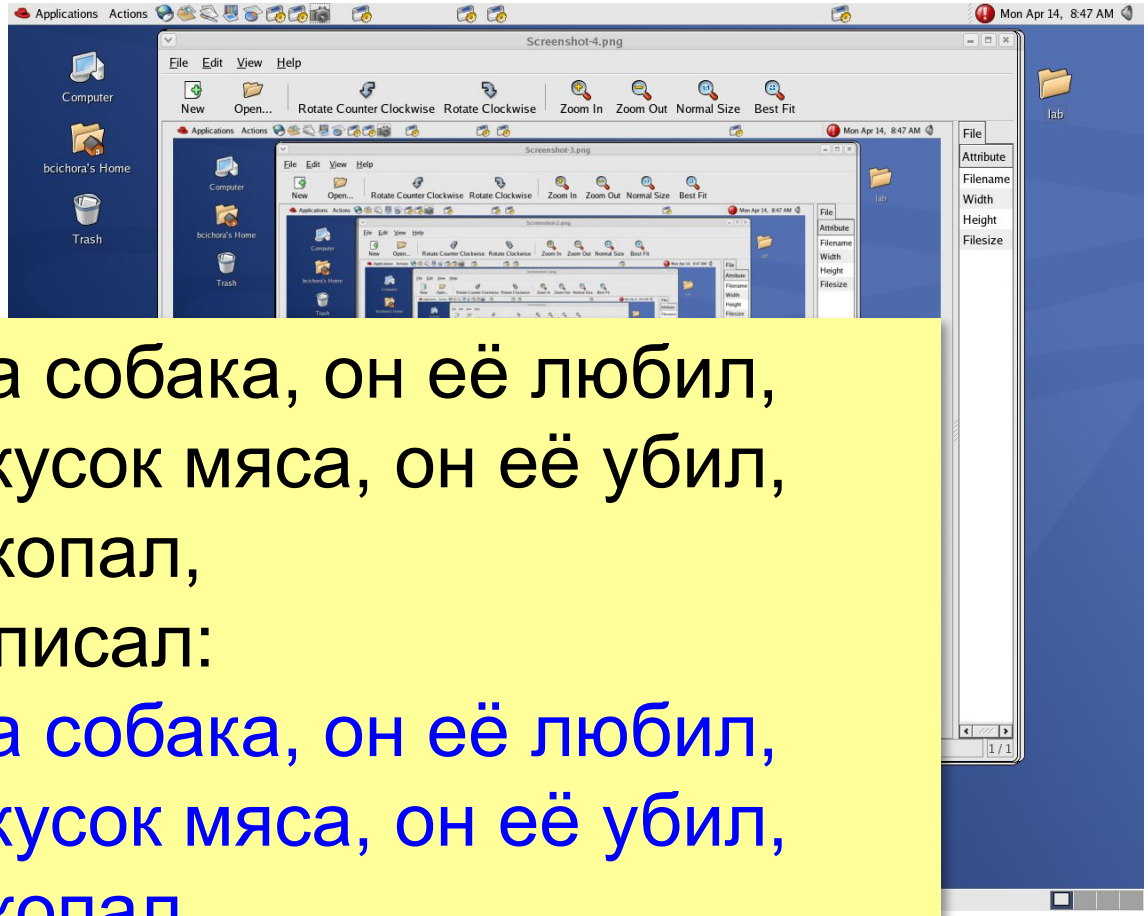
Пример:

```
Введите натуральное число:
19
Число 19 не гиперпростое.
```

Программирование на языке Паскаль

§ 61. Рекурсия

Что такое рекурсия?



У попа была собака, он её любил,
Она съела кусок мяса, он её убил,
В землю закопал,
Надпись написал:
У попа была собака, он её любил,
Она съела кусок мяса, он её убил,
В землю закопал,
Надпись написал:
…

Что такое рекурсия?

Натуральные числа:

- 1 – натуральное число
- если n – натуральное число, то $n+1$ – натуральное число

индуктивное определение

Рекурсия — это способ определения множества объектов через само это множество на основе заданных простых базовых случаев.

Числа Фибоначчи:

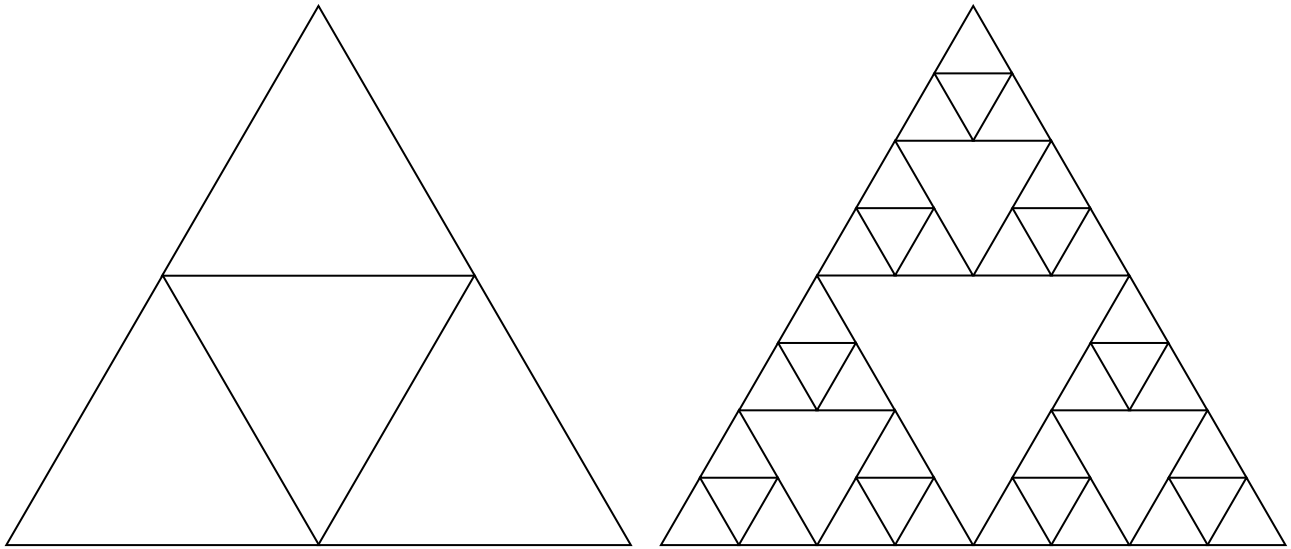
- $F_1 = F_2 = 1$
- $F_n = F_{n-1} + F_{n-2}$ при $n > 2$

1, 1, 2, 3, 5, 8, 13, 21, 34, …

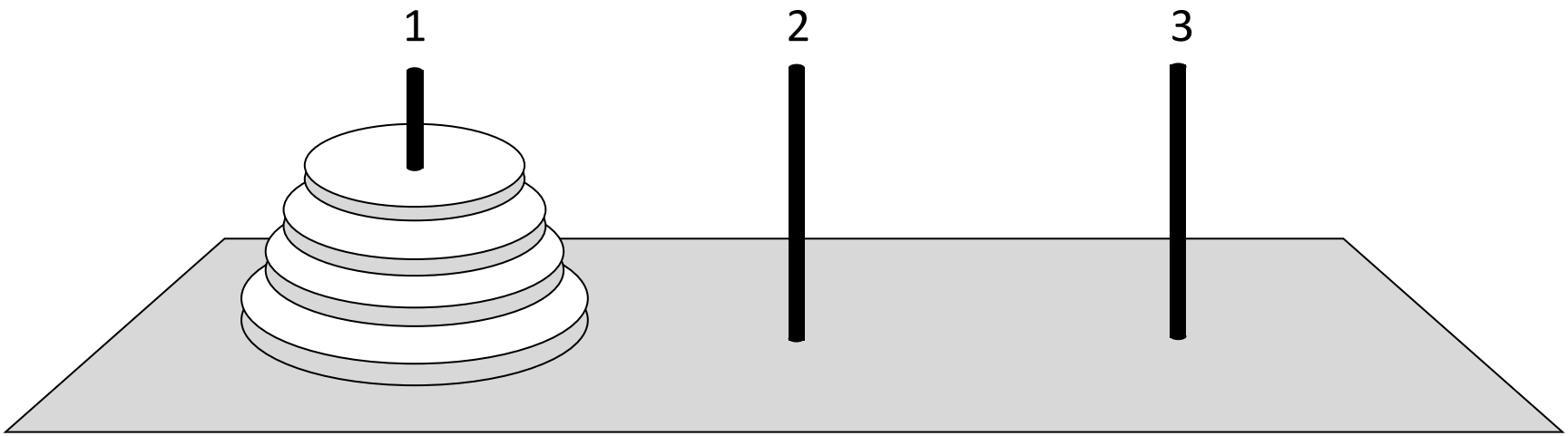
Фракталы

Фракталы – геометрические фигуры, обладающие самоподобием.

Треугольник Серпинского:



Ханойские башни



- за один раз переносится один диск
- класть только меньший диск на больший
- третий стержень вспомогательный

```
перенести (n, 1, 3)
```

```
перенести (n-1, 1, 2)
1 -> 3
перенести (n-1, 2, 3)
```

Ханойские башни – процедура

сколько | откуда | куда

```
proceduer Hanoi(n, k, m: integer);
var p: integer;
begin
  p := 6 - k - m;
  Hanoi(n-1, k, p);
  writeln(k, ' -> ', m);
  Hanoi(n-1, p, m)
end;
```

номер вспомогательного стержня (1+2+3=6!)

рекурсия

рекурсия

? Что плохо?

! **Рекурсия никогда не остановится!**

Ханойские башни – процедура

Рекурсивная процедура (функция) — это процедура (функция), которая вызывает сама себя напрямую или через другие процедуры и функции.

```
procedure Hanoi(n, k, m: integer);
var p: integer;
begin
  if n = 0 then exit;
  p := 6 - k - m;
  Hanoi(n-1, k, p);
  writeln(k, ' -> ', m);
  Hanoi(n-1, p, m)
end;
```

условие выхода из рекурсии

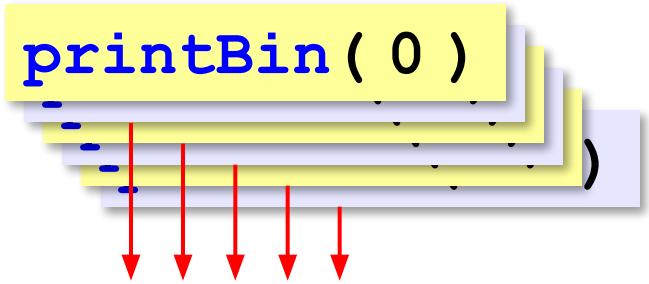
```
program HanoiTower;
...
begin
  Hanoi(4, 1, 3)
end.
```

Вывод двоичного кода числа

```
procedure printBin(n: integer);
begin
   if n = 0 then exit;
   printBin ( n div 2 );
   write( n mod 2 )
end;
```

условие выхода из рекурсии

напечатать все цифры, кроме последней



вывести последнюю цифру

? Как без рекурсии?

Вычисление суммы цифр числа

```
function sumDig(n: integer): integer;
var sum: integer;
нач
   sum:= n mod 10;
   if n >= 10 then
      sum:= sum + sumDig( n div 10 );
   sumDig:= sum
end;
```

последняя цифра

рекурсивный вызов

? Где условие окончания рекурсии?

```
sumDig( 1234 )
   4 + sumDig( 123 )
      4 + 3 + sumDig( 12 )
         4 + 3 + 2 + sumDig( 1 )
            4 + 3 + 2 + 1
```

Алгоритм Евклида

Алгоритм Евклида. Чтобы найти НОД двух натуральных чисел, нужно вычитать из большего числа меньшее до тех пор, пока меньшее не станет равно нулю. Тогда второе число и есть НОД исходных чисел.

```
function NOD(a, b: integer): integer;
begin
  if (a = 0) or (b = 0) then begin
    NOD:= a + b;
    exit
  end;
  if a > b then
      NOD:= NOD(a - b, b)
  else NOD:= NOD(a, b - a)
end;
```

условие окончания рекурсии

рекурсивные вызовы

Задачи

№13: Напишите рекурсивную функцию, которая вычисляет НОД двух натуральных чисел, используя модифицированный алгоритм Евклида.

Пример:

```
Введите два натуральных числа:
7006652 112307574
НОД(7006652,112307574)=1234.
```

№14: Напишите рекурсивную функцию, которая раскладывает число на простые сомножители.

Пример:

```
Введите натуральное число:
378
378 = 2*3*3*3*7
```

Задачи

№15: Дано натуральное число N. Требуется получить и вывести на экран количество всех возможных *различных* способов представления этого числа в виде суммы натуральных чисел (то есть, 1 + 2 и 2 + 1 – это один и тот же способ разложения числа 3). Решите задачу с помощью рекурсивной процедуры.

Пример:

```
Введите натуральное число:
4
Количество разложений: 4.
```

Как работает рекурсия?

Факториал:

$$N! = \begin{cases} 1, & N = 1 \\ N \cdot (N-1)!, & N > 1 \end{cases}$$

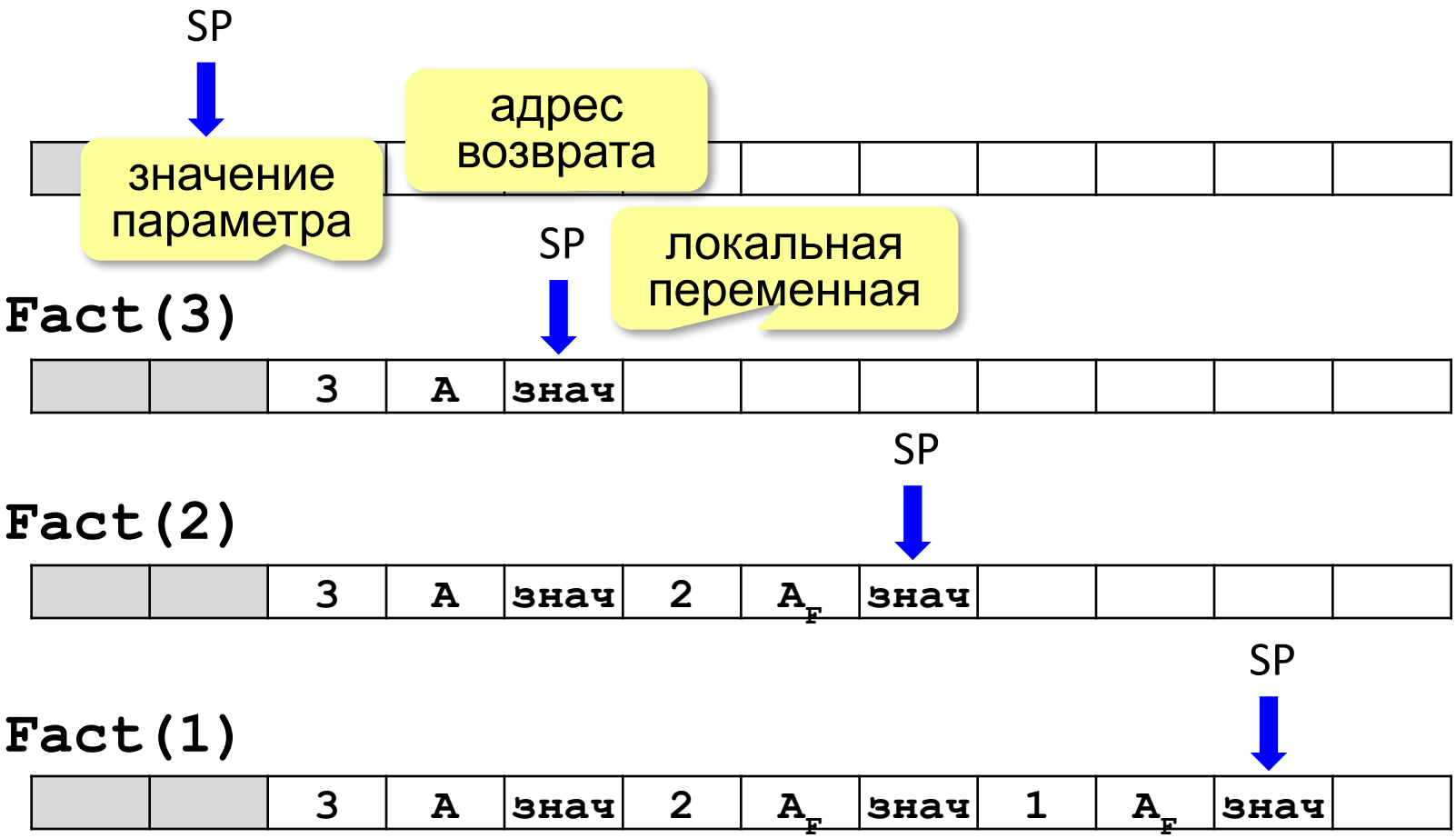
```
function Fact(N: integer): integer;
begin
  writeln('-> N = ', N);
  if N <= 1 then
       Fact:= 1
  else Fact:= N * Fact(N-1);
  writeln('<- N = ', N)
end;
```

```
-> N = 3
  -> N = 2
    -> N = 1
    <- N = 1
  <- N = 2
<- N = 3
```

? Как сохранить состояние функции перед рекурсивным вызовом?

Стек

Стек – область памяти, в которой хранятся локальные переменные и адреса возврата.



Рекурсия – «за» и «против»

- с каждым новым вызовом расходуется память в стеке (возможно переполнение стека)
- затраты на выполнение служебных операций при рекурсивном вызове

- программа становится более короткой и понятной

- возможно переполнение стека
- замедление работы

Любой рекурсивный алгоритм можно заменить нерекурсивным!

итерационный алгоритм

```
function Fact(N: integer):
                    integer;
var i, F: integer;
begin
  F:=1;
  for i:=1 to N do
    F:= F * i;
  Fact:= F
end;
```