Unidade 6: Ambiente de Programação

Prof. Daniel Caetano

Objetivo: Apresentar o uso de funções prontas do Portugol e do Python para efeturar cálculos mais complexos.

Bibliografia: ASCENCIO, 2007; MEDINA, 2006; SILVA, 2010; SILVA, 2006.

INTRODUÇÃO

Nas aulas anteriores tomamos contato com a base das linguagens Portugol e Python. Nesta aula, veremos algumas funções matemáticas mais avançadas do Portugol e do Python, que propiciam cálculos mais complexos e, portanto, tornam o computador mais interessante para o engenheiro.

Ao final desta aula, você saberá como usar o resto da divisão para executar tarefas úteis, além de aprender a utilizar diversas funções matemáticas pré-programadas nas linguagens.

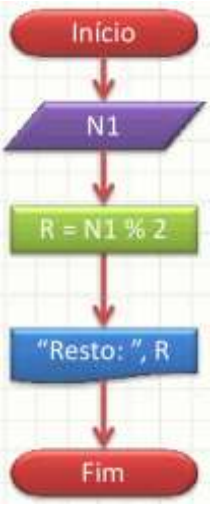
1. O RESTO DE DIVISÃO

Muitos alunos, em um primeiro momento, se questionam sobre a utilidade do resto de divisão (operador **%**). Parece um cálculo sem muita lógica, já que é pouco usado em nosso dia-a-dia, em que não fazemos cálculos de "divisão inteira".

Entretanto, na programação, a divisão inteira tem um papel muito importante, em especial na **determinação "par/ímpar"** e no **cálculo de conversão** entre bases ou escalas numéricas. Vejamos cada um destes casos.

1.1. DETERMINAÇÃO "PAR/ÍMPAR"

Determinar a paridade significa determinar se um valor é par ou ímpar. Matematicamente, pode-se dizer que **um número é par sempre que ele é divisível por 2**. Ora, ser divisível por 2 significa que o **resto da divisão por 2 é zero**! Assim, um algoritmo para determinar se um número é par ou ímpar pode ser descrito conforme o fluxograma a seguir. Se ele imprimir "0" significa que o número é par; se, por outro lado, ele imprimir "1", significa que o número é ímpar!



Em portugol, o algoritmo tem a seguinte forma:

```
ALGORITMO "Verifica se é par ou ímpar"
INICIO
real valor, resto

escreva("Digite um valor: ")
leia(valor)
resto <- valor % 2
escreva ("O resto é (0=par;1=ímpar): ", resto)
FIM
```

Em Python, este mesmo código tem a seguinte forma:

```
# Verifica se é par ou ímpar
valor = 0.0; resto = 0.0

valor = int( input("Digite um valor: ") )
resto = valor % 2
print ("O resto é (0=par;1=ímpar): ", resto)
```

1.2. CÁLCULOS DE CONVERSÃO

Vejamos agora como calcular uma conversão. Suponhamos que nos seja fornecido um número de segundos qualquer, por exemplo: 54346. Queremos representar este número em termos de horas : minutos : segundos.

Sabendo que:

a) Uma hora tem 3600 segundos;
b) Um minuto tem 60 segundos.

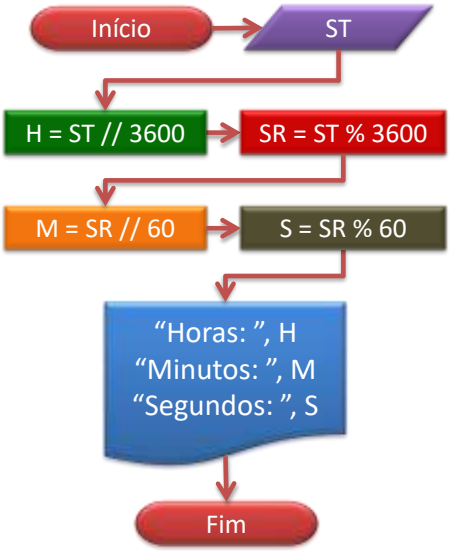
Podemos fazer essa conversão da seguinte forma:
Passo 1: Obter o número de horas com a divisão inteira por 3600
Passo 2: Calcular os segundos restantes com o resto de divisão por 3600
Passo 3: Calcular o número de minutos com a divisão inteira por 60
Passo 4: Calcular os segundos restantes com o resto de divisão por 60

Vejamos:

segundos_a = 54346	
horas = segundos_a // 3600	=> 15 (horas)
segundos_b = segundo_a % 3600	=> 346 (segundos)
minutos = segundos_b // 60	=> 5 (minutos)
segundos = segundos_b % 60	= > 46 (segundos)

Assim, o 54346s = 15h : 5min : 46s

O fluxograma deste algoritmo é o seguinte:



Em portugol, o algoritmo fica como é descrito a seguir. Observe a declaração das variáveis e a sequência de cálculos.

```
ALGORITMO "Converte segundos em h:min:s"
INICIO
inteiros segundos_totais,segundos_restantes
inteiros horas, minutos, segundos

escreva("Digite um valor (em segundos): ")
leia(segundos_totais)
horas <- segundos_totais \ 3600
segundos_restantes <- segundos_totais % 3600
minutos <- segundos_restantes \ 60
segundos <- segundos_restantes % 60
escreva ("Resultado: ", horas, ":",minutos, ":", segundos)
FIM
```

Em Python, este mesmo código tem a seguinte forma:

```
# Converte segundos em h:min:s
segundos_totais = 0; segundos_restantes = 0
horas = 0; minutos = 0; segundos = 0

segundos_totais = int( input ("Digite um valor (em segundos): ") )
horas = int (segundos_totais / 3600)
segundos_restantes = segundos_totais % 3600
minutos = int (segundos_restantes / 60)
segundos = segundos_restantes % 60
print ("Resultado: ", horas, ":", minutos, ":", segundos)
```

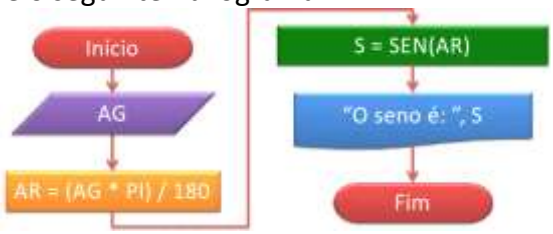
Por que é que eu posso usar a divisão tradicional? Na verdade, o ideal seria usar a divisão inteira; uma vez que o Python não tem esse operador, usamos a divisão tradicional. O truque aqui é "converter" o valor fracionário para inteiro, o que **despreza todos os dígitos após a vírgula**. NOTE que isso **não é o mesmo** que arredondar!

2. FUNÇÕES MATEMÁTICAS

Muitas vezes precisamos usar operações matemáticas na linguagem Python para que possamos implementar algoritmos que solucionem nossos problemas de engenharia. Funções como seno e cosseno, por exemplo, estão diposníveis para nosso uso. Abaixo, segue uma lista de funções, com seus nomes na versão Portugol e Python:

Portugol	Python	Comentário
-	math.ceil(x)	Devolve o valor de x arredondado para cima
cos(x)	math.cos(x)	Devolve o cosseno de x (com x em radianos)
-	math.exp(x)	Devolve o valor de e^x
abs(x)	math.fabs(x)	Devolve o valor absoluto (sem sinal) de x
-	math.floor(x)	Devovle o valor de x arredondado para baixo
logn(x)	math.log(x [,b])	Devolve o logaritmo de x na base b (padrão = e)
log(x)	math.log10(x)	Devolve o logaritmo em base 10 de x
exp(x, y)	math.pow(x, y)	Devolve o valor de x^y
sen(x)	math.sin (x)	Devolve o seno de x (com x em radianos)
raizq(x)	math.sqrt(x)	Devolve a raiz quadrada de x
tan(x)	math.tan(x)	Devolve a tangente de x (com x em radianos)
Pi	math.pi	Devolve o valor de PI (3,141592...)

Como usamos estas "funções" pré-programadas? Podemos usá-las de várias formas. Observe o seguinte fluxograma:



Observe o código abaixo, em portugol:

```
ALGORITMO "Teste de funções matemáticas"
INICIO
real angulo, angulo_radianos, resultado

// Lê ângulo em graus
escreva("Digite um ângulo entre 0 e 360: ")
leia(angulo)

// Calcula ângulo em radianos
angulo_radianos <- (angulo * pi) / 180

// Calcula o cosseno
resultado <- cos( angulo_radianos )
escreva ("O cosseno é: ", resultado)

// Calcula o seno
resultado <- sen( angulo_radianos )
Escreva ("O seno é: ", resultado)

// Calcula a tangente
resultado <- tan( angulo_radianos )
escreva ("A tangente é: ", resultado)
FIM
```

O mesmo código em Python fica assim:

```
import math
# Teste de funções matemáticas
angulo = 0.0; angulo_radianos = 0.0; resultado = 0.0

# Lê ângulo em graus
angulo = float(input("Digite um ângulo entre 0 e 360: "))
# Calcula ângulo em radianos
angulo_radianos = (angulo * math.pi) / 180.0
# Calcula o cosseno
resultado = math.cos( angulo_radianos )
print("O cosseno é: ", resultado)
# Calcula o seno
resultado = sin( angulo_radianos )
print("O seno é: ", resultado)
# Calcula a tangente
resultado = tan( angulo_radianos )
print ("A tangente é: ", resultado)
```

OBSERVE a necessidade de incluir a linha:

import math

Esta linha serve para indicar à linguagem Python que iremos utilizar as **funções matemáticas**. Sem essa linha, o programa **não** vai ser executado corretamente. Essas linhas "import" (que significa "importar") são necessárias porque a linguagem Python é uma

linguagem **extensível**, isto é, ela permite que eu acrescente funções a ela. Veremos futuramente como fazer isso.

Observe também que Python inclui uma função específica para converter de graus para radianos, chamada **math.radians()**. A versão do código a seguir faz usso dessa função:

```
import math
# Teste de funções matemáticas
angulo = 0.0; angulo_radianos = 0.0; resultado = 0.0

# Lê ângulo em graus
angulo = float(input("Digite um ângulo entre 0 e 360: "))
# Calcula ângulo em radianos
angulo_radianos = math.radians(angulo)
# Calcula o cosseno
resultado = math.cos( angulo_radianos )
print("O cosseno é: ", resultado)
# Calcula o seno
resultado = sin( angulo_radianos )
print("O seno é: ", resultado)
# Calcula a tangente
resultado = tan( angulo_radianos )
print ("A tangente é: ", resultado)
```

3. FUNÇÃO DE ARREDONDAMENTO

Ainda que em Portugol não exista uma função pronta para arredondamento, ela existe no Python. O pacote (ou biblioteca) math só tem funções de arredondamento para baixo (math.floor) e para cima (math.ceil). Por exemplo:

Número	math.floor	math.ceil
1,4	1,0	2,0
1,5	1,0	2,0

A função **round**, que não é do pacote math, arredonda o valor como na matemática. Essa função aceita dois parâmetros: **round(x, y)**. X é o valor a ser arredondado, e Y é o número de casas decimais após a vírgula.

Número	math.floor	math.ceil	round
1,4	1,0	2,0	1,0
1,5	1,0	2,0	2,0

Essa função pode ser usada da seguinte forma:

```
# Arredonda um número com 4 casas decimais
num = 0.0; arred = 0.0

# Lê número fracionário
num = float(input("Digite um número fracionário (ex: 1.578): ")

# Arredonda
arred = round(num, 4)

# Imprime resultado:
print ("O valor arredondado é: ", arred)
```

4. BIBLIOGRAFIA

ASCENCIO, A.F.G; CAMPOS, E.A.V. **Fundamentos da Programação de Computadores**. 2ed. Rio de Janeiro, 2007.

MEDINA, M; FERTIG, C. **Algoritmos e Programação: Teoria e Prática**. 2ed. São Paulo: Ed. Novatec, 2006.

SILVA, I.C.S; FALKEMBACH, G.M; SILVEIRA, S.R. **Algoritmos e Programação em Linguagem C**. 1ed. Porto Alegre: Ed. UniRitter, 2010.