L#6

Basics of Programming.
Procedures and functions

Course Basics of Programming Semester 1, FIIT

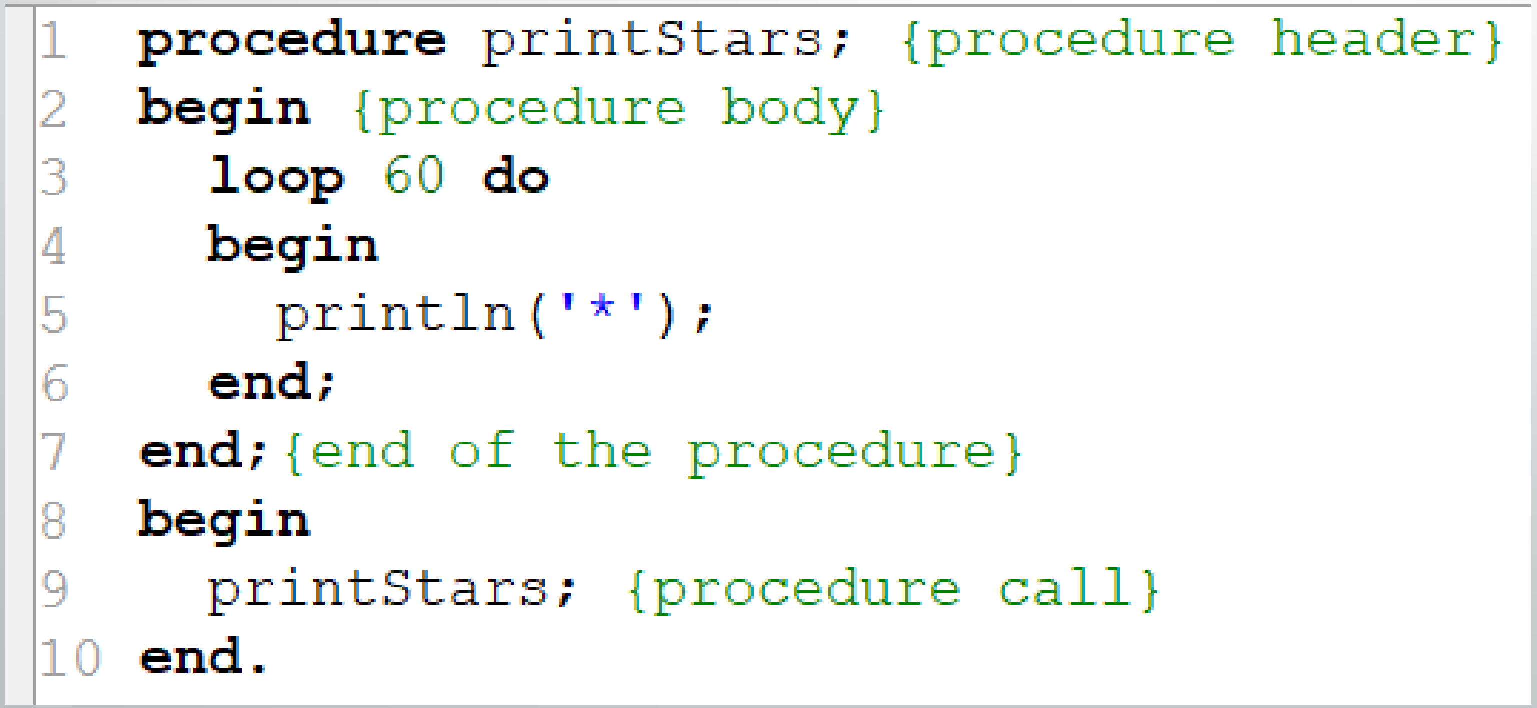
Mayer Svetlana Fyodorovna

Procedures: definitions

- Procedure is a sequence of program instructions that performs a specific task.
- Procedure is a subroutine of a main program.
- Procedure consists of header (name and parameters) and body. Header can be named a signature.
- Procedure must be defined once and can be called many times.

Parameterless procedure. Example

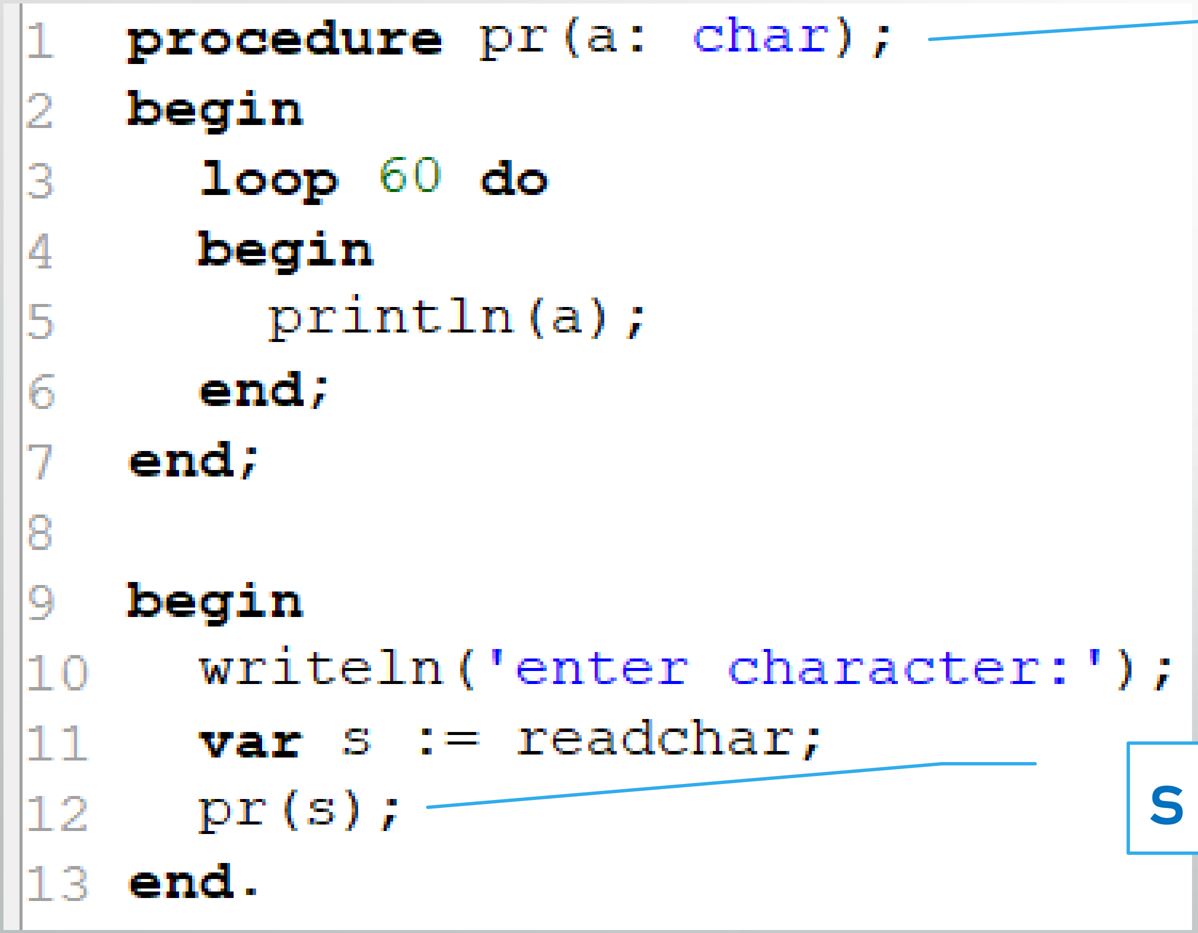
- Problem. Print 60 asterisks (star *), each on a new line. Use parameterless procedure
- Rule. Subroutine must be defined before the begin keyword of a main program



```
procedure printStars; {procedure header}
begin {procedure body}
  loop 60 do
  begin
    println('*');
  end;
end;{end of the procedure}
begin
  printStars; {procedure call}
end.
```

Procedure with parameters. Example

- Problem. Let's solve almost the same task , but we're going to print 60 entered character, each on a new line.



```
procedure pr(a: char);
begin
  loop 60 do
  begin
    println(a);
  end;
end;

begin
  writeln('enter character:');
  var s := readchar;
  pr(s);
end.
```

a - formal parameter

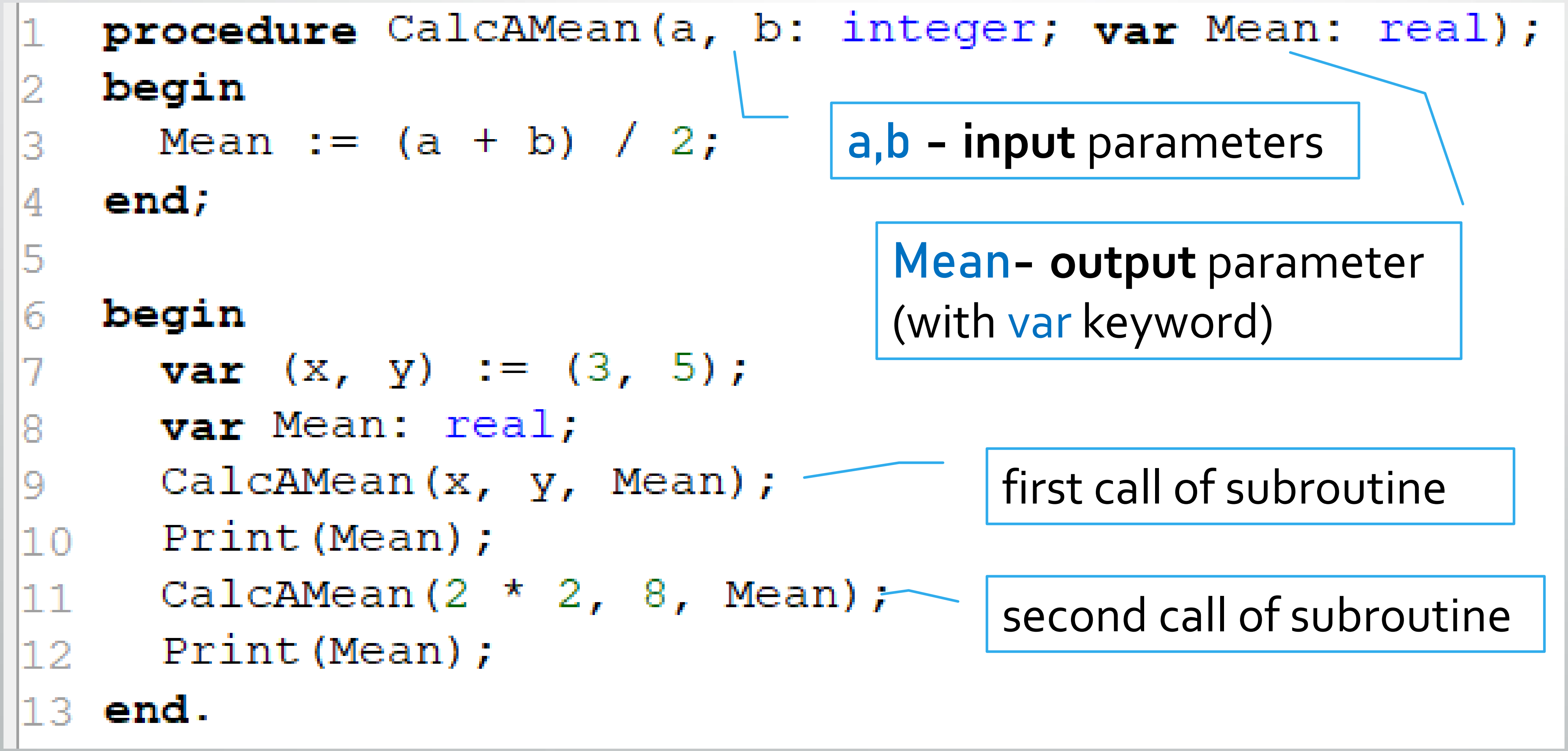
s - **actual** parameter or argument

Tasks

- To do: Lesson # 10, Tasks procedures 1, 2, 3, 4

Input and output parameters. Example

- Problem. Let's write a subroutine for calculating the arithmetic mean (average) of two entered integer values.

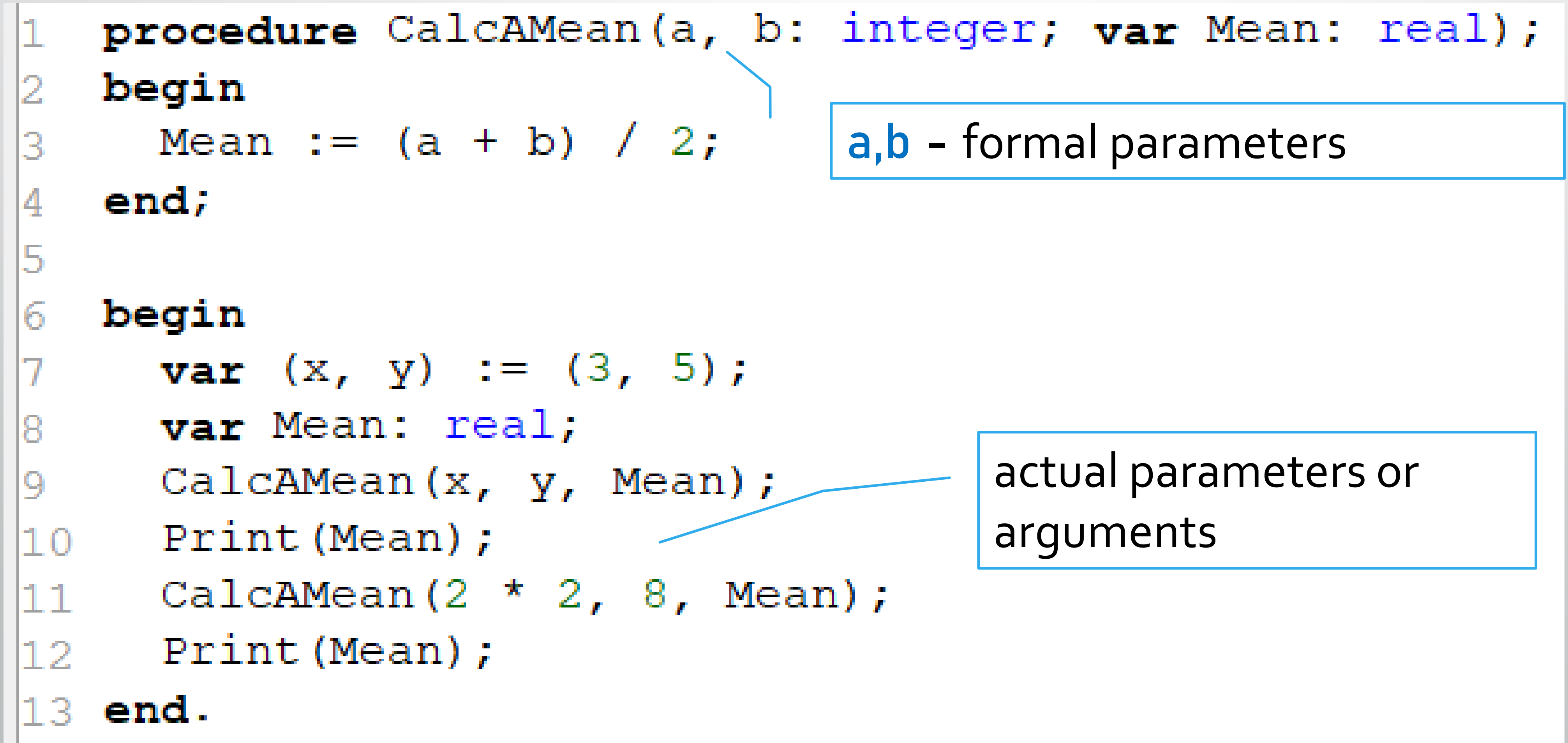


```
procedure CalcAMean(a, b: integer; var Mean: real);
begin
  Mean := (a + b) / 2;
end;

begin
  var (x, y) := (3, 5);
  var Mean: real;
  CalcAMean(x, y, Mean);
  Print(Mean);
  CalcAMean(2 * 2, 8, Mean);
  Print(Mean);
end.
```

a,b - input parameters
Mean- **output** parameter (with var keyword)
first call of subroutine
second call of subroutine

Formal & actual parameters. Example

- Problem. Let's write a subroutine for calculating the arithmetic mean of two entered integer values.
- An actual parameter can be constant or expression.



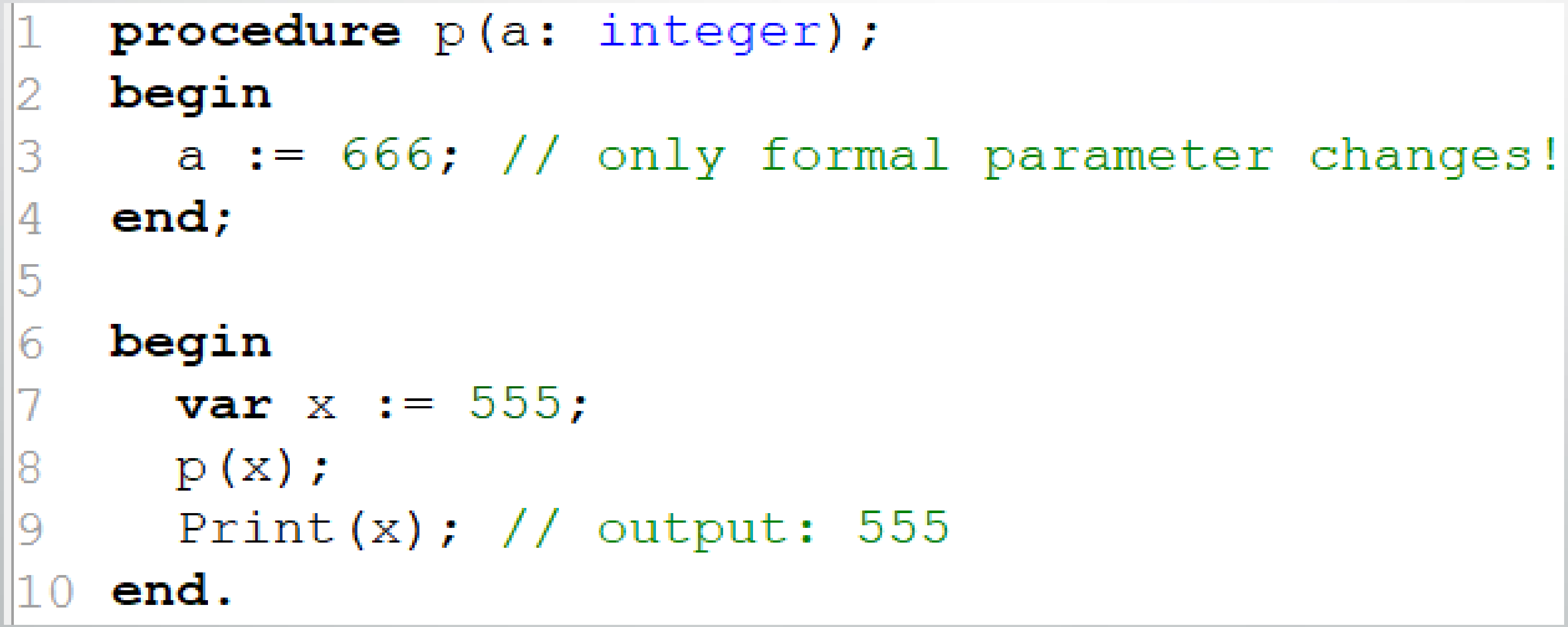
```
procedure CalcAMean(a, b: integer; var Mean: real);
begin
  Mean := (a + b) / 2;
end;

begin
  var (x, y) := (3, 5);
  var Mean: real;
  CalcAMean(x, y, Mean);
  Print(Mean);
  CalcAMean(2 * 2, 8, Mean);
  Print(Mean);
end.
```

a,b - formal parameters

actual parameters or arguments

Passing Arguments by Value and by Reference

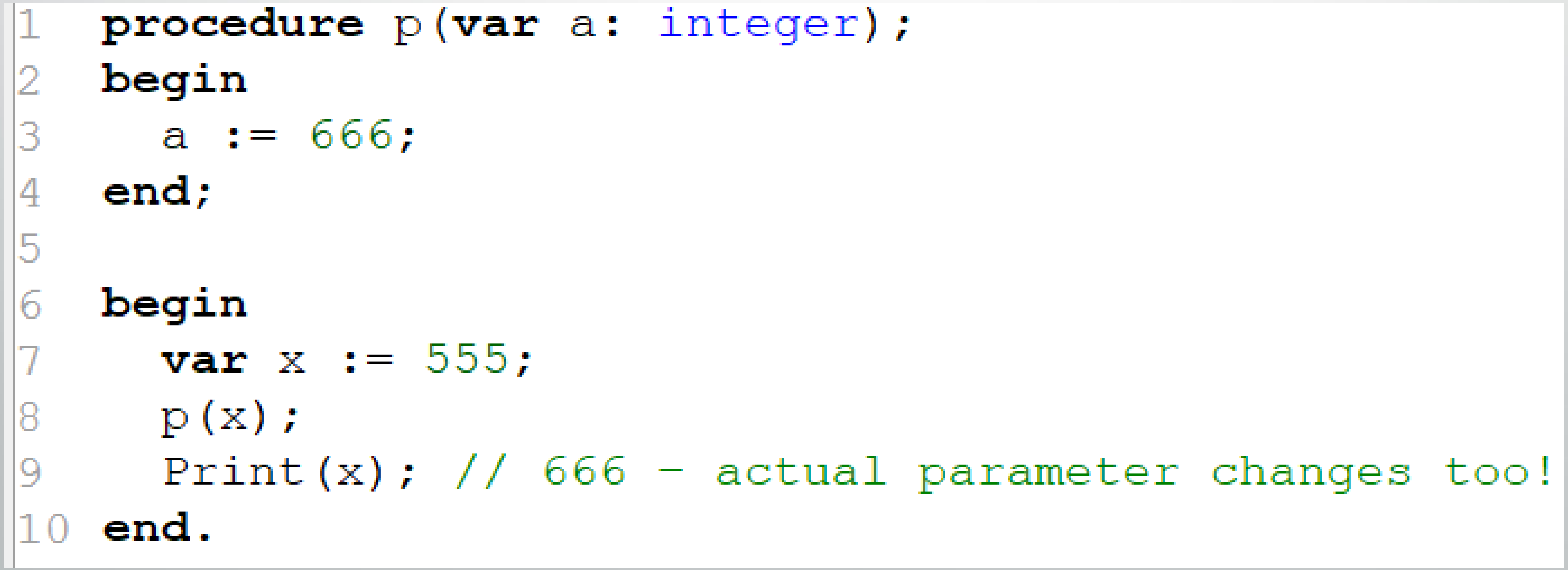
- In PascalABC.NET, you can pass an argument to a procedure by value or by reference.
- Passing by value: A value of an actual parameter is copied into corresponding formal parameter. Changing the formal parameter doesn't change an actual parameter



```
procedure p(a: integer);
begin
  a := 666; // only formal parameter changes!
end;

begin
  var x := 555;
  p(x);
  Print(x); // output: 555
end.
```

Passing Arguments by Value and by Reference

- In PascalABC.NET, you can pass an argument to a procedure by value or by reference.
- Passing by reference: Both the actual and formal parameters refer to the same location, so any changes made inside the procedure body are reflected in the actual parameters of procedure call.
- Changing the formal parameter changes the actual parameter too:



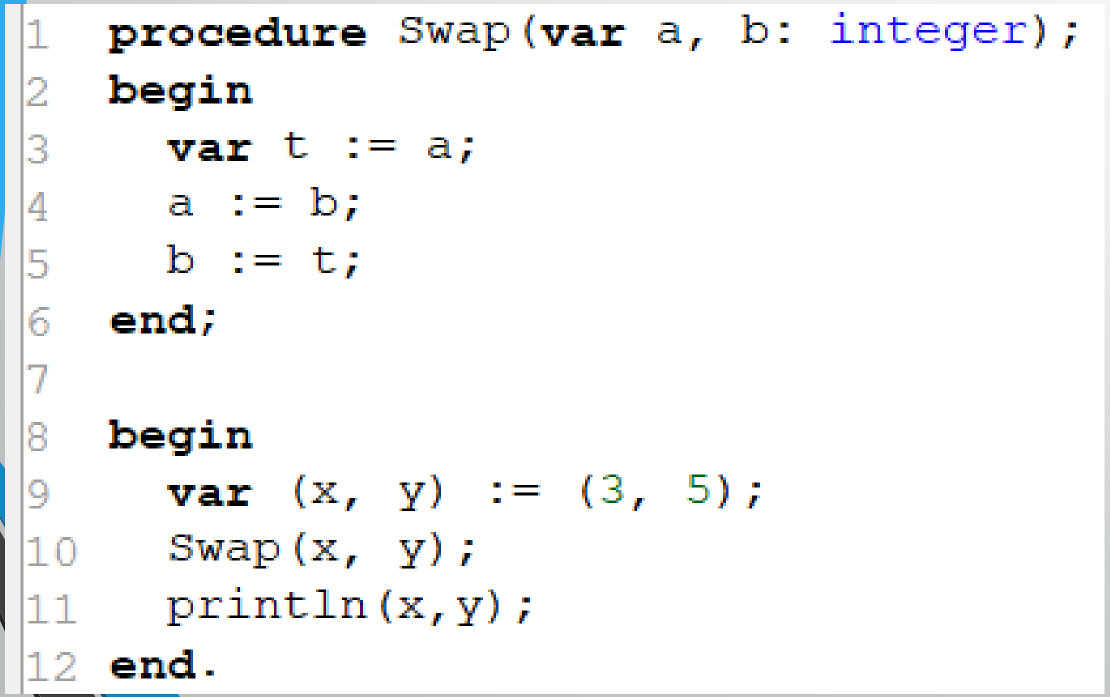
```pascal
procedure p(var a: integer);
begin
  a := 666;
end;

begin
  var x := 555;
  p(x);
  Print(x); // 666 - actual parameter changes too!
end.
```

Passing Arguments by Value and by Reference

- **Problem**. Create a procedure to swap the values of two variables:

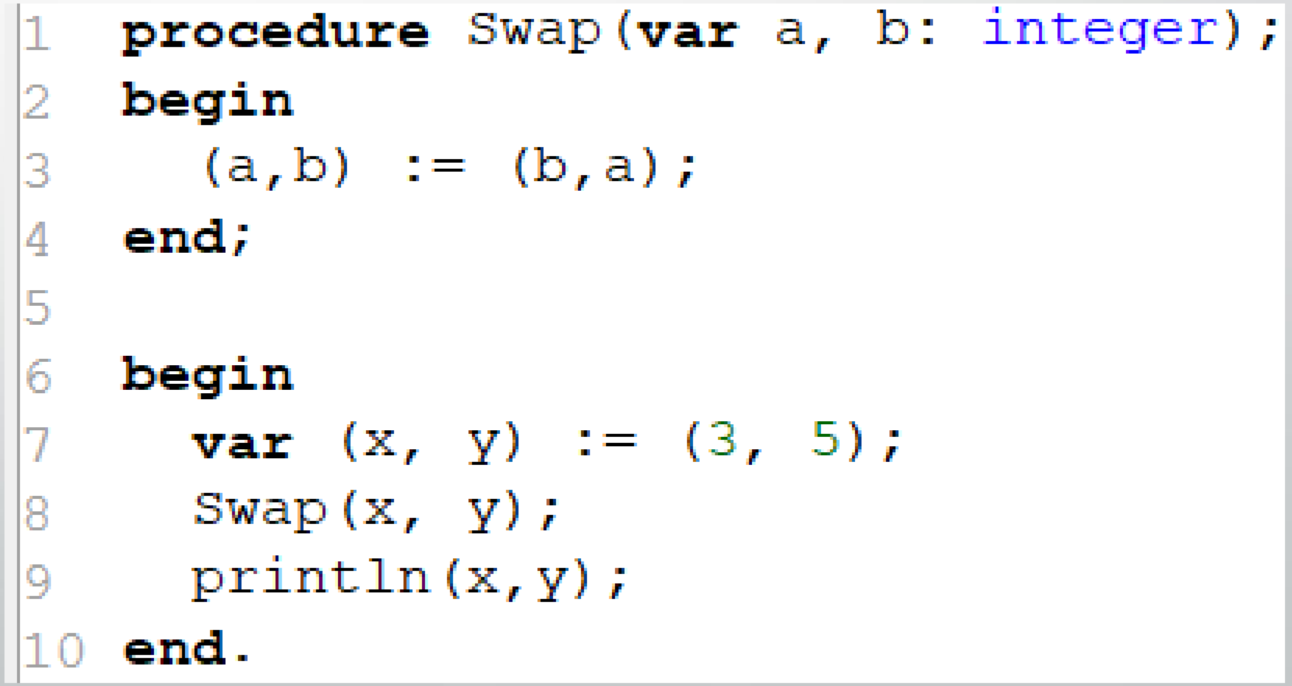
a = 10
b = 12
Result: a=12, b=10

Solution 1:



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

begin
  var (x, y) := (3, 5);
  Swap(x, y);
  println(x,y);
end.
```

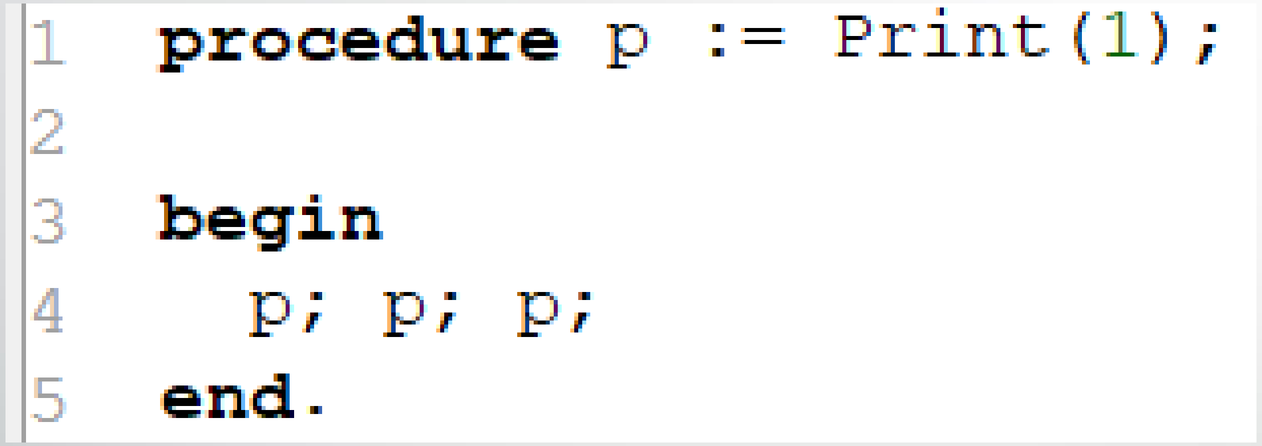
Solution 2:



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

begin
  var (x, y) := (3, 5);
  Swap(x, y);
  println(x,y);
end.
```

Short procedure definition

- If the body of the procedure consists of only one statement, we can use a short procedure definition



```
procedure p := Print(1);

begin
  p; p; p;
end.
```

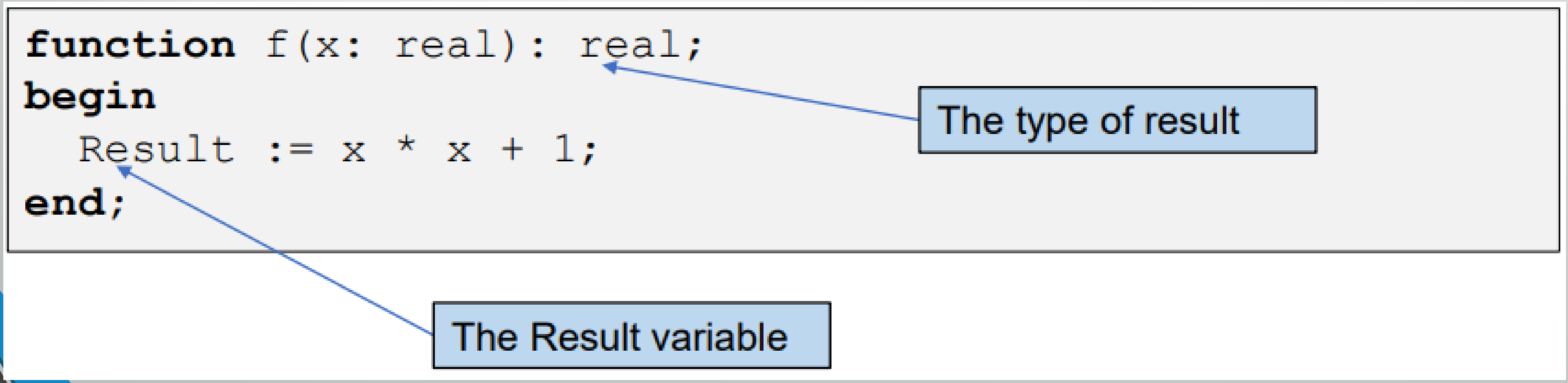
Tasks

- To do: Lesson # 10, Tasks procedures 5, 6, 7, 8, 9 Extra task 10

Functions

Functions

- A function is a kind of procedure that returns a value for use in an expression.
- The function definition differs from the procedure definition in two points:
  1. We have to write the Type of the value that function returns.
  2. The Result variable stores the result of the function



```
function f(x: real): real;
begin
  Result := x * x + 1;
end;
```

Functions

```
function f(x: real): real;
begin
  Result := x * x + 1;
end;
begin
  var x := 5;
  var r := Sqrt(x) + f(x);
end.
```

We use function call in an expression

Different ways to define a function

- declaration of formal parameters (those which values are passed from the main program to the function):

```
function ff( a, b: integer; x: real ): real;
```

- output parameters whose values become available in the main program (returned to the program)

```
function Max ( var a, b: integer): integer;
```

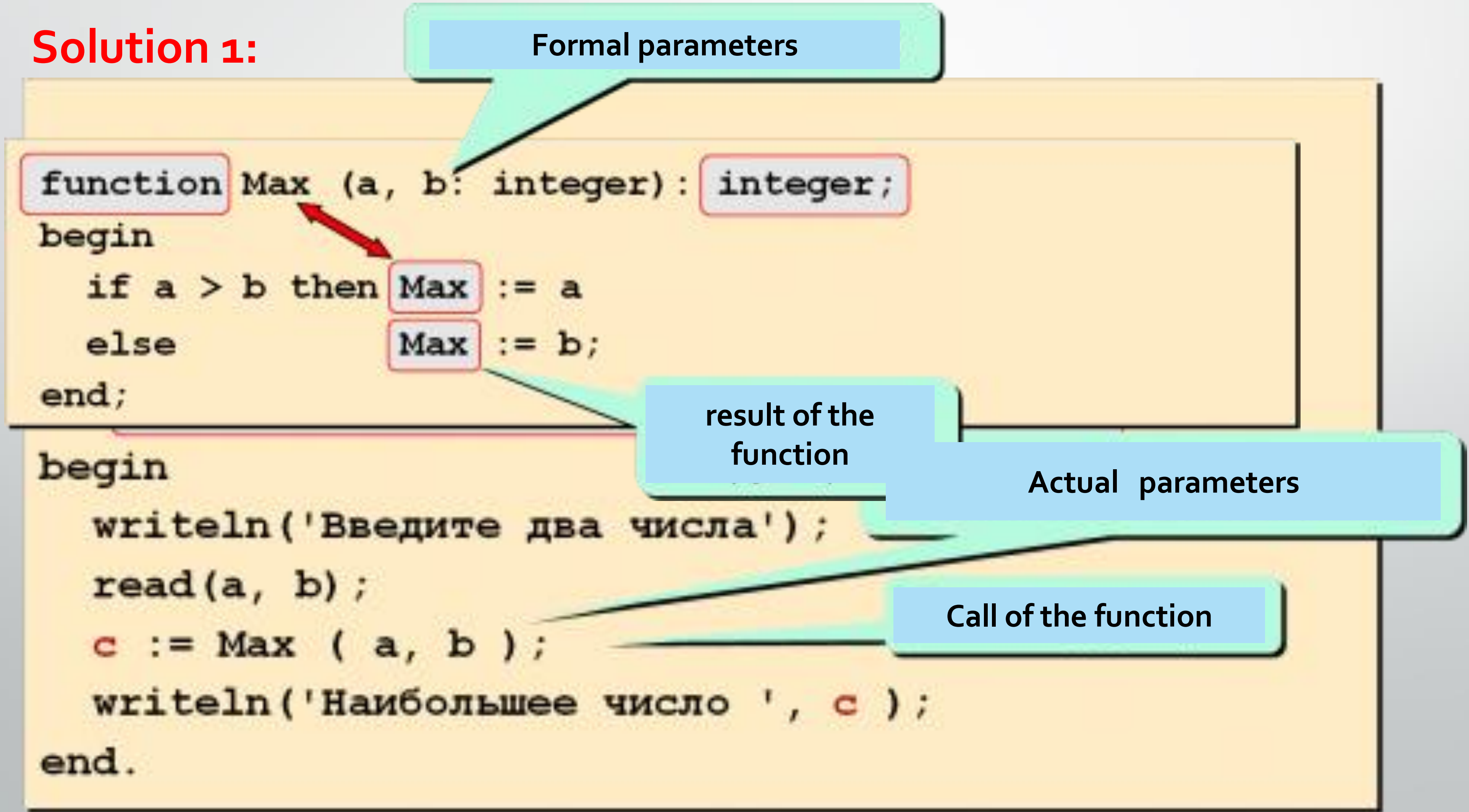
- The type of the function's return value is appended at the end of the function header, separated by a colon:

```
function Max (a, b: integer): integer;
```

Find the maximum among two numbers using the function

- To return a result to the main program the Result variable can be used, or the name of the function:

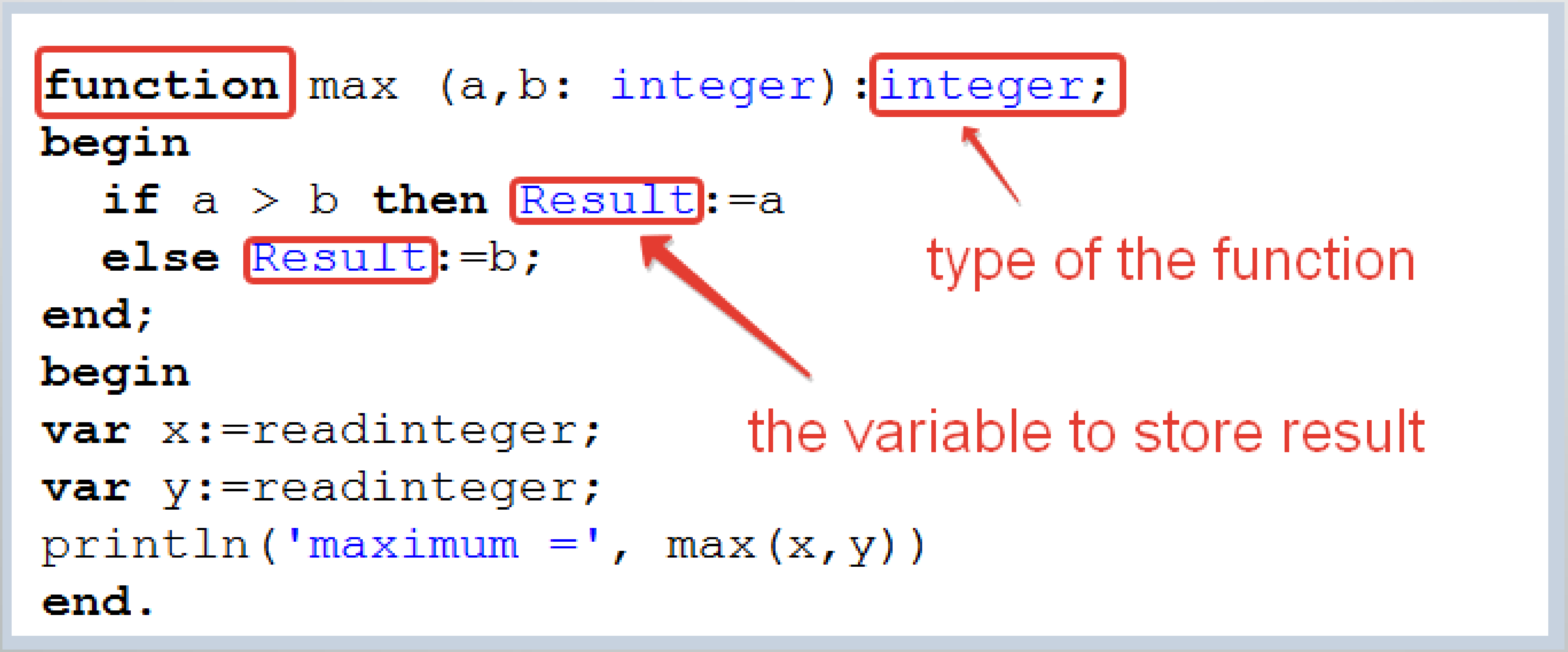
Solution 1:



```
function Max (a, b: integer): integer;
begin
  if a > b then Max := a
  else          Max := b;
end;
begin
  writeln('Введите два числа');
  read(a, b);
  c := Max ( a, b );
  writeln('Наибольшее число ', c );
end.
```

Find the maximum among two numbers using the function

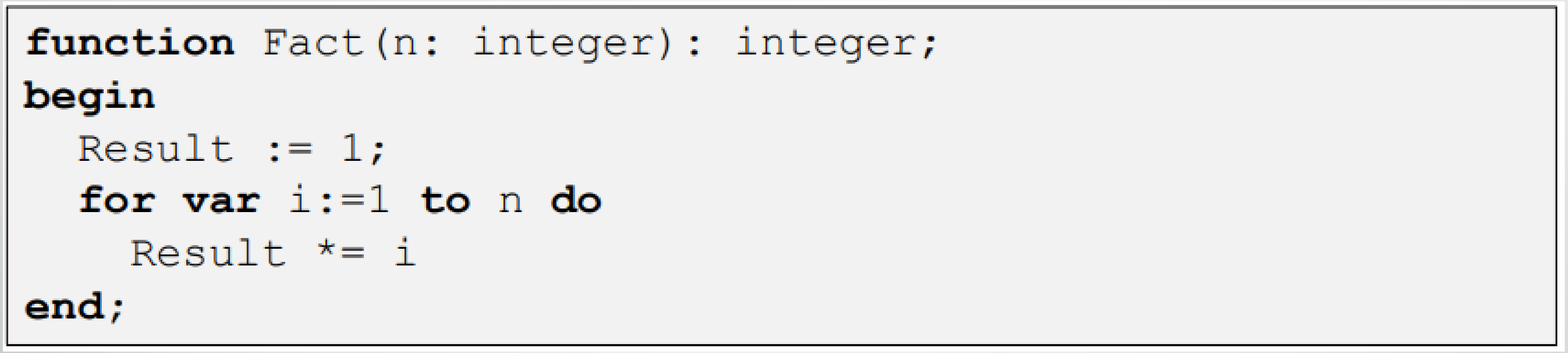
Solution 2:



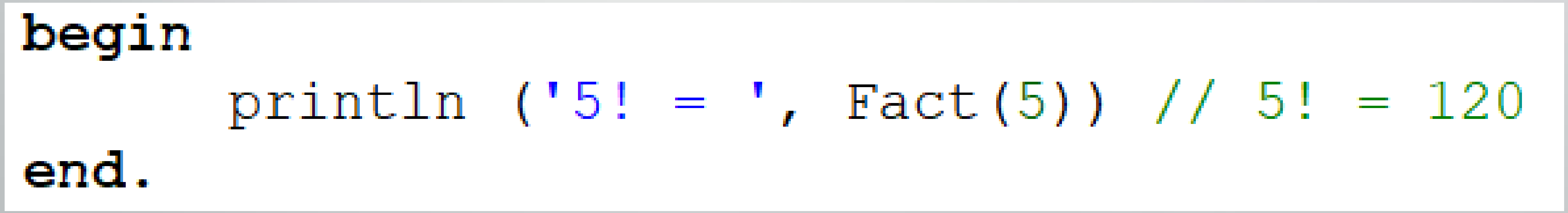
```
function max (a,b: integer):integer;
begin
   if a > b then Result:=a
   else Result:=b;
end;
begin
var x:=readinteger;
var y:=readinteger;
println('maximum =', max(x,y))
end.
```

Examples

- Body of function can have loops:



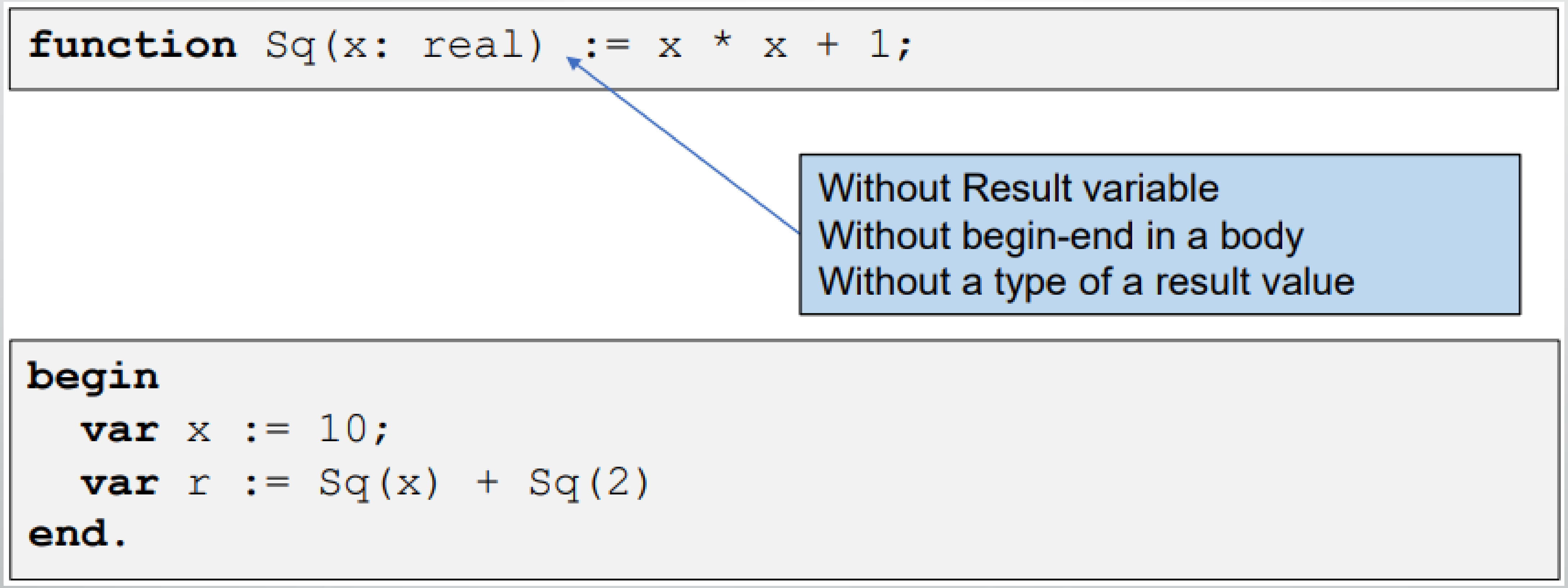
```
function Fact(n: integer): integer;
begin
  Result := 1;
  for var i:=1 to n do
    Result *= i
end;
```



```
begin
       println ('5! = ', Fact(5)) // 5! = 120
end.
```

Short function definition

- If result of the function is the only expression, we can use the short function definition



```
function Sq(x: real) := x * x + 1;
```

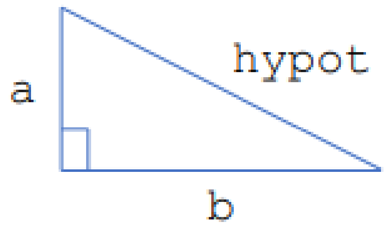
Without Result variable
Without begin-end in a body
Without a type of a result value

```
begin
  var x := 10;
  var r := Sq(x) + Sq(2)
end.
```

Examples

- The hypothenuse function.

```
function Hypot(a,b: real) := Sqrt(a*a + b*b);
```

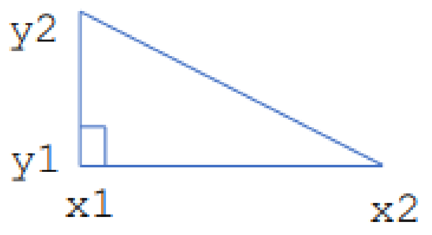


```
begin
  loop 5 do
  begin
    var (a,b) := ReadlnReal('Enter a,b:');
    Println(Hypot(a,b))
  end;
end.
```

Examples

- Function can be called inside another function:

```
function Hypot(a,b: real) := Sqrt(a*a + b*b);
```



```
function Hypot1(x1,y1,x2,y2: real) := Hypot(x2-x1,y2-y1);
```

```
begin
  loop 5 do
  begin
    var (x1,y1,x2,y2) := ReadlnReal4;
    Println(Hypot1(x1,y1,x2,y2));
  end;
end.
```

Tasks

- To do: Lesson # 10, Tasks functions 1, 2, 3, 4, 5, 6, 7

Q & A