



# ЗАПИСЬ ВСПОМОГАТЕЛЬНЫХ АЛГОРИТМОВ НА ЯЗЫКЕ ПАСКАЛЬ

## АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

9 класс



ИЗДАТЕЛЬСТВО

БИНОМ

# Ключевые слова

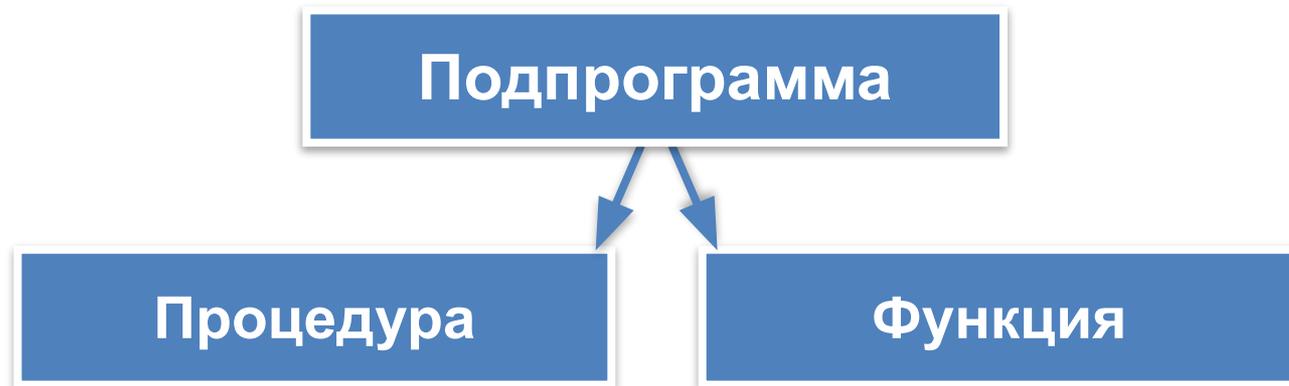
- подпрограмма
- процедура
- функция
- рекурсивная функция



# Подпрограммы

Запись вспомогательных алгоритмов в языках программирования осуществляется с помощью *подпрограмм*.

Структура описания подпрограммы аналогична структуре главной программы. Описание подпрограммы начинается с заголовка и заканчивается оператором **end**



# Процедуры

**Процедура** - подпрограмма, имеющая произвольное количество входных и выходных данных.

— Входные параметры:  
переменные, константы,  
выражения

**procedure** <имя\_процедуры> (<описание параметров-значений>; **var**: <описание параметров-переменных>);

**begin**

<операторы>

**end**;

— Выходные  
параметры

Для вызова процедуры достаточно указать её имя со списком фактических параметров.

# Алгоритм Евклида

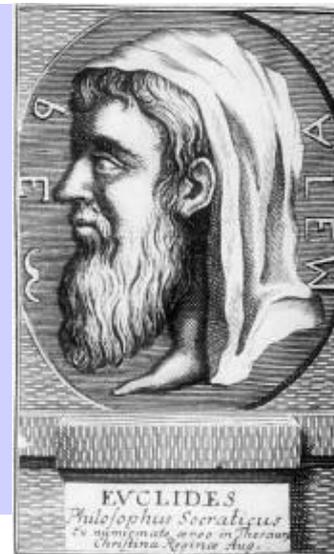
Алгоритм Евклида



Package

## Процедура для нахождения НОД

```
procedure nod (a, b: integer; var c: integer);  
begin  
  while a<>b do  
    if a>b then a:=a-b else b:=b-a;  
  c:=a  
end;
```



# Варианты вызова процедуры

`nod (36, 15, z)`

в качестве параметров-значений  
использованы константы

`nod (x, y, z)`

в качестве параметров-значений  
использованы имена переменных

`nod (x+ y, 15, z)`

в качестве параметров-значений  
использованы выражение и константа



Между фактическими и формальными параметрами должно быть полное соответствие по количеству, порядку следования и типу.

# Программа с процедурой

```
program n_6;
```

*Заголовок главной программы*

```
const m: array [1..6] of integer =(16, 32, 40, 64, 80, 128);
```

*Описание констант*

```
var l, x, y, z: integer;
```

*Раздел описания переменных*

```
procedure nod (a, b: integer; var c: integer);
```

```
begin
```

```
  while a<>b do
```

```
    if a>b then a:=a-b else b:=b-a;
```

```
  c:=a
```

```
end;
```

*Раздел описания подпрограммы*

```
begin
```

```
  x:=m[1];
```

```
  for i:=2 to 6 do
```

```
  begin
```

```
    y:=m[i];
```

```
    nod (x, y, z);
```

```
    x:=z
```

```
  end;
```

```
writeln ('НОД=', x)
```

```
end.
```

*Раздел описания операторов главной программы*

# Функции

**Функция** - подпрограмма, имеющая единственный результат, записываемый в ячейку памяти, имя которой совпадает с именем функции.

Перечень формальных параметров и их типов

**function** <имя\_функции> (<описание входных данных>):  
<тип\_функции>;

**begin**

<операторы>;

<имя\_функции> := <результат>

**end;**

Тип результата

В блоке функции обязательно должен присутствовать оператор **<имя\_функции>:=<результат>**.

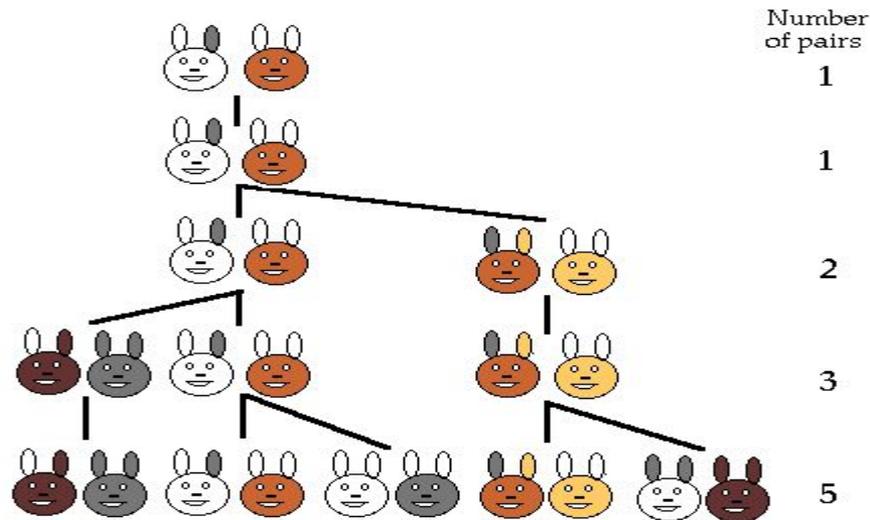
Для вызова функции достаточно указать её имя со списком фактических параметров в любом выражении, в условиях, (после слов if, while, until) или в операторе **write** главной программы.



# Последовательность Фибоначчи

В январе Саше подарили пару новорождённых кроликов. Через два месяца они дали первый приплод - новую пару кроликов, а затем давали приплод по паре кроликов каждый месяц.

Каждая новая пара также даёт первый приплод (пару кроликов) через два месяца, а затем - по паре кроликов каждый месяц. Сколько пар кроликов будет у Саши в декабре?



Числа 1, 1, 2, 3, 5, 8, ... образуют так называемую **последовательность Фибоначчи**, названную в честь итальянского математика, впервые решившего соответствующую задачу ещё в начале XIII века.

# Математическая модель

Пусть  $f(n)$  количество пар кроликов в месяце с номером  $n$ .

По условию задачи:

$$f(1) = 1,$$

$$f(2) = 1,$$

$$f(3) = 2.$$

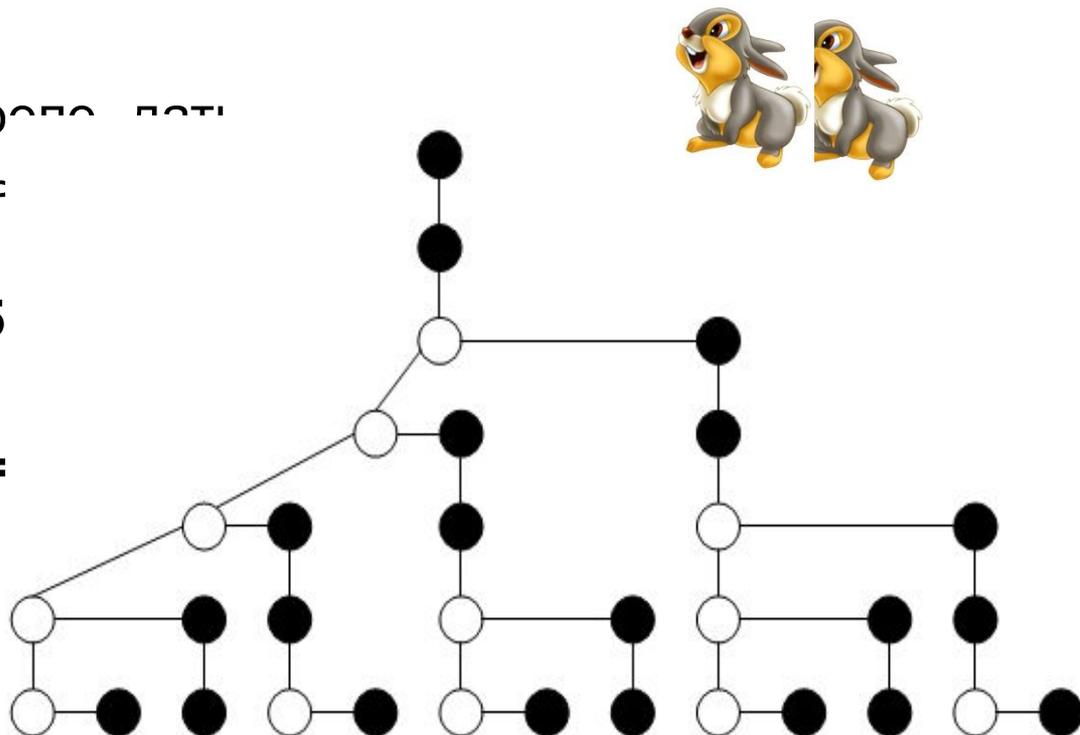
Из двух пар, имеющих в марте, дать приплод в апреле сможет только одна:  $f(4) = 3$ .

Из пар, имеющих в апреле, дать приплод в мае смогут только родившиеся в марте и ранее:

$$f(5) = f(4) + f(3) = 3 + 2 = 5$$

В общем случае:

$$f(n) = f(n-1) + f(n-2), n >=$$



# Функция

```
function f (n: integer): integer;  
begin  
  if (n=1) or (n=2) then f:=1  
  else f:=f(n-1)+f(n-2)  
end;
```

Полученная функция *рекурсивная* - в ней реализован способ вычисления очередного значения функции через вычисление её предшествующих значений.

$$\sqrt{3 + \sqrt{3 + \sqrt{3 + \sqrt{3 + \sqrt{3 + \dots}}}}}$$



# Самое главное

Запись вспомогательных алгоритмов в языках программирования осуществляется с помощью **подпрограмм**. В Паскале различают два вида подпрограмм: процедуры и функции.

**Процедура** - подпрограмма, имеющая произвольное количество входных и выходных данных.

**Функция** - подпрограмма, имеющая единственный результат, записываемый в ячейку памяти, имя которой совпадает с именем функции.



# Вопросы и задания

Напишите программу перестановки значений переменных  $a$ ,  $b$ ,  $c$  в порядке возрастания, т. е. так, чтобы  $a < b < c$ . Используйте

процедуру `swap` (var x, y: integer);

общего назначения для перестановки элементов массива.

15. Напишите программу, которая выводит на экран значение наибольшего

элемента массива.

16. Напишите программу, которая выводит на экран значение наибольшего

элемента массива.

17. Напишите программу, которая выводит на экран значение наибольшего

элемента массива.

Исходные данные вводятся с клавиатуры.

Пример входных данных	Пример выходных данных
1 2 3	1 2 3
2 1 3	1 2 3
3 1 2	1 2 3
2 3 1	1 2 3

# Опорный конспект

Запись вспомогательных алгоритмов в языках программирования осуществляется с помощью **подпрограмм**.

Подпрограмма

```
graph TD; A[Подпрограмма] --> B[Процедура]; A --> C[Функция]; B --> D[Подпрограмма, имеющая произвольное количество входных и выходных данных.]; C --> E[Подпрограмма, имеющая единственный результат, записываемый в ячейку памяти, имя которой совпадает с именем функции.];
```

Процедура

Подпрограмма, имеющая произвольное количество входных и выходных данных.

Функция

Подпрограмма, имеющая единственный результат, записываемый в ячейку памяти, имя которой совпадает с именем функции.