Unidade 6: Ambiente de Programação

Prof. Daniel Caetano

Objetivo: Apresentar o uso de funções prontas do Portugol e do C/C++ 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 C/C++. Nesta aula, veremos algumas funções matemáticas mais avançadas do Portugol e do C/C++, 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

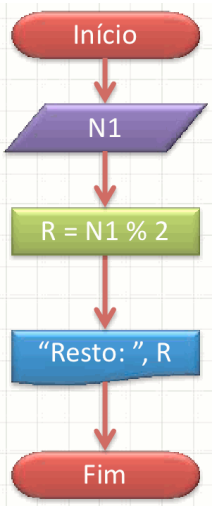
Muitos alunos, em um primeiro momento, se questionam sobre a utilidade do resto de divisão (operador **%**). Parece um cálculo sem muita lógica, já que é pouco usado em nosso dia-a-dia, em que não fazemos cálculos de "divisão inteira".

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

Em C/C++, este mesmo código tem a seguinte forma:

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int valor, resto;

      cout << "Digite um valor: ";
      cin >> valor;
      resto = valor % 2;
      cout << "O resto é (0=par;1=ímpar): " << resto << endl;

getchar();
}
```

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:

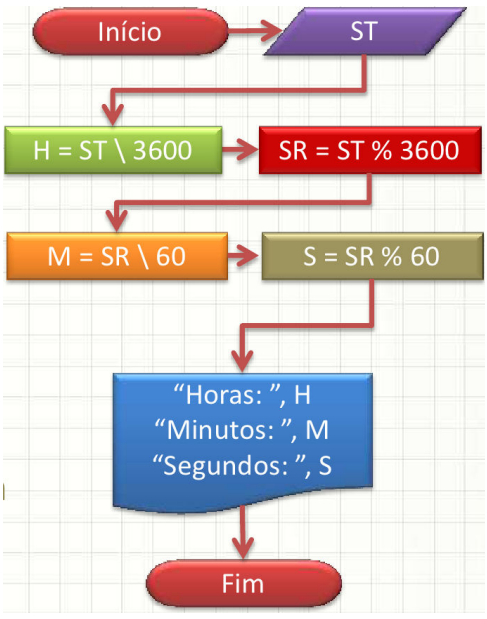
Passo 1: Obter o número de horas com a divisão inteira por 3600
Passo 2: Calcular os segundos restantes com o resto de divisão por 3600
Passo 3: Calcular o número de minutos com a divisão inteira por 60
Passo 4: Calcular os segundos restantes com o resto de divisão por 60

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"
VAR
      segundos, segundos_restantes, segundos_totais : inteiro
      minutos : inteiro
      horas : inteiro
INICIO
      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: ")
      escreva (horas, "h : ")
      escreva (minutos, "min : ")
      escreval (segundos, "s")
FIMALGORITMO
```

Em C/C++, este mesmo código tem a seguinte forma:

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int segundos_restantes, segundos_totais, segundos;
      int minutos;
      int horas;

      cout << "Digite um valor (em segundos): ";
      cin >> segundos_totais;

      horas = segundos_totais / 3600;
      segundos_restantes = segundos_totais % 3600;
      minutos = segundos_restantes / 60;
      segundos = segundos_restantes % 60;

      cout << "Resultado: ";
      cout << horas << "h : ";
      cout << minutos << "min : ";
      cout << segundos << "s" << endl;

getchar();
}
```

Por que é que eu posso usar a divisão tradicional? Na verdade, o ideal seria usar a divisão inteira; uma vez que o C/C++ não tem esse operador, usamos a divisão tradicional. Isso funciona porque as variáveis com que estamos trabalhando são **inteiras**, isto é, elas não armazenam a parte fracionária dos cálculos. Isso significa que, quando guardamos um número fracionário em uma variável inteira, **a parte fracionária é perdida**! NOTE que isso **não é o mesmo** que arredondar!

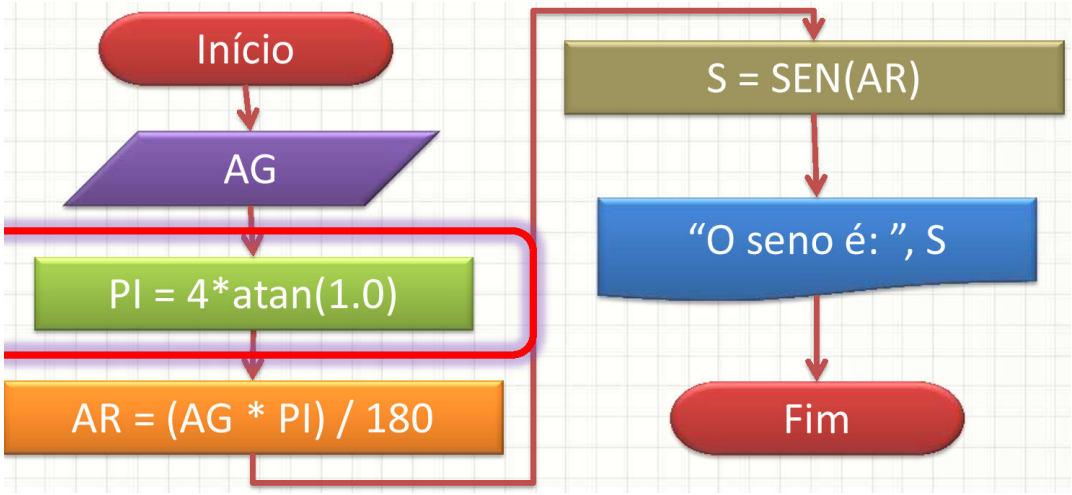
2. FUNÇÕES MATEMÁTICAS

Muitas vezes precisamos usar operações matemáticas na linguagem C/C++ 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 C/C++:

Portugol	C/C++	Comentário
-	ceil(x)	Devolve o valor de x arredondado para cima
cos(x)	cos(x)	Devolve o cosseno de x (com x em radianos)
-	exp(x)	Devolve o valor de e^x
abs(x)	abs(x)	Devolve o valor absoluto (sem sinal) de x
-	floor(x)	Devovle o valor de x arredondado para baixo
logn(x)	log(x)	Devolve o logaritmo natural de x
log(x)	log10(x)	Devolve o logaritmo em base 10 de x
exp(x, y)	pow(x, y)	Devolve o valor de x^y
sen(x)	sin (x)	Devolve o seno de x (com x em radianos)
raizq(x)	sqrt(x)	Devolve a raiz quadrada de x
tan(x)	tan(x)	Devolve a tangente de x (com x em radianos)
Pi	4.0 * atan(1.0)	Devolve o valor de PI (3,141592...)

Como usamos estas "funções" pré-programadas?

Podemos usá-las de várias formas. Observe o seguinte fluxograma:



(A operação do retângulo verde é desnecessária no Portugol!)

Observe o código abaixo, em portugol:

```
ALGORITMO "Teste de funções matemáticas"
VAR
      angulo, angulo_radianos, resultado : real
INICIO
      // Pega 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 )
      escreval ("O cosseno é: ", resultado)
      // Calcula o seno
      resultado <- sen( angulo_radianos )
      escreval ("O seno é: ", resultado)
      // Calcula a tangente
      resultado <- tan( angulo_radianos )
      escreval ("A tangente é: ", resultado)
FIMALGORITMO
```

O mesmo código em C/C++ fica assim:

```
#include <stdio>
#include <iostream>
#include <math>

using namespace std;
int main(void) {

      float angulo, angulo_radianos, resultado, pi;

      // Pega ângulo em graus
      cout << "Digite um ângulo entre 0 e 360: ";
      cin >> angulo;

      // Calcula PI
      pi = 4.0 * atan(1.0);

      // Calcula ângulo em radianos
      angulo_radianos = (angulo * pi) / 180.0;

      // Calcula o cosseno
      resultado = cos( angulo_radianos );
      cout << "O cosseno é: " << resultado << endl;
      // Calcula o seno
      resultado = sin( angulo_radianos );
      cout << "O seno é: " << resultado << endl;
      // Calcula a tangente
      resultado = tan( angulo_radianos );
      cout << "A tangente é: " << resultado << endl;

getchar();
}
```

OBSERVE a necessidade de incluir a linha:

#include <math.h>

Esta linha serve para indicar à linguagem C/C++ que iremos utilizar as **funções matemáticas**. Sem essa linha, o programa **não** vai ser gerado. Essas linhas "include" (que significa "incluir") são necessárias porque a linguagem C/C++ é uma linguagem **extensível**, isto é, ela permite que eu acrescente funções a ela. Veremos futuramente como fazer isso.

3. FUNÇÃO DE ARREDONDAMENTO

Ainda que em Portugol não exista uma função pronta para arredondamento, ela existe no C/C++. Entretanto, o C/C++ só tem funções de arredondamento para baixo (floor) e para cima (ceil). Por exemplo:

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

Elas podem ser usadas da seguinte forma:

```
#include <stdio>
#include <iostream>
#include <math>

using namespace std;
int main(void) {

      float num, arred;

      // Pega número fracionário
      cout << "Digite um número fracionário (ex: 1.578): ";
      cin >> num;

      // Arredonda
      arred = floor(num);

      // Imprime resultado:
      cout << "O valor arredondado é: " << arred << endl;

getchar();
}
```

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.