BIS0005 - Bases Computacionais da Ciência

Aula 07 - Lógica de programação: Estruturas de repetição

Saul Leite
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Q2 2018

Introdução

Estruturas de controle permitem o controle do fluxo de execução dos comandos. Existem três estruturas básicas de controle:

1. Sequencial (aula 05)

3. Condicional ou Desvio (if) (aula 06)

4. Repetição (nesta aula)

Laços (Estruturas de repetição)

Usado em situações em que é necessário repetir um determinado trecho de um programa várias vezes.

Um exemplo bem simples: suponha que queremos imprimir “oi” 5 vezes.

Laços (Estruturas de repetição)

Usado em situações em que é necessário repetir um determinado trecho de um programa várias vezes.

Um exemplo bem simples: suponha que queremos imprimir “oi” 5 vezes.

Uma solução seria o seguinte:

```
cat("oi\n")
cat("oi\n")
cat("oi\n")
cat("oi\n")
cat("oi\n")
```

Estrutura de Repetição

De forma alternativa, podemos fazer isso com uma estrutura de repetição!

Veremos duas formas de fazer repetições:

- **Enquanto-faça** ou **while** em R;
- **Para-cada** ou **for** em R.

Estrutura de Repetição (While)

Estrutura de Repetição **Enquanto-faça** ou **while**

enquanto ***Condição*** **faça**

```
    #Comandos para serem repetidos
    Comando a
    Comando b
    ...
    Comando z
```

fim

Resto do programa

Os comandos dentro do bloco irão se repetir **enquanto** a **Condição** for VERDADEIRA.

Obs.: Note que a **Condição** é um predicado lógico (expressões que devolvem um valor VERDADEIRO OU FALSO) – o mesma da aula passada.

Estrutura de Repetição (While): Exemplo 1

Vamos imprimir "oi" 5 vezes:

```
i <- 1
while( i <= 5 )
{
  cat("oi\n")
  i <- i + 1
}
```

Obs.: Tudo que está contido dentro das **chaves** $\{\}$ será repetido!

Obs.: Note que a variável **i** está contando o número de vezes que estamos repetindo! Chamamos estas variáveis de **contadores**.

Estrutura de Repetição (While): Exemplo 2

Vamos fazer um programa para imprimir todos os números pares de 1 até 20:

```
i <- 1
while( i <= 20 )
{
  if( i %% 2 == 0 ){
    cat("O numero ",i," é par\n")
  }
  i <- i + 1
}
```

Obs.: Note que usamos uma variável **contadora** novamente!

Estrutura de Repetição

Estruturas de repetição também são muito úteis para **verificar valores entrados** em programas.

Por exemplo, suponha que no nosso programa, precisamos ler um valor X positivo.

Podemos continuar solicitando a informação até que o valor seja entrado de forma correta!

Estrutura de Repetição (While): Exemplo 3

No exemplo abaixo, o valor de X continua sendo solicitado para o usuário enquanto ele não fornecer um valor positivo.

```
x <- as.numeric(readline("Digite um número positivo: "))

while( x <= 0 ){

  cat("O número não é positivo\n")
  x <- as.numeric(readline("Digite um número positivo: "))

}

cat("Numero entrado foi: ",x,"\n")
```

Estrutura de Repetição

Estruturas de repetição também são úteis para fazer somas.

Suponha que precisamos calcular a seguinte soma:

$$S = 1 + 2 + 3 + 4 + 5 + 6$$

Como fazer isso usando uma estrutura de repetição?

Estrutura de Repetição

A ideia é fazer a somas iterativamente, somando um termo de cada vez. Isso é feito usando uma **variável acumuladora**.

Estrutura de Repetição

A ideia é fazer a somas iterativamente, somando um termo de cada vez. Isso é feito usando uma **variável acumuladora**.

Idéia: **Somar termo a termo**:

```
S <- 1
S <- S + 2    # S <-  1 + 2   (= 3)
S <- S + 3    # S <-  3 + 3   (= 6)
S <- S + 4    # S <-  6 + 4   (= 10)
S <- S + 5    # S <- 10 + 5   (= 15)
S <- S + 6    # S <- 15 + 6   (= 21)
cat("O resultado da soma é: ",S,"\n")
```

A variável **S** está acumulando os valores.

Estrutura de Repetição (While): Exemplo 4

A ideia é fazer a somas iterativamente, somando um termo de cada vez. Isso é feito usando uma **variável acumuladora**.

Idéia: **Somar termo a termo**:

```
S <- 1
i <- 2
while( i <= 6 ){
  S <- S + i
  i <- i + 1
}
cat("O resultado da soma é: ",S,"\n")
```

A variável **S** está acumulando os valores.

Obs.: Note que usamos uma variável **contadora** novamente!

Estrutura de Repetição (For)

Estrutura de Repetição **Para-cada** ou **for**

para *cada* ***elemento no conjunto*** **faça**

```
    #Comandos para serem repetidos
    Comando a
    Comando b
    ...
    Comando z
```

fim

Resto do programa

Estas estruturas são muito úteis quando precisamos usar **contadores**.

Estrutura de Repetição (For): Exemplo 1 - revisitado

Vamos imprimir a palavra "oi" 5 vezes usando o **for**:

```
for( i in 1:5 ){

  cat("oi\n")

}
```

Os comandos entre **chaves** $\{\}$ são repetidos para cada elemento **i** do **conjunto {1,2,3,4,5}**.

Obs.: Lembre-se que 1:5 gera um vetor com elementos:

```
## [1] 1 2 3 4 5
```

Estrutura de Repetição (For): Exemplo 4 - revisitado

Vamos utilizar o *for* para fazer a seguinte soma:

$$S = 1 + 2 + 3 + 4 + 5 + 6$$

Estrutura de Repetição (For): Exemplo 4 - revisitado

Vamos utilizar o *for* para fazer a seguinte soma:

$$S = 1 + 2 + 3 + 4 + 5 + 6$$

```
S <- 1
for( i in 2:6 ){

  S <- S + i

}
cat("O resultado da soma é: ",S,"\n")
```

Obs.: Note que nós precisamos do **acumulador**.

EXEMPLOS

Exemplo 5

Faça **um programa** que lê dois valores entrados x e y e soma todos os números entre estes dois valores.

Exemplos de entrada e saída:

```
Exemplo1: 9, 12
Saída: A soma dos valores é = 42

Exemplo: 100, 1
Saída: A soma dos valores é = 5050
```

Exemplo 5

Solução:

```
x <- as.numeric(readline("Digite o valor de x: "))
y <- as.numeric(readline("Digite o valor de y: "))

S <- 0
for( i in x:y ){

  S <- S + i
}

cat("A soma dos valores é =",S,"\n")
```

Exemplo 6

Defina uma **função** fatorial que recebe como argumento um número e calcula o fatorial deste número. Lembre-se que o fatorial de um número n é dado por:

$$n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdots (n-1) \cdot n$$

Exemplos de chamada da função:

```
#Exemplo1:
fatorial(3)
```

```
## [1] 6
```

```
#Exemplo2:
fatorial(5)
```

```
## [1] 120
```

Exemplo 6

Solução:

```
fatorial <- function(n){

  M <- 1
  if( n != 0 ){
    for(i in 1:n){
      M <- M * i
    }
  }
  return(M)
}
```

Obs.: Esta função já existe na linguagem R, é chamada de *factorial*.

Exemplo 7

A expressão abaixo é chamada de Fórmula de Leibniz para π:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots$$

O exercício consiste em fazer um programa que pede ao usuário um valor para N e faz a soma acima até o N-ésimo termo. Ou seja, se $N = 4$, iremos fazer:

$$S = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7}$$

Note que pelo resultado no parágrafo acima, essa soma deve tender a $\pi/4$ quando N é muito grande!

Exemplo 7

Solução:

```
n <- as.numeric(readline("Digite um valor para N: "))

S <- 1
for( i in 2:n ){

  if( i %% 2 == 0 ){
    sinal = -1
  } else {
    sinal = 1
  }
  S <- S + sinal/(i*2-1)
}

cat("Temos a seguinte aproximação para pi: ",S*4,"\n")
```

Exemplo 8

Defina uma **função** que calcula o maior divisor comum (MDC) entre dois números x e y usando o algoritmo de Euclides abaixo:

Algoritmo de Euclides para MDC, suponha que $x > y$:

1. Divida x por y;
2. Guarde o resultado da divisão inteira em div;
3. Guarde o resto da divisão inteira em $resto$;
4. Se o $resto$ for zero, o MDC é y;
5. Se não, faça:
   - $x \leftarrow y$;
   - $y \leftarrow resto$;
   - volte ao passo 1;

Exemplo 8

Solução:

```
MDC <- function(x,y){

  while(TRUE){

      div   <- x %/% y
      resto <- x %% y

      if( resto == 0 ){
        break;
      } else {
        x <- y
        y <- resto
      }
  }
  return(y)
}
```

Obs.: O comando *break* interrompe a repetição.

ATIVIDADE EM SALA

Exercicio 1

Crie um programa que faz a soma dos números **ímpares** da sequência de inteiros contida no intervalo $[x, y]$. O seu programa deve solicitar os valores de x e y. Fique atento para tratar o caso em que $x > y$ corretamente.

O resultado da soma deve ser impresso na tela.

```
Entrada 1: 8, 101
Saída: 2585

Entrada 1: 90, 3
Saída: 2024
```

Exercício 2

Crie uma **função** *soma* que recebe por argumento o valor de n e retorna o valor da seguinte soma:

$$S = \frac{1}{n} + \frac{2}{n-1} + \frac{3}{n-2} + \frac{4}{n-3} + \cdots + \frac{n}{1}$$

Exemplos execução e saída:

```
soma(10)
```

```
## [1] 22.21865
```

```
soma(15)
```

```
## [1] 38.09166
```

Exercício 3

Faça uma **função** *arctan* que recebe o número real $x \in [0, 1]$ como argumento e retorna uma aproximação do arco tangente de x (em radianos) através da série:

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

Exemplos execução e saída:

```
arctan(0.9)
```

```
## [1] 0.7328151
```

```
arctan(0.2)
```

```
## [1] 0.1973956
```

Exercício 4

Crie um programa que lê do teclado um número inteiro positivo x e imprime na tela se o número é primo ou não.

Obs.: O programa deve continuar solicitando o valor de x se o valor entrado não for positivo.

Exercício 5

Faça um programa que calcula o valor de S dado abaixo.

$$S = 1 + \frac{3}{2} + \frac{5}{3} + \cdots + \frac{99}{50}$$

Resposta: 95.50079

Exercício 6

Faça um programa que calcula o valor de S dado abaixo.

$$S = \frac{5}{1} - \frac{6}{4} + \frac{7}{9} - \frac{8}{16} + \frac{9}{25} - \frac{10}{36} + \cdots - \frac{14}{100}$$

Resposta: 3.917484

Exercício 7

Faça um programa que lê do teclado um número natural n e uma sequencia de n números naturais e verifica quantos são múltiplos de 3 e 5.

Referências

- Aulas dos Profs. David Correa Martins Jr, Jesús P. Mena-Chalco.
- Livro Bases Computacionais da Ciência.