

```
:whileNotValidDpiNumber
set /p dpi=With what dpi should it be exported (e.g. 300)?
if "%dpi%" EQU "q" exit /b
if %dpi% NEQ %dpi% (
echo Invalid input! Please input an actual number.
goto :whileNotValidDpiNumber
)
echo.

:: count how many files we need to convert before converting!
set /a total=0
for /R %i in (*.sourceType) do (set /a total+=1)
echo Found %total% file(s) of type *.sourceType in the current folder (%dp0)

echo.

set /a count=0
:: Running through all files found with the defined ending
if %inkscapeMajorVersion% NEQ 0 (
:: Inkscape 1.0 and newer
for /R %i in (*.sourceType) do (
set /a count+=1
:: Create out folder if it does not exist
if not exist "%~di%-piout" mkdir "%~di%-piout"
echo %i -^> "%~di%-piout\%~ni%.outputType% [!count!/%total%]
%inkscapePath% --export-filename="%~di%-piout\%~ni%.outputType%" --export-dpi=%dpi% %i"
)
) else (
```

```
<meta name="google-site-verification"
content="c3kub-K2Q2F6151pcwPw4ilad-JA_dHynw"
<meta name="google-site-verification"
content="KT5gshbwaagLkxwQbHbWwC2Kz3q5Cwz"
<meta name="google-site-verification"
content="ZbhytFw4w4w-d0TmdJw2PSSvrendw4w"
<meta name="google-site-verification"
content="5x5K5bWwC2Kz3q5Cwz"
<meta name="google-site-verification"
content="4p1b7-x88wJ5chpWwC2Kz3q5Cwz"
```



Funções

Também chamados de **subalgoritmos**.

É o segundo tipo de **rotinas**.

São trechos de algoritmos que executam um ou mais cálculos determinados.



Funções

Ao invés de escrever um código grande, pode-se escrever vários algoritmos menores, e assim como os **procedimentos**, chamar a função para ser executada (**Modularização**).

Mas qual a diferença entre função e procedimento?



Funções

Funções **podem retornar algum valor** do cálculo para o algoritmo principal.

Procedimentos não retornam nada, só são executados (por isso também são chamados de **subrotinas**).



Funções

Podem ser **predefinidas** pela linguagem de programação usada **ou criadas** pelo programador.

Raiz Quadrada

Criar um algoritmo que calcule o valor da raiz quadrada de um número.



Raizq(valor : real) : real

Nome
da função

Parâmetro :
Tipo do parâmetro

Tipo
de retorno


```
algoritmo "Uso_de_Funcao_Predefinida"  
var  
    numero, raiz: real  
inicio  
    escreva("Digite um número: ")  
    leia(numero)  
    raiz <- Raizq(numero)  
    escreva("A raiz quadrada do número digitado é: ", raiz)  
fimalgoritmo
```


Exemplo Prático (Pré-definidas)

```
algoritmo "Uso_de_Funcao_Predefinida"  
var  
    numero, raiz: real  
inicio  
    escreva("Digite um número: ")  
    leia(numero)  
    raiz <- Raizq(numero)  
    escreva("A raiz quadrada do número digitado é: ", raiz)  
fimalgoritmo
```

Chamada da
função passando
a variável
numero como
parâmetro da
função **Raizq**

Exemplo Prático (Pré-definidas)

```
algoritmo "Uso_de_Funcao_Predefinida"  
var  
    numero, raiz: real  
inicio  
    escreva("Digite um número: ")  
    leia(numero)  
    raiz <- Raizq(numero)  
    escreva("A raiz quadrada do número digitado é: ", raiz)  
fimalgoritmo
```

Atribuindo à variável **raiz** o retorno da função **Raizq**

Compr(c : caractere)



Nome
da função



Parâmetro :
Tipo do parâmetro

OBS: o retorno será do tipo inteiro.

Contador de Letras

Criar um algoritmo que, digitando uma palavra qualquer retorne a quantidade de letras.



algoritmo "Uso_de_Funcao_Predefinida"

var

palavra: caractere

qtd: inteiro

inicio

escreva("Digite uma palavra qualquer: ")

leia(palavra)

qtd <- Compr(palavra)

escreva("A palavra que você digitou possui ", qtd, " caracteres")

fimalgoritmo

Rand

Retorna um número real gerado aleatoriamente, maior ou igual a zero e menor que um (zero a um).

Randi (Limite)

Retorna um número inteiro gerado aleatoriamente, maior ou igual a zero e menor que o limite.

Maiusc(c:caractere)

Retorna um valor caractere contendo a expressão em maiúscula.

Minusc(c:caractere)

Retorna um valor caractere contendo a expressão em minúscula.

Copia(c:caractere,posini,posfin:inteiro):caractere

Copia um determinado trecho de uma palavra.

Pos("letra/caractere", c:caractere):inteiro

Retorna a posição de uma letra (caractere).



Criando Funções

A criação das funções devem ser sempre realizadas **dentro da seção de variáveis (var)**

Este tipo de subalgoritmo sempre retorna **apenas um valor** para o algoritmo que o chamou.

```
algoritmo <nome do algoritmo>
var
    <declaração das variáveis globais>
    <definição das funções>
inicio
    <lista de comandos>
finalgoritmo
```

```
funcao <nome do funcao> (<parâmetros>) <tipo de retorno>
```

Dobro de um número

Criar uma função para calcular o dobro de um número passado por parâmetro.



```
algoritmo "FUNÇÃO DOBRO"  
var  
  
    numero, resultado: real  
    total: real  
funcao Dobro(valor:real): real  
  
inicio  
    total <- valor*2  
    retorne total  
fimfuncao  
  
inicio  
  
    Escreva("Digite um número para calcular o dobro: ")  
    Leia(numero)  
    resultado <- Dobro(numero)  
  
    Escreva("O dobro é: ", resultado)  
  
fimalgoritmo
```

Exemplo Prático

```
algoritmo "FUNÇÃO DOBRO"
```

```
var
```

```
    numero, resultado: real
```

```
    total: real
```

```
funcao Dobro(valor:real): real
```

```
inicio
```

```
    total <- valor*2
```

```
    retorne total
```

```
fimfuncao
```

```
inicio
```

```
    Escreva("Digite um número para calcular o dobro: ")
```

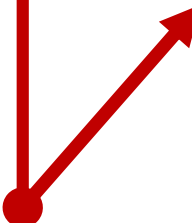
```
    Leia(numero)
```

```
    resultado <- Dobro(numero)
```

```
    Escreva("O dobro é: ", resultado)
```

```
fimalgoritmo
```


Declaração da
função



Exemplo Prático

```
algoritmo "FUNÇÃO DOBRO"  
var  
  
    numero, resultado: real  
    total: real  
funcao Dobro(valor:real): real  
  
inicio  
    total <- valor*2  
    retorne total  
fimfuncao
```

Algoritmo
principal,
comandos



```
inicio  
  
    Escreva("Digite um número para calcular o dobro: ")  
    Leia(numero)  
    resultado <- Dobro(numero)  
  
    Escreva("O dobro é: ", resultado)  
  
fimalgoritmo
```

```
algoritmo "FUNÇÃO DOBRO"
```

```
var
```

```
    numero, resultado: real
```

```
    total: real
```

```
funcao Dobro(valor:real): real
```

```
inicio
```

```
    total <- valor*2
```

```
    retorne total
```

```
fimfuncao
```

```
inicio
```

```
    Escreva("Digite um número para calcular o dobro: ")
```

```
    Leia(numero)
```

```
    resultado <- Dobro(numero)
```

```
    Escreva("O dobro é: ", resultado)
```

```
fimalgoritmo
```

Nome da função



```
algoritmo "FUNÇÃO DOBRO"
```

```
var
```

```
    numero, resultado: real
```

```
    total: real
```

```
funcao Dobro(valor:real): real
```

```
inicio
```

```
    total <- valor*2
```

```
    retorne total
```

```
fimfuncao
```

```
inicio
```

```
    Escreva("Digite um número para calcular o dobro: ")
```

```
    Leia(numero)
```

```
    resultado <- Dobro(numero)
```

```
    Escreva("O dobro é: ", resultado)
```

```
fimalgoritmo
```

Parâmetro: tipo
do parâmetro

```
algoritmo "FUNÇÃO DOBRO"
```

```
var
```

```
    numero, resultado: real
```

```
    total: real
```

```
funcao Dobro(valor:real): real
```

Tipo de retorno



```
inicio
```

```
    total <- valor*2
```

```
    retorne total
```

```
fimfuncao
```

```
inicio
```

```
    Escreva("Digite um número para calcular o dobro: ")
```

```
    Leia(numero)
```

```
    resultado <- Dobro(numero)
```

```
    Escreva("O dobro é: ", resultado)
```

```
fimalgoritmo
```

```
algoritmo "FUNÇÃO DOBRO"  
var  
  
    numero, resultado: real  
    total: real  
funcao Dobro(valor:real): real
```

```
inicio
```

```
    total <- valor*2  
    retorne total
```

```
fimfuncao
```

```
inicio
```

```
    Escreva("Digite um número para calcular o dobro: ")  
    Leia(numero)  
    resultado <- Dobro(numero)
```

```
    Escreva("O dobro é: ", resultado)
```

```
fimalgoritmo
```



Comandos

algoritmo "FUNÇÃO DOBRO"

var

numero, resultado: real

total: real

funcao Dobro(valor:real): real

inicio

total <- valor*2

retorne total

fimfuncao

inicio

Escreva("Digite um número para calcular o dobro: ")

Leia(numero)

resultado <- Dobro(numero)

Escreva("O dobro é: ", resultado)

fimalgoritmo

 Retorno

Dobro e Triplo de um número

Criar duas funções para calcular o dobro e o triplo de um número passado por parâmetro.

(um algoritmo pode ter mais de uma função)




```
algoritmo "FUNÇÃO DOBRO E TRIPLO"
var

numero: real
total: real

    funcao Dobro(valor:real): real
    inicio
    total <- valor*2
    retorne total
    fimfuncao

    funcao Triplo(valor:real): real
    inicio
    total <- valor*3
    retorne total
    fimfuncao

inicio |

Escreva("Digite um número: ")
Leia(numero)

EscrevaL("O dobro é: ", Dobro(numero))
EscrevaL("O triplo é: ", Triplo(numero))

fimalgoritmo
```



Criando Funções

As funções podem ter mais de um parâmetro.

- ✓ **Parâmetros de um mesmo tipo** são separados por **vírgula**
- ✓ **Parâmetros de tipos diferentes** são separados por **ponto e vírgula**

Média

Criar uma função que receba três valores reais como parâmetro e retorne a média desses valores



```
algoritmo "FUNÇÃO SOMA VÁRIOS"
var
    N1, N2, N3, resultado: real
    calculoMedia: real
    |
    funcao Media(N1, N2, N3: real): real

        inicio
            calculoMedia <- (N1+N2+N3)/3
            retorne calculoMedia
        fimfuncao

inicio

    EscrevaL("Digite três números: ")
    Leia(N1)
    Leia(N2)
    Leia(N3)

    resultado <- Media(N1,N2,N3)

    EscrevaL("A média é: ", resultado)

fimalgoritmo
```