Alfredo's MAC0110 Journal

Alfredo Goldman

March 15, 2020

0.1 Aula 05 - *<2020-03-16 seg>*

Nessa aula, em meio ao caos de uma pandemia mundial, vamos aprender um novo comando. O desvio condicional, através dele é possível alterar o fluxo de execução de um programa. Até o momento não tínhamos comentado isso explicitamente, mas a ordem de execução de instruções segue a ordem em que elas estão. Vejamos o exemplo abaixo:

```
println("Oi")
println("um")
println("dois")
```

A ordem de impressão será Oi, um e dois.

Da mesma forma não temos problema ao executar o código abaixo.

```
denominador = 0
denominador += 2
30 / denominador
```

Apesar da variável denominador começar inicialmente com 0, antes de se fazer a divisão, ela estará valendo 2.

Como é de se esperar nem sempre queremos que essa ordem seja respeitada. Observe o seguinte exemplo:

```
pandemia = true
println("Vou sair de casa?")
if pandemia == true
   println("So vou sair de casa se for essencial")
end
```

O exemplo acima é claro, se uma condição for verdadeira, o código que está no escopo do if (isso é entre a condição e o end) será executado.

Um outro exemplo:

```
denominador = 0
if denominador != 0
   println("sei fazer a divisao se nao for por zero")
   println("o resultado da divisao de 30 por ", denominador, " e igual a ", 30/denominador)
end
```

Situações muito comuns em computação devem ser favorecidas pela linguagem, nesse caso do if, é muito comum termos duas ou mais situações. Nesse sentido em Julia podemos também ter alternativas como abaixo:

```
pandemia = true
println("Vou sair de casa?")
if pandemia == true
   println("So vou sair de casa se for essencial")
else
   println("Balada liberada")
end
```

No caso de termos mais de uma altenativa, não basta termos só uma condição, nesse caso temos que usar elseif.

```
pandemia = true
tenhoqueestudar = false
println("Vou sair de casa?")
if pandemia == true
   println("So vou sair de casa se for essencial")
elseif tenhoqueestudar == true
   println("Melhor ficar em casa")
else
   println("Balada liberada")
end
```

Conhecendo o if, agora, escreva uma função que recebe os coeficientes, a, b e c de uma equação de segundo grau e imprime as suas raízes reais.

Espaço para a solução aqui :)

Vamos agora a parte mais importante da aula, lembrando que até o momento aprendemos:

1. valores
2. varíaveis e alguns dos seus tipos
3. Alguma funções já prontas como div(), typeof(), parse(), string(), println(), etc
4. como fazer as nossas funções com a palavra reservada function e que termina por end
   (a) lembrando que a função pode ou não devolver algo através do return
   (b) lembrando também que uma função pode chamar outra função
5. como mudar o fluxo de execução normal com o if, elseif

   Mas, agora vem a dúvida, uma função pode se chamar?

```
function imprime()
  println("Mensagem")
  imprime()
end
```

   O resultado da função é curioso, ela vai ficar se chamando até uma memória do computador acabar (para quem conhece é a pilha ou stack). Mas, será que podemos usar isso de forma mais inteligente ao nosso favor? Isso é, em algum momento a função teria que parar de se chamar de forma a não acabar com erro.

   Uma forma de se fazer isso é através de um comando como o if, que pode ou não seguir chamando a função, mas para isso vamos precisar receber um parâmetro.

```
function countdown(n)
  println(n)
  if n > 0
    countdown(n-1)
  else
    println("Acabou")
  end
end
countdown(10)
```

   Para entender um pouco melhor o que acontece acima, vamos colocar mais umas impressões.

```
function countdown(n)
  println(n)
  if n > 0
    println("Vou chamar countdown com n = ", n - 1)
    countdown(n-1)
    println("Voltei da chamada com n = ", n - 1)
  else
    println("Acabou")
  end
end
countdown(10)
```

   Observando a sequência de chamadas, fica claro como funciona o computador, de alguma forma, cada uma das chamadas é empilhada (colocada em um estrutura como uma pilha, de livros, mas no caso de chamadas de função), sendo que no final são desempilhadas.

   Vou desenhar para o caso countdown(5).

   Essa estrutura é bem poderosa, pois permite que operações sejam executadas um número controlado de vezes. Voltando ao countdown, imagine que ao invés de imprimir uma mensagem quiséssemos fazer uma conta com o que será devolvido.

```
function soma(n)
  if n > 0
    return n + soma(n - 1)
  else
    return 1
  end
end
soma(10)
```

   Essa estrutura é bastante poderosa e pode ser usada para o cálculo de produto, nesse caso, a mudança é bem pequena.

   Da mesma forma segue um exemplo para o cálculo dos n primeiros elementos da soma hârmonica.

```
function somaharmonica(atual, n)
  if atual >= n
    return 1.0 / atual
  else
    return 1.0 / atual + somaharmonica(atual + 1, n)
  end
end
somaharmonica(1, 10)
```

O miniEP para a próxima semana será:

(a) computar a soma dos n primeiros inversos dos quadrados

(b) computar os n primeiros elementos da soma harmônica alternada, onde os elementos 1/n são somados com sinais alternantes. Para isso um dos parâmetros a serem passados será o sinal.