# Etapa 5 — Estruturas de Repetição (Loops)

## Trilha: Lógica de Programação

## Introdução

As **estruturas de repetição (loops)** são fundamentais na lógica de programação.
Elas permitem que **um mesmo conjunto de comandos seja executado várias vezes**, com base em uma **condição lógica** ou em um **intervalo definido**.

Sem elas, teríamos que repetir manualmente trechos de código, o que tornaria os algoritmos longos e difíceis de manter.

## Tipos de Estruturas de Repetição

Existem três tipos principais de estruturas de repetição:

1. **Enquanto (While Loop)**
   → Executa enquanto uma condição for **verdadeira**.

2. **Repita...até (Do...Until Loop)**
   → Executa até que uma condição se torne **verdadeira** (ou seja, testa no final).

3. **Para (For Loop)**
   → Executa um número **fixo de vezes**, controlado por um contador.

Cada tipo tem uma **situação ideal de uso**, como veremos a seguir.

## Estrutura “Enquanto” (While Loop)

### Funcionamento

A estrutura **“enquanto”** verifica **a condição antes** de executar o bloco de comandos.
Se a condição for **falsa logo no início**, o bloco **não é executado nenhuma vez**.

### Sintaxe:

```
enquanto (condicao) faca

  // ações repetidas
```

fimenquanto

### Exemplo:

```
inicio
  contador ← 1
  enquanto contador <= 5 faca
     escreva contador
     contador ← contador + 1
  fimenquanto
fim
```

### Saída:

1

2

3

4

5

### Uso comum:

- Leitura de dados até que o usuário digite um valor de parada (como “0”).
- Processamento de informações enquanto uma condição se mantiver ativa.

## Estrutura “Repita...até” (Do...Until Loop)

### Funcionamento

O **teste da condição é feito no final** da execução.
Por isso, o bloco **é executado pelo menos uma vez**, mesmo que a condição seja falsa desde o início.

## Sintaxe:

```
repita
   // ações
ate (condicao)
```

## Exemplo:

```
inicio
  contador ← 1
  repita
     escreva contador
     contador ← contador + 1
  ate (contador > 5)
fim
```

## Saída:

1

2

3

4

5

## Uso comum:

- Exibir menus que devem aparecer **ao menos uma vez** antes da verificação da saída.

- Coletar dados de um usuário, repetindo enquanto uma condição não for atingida.

# Estrutura “Para” (For Loop)

## Funcionamento

A estrutura **“para”** é usada quando sabemos **exatamente quantas vezes** o loop precisa se repetir.
O contador é inicializado, incrementado automaticamente e testado a cada iteração.

## Sintaxe:

```
para contador de inicio ate fim faca
  // ações
fimpara
```

## Exemplo:

```
inicio
  para i de 1 ate 5 faca
    escreva "Valor:", i
  fimpara
fim
```

## Saída:

```
Valor: 1
Valor: 2
Valor: 3
Valor: 4
```

Valor: 5

### Uso comum:

- Contagens definidas (como gerar uma tabuada ou somar números).
- Processar listas e vetores com tamanho conhecido.

## Controle de Incremento (Step)

Em muitos casos, queremos **controlar o passo do contador** — ou seja, quanto ele aumenta ou diminui a cada iteração.

### Exemplo com passo crescente:

para i de 1 ate 10 passo 2 faca

  escreva i

fimpara

Saída: `1, 3, 5, 7, 9`

### Exemplo com passo decrescente:

para i de 10 ate 1 passo -1 faca

  escreva i

fimpara

Saída: `10, 9, 8, 7, ... 1`

## Interrupção de Repetição (Break)

Em alguns algoritmos, é possível **encerrar o loop antes que termine naturalmente** usando o comando **“saia” (break)**.

### Exemplo:

```
inicio
  para i de 1 ate 10 faca
    se i = 5 entao
      saia
    fimse
    escreva i
  fimpara
fim
```

**Saída:**

1

2

3

4

**Uso comum:**

- Interromper uma busca quando um elemento é encontrado.
- Parar a execução diante de uma condição especial.

# Erros Comuns em Loops

1. ❌ Esquecer de atualizar a variável de controle (em `enquanto`).
2. ❌ Criar **loops infinitos** (condição nunca se torna falsa).
3. ❌ Escrever condições incorretas com operadores lógicos.

Sempre garanta que a condição **se tornará falsa em algum momento**.
Teste os loops com poucos valores antes de aplicá-los em casos reais.

# Exemplo Prático — Tabuada

```
algoritmo "tabuada"
inicio
  leia numero
  para i de 1 ate 10 faca
     resultado ← numero * i
     escreva numero, "x", i, "=", resultado
  fimpara
fim
```

**Saída (para número = 3):**

3 x 1 = 3

3 x 2 = 6

3 x 3 = 9

...

3 x 10 = 30

# Dicas de Lógica

| Situação | Estrutura Ideal |
| --- | --- |
| Número de repetições fixo | `para` |
| Condição depende de um valor | `enquanto` |

| | |
|---|---|
| Execução mínima de 1 vez | `repita...ate` |

# Conclusão

As **estruturas de repetição** são uma das partes mais importantes da lógica de programação.
Com elas, é possível **automatizar tarefas**, **reaproveitar código** e **otimizar a execução** de algoritmos.

Dominar o uso de `para`, `enquanto` e `repita...ate` é essencial antes de avançar para estruturas de dados como **vetores e matrizes**.

# Perguntas de Múltipla Escolha

**1. Qual das estruturas testa a condição *após* executar o bloco?**

a) enquanto
b) para
c) repita...ate
d) saia

**2. Qual é a estrutura mais indicada quando sabemos o número de repetições?**

a) enquanto
b) repita...ate
c) para
d) escolha

**3. O que acontece se a condição de um “enquanto” nunca for falsa?**

a) O programa termina automaticamente
b) O loop nunca executa
c) O programa entra em loop infinito
d) O contador é resetado

**4. Qual comando é usado para sair de um loop antes do final?**

a) pare
b) saia
c) continue
d) encerre

**5. Em qual estrutura o bloco é sempre executado pelo menos uma vez?**

a) enquanto
b) para
c) repita...ate
d) se...entao

## Gabarito

1. c
2. c
3. c
4. b
5. c