MAC 113 — Introdução à Ciência da Computação

Aula 7

Nelson Lago

1°/2025

Previously on MAC 113...

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 e condições

- **Em ambos os casos, "algo" precisa acontecer para indicar que as repetições chegaram ao fim (o objetivo foi atingido ou todos os elementos do conjunto já foram processados)**

  (ok, às vezes queremos repetir indefinidamente, mas vamos ignorar isso por enquanto)

- **As repetições são controladas por algum tipo de *condição* baseada no estado de uma *variável***

```
chute <- readline("Adivinhe qual é minha cor favorita: ")
while (chute != "rosa choque") {
    chute <- readline("Errou! Tente novamente: ")
}
cat("Acertou!", "\n")
```

Repetições e condições

- A **condição** é um **booleano**

```
chute <- readline("Adivinhe qual é minha cor favorita: ")
while (chute != "rosa choque") {
    chute <- readline("Errou! Tente novamente: ")
}
cat("Acertou!", "\n")
```

```
chute <- readline("Adivinhe qual é minha cor favorita: ")
if (chute != "rosa choque") {
    chute <- readline("Errou! Tente novamente: ")
    vaidinovo!!!!
}
cat("Acertou!", "\n")
```

Partes mínimas de um laço

- **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

```
chute <- readline("Adivinhe qual é minha cor favorita: ")
while (chute != "rosa choque") {
    chute <- readline("Errou! Tente novamente: ")
}
cat("Acertou!", "\n")
```

Repetições até atingir um resultado

- **Caso comum:**
  - branco sujo
  - amarelo icterícia
  - roxo hematoma
  - **rosa choque**

*A **variável** muda em todas as iterações, mas a **condição** só muda na última iteração*

- **Caso sortudo:**
  - **rosa choque**

***Nenhuma** iteração, então nem a variável nem a condição mudam*

- **Caso absurdo:**
  - roxo hematoma
  - roxo hematoma
  - roxo hematoma
  - **rosa choque**

*A variável **e** a condição só mudam na última iteração*

A variável sempre precisa mudar (ao menos) na última iteração!

Exercícios

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
main <- function() {









}
main()
```

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))









    cat(fatorial, "\n")
}
main()
```

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))

    while (n > 1) {



    }
    cat(fatorial, "\n")
}
main()
```

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))
    fatorial <- 1
    while (n > 1) {



    }
    cat(fatorial, "\n")
}
main()
```

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))
    fatorial <- 1
    while (n > 1) {

        n <- n - 1
    }
    cat(fatorial, "\n")
}
main()
```

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))
    fatorial <- 1
    while (n > 1) { # Ou será >= 1 ?

        n <- n - 1
    }
    cat(fatorial, "\n")
}
main()
```

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))
    fatorial <- 1
    while (n > 1) { # Ou será >= 1 ?
        fatorial <- fatorial * n
        n <- n - 1
    }
    cat(fatorial, "\n")
}
main()
```

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))
    fatorial <- 1
    while (n > 1) { # Ou será >= 1 ?
        fatorial <- fatorial * n
        n <- n - 1
    }
    cat(fatorial, "\n")
}
main()
```

É mais comum usar o valor da variável recebido
no início do laço e atualizar seu valor no final

(“principle of least surprise”)

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {









}
main()
```

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {









    cat("A soma dos", n, "primeiros inteiros é", soma, "\n")
}
main()
```

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))









    cat("A soma dos", n, "primeiros inteiros é", soma, "\n")
}
main()
```

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))



    while (n > 0) {



    }
    cat("A soma dos", n, "primeiros inteiros é", soma, "\n")
}
main()
```

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))



    while (n > 0) {

        n <- n - 1
    }
    cat("A soma dos", n, "primeiros inteiros é", soma, "\n")
}
main()
```

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))



    while (n > 0) {
        soma <- soma + n
        n <- n - 1
    }
    cat("A soma dos", n, "primeiros inteiros é", soma, "\n")
}
main()
```

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))

    soma <- 0
    while (n > 0) {
        soma <- soma + n
        n <- n - 1
    }
    cat("A soma dos", n, "primeiros inteiros é", soma, "\n")
}
main()
```

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))

    soma <- 0
    while (n > 0) {
        soma <- soma + n
        n <- n - 1
    }
    cat("A soma dos", n, "primeiros inteiros é", soma, "\n")
}
main()
```

Exercício — somatório

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))
    val <- n
    soma <- 0
    while (val > 0) {
        soma <- soma + val
        val <- val - 1
    }
    cat("A soma dos", n, "primeiros inteiros é", soma, "\n")
}
main()
```

Exercício — outro somatório 1/2

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma (repetições até atingir um resultado)

```
main <- function() {









}
main()
```

Exercício — outro somatório 1/2

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma (repetições até atingir um resultado)

```
main <- function() {
    n <- as.integer(readline("Digite um número (zero para sair): "))









    cat("A soma dos números é", soma, "\n"))
}
main()
```

Exercício — outro somatório 1/2

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma (repetições até atingir um resultado)

```
main <- function() {
    n <- as.integer(readline("Digite um número (zero para sair): "))

    while (n != 0) {



    }
    cat("A soma dos números é", soma, "\n"))
}
main()
```

Exercício — outro somatório 1/2

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma (repetições até atingir um resultado)

```
main <- function() {
    n <- as.integer(readline("Digite um número (zero para sair): "))

    while (n != 0) {

        n <- as.integer(readline("Digite um número (zero para sair): "))
    }
    cat("A soma dos números é", soma, "\n"))
}
main()
```

Exercício — outro somatório 1/2

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma (repetições até atingir um resultado)

```
main <- function() {
    n <- as.integer(readline("Digite um número (zero para sair): "))

    while (n != 0) {
        soma <- soma + n
        n <- as.integer(readline("Digite um número (zero para sair): "))
    }
    cat("A soma dos números é", soma, "\n"))
}
main()
```

Exercício — outro somatório 1/2

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma (repetições até atingir um resultado)

```
main <- function() {
    n <- as.integer(readline("Digite um número (zero para sair): "))
    soma <- 0
    while (n != 0) {
        soma <- soma + n
        n <- as.integer(readline("Digite um número (zero para sair): "))
    }
    cat("A soma dos números é", soma, "\n"))
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {









}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {









    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {
    pares <- 0









    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {
    pares <- 0 # elemento neutro









    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {
    pares <- 0 # elemento neutro
    n <- as.integer(readline("Digite um número (zero para sair): "))









    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {
    pares <- 0 # elemento neutro
    n <- as.integer(readline("Digite um número (zero para sair): "))
    while (n != 0) {









    }
    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {
    pares <- 0 # elemento neutro
    n <- as.integer(readline("Digite um número (zero para sair): "))
    while (n != 0) {




        n <- as.integer(readline("Digite um número (zero para sair): "))
    }
    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {
    pares <- 0 # elemento neutro
    n <- as.integer(readline("Digite um número (zero para sair): "))
    while (n != 0) {
        if (n %% 2 == 0) {

        }
        n <- as.integer(readline("Digite um número (zero para sair): "))
    }
    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {
    pares <- 0 # elemento neutro
    n <- as.integer(readline("Digite um número (zero para sair): "))
    while (n != 0) {
        if (n %% 2 == 0) {
            pares <- pares + 1
        }
        n <- as.integer(readline("Digite um número (zero para sair): "))
    }
    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Exercício — contagem de pares

Leia uma série de números terminada por zero fornecida pelo usuário e diga quantos deles são pares (repetições até atingir um resultado)

```
main <- function() {
    pares <- 0
    n <- 1 # 1 é ímpar :)
    while (n != 0) {
        if (n %% 2 == 0) {
            pares <- pares + 1
        }
        n <- as.integer(readline("Digite um número (zero para sair): "))
    }
    cat("Você digitou", pares, "números pares", "\n")
}
main()
```

Rápido como uma tartaruga

A biblioteca **TurtleGraphics** permite desenhar figuras simples, mas infelizmente tem problemas de compatibilidade com o colab. É possível contornar esse problema com isto:

```
install.packages("TurtleGraphics")
library("TurtleGraphics")
fakedraw <- function() {invisible(NULL)}
environment(fakedraw) <- asNamespace("TurtleGraphics")
assignInNamespace(".turtle_draw",fakedraw, ns="TurtleGraphics")
assignInNamespace(".turtle_undraw",fakedraw, ns="TurtleGraphics")
```

Rápido como uma tartaruga

Podemos usar a biblioteca **TurtleGraphics** para desenhar um quadrado assim:

Rápido como uma tartaruga

Podemos usar a biblioteca **TurtleGraphics** para desenhar um quadrado assim:

```
main <- function() {









}
main()
```

Rápido como uma tartaruga

Podemos usar a biblioteca **TurtleGraphics** para desenhar um quadrado assim:

```
main <- function() {
    turtle_init(300,300)









}
main()
```

Rápido como uma tartaruga

Podemos usar a biblioteca **TurtleGraphics** para desenhar um quadrado assim:

```
main <- function() {
    turtle_init(300,300)
    turtle_forward(100)









}
main()
```

Rápido como uma tartaruga

Podemos usar a biblioteca **TurtleGraphics** para desenhar um quadrado assim:

```
main <- function() {
    turtle_init(300,300)
    turtle_forward(100)
    turtle_right(90)









}
main()
```

Rápido como uma tartaruga

Podemos usar a biblioteca **TurtleGraphics** para desenhar um quadrado assim:

```
main <- function() {
    turtle_init(300,300)
    turtle_forward(100)
    turtle_right(90)
    turtle_forward(100)
    turtle_right(90)



}
main()
```

Rápido como uma tartaruga

Podemos usar a biblioteca **TurtleGraphics** para desenhar um quadrado assim:

```
main <- function() {
    turtle_init(300,300)
    turtle_forward(100)
    turtle_right(90)
    turtle_forward(100)
    turtle_right(90)
    turtle_forward(100)
    turtle_right(90)
    turtle_forward(100)
}
main()
```

Exercício — desenhando um quadrado

Modifique o programa anterior para desenhar um quadrado usando um laço

Exercício — desenhando um quadrado

Modifique o programa anterior para desenhar um quadrado usando um laço

```
main <- function() {
    turtle_init(300,300)









}
main()
```

Exercício — desenhando um quadrado

Modifique o programa anterior para desenhar um quadrado usando um laço

```
main <- function() {
    turtle_init(300,300)

    while (i <= 4) {




    }
}
main()
```

Exercício — desenhando um quadrado

Modifique o programa anterior para desenhar um quadrado usando um laço

```
main <- function() {
    turtle_init(300,300)
    i <- 1
    while (i <= 4) {



        i <- i + 1
    }
}
main()
```

Exercício — desenhando um quadrado

Modifique o programa anterior para desenhar um quadrado usando um laço

```
main <- function() {
    turtle_init(300,300)
    i <- 1
    while (i <= 4) {
        turtle_forward(100)

        i <- i + 1
    }
}
main()
```

Exercício — desenhando um quadrado

Modifique o programa anterior para desenhar um quadrado usando um laço

```
main <- function() {
    turtle_init(300,300)
    i <- 1
    while (i <= 4) {
        turtle_forward(100)
        turtle_right(90)
        i <- i + 1
    }
}
main()
```

Exercício — desenhando um círculo

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe um círculo

```
main <- function() {
    turtle_init(300,300)









}
main()
```

Exercício — desenhando um círculo

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe um círculo

```
main <- function() {
    turtle_init(300,300)

    while (i <= 360) {




    }
}
main()
```

Exercício — desenhando um círculo

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe um círculo

```
main <- function() {
    turtle_init(300,300)
    i <- 1
    while (i <= 360) {



        i <- i + 1
    }
}
main()
```

Exercício — desenhando um círculo

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe um círculo

```
main <- function() {
    turtle_init(300,300)
    i <- 1
    while (i <= 360) {
        turtle_forward(1)

        i <- i + 1
    }
}
main()
```

Exercício — desenhando um círculo

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe um círculo

```
main <- function() {
    turtle_init(300,300)
    i <- 1
    while (i <= 360) {
        turtle_forward(1)
        turtle_right(1)
        i <- i + 1
    }
}
main()
```

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{360^\circ}{5} = 72^\circ$

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{360^\circ}{5} = 72^\circ$

```
main <- function() {
    turtle_init(300,300)









}
main()
```

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{360^\circ}{5} = 72^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)









}
main()
```

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{360^\circ}{5} = 72^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {




}
main()
```

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{360^\circ}{5} = 72^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {
        turtle_forward(100)




}
main()
```

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{360^\circ}{5} = 72^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {
        turtle_forward(100)
        turtle_right(72)



}
main()
```

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{360^\circ}{5} = 72^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {
        turtle_forward(100)
        turtle_right(72)
        i <- i + 1
    }
}
main()
```

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: ☆

- $\frac{360^\circ}{5} = 72^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {
        turtle_forward(100)
        turtle_right(72)
        i <- i + 1
    }
}
main()
```

Exercício — desenhando uma estrela 1/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: ☆

- $\frac{360^\circ}{5} = 72^\circ$

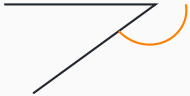
```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {
        turtle_forward(100)
        turtle_right(72)
        i <- i + 1
    }
}
main()
```

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3



Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 2/3

Exercício — desenhando uma estrela 3/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

```
main <- function() {
    turtle_init(300,300)









}
main()
```

Exercício — desenhando uma estrela 3/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{2 \cdot 360^\circ}{5} = \frac{720^\circ}{5} = 144^\circ$

```
main <- function() {
    turtle_init(300,300)









}
main()
```

Exercício — desenhando uma estrela 3/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{2 \cdot 360^\circ}{5} = \frac{720^\circ}{5} = 144^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)









}
main()
```

Exercício — desenhando uma estrela 3/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{2 \cdot 360^\circ}{5} = \frac{720^\circ}{5} = 144^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1









}
main()
```

Exercício — desenhando uma estrela 3/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{2 \cdot 360^\circ}{5} = \frac{720^\circ}{5} = 144^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {




}
main()
```

Exercício — desenhando uma estrela 3/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{2 \cdot 360^\circ}{5} = \frac{720^\circ}{5} = 144^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {
        turtle_forward(100)




}
main()
```

Exercício — desenhando uma estrela 3/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: 

- $\frac{2 \cdot 360^\circ}{5} = \frac{720^\circ}{5} = 144^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {
        turtle_forward(100)
        turtle_right(144)



}
main()
```

Exercício — desenhando uma estrela 3/3

Usando os comandos vistos da biblioteca **TurtleGraphics**, desenhe uma estrela de cinco pontas: ☆

- $\frac{2 \cdot 360^\circ}{5} = \frac{720^\circ}{5} = 144^\circ$

```
main <- function() {
    turtle_init(300,300)
    turtle_right(90)
    i <- 1
    while (i <= 5) {
        turtle_forward(100)
        turtle_right(144)
        i <- i + 1
    }
}
main()
```