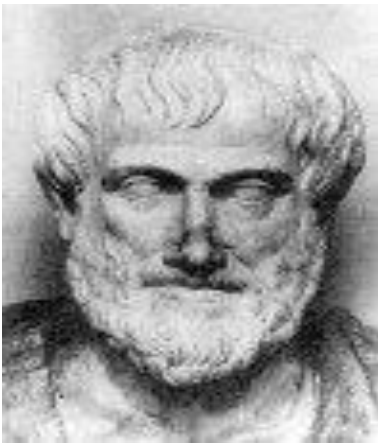


# *подпрограммы*

*Разработка: Барбаровой А.Л. –  
учителя информатики  
теор. лицей      п.Светлый*



**«Ум заключается не  
только в знании, но и в  
умении прилагать знания  
на деле.»**

*Аристотель.*

Тема :

*подпрограммы*

## Субкомпетенции :

1. Обработка данных с помощью стандартных подпрограмм и подпрограмм, определённых пользователем.
2. Организация передачи данных между вызывающей программой\подпрограммой и вызываемой подпрограммой. .
3. Структурное проектирование алгоритма и программы.

В программах часто приходится повторять некоторые аналогичные действия многократно.

Используя подпрограммы, можно единожды описать действия в подпрограмме, а затем лишь только вызывать ее.

Такой принцип дефрагментации программы называется **нисходящим программированием** и соответствует принципам структурного программирования, в основу которого и положено понятие подпрограммы.

# Что такое подпрограмма?

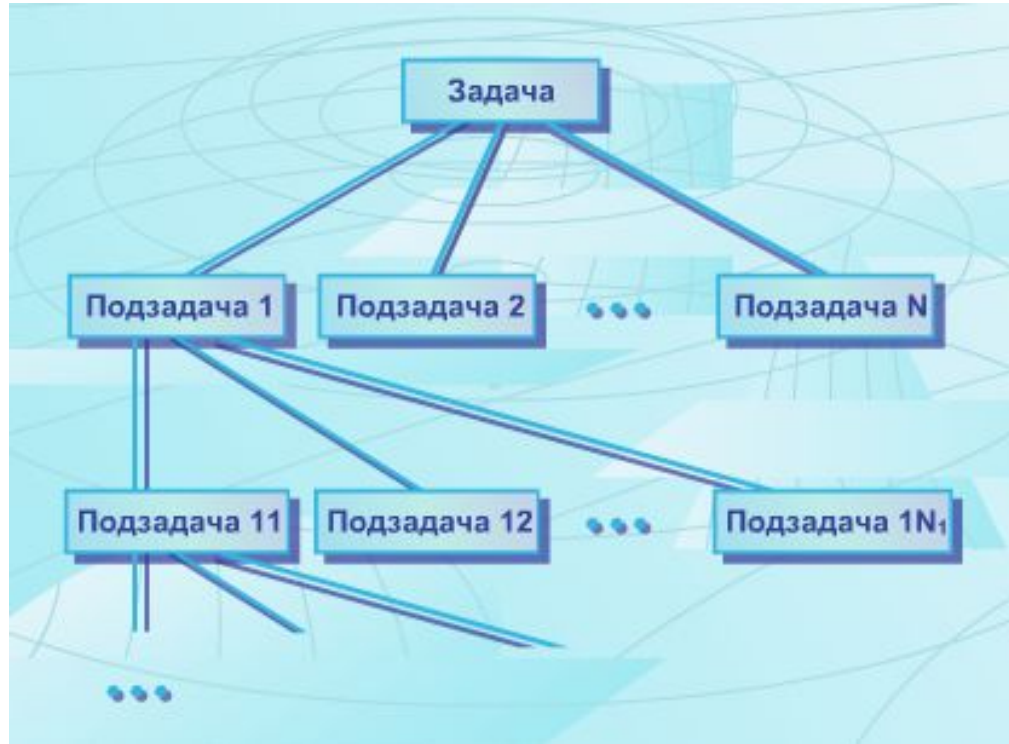
**Подпрограмма** — повторяющаяся группа операторов, оформленная в виде самостоятельной программной единицы.

Записывается однократно, а в соответствующих местах программы обеспечивается обращение к ней (ссылка).

## Для чего используют подпрограммы?

Подпрограммы используют, чтобы **сократить объем и улучшить структуру программы** с точки зрения наглядности и читаемости, **уменьшить вероятность ошибок** и **облегчить процесс отладки** программы.

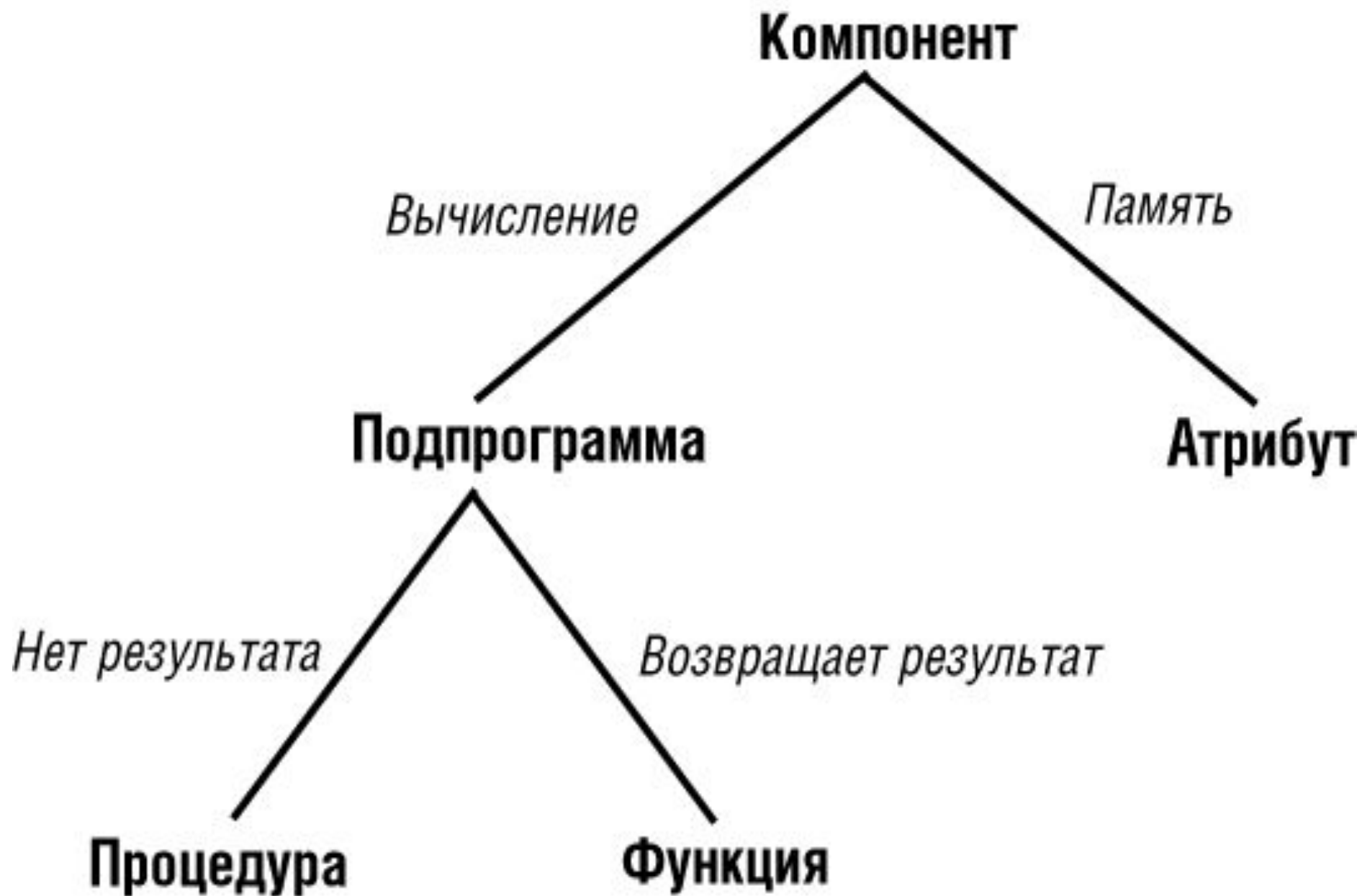
# Принцип нисходящего программирования



При создании программы для решения сложной задачи выполняется разделение этой задачи на подзадачи, этих подзадач – на более мелкие подзадачи и так далее до тех пор, пока подзадачи не станут легко программируемыми.

Для такой организации используются **подпрограммы**.





# В языке Паскаль подпрограммы реализуются в виде процедур и функций.

процедура	функция
предназначена	предназначена
для выполнения законченной последовательности действий	для выполнения законченной последовательности действий, результатом которой является один параметр
заголовок <b>procedure</b> <ИМЯ>(<список формальных параметров>);	заголовок <b>function</b> <ИМЯ > ( <список формальных параметров>): <тип >
обращение к процедуре <u>&lt;ИМЯ&gt;</u> (< список фактических параметров >);	обращение к функции <ИМЯ> (< список фактических параметров >);
осуществляется в любом месте программы	функция записывается только в выражениях (т.е. справа от знака :=)

**При вызове процедуры или функции  
формальные параметры, указанные в  
заголовке, ...**

заменяются фактическими  
параметрами в порядке их следования.

**Формальные параметры — это**

**переменные**, формально присутствующие в процедуре и определяющие тип и место подстановки фактических параметров.

**Фактические параметры — это**

**реальные объекты программы**, заменяющие в теле процедуры при ее вызове формальные параметры.

# Соответствие между фактическими и формальными параметрами должно быть следующим:

- число фактических параметров должно быть равно числу формальных параметров;
- соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу.

# Процедуры



```
graph TD; A[Процедуры] --> B[Без параметров]; A --> C[С параметрами]; C --> D[Параметры – переменные]; C --> E[Параметры – значения];
```

Без параметров

С параметрами

Параметры –  
переменные

Параметры –  
значения

**Параметры - значения** в основной программе не меняются.

Используются для передачи исходных данных в подпрограмму (отсутствует слово `var`).

**Параметры - переменные** подпрограмма может изменить в основной программе.

Используются для определения результатов выполнения процедуры (обязательно `var`).

# Пример 1

```
program primer1;  
  procedure ok1;  
    begin  
      writeln ('Это подпрограмма.');    end;  
  procedure ok2;  
    begin  
      writeln ('Еще одна подпрограмма.');    end;  
begin  
  writeln ('Привет!');  
  ok1;  
  writeln ('Снова привет!');  
  ok1;  
  writeln ('А это что такое?');  
  ok2;  
end.
```

Привет!

Это подпрограмма.

Снова привет!

Это подпрограмма.

А это что такое?

Еще одна  
подпрограмма.



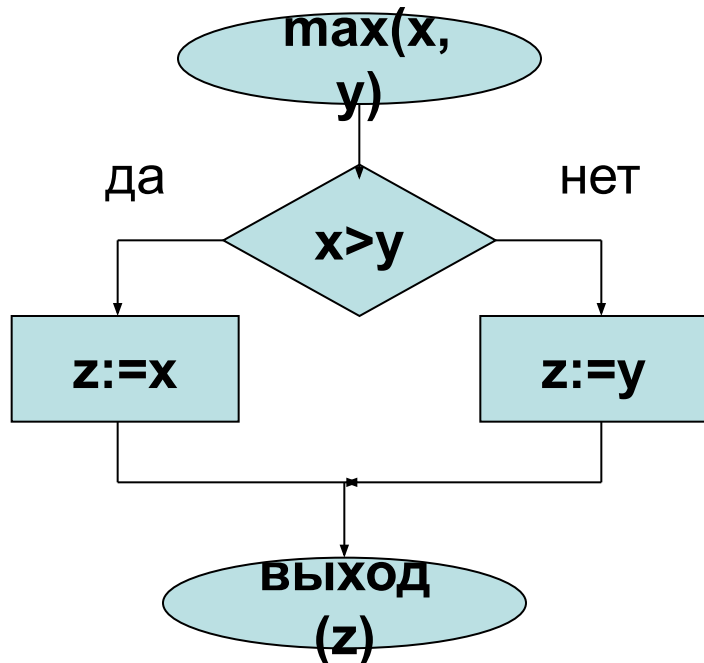
Найти большее из трех данных чисел, используя подпрограмму нахождения большего из двух.

## Пример 2

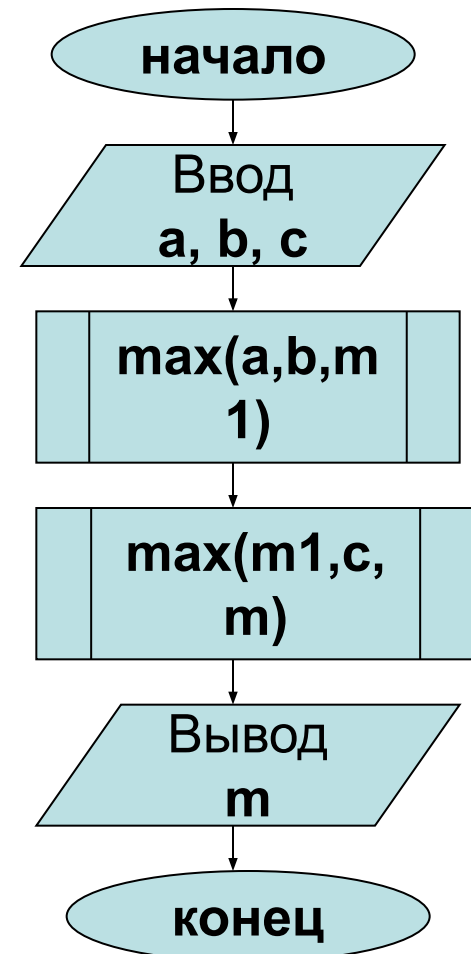
Основной алгоритм

### I. Используем процедуру

Вспомогательный алгоритм



$x, y, z$  - Формальные параметры



$a, b, c$  - Фактические параметры

## Используем процедуру

## Пример 2

```
program pr2;  
  var a, b, c, m, m1: real;  
  procedure max(x, y: real; var z: real);  
    begin  
      if x>y then z:=x else z:=y  
    end;  
begin  
  writeln('a='); readln (a);  
  writeln('b='); readln (b);  
  writeln('c='); readln (c);  
  max(a, b, m1);  
  max(m1, c, m);  
  writeln ('max= ', m);  
end.
```

параметры переменные

параметры значения

m1 – большее из a и b

m – большее из m1 и c

x, y, z – x, y, z – локальные  
переменные.

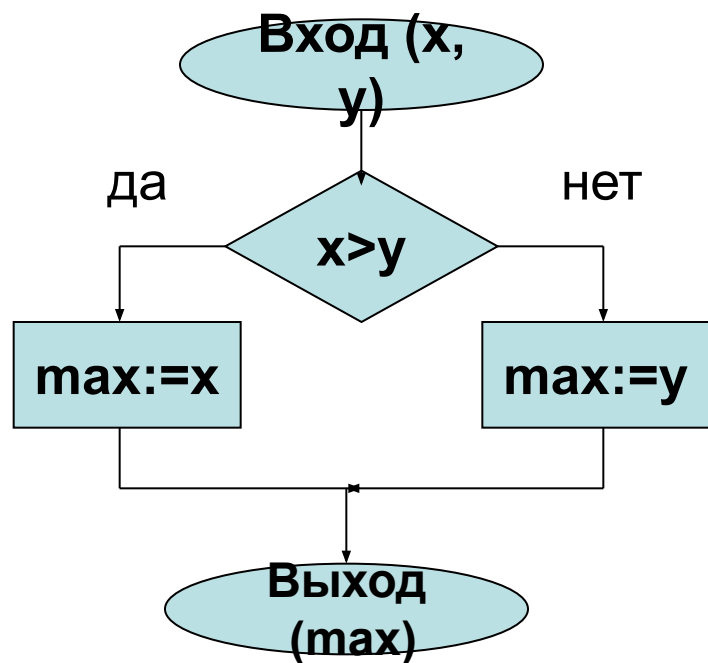
a, b, c, m, m1 a, b, c, m, m1 –  
глобальные переменные

Найти большее из трех данных чисел, используя подпрограмму нахождения большего из двух.

## Пример 2

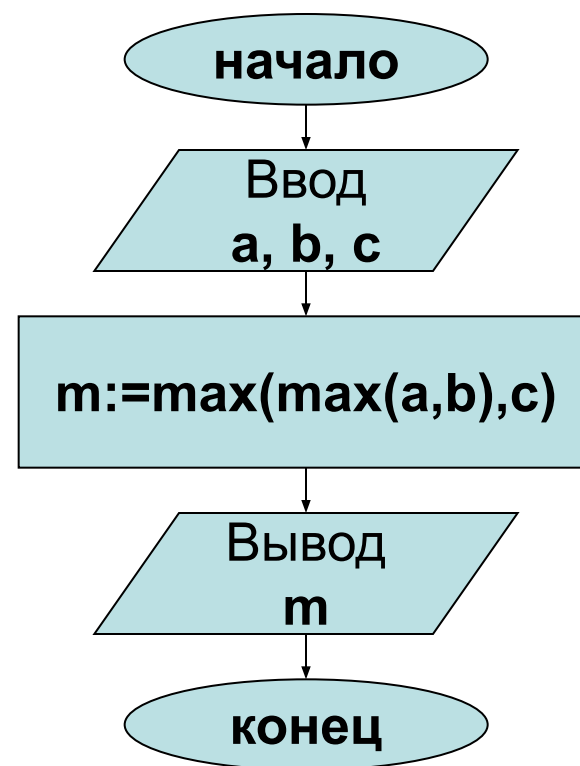
### II. Используем функцию

Вспомогательный алгоритм



$x, y, z$  - Формальные параметры

Основной алгоритм



$a, b, c$  - Фактические параметры

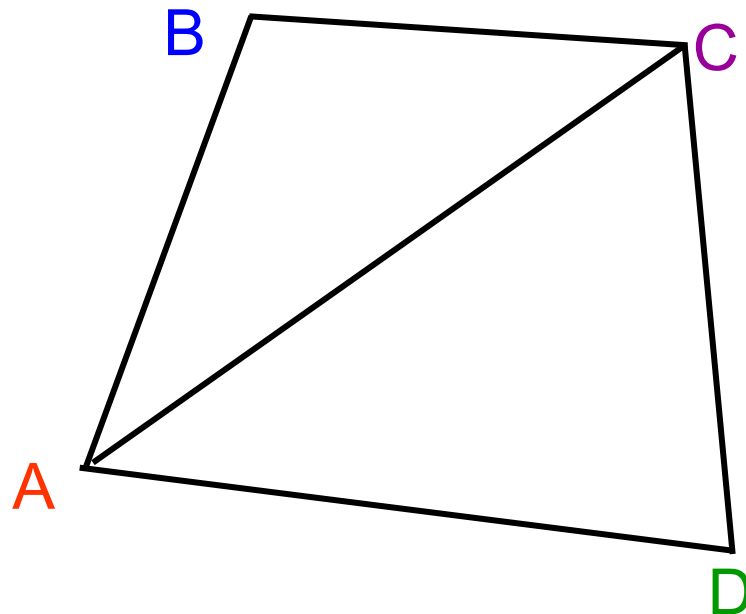
## II. Используем функцию

## Пример 2

```
program pr2;  
  var  a, b, c, m, m1: real;  
  function max(x, y: real): real;  
    begin  
      if x>y then max:=x else max:=y  
    end;  
  begin  
    writeln('a='); readln (a);  
    writeln('b='); readln (b);  
    writeln('c='); readln (c);  
    m:=max(max(a, b), c);  
    writeln ('max= ', m);  
  end.
```

# Пример 3

Составить программу для вычисления площади выпуклого 4-угольника, заданного длинами его сторон и диагональю.



Диагональ делит 4-угольник на два 3-угольника, к которым применима формула Герона:

$$s = \sqrt{p(p-a)(p-b)(p-c)},$$

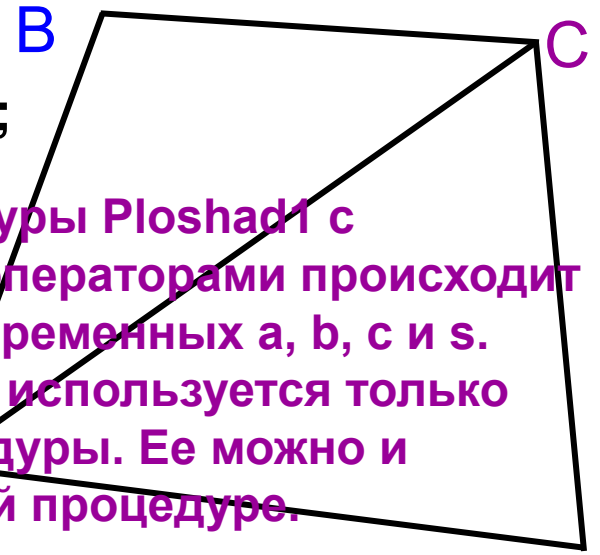
$a, b, c$  – длины сторон  $\Delta$ ,

$$p = \frac{a+b+c}{2}$$

```

program Prog1;
uses CRT;
var AB, BC, CD, DA, AC, S1, S2, S, a, b, c, p: real;
Procedure Ploshad1;
begin
  p:=(a+b+c)/2;
  s:=sqrt(p*(p-a)*(p-b)*(p-c));
end;
begin
Clrscr;

```



Связь процедуры Ploshad1 с остальными операторами происходит с помощью переменных a, b, c и s. Переменная p используется только внутри процедуры. Ее можно и описать в этой процедуре.

```

Writeln('Задайте стороны 4-х угольника ABCD и его диагональ AC');
readln (AB, BC, CD, DA, AC);

```

```

a:=AB; b:=BC; c:=AC;

```

```

Ploshad1;

```

```

S1:=s;

```

```

a:=DA; b:=AC; c:=CD;

```

```

Ploshad1;

```

```

S2:=s;

```

```

Writeln ('Площадь ABCD= ', S1+S2:8:2);

```

```

readln;

```

```

end.

```

2 обращения к процедуре

Команды присваивания, задающие значения a, b, c перед каждым вызовом процедуры

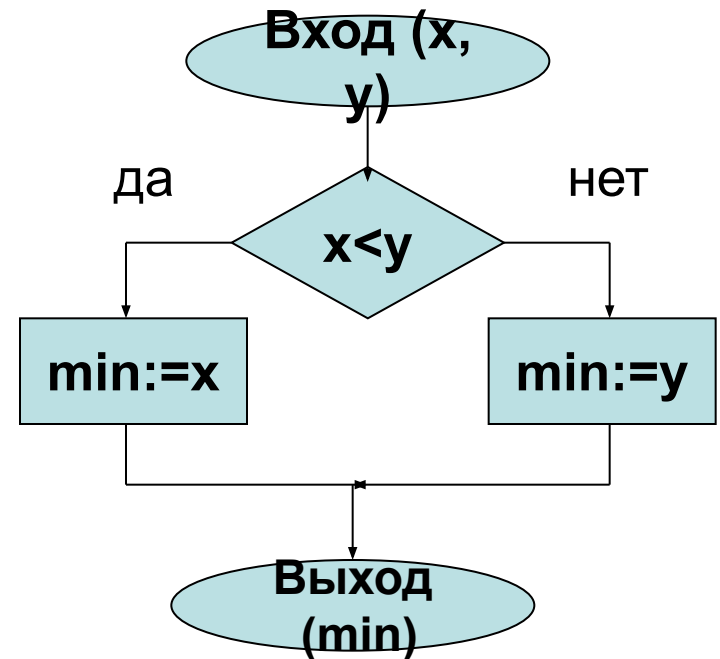
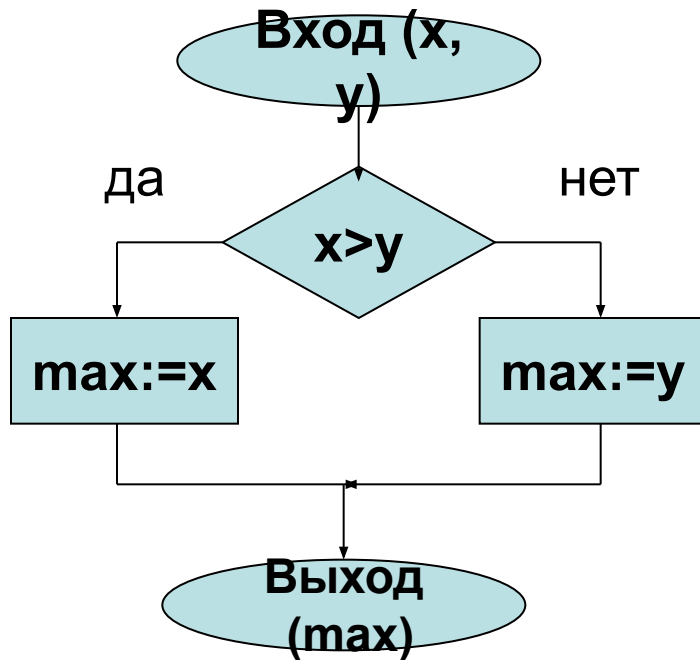
Команды присваивания для сохранения результатов

$$m := \frac{2 \max(a, b) + \min(a + 3, b)}{\min(c, a - b)}$$

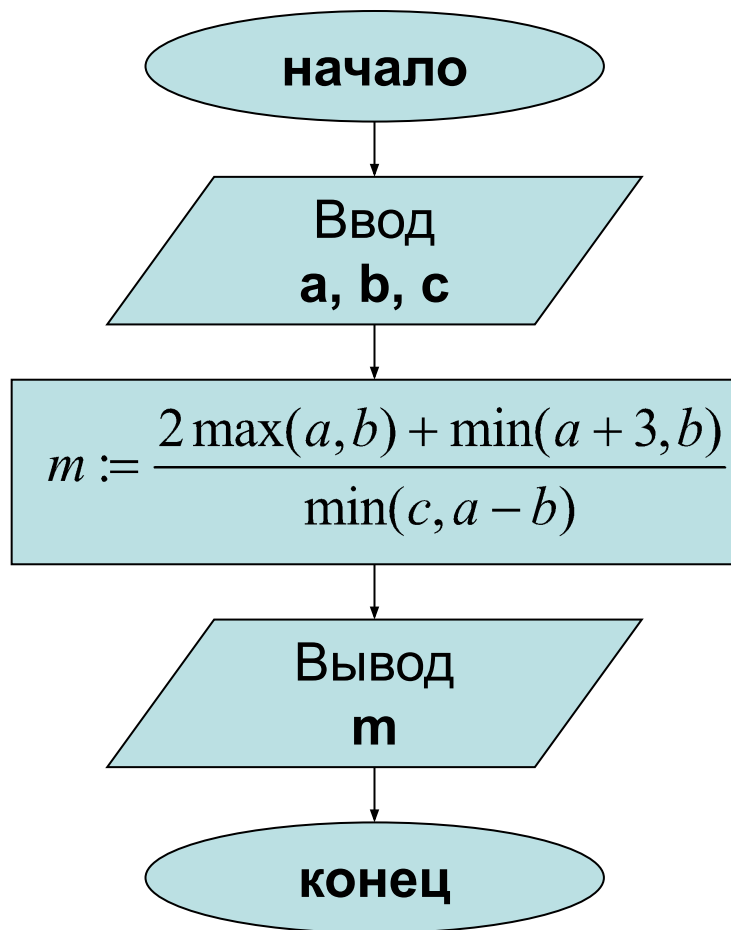
Найти  $m$ , используя подпрограммы нахождения большего и меньшего из двух чисел.

Вспомогательные алгоритмы

Пример 4



## Основной алгоритм



```
program pr2;
```

```
var a, b, c, m: real;
```

```
function max(x, y: real): real;
```

```
begin
```

```
    if x>y then max:=x else max:=y
```

```
end;
```

```
function min(x, y: real): real;
```

```
begin
```

```
    if x<y then min:=x else min:=y
```

```
end;
```

```
begin
```

```
    writeln('a='); readln (a);
```

```
    writeln('b='); readln (b);
```

```
    writeln('c='); readln (c);
```

```
    m:=(2*max(a, b)+min(a+3,b))/min(c,a-b);
```

```
    writeln ('max= ', m);
```

```
end.
```



# Объявление переменных

- **Глобальные переменные** - переменные, объявленные в основной программе, доступны всем операторам программы, а также операторам процедур и функций.
- **Локальные переменные** - переменные, объявленные в процедуре или функции. Они доступны только операторам процедур или функций.

# Структура функции

**Function** <имя> (<параметры>):<тип результата>;

**const** ...;

.....

**var** ... ;

Блок описания локальных переменных

**Begin**

<операторы>

**имя:= выражение;**

**End;**

В разделе операторов должен находиться, хотя бы один оператор, присваивающий имени функции значение.

# Пример:

```
program primer1;
```

```
var
```

```
    r, c, q : real;    ← Глобальные переменные
```

```
function inper (  a: real; b: real) :real;
```

```
var
```

```
    x,y: real;    ← Локальные переменные
```

```
    begin
```

```
    <операторы функции>;
```

```
    end;
```

```
begin
```

```
(основная программа)
```

```
end.
```

Спасибо за  
внимание!

