

MAC 113 — Introdução à Ciência da Computação

Aula 9

Nelson Lago

1º/2025



Exercício — adivinhando números 1/4

Faça um programa que escolhe um número de 1 a 10 e pede para o usuário tentar adivinhá-lo até que o usuário tenha sucesso (repetições até atingir um resultado)

```
main <- function() {  
  n <- sample(1:10, 1)  
  chute <- as.integer(readline("Escolhi um número de 1 a 10; tente adivinhá-lo! "))  
  while (chute != n) {  
    chute <- as.integer(readline("Errou! Tente de novo: "))  
  }  
  cat("Parabéns, você acertou!", "\n")  
}  
main()
```

Exercício — adivinhando números 2/4

Modifique o programa anterior para que ele informe o usuário se o número correto é maior ou menor que o número informado

Exercício — adivinhando números 2/4

Modifique o programa anterior para que ele informe o usuário se o número correto é maior ou menor que o número informado

```
main <- function() {  
  n <- sample(1:10, 1)  
  chute <- as.integer(readline("Escolhi um número de 1 a 10; tente adivinhá-lo! "))  
  while (chute != n) {  
  
    chute <- as.integer(readline(  
    })  
  }  
  cat("Parabéns, você acertou!", "\n")  
}  
main()
```

Exercício — adivinhando números 2/4

Modifique o programa anterior para que ele informe o usuário se o número correto é maior ou menor que o número informado

```
library(glue)
main <- function() {
  n <- sample(1:10, 1)
  chute <- as.integer(readline("Escolhi um número de 1 a 10; tente adivinhá-lo! "))
  while (chute != n) {

    chute <- as.integer(readline(glue("Errou, o número é {msg}! Tente de novo: ")))
  }
  cat("Parabéns, você acertou!", "\n")
}
main()
```

Exercício — adivinhando números 2/4

Modifique o programa anterior para que ele informe o usuário se o número correto é maior ou menor que o número informado

```
library(glue)
main <- function() {
  n <- sample(1:10, 1)
  chute <- as.integer(readline("Escolhi um número de 1 a 10; tente adivinhá-lo! "))
  while (chute != n) {
    if (n > chute) {

    }

  }
  chute <- as.integer(readline(glue("Errou, o número é {msg}! Tente de novo: ")))
}
cat("Parabéns, você acertou!", "\n")
}
main()
```

Exercício — adivinhando números 2/4

Modifique o programa anterior para que ele informe o usuário se o número correto é maior ou menor que o número informado

```
library(glue)
main <- function() {
  n <- sample(1:10, 1)
  chute <- as.integer(readline("Escolhi um número de 1 a 10; tente adivinhá-lo! "))
  while (chute != n) {
    if (n > chute) {
      msg <- "maior"
    }

    }
    chute <- as.integer(readline(glue("Errou, o número é {msg}! Tente de novo: ")))
  }
  cat("Parabéns, você acertou!", "\n")
}
main()
```

Exercício — adivinhando números 2/4

Modifique o programa anterior para que ele informe o usuário se o número correto é maior ou menor que o número informado

```
library(glue)
main <- function() {
  n <- sample(1:10, 1)
  chute <- as.integer(readline("Escolhi um número de 1 a 10; tente adivinhá-lo! "))
  while (chute != n) {
    if (n > chute) {
      msg <- "maior"
    } else {
      msg <- "menor"
    }
    chute <- as.integer(readline(glue("Errou, o número é {msg}! Tente de novo: ")))
  }
  cat("Parabéns, você acertou!", "\n")
}
main()
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n <- as.integer(readline("Digite o número n: "))  
d <- as.integer(readline("Digite o algarismo d: "))
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n <- as.integer(readline("Digite o número n: "))  
d <- as.integer(readline("Digite o algarismo d: "))
```

```
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))

conta <- 0 # elemento neutro

cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))

conta <- 0 # elemento neutro
while (n > 0) {

}

cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))

conta <- 0 # elemento neutro
while (n > 0) {

    n <- n %% 10

}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))

conta <- 0 # elemento neutro
while (n > 0) {
  este <- n %% 10

  n <- n %% 10

  if (este == d) {
    conta <- conta + 1
  }
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))

conta <- 0 # elemento neutro
while (n > 0) {
  este <- n %% 10
  if (este == d) {
    conta <- conta + 1
  }
  n <- n %/% 10
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).



```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))

conta <- 0 # elemento neutro
while (n > 0) {
  este <- n %% 10
  if (este == d) {
    conta <- conta + 1
  }
  n <- n %/% 10
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))

val <- n
conta <- 0 # elemento neutro
while (val > 0) {
  este <- val %% 10
  if (este == d) {
    conta <- conta + 1
  }
  val <- val %/% 10
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

E se n for zero?

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))

val <- n
conta <- 0 # elemento neutro
while (val > 0) {
  este <- val %% 10
  if (este == d) {
    conta <- conta + 1
  }
  val <- val %/% 10
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

E se n for zero?

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))
if (                ) {                }
val <- n
conta <- 0 # elemento neutro
while (val > 0) {
  este <- val %% 10
  if (este == d) {
    conta <- conta + 1
  }
  val <- val %/% 10
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

E se n for zero?

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))
if (n == 0) {
  # caso n seja zero, a contagem é zero
  conta <- 0
} else {
  val <- n
  conta <- 0 # elemento neutro
  while (val > 0) {
    este <- val %% 10
    if (este == d) {
      conta <- conta + 1
    }
    val <- val %/% 10
  }
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

E se n for zero?

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))
if (n == 0) { conta <- 1 }
val <- n
conta <- 0 # elemento neutro
while (val > 0) {
  este <- val %% 10
  if (este == d) {
    conta <- conta + 1
  }
  val <- val %/% 10
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — contagem de dígitos

Dado um natural $n \geq 0$ e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

E se n for zero?

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))
if (n == 0 && d == 0) { conta <- 1 }
val <- n
conta <- 0 # elemento neutro
while (val > 0) {
  este <- val %% 10
  if (este == d) {
    conta <- conta + 1
  }
  val <- val %/% 10
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

k: 25

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3

x: 5

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3

x: 5

1ª iteração

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3

x: 5

1ª iteração

k: 20

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3

x: 5

1ª iteração

k: 20 a: 23

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3

x: 5

1ª iteração

k: 20 a: 23 **saída: 35**

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14**

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21**

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53 **saída: 42**

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53 **saída: 42** x: 4

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53 **saída: 42** x: 4

5ª iteração

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53 **saída: 42** x: 4

5ª iteração

k: 0

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53 **saída: 42** x: 4

5ª iteração

k: 0 a: 53

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53 **saída: 42** x: 4

5ª iteração

k: 0 a: 53 **saída: 28**

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53 **saída: 42** x: 4

5ª iteração

k: 0 a: 53 **saída: 28** x: 4

Exercício — pensando como o computador

O que este programa imprime se o usuário escolhe o número 5?

```
main <- function() {  
  k <- 25  
  a <- 3  
  x <- as.integer(readline("Digite um inteiro positivo: "))  
  i <- 1  
  while (i <= 5) {  
    k <- k - 5  
    a <- a + k  
    cat(x * 7, "\n")  
    x <- a %% 7  
    i <- i + 1  
  }  
}  
main()
```

k: 25 a: 3 x: 5

1ª iteração

k: 20 a: 23 **saída: 35** x: 2

2ª iteração

k: 15 a: 38 **saída: 14** x: 3

3ª iteração

k: 10 a: 48 **saída: 21** x: 6

4ª iteração

k: 5 a: 53 **saída: 42** x: 4

5ª iteração

k: 0 a: 53 **saída: 28** x: 4

Não há mais iterações

and now for something completely different

Nomes (funções)

- *Nomes* permitem que pensemos mais sobre o problema a ser resolvido e menos sobre as idiossincrasias do computador
- Variáveis são **expressões** (têm um valor)
- Nomes também podem ser usados para representar “ações” (**como calcular** o valor de uma expressão)
 - ▶ Por exemplo, `sqrt()` é o **nome** que usamos para representar o cálculo da raiz quadrada
- Nomes desse tipo recebem um valor, realizam algum processamento e devolvem outro valor correspondente
 - ▶ Como as funções da matemática
- E, por isso, também são chamados de **funções**

Nomes (funções)

```
library(glue)
v <- as.numeric(readline("Velocidade do vento (Km/h): "))
t <- as.numeric(readline("Temperatura (°C): "))
u <- as.numeric(readline("Umidade relativa (%): ")) / 100
sensação <- t + 2.015 * u * exp(17.27*t/(237.7+t)) - 0.194*v - 4
cat(glue("Sensação térmica: {round(sensação, 1)}°C"), "\n")
```

- Aqui, *sensação* representa o *resultado* da expressão
- Mas podemos representar também o *cálculo* da expressão

```
library(glue)
sensação <- function(v, t, u) {
  return(round(t + 2.015 * u * exp(17.27*t/(237.7+t)) - 0.194*v - 4, 1))
}
v <- as.numeric(readline("Velocidade do vento (Km/h): "))
t <- as.numeric(readline("Temperatura (°C): "))
u <- as.numeric(readline("Umidade relativa (%): ")) / 100
cat(glue("Sensação térmica: {sensação(v, t, u)}°C"), "\n")
```

Nomes (funções)

```
library(glue)
sensação <- function(v, t, u) {
  ➡ resultado = t + 2.015 * u * exp(17.27*t/(237.7+t)) - 0.194*v - 4
  ➡ return(round(resultado, 1))
}
v <- as.numeric(readline("Velocidade do vento (Km/h): "))
t <- as.numeric(readline("Temperatura (°C): "))
u <- as.numeric(readline("Umidade relativa (%): ")) / 100
cat(glue("Sensação térmica: {sensação(v, t, u)}°C"), "\n")
```

- A função define **como** realizar uma operação
- A **chamada** à função é uma expressão (tem um valor)
- O começo e o fim da função são identificados por { e }
- A indentação em R é apenas uma **convenção**
 - ▶ Mas é **muito** importante e comum a todas as linguagens de programação
 - » Em python não é apenas convenção, é obrigatório

Função `main()`

- Vamos nos acostumar a fazer nossos programas dentro de uma função chamada `main()`

```
library(glue)
sensação <- function(v, t, u) {
  resultado = round(t + 2.015 * u * exp(17.27*t/(237.7+t)) - 0.194*v - 4, 1)
  return(resultado)
}
main <- function() {
  v <- as.numeric(readline("Velocidade do vento (Km/h): "))
  t <- as.numeric(readline("Temperatura (°C): "))
  u <- as.numeric(readline("Umidade relativa (%): ")) / 100
  cat(glue("Sensação térmica: {sensação(v, t, u)}°C"), "\n")
}
main()
```

- Isso também é apenas uma convenção, mas vai facilitar a vida mais à frente 😊

Exercício – área do triângulo

A área de um triângulo de lados a, b, c é dada por

$$\sqrt{s \cdot (s - a) \cdot (s - b) \cdot (s - c)}, \text{ onde } s = \frac{a+b+c}{2}$$

Modifique o programa abaixo para obter os valores de a, b, c do usuário e calcular a área usando uma função

```
area <- function(a, b, c) {  
  s <- (a + b + c) / 2  
  return(sqrt(s * (s-a) * (s-b) * (s-c)))  
}  
main <- function() {  
  a <- as.integer(readline("Digite o primeiro lado: "))  
  b <- as.integer(readline("Digite o segundo lado: "))  
  c <- as.integer(readline("Digite o terceiro lado: "))  
  cat("A área do triângulo dado é", area(a, b, c), "\n")  
}  
main()
```

Exercício

Cálculo do número de horas vividas por uma pessoa

Exercício

Cálculo do número de horas vividas por uma pessoa

```
main <- function() {  
  dia <- as.integer(readline("Digite o dia inicial: "))  
  mês <- as.integer(readline("Digite o mês inicial: "))  
  ano <- as.integer(readline("Digite o ano inicial: "))  
  hora <- as.integer(readline("Digite a hora inicial: "))  
  dia_fim <- as.integer(readline("Digite o dia final: "))  
  mês_fim <- as.integer(readline("Digite o mês final: "))  
  ano_fim <- as.integer(readline("Digite o ano final: "))  
  hora_fim <- as.integer(readline("Digite a hora final: "))  
  ...  
  cat("Total de horas vividas:", horas, "\n")  
}  
main()
```

Exercício

Exercício

```
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
```

```
}
```

Exercício

```
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {  
  horas <- horas + 24  
  dia <- dia + 1
```

```
}
```

Exercício

```
horas <-  
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {  
  horas <- horas + 24  
  dia <- dia + 1  
  
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1

}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1

  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }

}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1

  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30

  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if (
  ) {

}

  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if (
    dias_no_mês <- 31
  ) {

    if (dia > dias_no_mês) {
      dia <- 1
      mês <- mês + 1
    }
    if (mês > 12) {
      mês <- 1
      ano <- ano + 1
    }
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) ) {
    dias_no_mês <- 31
  }

  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) || ) {
    dias_no_mês <- 31
  }

  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) || (mês %% 2 == 0 && mês > 7)) {
    dias_no_mês <- 31
  }

  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) || (mês %% 2 == 0 && mês > 7)) {
    dias_no_mês <- 31
  }
  if (mês == 2) {

  }
  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) || (mês %% 2 == 0 && mês > 7)) {
    dias_no_mês <- 31
  }
  if (mês == 2) {
    dias_no_mês <- 28
  }
  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) || (mês %% 2 == 0 && mês > 7)) {
    dias_no_mês <- 31
  }
  if (mês == 2) {
    dias_no_mês <- 28
    if (
    ) {
      dias_no_mês <- 29
    }
  }
  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) || (mês %% 2 == 0 && mês > 7)) {
    dias_no_mês <- 31
  }
  if (mês == 2) {
    dias_no_mês <- 28
    if ((ano %% 4 == 0 && ano %% 100 != 0) || ano %% 400 == 0) {
      dias_no_mês <- 29
    }
  }
  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) || (mês %% 2 == 0 && mês > 7)) {
    dias_no_mês <- 31
  }
  if (mês == 2) {
    dias_no_mês <- 28
    if ((ano %% 4 == 0 && ano %% 100 != 0) || ano %% 400 == 0) {
      dias_no_mês <- 29
    }
  }
  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  dias_no_mês <- 30
  if ((mês %% 2 != 0 && mês <= 7) || (mês %% 2 == 0 && mês > 7)) {
    dias_no_mês <- 31
  }
  if (mês == 2) {
    dias_no_mês <- 28
    if ((ano %% 4 == 0 && ano %% 100 != 0) || ano %% 400 == 0) {
      dias_no_mês <- 29
    }
  }
  if (dia > dias_no_mês) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
```

```
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
```

```
horas <- horas + hora_fim
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
```

```
    if (dia > ndias(mês, ano)) {
      dia <- 1
      mês <- mês + 1
    }
    if (mês > 12) {
      mês <- 1
      ano <- ano + 1
    }
  }
  horas <- horas + hora_fim
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim
```

...

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim
...
ndias <- function(    ) {

}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim
...
ndias <- function(m, a) {

}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim

...
ndias <- function(m, a) {
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7))
    return(31)
  else if (m == 2) {
    if (a %% 4 == 0 && (a %% 100 != 0 || a %% 400 == 0))
      return(29)
    else
      return(28)
  }
  else
    return(30)
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim

...
ndias <- function(m, a) {
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }

}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim

...
ndias <- function(m, a) {
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }
  if (m == 2) {

  }

}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim

...
ndias <- function(m, a) {
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }
  if (m == 2) {

    return (28)
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim

...
ndias <- function(m, a) {
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }
  if (m == 2) {
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0)
      return (28)
  }
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim

...
ndias <- function(m, a) {
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }
  if (m == 2) {
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }
    return (28)
  }
}
```

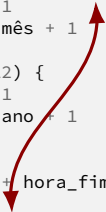
Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim

...
ndias <- function(m, a) {
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }
  if (m == 2) {
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }
    return (28)
  }
  return (30)
}
```

Exercício

```
horas <- -1 * hora
while (dia < dia_fim || mês < mês_fim || ano < ano_fim) {
  horas <- horas + 24
  dia <- dia + 1
  if (dia > ndias(mês, ano)) {
    dia <- 1
    mês <- mês + 1
  }
  if (mês > 12) {
    mês <- 1
    ano <- ano + 1
  }
}
horas <- horas + hora_fim
...
ndias <- function(m, a) {
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }
  if (m == 2) {
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }
    return (28)
  }
  return (30)
}
```



Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida
- Uma das vantagens é que elas podem ser usadas em contextos diferentes para realizar a computação correspondente àquela ideia

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida
- Uma das vantagens é que elas podem ser usadas em contextos diferentes para realizar a computação correspondente àquela ideia
- Portanto...

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida
- Uma das vantagens é que elas podem ser usadas em contextos diferentes para realizar a computação correspondente àquela ideia
- Portanto...
- Elas precisam ser capazes de funcionar de maneira independente do contexto

**Cada um com seu cada um e ninguém se mete
no cada um dos outros**

Nomes e memória

- Em R, só existe quem tem nome:

Nomes e memória

- Em R, só existe quem tem nome:

```
saudação <- "Olá!"
```

```
saudação <- "Oi!"
```

Nomes e memória

- Em R, só existe quem tem nome:

```
saudação <- "olá!"  
saudação <- "oi!"
```

- O texto **"olá!"** não existe mais em nenhum lugar da memória após a segunda linha ser executada!

Nomes e memória

- Em R, só existe quem tem nome:

```
saudação <- "Olá!"  
saudação <- "Oi!"
```

- O texto **"Olá!"** não existe mais em nenhum lugar da memória após a segunda linha ser executada!

```
x <- 1  
x <- x + 1
```

Nomes e memória

- Em R, só existe quem tem nome:

```
saudação <- "Olá!"  
saudação <- "Oi!"
```

- O texto **"Olá!"** não existe mais em nenhum lugar da memória após a segunda linha ser executada!

```
x <- 1  
x <- x + 1
```

- O número **1** não existe mais em nenhum lugar da memória após a segunda linha ser executada!

Nomes e memória

- Em R, só existe quem tem nome:

Nomes e memória

- Em R, só existe quem tem nome:

```
x <- 1  
y <- x  
x <- x + 1
```

Nomes e memória

- Em R, só existe quem tem nome:

```
x <- 1  
y <- x  
x <- x + 1
```

- O número **1** *continua existindo* na memória após a terceira linha ser executada

Nomes e memória

- Em R, só existe quem tem nome:

```
x <- 1  
y <- x  
x <- x + 1
```

- O número **1** *continua existindo* na memória após a terceira linha ser executada
 - ▶ Mas o que acontece na segunda linha?!?!?

Nomes e memória

- Em R, só existe quem tem nome:

```
x <- 1  
y <- x  
x <- x + 1
```

- O número **1** *continua existindo* na memória após a terceira linha ser executada
 - ▶ Mas o que acontece na segunda linha?!?!?
 - » Temos duas “cópias” do número 1 na memória?

Nomes e memória

- Em R, só existe quem tem nome:

```
x <- 1  
y <- x  
x <- x + 1
```

- O número 1 *continua existindo* na memória após a terceira linha ser executada
 - ▶ Mas o que acontece na segunda linha?!?!?
 - » Temos duas “cópias” do número 1 na memória?
- DEPENDE

Nomes e memória

- Em R, só existe quem tem nome:

```
x <- 1  
y <- x  
x <- x + 1
```

- O número **1** *continua existindo* na memória após a terceira linha ser executada
 - ▶ Mas o que acontece na segunda linha?!?!?
 - » Temos duas “cópias” do número 1 na memória?
- **DEPENDE**
 - ▶ Em R, sim!

Nomes e memória

- Em R, só existe quem tem nome:

```
x <- 1  
y <- x  
x <- x + 1
```

- O número **1** *continua existindo* na memória após a terceira linha ser executada
 - ▶ Mas o que acontece na segunda linha?!?!?
 - » Temos duas “cópias” do número 1 na memória?
- **DEPENDE**
 - ▶ Em R, sim!
 - ▶ Na terceira linha, voltamos a ter apenas uma cópia

Nomes e memória

- Em R, só existe quem tem nome:

```
x <- 1  
y <- x  
x <- x + 1
```

- O número 1 *continua existindo* na memória após a terceira linha ser executada
 - ▶ Mas o que acontece na segunda linha?!?!?
 - » Temos duas “cópias” do número 1 na memória?
- DEPENDE
 - ▶ Em R, sim!
 - ▶ Na terceira linha, voltamos a ter apenas uma cópia
 - » (R na verdade evita copiar quando não é necessário, mas do ponto de vista do programador é “como se” sempre houvesse a cópia)

O que são “variáveis”?

O que são “variáveis”?

- Nomes que usamos para acessar dados armazenados em algum lugar na memória do computador?

O que são “variáveis”?

- Nomes que usamos para acessar dados armazenados em algum lugar na memória do computador?
- Conceitos abstratos que representam uma informação relevante para a computação que estamos realizando?

O que são “variáveis”?

- Nomes que usamos para acessar dados armazenados em algum lugar na memória do computador?
- Conceitos abstratos que representam uma informação relevante para a computação que estamos realizando?

Ambos

O que são “variáveis”?

- Nomes que usamos para acessar dados armazenados em algum lugar na memória do computador?
- Conceitos abstratos que representam uma informação relevante para a computação que estamos realizando?

Ambos

- Mas cada linguagem dá mais ênfase a este ou àquele ponto de vista

Variáveis e memória

- **Em python**

- ▶ `x = x + 1` não altera o número 1 armazenado na memória, mas sim o valor associado à variável `x` (o número 2 “nasce” e recebe o nome `x`, enquanto o número 1 é “jogado fora”)

Variáveis e memória

- Em python

- ▶ $x = x + 1$ não altera o número 1 armazenado na memória, mas sim o valor associado à variável x (o número 2 “nasce” e recebe o nome x , enquanto o número 1 é “jogado fora”)
 - » *Pensando em variáveis como conceitos, isso faz sentido: o valor da variável é o conceito, e ele foi modificado; o número armazenado é **outro** número*

Variáveis e memória

- Em python

- ▶ $x = x + 1$ não altera o número 1 armazenado na memória, mas sim o valor associado à variável x (o número 2 “nasce” e recebe o nome x , enquanto o número 1 é “jogado fora”)
 - » *Pensando em variáveis como conceitos, isso faz sentido: o valor da variável é o conceito, e ele foi modificado; o número armazenado é **outro** número*

- Em outras linguagens, como R e C, as coisas são diferentes:

Variáveis e memória

- **Em python**

- ▶ $x = x + 1$ não altera o número 1 armazenado na memória, mas sim o valor associado à variável x (o número 2 “nasce” e recebe o nome x , enquanto o número 1 é “jogado fora”)
 - » *Pensando em variáveis como conceitos, isso faz sentido: o valor da variável é o conceito, e ele foi modificado; o número armazenado é **outro** número*

- **Em outras linguagens, como R e C, as coisas são diferentes:**

- ▶ $x \leftarrow y$ guarda duas “cópias” do mesmo número em lugares diferentes da memória

Variáveis e memória

- **Em python**

- ▶ $x = x + 1$ não altera o número 1 armazenado na memória, mas sim o valor associado à variável x (o número 2 “nasce” e recebe o nome x , enquanto o número 1 é “jogado fora”)
 - » *Pensando em variáveis como conceitos, isso faz sentido: o valor da variável é o conceito, e ele foi modificado; o número armazenado é **outro** número*

- **Em outras linguagens, como R e C, as coisas são diferentes:**

- ▶ $x \leftarrow y$ guarda duas “cópias” do mesmo número em lugares diferentes da memória
- ▶ $x \leftarrow x + 1$ procura o lugar na memória em que a variável x está armazenada e altera o valor que está ali

Variáveis e memória

- **Em python**

- ▶ $x = x + 1$ não altera o número 1 armazenado na memória, mas sim o valor associado à variável x (o número 2 “nasce” e recebe o nome x , enquanto o número 1 é “jogado fora”)
 - » *Pensando em variáveis como conceitos, isso faz sentido: o valor da variável é o conceito, e ele foi modificado; o número armazenado é **outro** número*

- **Em outras linguagens, como R e C, as coisas são diferentes:**

- ▶ $x \leftarrow y$ guarda duas “cópias” do mesmo número em lugares diferentes da memória
- ▶ $x \leftarrow x + 1$ procura o lugar na memória em que a variável x está armazenada e altera o valor que está ali
 - » *Pensando em variáveis como nomes para um dado na memória, isso faz sentido: o conteúdo daquele espaço de memória foi modificado*

E por que eu preciso me preocupar com isso?

E por que eu preciso me preocupar com isso?

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
val <- 10  
cat(metade(val), "é a metade de", val, "\n")
```

E por que eu preciso me preocupar com isso?

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
val <- 10  
cat(metade(val), "é a metade de", val, "\n")
```

- Na chamada a **metade()**, o número **10**, que está armazenado em **val**, é **copiado** para o nome **x**

E por que eu preciso me preocupar com isso?

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
val <- 10  
cat(metade(val), "é a metade de", val, "\n")
```

- Na chamada a **metade()**, o número **10**, que está armazenado em **val**, é **copiado** para o nome **x**
 - ▶ O nome **x** só existe **dentro** da função **metade()**!

E por que eu preciso me preocupar com isso?

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
val <- 10  
cat(metade(val), "é a metade de", val, "\n")
```

- Na chamada a **metade()**, o número **10**, que está armazenado em **val**, é **copiado** para o nome **x**
 - ▶ O nome **x** só existe **dentro** da função **metade()**!
 - ▶ Quando a função termina, o nome **x** e seu valor deixam de existir

E por que eu preciso me preocupar com isso?

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
val <- 10  
cat(metade(val), "é a metade de", val, "\n")
```

- Na chamada a **metade()**, o número **10**, que está armazenado em **val**, é **copiado** para o nome **x**
 - ▶ O nome **x** só existe **dentro** da função **metade()**!
 - ▶ Quando a função termina, o nome **x** e seu valor deixam de existir
 - ▶ Quando o valor de **x** é modificado dentro da função, isso não altera nenhuma variável **fora** da função

E por que eu preciso me preocupar com isso?

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
val <- 10  
cat(metade(val), "é a metade de", val, "\n")
```

- Na chamada a **metade()**, o número **10**, que está armazenado em **val**, é **copiado** para o nome **x**
 - ▶ O nome **x** só existe **dentro** da função **metade()**!
 - ▶ Quando a função termina, o nome **x** e seu valor deixam de existir
 - ▶ Quando o valor de **x** é modificado dentro da função, isso não altera nenhuma variável **fora** da função
 - » *E isso continua valendo mesmo que, “por coincidência”, o nome “de fora” e o nome “de dentro” sejam iguais*

E por que eu preciso me preocupar com isso?

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
x <- 10  
cat(metade(x), "é a metade de", x, "\n")
```

- Na chamada a **metade()**, o número **10**, que está armazenado em **val**, é **copiado** para o nome **x**
 - ▶ O nome **x** só existe **dentro** da função **metade()**!
 - ▶ Quando a função termina, o nome **x** e seu valor deixam de existir
 - ▶ Quando o valor de **x** é modificado dentro da função, isso não altera nenhuma variável **fora** da função
 - » *E isso continua valendo mesmo que, “por coincidência”, o nome “de fora” e o nome “de dentro” sejam iguais*

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida
- Uma das vantagens é que elas podem ser usadas em contextos diferentes para realizar a computação correspondente àquela ideia

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida
- Uma das vantagens é que elas podem ser usadas em contextos diferentes para realizar a computação correspondente àquela ideia
- Portanto...

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida
- Uma das vantagens é que elas podem ser usadas em contextos diferentes para realizar a computação correspondente àquela ideia
- Portanto...
- Elas precisam ser capazes de funcionar de maneira independente do contexto

**Cada um com seu cada um e ninguém se mete
no cada um dos outros**

- Não confunda `cat()` e `return`!

- Não confunda `cat()` e `return`!
 - ▶ `cat()` e `readline()` → comunicação com o “mundo”

- Não confunda **cat()** e **return**!
 - ▶ **cat()** e **readline()** → comunicação com o “mundo”
 - ▶ **return** e os parâmetros das funções → comunicação entre partes diferentes do programa

- Não confunda **cat()** e **return**!

- ▶ **cat()** e **readline()** → comunicação com o “mundo”
- ▶ **return** e os parâmetros das funções → comunicação entre partes diferentes do programa



- Não confunda **cat()** e **return**!

- ▶ **cat()** e **readline()** → comunicação com o “mundo”
- ▶ **return** e os parâmetros das funções → comunicação entre partes diferentes do programa

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
val <- 10  
cat(metade(val), "é a metade de", val, "\n")
```

Funções

- Não confunda **cat()** e **return**!

- ▶ **cat()** e **readline()** → comunicação com o “mundo”
- ▶ **return** e os parâmetros das funções → comunicação entre partes diferentes do programa

```
metade <- function(x) {  
  x <- x / 2  
  return (x)  
}  
val <- 10  
cat(metade(val), "é a metade de", val, "\n")
```

```
metade <- function(x) {  
  y <- x / 2  
  cat(y, "é a metade de", x, "\n")  
}  
metade(10)
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }  
    return (28)  
  }  
  return (30)  
}
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
cat("Fevereiro de 2018 teve", ndias(2, 2018), "dias", "\n")
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
cat("Fevereiro de 2018 teve", ndias(2, 2018), "dias", "\n")  
month <- as.integer(readline("Mês? "))  
year <- as.integer(readline("Ano? "))
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
cat("Fevereiro de 2018 teve", ndias(2, 2018), "dias", "\n")  
month <- as.integer(readline("Mês? "))  
year <- as.integer(readline("Ano? "))  
cat("0 mês", month, "do ano", year, "tem", ndias(month, year), "dias", "\n")
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
cat("Fevereiro de 2018 teve", ndias(2, 2018), "dias", "\n")  
month <- as.integer(readline("Mês? "))  
year <- as.integer(readline("Ano? "))  
cat("0 mês", month, "do ano", year, "tem", ndias(month, year), "dias", "\n")  
m <- as.integer(readline("Mês? "))  
a <- as.integer(readline("Ano? "))
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if ((a %% 4 == 0 && a %% 100 != 0) || a %% 400 == 0) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
cat("Fevereiro de 2018 teve", ndias(2, 2018), "dias", "\n")  
month <- as.integer(readline("Mês? "))  
year <- as.integer(readline("Ano? "))  
cat("0 mês", month, "do ano", year, "tem", ndias(month, year), "dias", "\n")  
m <- as.integer(readline("Mês? "))  
a <- as.integer(readline("Ano? "))  
cat("0 mês", m, "do ano", a, "tem", ndias(m, a), "dias", "\n")
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if (bissexto(a)) { return (29) }  
    return (28)  
  }  
  return (30)  
}
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if (bissexto(a)) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
bissexto <- function( ) {  
  
}
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if (bissexto(a)) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
bissexto <- function(ano) {  
  
}
```

Exercício

```
ndias <- function(m, a) {  
  if ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7)) { return (31) }  
  if (m == 2) {  
    if (bissexto(a)) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
  
bissexto <- function(ano) {  
  return ((ano %% 4 == 0 && ano %% 100 != 0) || ano %% 400 == 0)  
}
```

Exercício

```
ndias <- function(m, a) {  
  if mês_longo(m) { return (31) }  
  if (m == 2) {  
    if (bissexto(a)) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
bissexto <- function(ano) {  
  return ((ano %% 4 == 0 && ano %% 100 != 0) || ano %% 400 == 0)  
}
```

Exercício

```
ndias <- function(m, a) {  
  if mês_longo(m) { return (31) }  
  if (m == 2) {  
    if (bissexto(a)) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
  
bissexto <- function(ano) {  
  return ((ano %% 4 == 0 && ano %% 100 != 0) || ano %% 400 == 0)  
}  
  
mês_longo <- function(m) {  
  return ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7))  
}
```

Exercício

```
main <- function() {  
  dia <- as.integer(readline("Digite o dia inicial: "))  
  mês <- as.integer(readline("Digite o mês inicial: "))  
  ano <- as.integer(readline("Digite o ano inicial: "))  
  hora <- as.integer(readline("Digite a hora inicial: "))  
  dia_fim <- as.integer(readline("Digite o dia final: "))  
  mês_fim <- as.integer(readline("Digite o mês final: "))  
  ano_fim <- as.integer(readline("Digite o ano final: "))  
  hora_fim <- as.integer(readline("Digite a hora final: "))  
  ...  
  cat("Total de horas vividas:", horas, "\n")  
}  
main()
```

Exercício

```
main <- function() {  
  dia <- as.integer(readline("Digite o dia inicial: "))  
  mês <- as.integer(readline("Digite o mês inicial: "))  
  ano <- as.integer(readline("Digite o ano inicial: "))  
  hora <- as.integer(readline("Digite a hora inicial: "))  
  dia_fim <- as.integer(readline("Digite o dia final: "))  
  mês_fim <- as.integer(readline("Digite o mês final: "))  
  ano_fim <- as.integer(readline("Digite o ano final: "))  
  hora_fim <- as.integer(readline("Digite a hora final: "))  
  horas <- calcula_horas(dia, mês, ano, hora, dia_fim, mês_fim, ano_fim, hora_fim)  
  cat("Total de horas vividas:", horas, "\n")  
}  
main()
```

Exercício

```
calcula_horas <- function(dia, mês, ano, hora, dfim, mfim, afim, hfim) {  
  total <- -1 * hora  
  while (dia < dfim || mês < mfim || ano < afim) {  
    total <- total + 24  
    dia <- dia + 1  
    if (dia > ndias(mês, ano)) {  
      dia <- 1  
      mês <- mês + 1  
    }  
    if (mês > 12) {  
      mês <- 1  
      ano <- ano + 1  
    }  
  }  
  return (total + hfim)  
}
```

Exercício

```
ndias <- function(m, a) {  
  if mês_longo(m) { return (31) }  
  if (m == 2) {  
    if (bissexto(a)) { return (29) }  
    return (28)  
  }  
  return (30)  
}  
  
bissexto <- function(ano) {  
  return ((ano %% 4 == 0 && ano %% 100 != 0) || ano %% 400 == 0)  
}  
  
mês_longo <- function(m) {  
  return ((m %% 2 != 0 && m <= 7) || (m %% 2 == 0 && m > 7))  
}
```