MAC 113 — Introdução à Ciência da Computação

Aula 6

Nelson Lago

1°/2025

Previously on MAC 113...

Execução condicional

```r
n <- as.integer(readline("Digite um número natural: "))
if (n %% 2 == 0) {
    cat("O número", n, "é par!", "\n")
}
cat("Ahazei!", "\n")
```

- **`n %% 2 == 0` → TRUE ou FALSE**
  - Embora possamos ler "se condição", na verdade R faz "se o valor da expressão é verdadeiro (**TRUE**)"
  - É **como se** R executasse `if (condição == TRUE)`
- **O bloco a ser executado ou não é definido por { e }**
  - Novamente, a indentação é uma convenção, mas na prática é essencial

Execução condicional – `else`

```r
n <- as.integer(readline("Digite um número natural: "))
if (n %% 2 == 0) {
    cat("O número", n, "é par!", "\n")
} else {
    cat("O número", n, "é ímpar!", "\n")
}
cat("Ahazei!", "\n")
```

- **Há dois "lados" do condicional**
  - Sem uma variável ("n"), o programa sempre executaria o mesmo "lado"
- **O estado da variável só importa no momento do teste**
  - Se ela mudar em seguida, não afeta o condicional
- **Com `else`, os "lados" são mutuamente excludentes**
  - **Um** e **apenas um** deles é executado

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```r
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
if (                                                    )
    { cat("Verão!", "\n") }
if (                                                      )
    { cat("Outono!", "\n") }
if (                                                      )
    { cat("Inverno!", "\n") }
if (                                                        )
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
if (mês < 3                                                    )
    { cat("Verão!", "\n") }
if (                                                             )
    { cat("Outono!", "\n") }
if (                                                             )
    { cat("Inverno!", "\n") }
if (                                                               )
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
if (mês < 3 || mês == 3 && dia <= 20                          )
    { cat("Verão!", "\n") }
if (                                                            )
    { cat("Outono!", "\n") }
if (                                                            )
    { cat("Inverno!", "\n") }
if (                                                              )
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
if (mês < 3 || mês == 3 && dia <= 20 || mês == 12 && dia >= 21)
    { cat("Verão!", "\n") }
if (                                                    )
    { cat("Outono!", "\n") }
if (                                                    )
    { cat("Inverno!", "\n") }
if (                                                      )
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
if (mês < 3 || mês == 3 && dia <= 20 || mês == 12 && dia >= 21)
    { cat("Verão!", "\n") }
if (mês > 3 && mês < 6 || mês == 3 && dia >= 21 || mês == 6 && dia <= 20)
    { cat("Outono!", "\n") }
if (                                                              )
    { cat("Inverno!", "\n") }
if (                                                                )
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
if (mês < 3 || mês == 3 && dia <= 20 || mês == 12 && dia >= 21)
    { cat("Verão!", "\n") }
if (mês > 3 && mês < 6 || mês == 3 && dia >= 21 || mês == 6 && dia <= 20)
    { cat("Outono!", "\n") }
if (mês > 6 && mês < 9 || mês == 6 && dia >= 21 || mês == 9 && dia <= 22)
    { cat("Inverno!", "\n") }
if (                                                                  )
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
if (mês < 3 || mês == 3 && dia <= 20 || mês == 12 && dia >= 21)
    { cat("Verão!", "\n") }
if (mês > 3 && mês < 6 || mês == 3 && dia >= 21 || mês == 6 && dia <= 20)
    { cat("Outono!", "\n") }
if (mês > 6 && mês < 9 || mês == 6 && dia >= 21 || mês == 9 && dia <= 22)
    { cat("Inverno!", "\n") }
if (mês > 9 && mês < 12 || mês == 9 && dia >= 23 || mês == 12 && dia <= 20)
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```r
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
dia_no_ano <- (mês-1) * 30 + dia
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
dia_no_ano <- (mês-1) * 30 + dia
if (dia_no_ano <= 80 || dia_no_ano >= 351)
    { cat("Verão!", "\n") }
if (dia_no_ano > 80 && dia_no_ano <= 170)
    { cat("Outono!", "\n") }
if (dia_no_ano > 170 && dia_no_ano <= 262)
    { cat("Inverno!", "\n") }
if (dia_no_ano > 262 && dia_no_ano <= 350)
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
dia_no_ano <- (mês-1) * 30 + dia
if (dia_no_ano <= 80 || dia_no_ano >= 351)
    { cat("Verão!", "\n") }
if (dia_no_ano > 80 && dia_no_ano <= 170)
    { cat("Outono!", "\n") }
if (dia_no_ano > 170 && dia_no_ano <= 262)
    { cat("Inverno!", "\n") }
if (dia_no_ano > 262 && dia_no_ano <= 350)
    { cat("Primavera!", "\n") }
```

Exercício — estações do ano

Dada uma data (dia e mês), informe se a data acontece no verão (21 de dezembro a 20 de março), outono (21 de março a 20 de junho), inverno (21 de junho a 22 de setembro) ou primavera (23 de setembro a 20 de dezembro). Use números de 1 a 12 para representar os meses do ano.

```
dia <- as.integer(readline("Digite o dia: "))
mês <- as.integer(readline("Digite o mês (1 a 12): "))
dia_no_ano <- (mês-1) * 30 + dia
if (dia_no_ano <= 80 || dia_no_ano >= 351)
    { cat("Verão!", "\n") }
if (dia_no_ano > 80 && dia_no_ano <= 170)
    { cat("Outono!", "\n") }
if (dia_no_ano > 170 && dia_no_ano <= 262)
    { cat("Inverno!", "\n") }
if (dia_no_ano > 262 && dia_no_ano <= 350)
    { cat("Primavera!", "\n") }
```

Truque sujo!

(funciona, mas...)

and now for something completely different

Repetições

Por que computação?

O computador é extremamente rápido, mas

- **É uma ferramenta com o mesmo nível de "inteligência" que um martelo**
  - Tudo tem que ser esmiuçado nos mínimos detalhes
    - » *"Vá à padaria e compre três pães"*
    - » *"Localize esta palavra no texto"*
    - » *...*

Não é mais fácil fazer manualmente?

Por que computação?

Às vezes, sim 🙂 mas:

- **Sistemas de controle**
- **Comunicação**
- **...**
- **Repetições**

Repetições “externas” e “internas”

- **Algumas repetições são externas ao programa**
  - Calculadora (o usuário faz inúmeros cálculos)
  - Jogo (cada partida é uma “repetição”)
  - ...
- **Mas, em geral, qualquer programa não-trivial vai realizar repetições internamente**
  - Xadrez (cada jogada é uma repetição)
  - Procurar uma palavra em um texto (várias comparações)
  - Apresentar uma foto na tela (cada pixel precisa ser “pintado” com a cor adequada)
  - ...

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)**

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)

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***

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**

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")
```

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: ")

}
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

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*
     - » *...*

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 até atingir um resultado

```
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

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

- **A *condição* precisa mudar ao menos na última iteração!**
  - A *condição* testada é “a pessoa chutou `rosa choque`”

Repetições até atingir um resultado

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

- **A *condição* precisa mudar ao menos na última iteração!**
  - A *condição* testada é "a pessoa chutou `rosa choque`"
- **Sem variável, o programa nunca pararia de repetir**
  - (aqui, a variável é "`chute`")

Repetições até atingir um resultado

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

- **A *condição* precisa mudar ao menos na última iteração!**
  - A *condição* testada é "a pessoa chutou `rosa choque`"
- **Sem variável, o programa nunca pararia de repetir**
  - (aqui, a variável é "`chute`")
    - » *(Onde está "* `== TRUE` *"?)*

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

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**

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*

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**

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*

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**

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*

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!

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*
     - » *...*

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 sobre os elementos de um conjunto

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

Repetições sobre os elementos de um conjunto

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

- **O *conjunto* são os números 1–10 (às vezes a ordem importa!)**

Repetições sobre os elementos de um conjunto

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

- **O *conjunto* são os números 1–10 (às vezes a ordem importa!)**
- **A *condição* precisa mudar ao menos na última iteração!**
  - (indicando que todos os elementos do conjunto foram processados)

Repetições sobre os elementos de um conjunto

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

- **O *conjunto* são os números 1–10 (às vezes a ordem importa!)**
- **A *condição* precisa mudar ao menos na última iteração!**
  - (indicando que todos os elementos do conjunto foram processados)
    - » *A condição testada é "*n *é maior que zero"*

Repetições sobre os elementos de um conjunto

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

- **O *conjunto* são os números 1–10 (às vezes a ordem importa!)**
- **A *condição* precisa mudar ao menos na última iteração!**
  - (indicando que todos os elementos do conjunto foram processados)
    - » *A condição testada é "*`n` *é maior que zero"*
- **Sem variável, o programa nunca pararia de repetir**
  - (aqui, a variável é "n")

Repetições sobre os elementos de um conjunto

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

- **O *conjunto* são os números 1–10 (às vezes a ordem importa!)**
- **A *condição* precisa mudar ao menos na última iteração!**
  - (indicando que todos os elementos do conjunto foram processados)
    - » *A condição testada é "*`n` *é maior que zero"*
- **Sem variável, o programa nunca pararia de repetir**
  - (aqui, a variável é "n")
    - » *(Onde está "* `== TRUE` *"?)*

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

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

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

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

```
main <- function() {









}
main()
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

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









}
main()
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

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









}
main()
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

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




}
main()
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

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




}
main()
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

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




}
main()
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

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



}
main()
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

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

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))
    fatorial <- n
    while (n > 2) {
        n <- n - 1
        fatorial <- fatorial * n
    }
    cat(fatorial, "\n")
}
main()
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

```
main <- function() {
    n <- as.integer(readline("Digite um inteiro positivo: "))
    fatorial <- n
    while (n > 2) { # Número mágico
        n <- n - 1
        fatorial <- fatorial * n
    }
    cat(fatorial, "\n")
}
main()
```

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: "))









}
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









}
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) {




}
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 ?




}
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




}
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



}
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)

```r
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: "))

    soma <- 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: "))

    soma <- 0
    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: "))

    soma <- 0
    while (n > 0) {
        soma <- soma + n

    }
    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)

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









}
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









}
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) {




}
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




}
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): "))



}
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()
```