MAC 113 — Introdução à Ciência da Computação

Aula 19

Nelson Lago

1°/2025

Previously on MAC113...

Repetições encaixadas

- **Repetições usam uma variável de controle**
- **Quando essa variável corresponde a um conjunto, as repetições permitem trabalhar sobre os elementos desse conjunto**
  - De maneira unidimensional!
  - Números naturais, nomes de pessoas, notas de alunos...
- **Como trabalhar com conjuntos de maneira bidimensional?**
  - Pontos de um plano, tabelas...
- **Repetições encaixadas**

Repetições encaixadas

```
while (condição1) {
    while (condição2) {
        …
    }
}
```

- **A cada "rodada" do laço mais externo, o laço interno é executado várias vezes (até sua condição deixar de ser verdadeira)**
- **Para isso funcionar, `condição2` deve voltar a ser verdadeira a cada nova rodada do laço mais externo**
- **Duas variáveis de controle**
  - Um olho no peixe, outro no gato 😉

Exemplo — tabuada

```
tabuada_do <- 1
while (tabuada_do <= 10) {
    n <- 1
    while (n <= 10) {
        cat(format(tabuada_do * n, width=3), "")
        n <- n + 1
    }
    cat("\n")
    tabuada_do <- tabuada_do + 1
}
```

```
  1   2   3   4   5   6   7   8   9  10
  2   4   6   8  10  12  14  16  18  20
  3   6   9  12  15  18  21  24  27  30
  4   8  12  16  20  24  28  32  36  40
  5  10  15  20  25  30  35  40  45  50
  6  12  18  24  30  36  42  48  54  60
  7  14  21  28  35  42  49  56  63  70
  8  16  24  32  40  48  56  64  72  80
  9  18  27  36  45  54  63  72  81  90
 10  20  30  40  50  60  70  80  90 100
```

Tipos de repetição

- **Dois tipos fundamentais de repetição**
  1. Repetições até atingir um resultado
     - » *Encontrar o próximo primo*
     - » *Reiniciar o jogo até o usuário escolher "sair"*
     - » *...*
  2. Repetições sobre os elementos de um conjunto
     - » *Apresentar todos os pixels de uma foto na tela*
     - » *Trocar todas as letras de um texto para maiúsculas*
     - » *...*

Repetições com coleções

```
primos <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)
i <- 1
while (i < length(primos)) {
    cat("O número", primos[i], "é primo", "\n")
    i <- i + 1
}
```

- **`i` é a variável de controle do laço**
- **Mas! Não usamos o valor de `i` "de verdade"**
  - Só usamos `i` para acessar os elementos do vetor

```
primos <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)
for (p in primos) {
    cat("O número", p, "é primo", "\n")
}
```

- **outro jeito de fazer repetições sobre os elementos de um conjunto**

Repetições com coleções

```r
primos <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)
for (p in primos) {
    cat("O número", p, "é primo", "\n")
}
```

- **Um laço correto precisa**
  - Inicializar a variável de controle antes do início do laço
  - Verificar a condição adequada a cada iteração para que as repetições aconteçam o número correto de vezes
  - Alterar o valor da variável de acordo com a lógica do programa (no mínimo, na última iteração) para garantir que o laço termine
- **Os laços com `for` "escondem" esses passos**
  - (R faz automaticamente para você)

Repetições com coleções

```r
primos <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)
for (p in primos) {
    cat("O número", p, "é primo", "\n")
}
```

```r
primos <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)
i <- 1
while (i <= length(primos)) {
    p <- primos[i]
    cat("O número", p, "é primo", "\n")
    i <- i + 1
}
```

Repetições com coleções

```r
cat("Prepare-se para o grito:", "\n")
n <- 10
while (n > 0) {
    cat(n, "\n")
    n <- n -1
}
cat("AAAHHH!!!!", "\n")
```

- **Repetições sobre o conjunto dos números 10–1**
- **Podemos criar um vector com esses números!**

```r
cat("Prepare-se para o grito:", "\n")
for (n in 10:1) {
    cat(n, "\n")
}
cat("AAAHHH!!!!", "\n")
```

Tipos de laços diferentes “combinam melhor” com sintaxes diferentes

Tipos de repetição

- **Quando a quantidade de repetições é desconhecida, `while` é uma boa escolha:**
  - `while (! achei)`
  - `while (encontrados < 10)`
  - `while (usuárioQuerJogar)`
- **Quando queremos manipular os elementos de uma coleção, `for (... in ...)` é uma boa escolha:**
  - `for (p in primos)`
  - `for (canção in canções)`
- **Quando queremos manipular uma lista pré-definida de números *ou* os *índices* dos elementos de uma coleção, `for (n in X:Y)` é uma boa escolha:**
  - `for (n in 1:10)`
  - `for (n 1:length(meu_vec))`

Repetições com coleções

- **Muitas vezes, estamos interessados apenas nos elementos do conjunto**

```
primos <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)
for (p in primos) {
    cat("O número", p, "é primo", "\n")
}
```

- **Mas às vezes estamos interessados na posição (índice) dos elementos também**

```
primos <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)
for (i in 1:length(primos)) {
    cat(glue("O {i}o primo é {primos[i]}"), "\n")
}
```

And now for something somewhat different

Exercício – funções vetoriais

Escreva uma função que recebe um vetor de números e devolve o maior elemento

```
maior <- function(v) {
    m <- v[1]
    for (elem in v) {
        if (elem > m) { m <- elem }
    }
    return (m)
}
```

Exercício – funções vetoriais

Escreva uma função que recebe um vetor de números e devolve o menor elemento

```
menor <- function(v) {
    m <- v[1]
    for (elem in v) {
        if (elem < m) { m <- elem }
    }
    return (m)
}
```

Exercício – funções vetoriais

Dada uma lista com a idade dos alunos de MAC113, imprima a diferença entre a maior e a menor idades da turma

```
main <- function() {
    idades <- c(18, 18, 19, 18, 20, 18, 19, 17, 18, 19)
    cat("A diferença das idades é", maior(idades) - menor(idades), "\n")
}
main()
```

Exercício – funções vetoriais

Escreva uma função que recebe um vetor de números e devolve a soma dos seus elementos

```
somatório <- function(v) {
    s <- 0
    for (n in v) { s <- s + n }
    return (s)
}
```

Exercício – funções vetoriais

Escreva uma função que recebe um vetor de números
e devolve a média dos seus elementos

```
média <- function(v) {
    return (somatório(v) / length(v))
}
```

Exercício – funções vetoriais

Dada uma lista com as notas dos alunos de MAC113, imprima a média da turma

```
main <- function() {
    notas <- c(6.3, 9.1, 7.7, 6.8, 9.6, 8.4)
    cat("A média da turma é", média(notas), "\n")
}
main()
```

Exercício – funções vetoriais

Escreva uma função que recebe um vetor de números e devolve **TRUE** ou **FALSE** indicando se algum dos números é par

```
tem_par <- function(v) {
    for (n in v) {
        if (n %% 2 == 0) { return (TRUE) }
    }
    return (FALSE)
}
```

Exercício – funções vetoriais

Escreva uma função que recebe dois vetores de números do mesmo tamanho e devolve um vetor com a soma dos elementos dos vetores um a um

```
soma <- function(v1, v2) {
    s <- numeric(length(v1))
    for (i in 1:length(v1)) { s[i] <- v1[i] + v2[i] }
    return (s)
}
```

Exercício – funções vetoriais

Dadas duas listas de números, diga se a soma dos elementos das listas um a um resulta em algum número par

```
main <- function() {
    v1 <- c(1, 3, 5)
    v2 <- c(2, 4, 7)
    s <- soma(v1, v2)
    if (tem_par(s)) {
        cat("A soma inclui números pares\n")
    } else {
        cat("A soma não inclui números pares\n")
    }
}
main()
```

Funções vetoriais

- Quando queremos fazer algo envolvendo vetores como um todo, funções vetoriais são muito úteis
- Tão úteis que R possui várias funções vetoriais prontas!

```
x <- c(1, 5, 3)
cat(max(x), mean(x), "\n")
```

```
5 3
```

- Tão úteis que, em R, muitas funções e operadores que podem processar dados "simples" são também funções vetoriais!

```
x <- c(1, 2, 3)
y <- c(4, 5, 6)
z <- x + y
cat(z, "\n")
```

```
5 7 9
```

paste()

- **Uma função vetorial interessante é `paste()`**
  - Ela processa os elementos de um vetor e gera um vetor de strings correspondente

```
frutas <- c("caqui", "maçã", "abacate")
cat(paste("Eu gosto de", frutas), "\n")
```

```
Eu gosto de caqui Eu gosto de maçã Eu gosto de abacate
```

Exercício – cartas do baralho

Usando a função `paste()`, escreva um programa que imprime o nome de todas as cartas do baralho (“ás de paus, 2 de paus,..., 10 de copas, valete de copas...”)

```
main <- function() {
    cartas <- c()
    nums <- c("ás", as.character(2:10), "valete", "dama", "rei")
    naipes <- c("ouros", "copas", "paus", "espadas")
    for (naipe in naipes) {
        cartas <- c(cartas, paste(nums, "de", naipe))
    }
    cartas <- c(cartas, "coringa vermelho", "coringa preto")
    cat(cartas, "\n")
}
main()
```