



ЗАПИСЬ ВСПОМОГАТЕЛЬНЫХ АЛГОРИТМОВ НА ЯЗЫКЕ ПАСКАЛЬ

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

9 класс

Ключевые слова

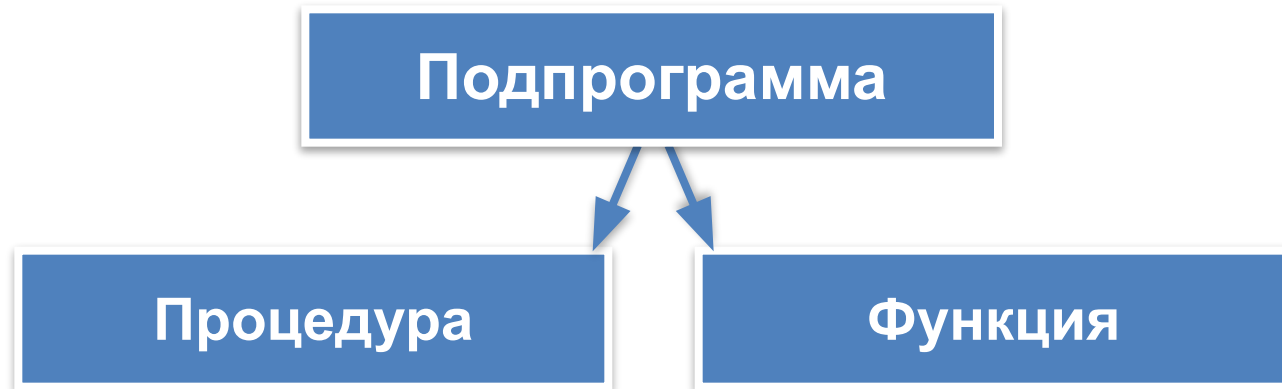
- подпрограмма
- процедура
- функция
- рекурсивная функция



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

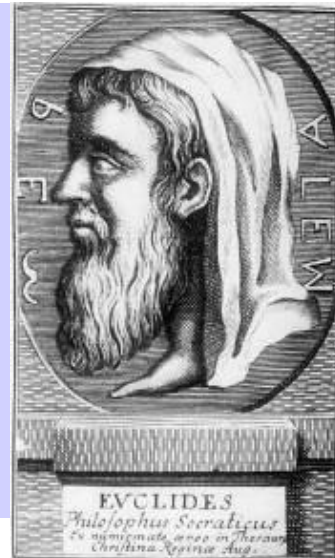
Алгоритм Евклида



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.

Функция поиска максимального из 2-х

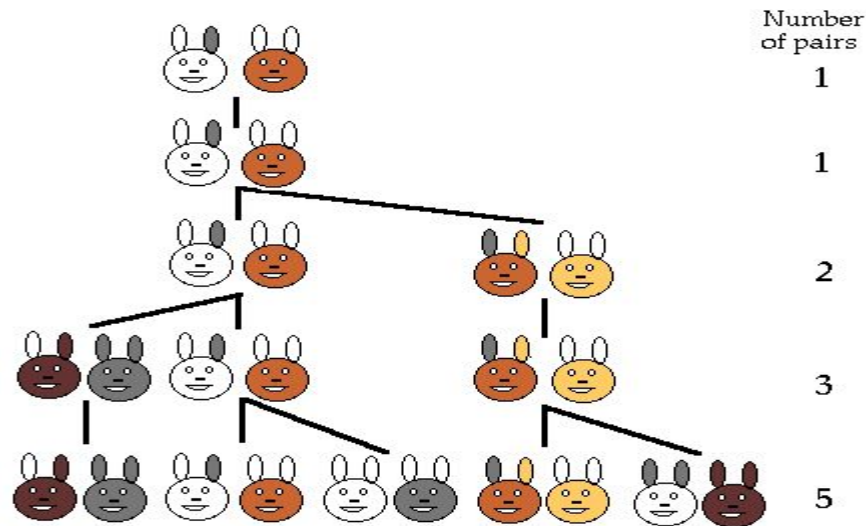
program n_7;	<i>Заголовок главной программы</i>
var a, b, c, d, f: integer;	<i>Описание переменных</i>
function max (x, y: integer): integer; begin if x>y then max:=x else max:=y; end;	<i>Раздел описания подпрограммы</i>
begin readln (a, b, c, d); f:= max(max(a, b), max(c, d)); writeln ('f=', f); end.	<i>Раздел операторов главной программы (поиск максимального из 4-х чисел)</i>



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.

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

1. Для чего используются подпрограммы?
2. В чём основное различие процедур и функций?
3. Напишите программу вычисления наименьшего общего кратного следующих четырёх чисел: 36, 54, 18 и 15.
Используйте процедуру вычисления наибольшего общего делителя двух чисел.

Вопросы и задания

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

```
procedure swap (var x, y: integer);  
  var m: integer;  
  begin  
    m:=x;  
    x:=y;  
    y:=m  
  end;
```

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

Вопросы и задания

5. Напишите программу поиска наибольшего из четырёх чисел с использованием подпрограммы поиска наибольшего из трёх чисел.
6. Видоизмените программу сортировки массива выбором так, чтобы в ней использовалась процедура выбора наибольшего элемента массива.
7. Напишите программу вычисления выражения:
$$s = 1! + 2! + 3! + \dots + n!$$
Используйте функцию вычисления факториала.

Вопросы и задания

8. Напишите программу вычисления выражения:

$$s = x^3 + x^5 + x^n,$$

где x и n вводятся с клавиатуры.

Используйте функцию вычисления степени.

9. Напишите функцию, вычисляющую длину отрезка по координатам его концов. С помощью этой функции напишите программу, вычисляющую периметр треугольника по координатам его вершин.

10. Напишите функцию, вычисляющую площадь треугольника по целочисленным координатам его вершин.

С помощью этой функции вычислите площадь четырёхугольника по координатам его вершин.

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.