

# Construções de repetição: for, do (e while).

David Déharbe



## **Roteiro da aula**

- **Exercícios**
- **A construção `for`;**
- **A construção `do`;**
- **Exercícios.**

## Exercícios

- Escrever um programa que lê um número inteiro, digamos  $x$ , um número natural, digamos  $n$ , e imprime uma mensagem de erro quando  $n = x = 0$  e, caso contrário imprime  $x^n$  ( $x$  elevado à potência  $n$ ).
- Escrever um programa que lê dois números inteiros, digamos  $q$  e  $r$ , um número natural, digamos  $n$ , e imprime os  $n$  primeiros termos da progressão aritmética de razão  $r$  e de valor inicial  $q$ .
- Escrever um programa que lê dois números inteiros, digamos  $q$  e  $r$ , um número natural, digamos  $n$ , e imprime os  $n$  primeiros termos da progressão geométrica de razão  $r$  e de valor inicial  $q$ .
- Escrever um programa que lê um número natural, digamos  $n$ , e lê  $n$  números, digamos  $v_1 \dots v_n$ , e imprime a soma  $v_1 + \dots + v_n$ .

## A construção `for`: motivação

- Para programar uma repetição por um determinado número de vezes, usamos:

- o seguinte padrão:

```
int i; // variável contador
i = 1;
while (i <= n) {
    /* processamento desejado */
    i = i + 1;
}
```

- Esse padrão se repete tanto que há uma construção que permite expressar ele de forma mais sucinta: `for`.

## A construção for: exemplo

- O padrão com while:

```
int i; // variável contador
i = 1;
while (i <= n) {
    /* processamento desejado */
    i = i + 1;
}
```

- O padrão com for:

```
int i;
for (i = 1; i <= n; i = i + 1)
    /* processamento desejado */
```

## A construção for: exemplo

- O padrão com while:

```
int i; // variável contador
i = 1;
while (i <= n) {
    /* processamento desejado */
    i = i + 1;
}
```

- O padrão com for:

```
int i;
for (i = 1; i <= n; i = i + 1)
    /* processamento desejado */
```

Inicialização

## A construção for: exemplo

- O padrão com while:

```
int i; // variável contador
i = 1;
while (i <= n) {
    /* processamento desejado */
    i = i + 1;
}
```

- O padrão com for:

```
int i;
for (i = 1; i <= n; i = i + 1)
    /* processamento desejado */
```

Inicialização

Guarda

## A construção for: exemplo

- O padrão com while:

```
int i; // variável contador
i = 1;
while (i <= n) {
    /* processamento desejado */
    i = i + 1;
}
```

- O padrão com for:

```
int i;
for (i = 1; i <= n; i = i + 1)
    /* processamento desejado */
```

Inicialização

Guarda

Incremento

## A construção for: exemplo

- O padrão com while:

```
int i; // variável contador
i = 1;
while (i <= n) {
    /* processamento desejado */
    i = i + 1;
}
```

- O padrão com for:

```
int i;
for (i = 1; i <= n; i = i + 1)
    /* processamento desejado */
```

Inicialização

Guarda

Incremento

Corpo

## A construção for: sintaxe

- <comando for> ::=  
**for ( <expressão> ; <expressão> ; <expressão> )**  
**<comando>**

Exemplo:

```
int i;  
for (i = 1; i <= n; i = i + 1) {  
    printf("%i\n", i);  
}
```

## A construção for: sintaxe

• <comando for> ::=

for ( <expressão> ; <expressão> ; <expressão> )  
<comando>

Inicialização

Exemplo:

```
int i;  
for (i = 1; i <= n; i = i + 1) {  
    printf("%i\n", i);  
}
```

## A construção for: sintaxe

- <comando for> ::=  
for ( <expressão> ; <expressão> ; <expressão> )  
    <comando>

Inicialização

Guarda

Exemplo:

```
int i;  
for (i = 1; i <= n; i = i + 1) {  
    printf("%i\n", i);  
}
```

## A construção for: sintaxe

- <comando for> ::=  
`for ( <expressão> ; <expressão> ; <expressão> )`  
`<comando>`

Inicialização

Guarda

Incremento

Exemplo:

```
int i;
for (i = 1; i <= n; i = i + 1) {
    printf("%i\n", i);
}
```

## A construção for: sintaxe

• <comando for> ::=

**for** (**<expressão>** ; **<expressão>** ; **<expressão>** )  
**<comando>**

Inicialização

Guarda

Incremento

Corpo

Exemplo:

```
int i;
for (i = 1; i <= n; i = i + 1) {
    printf("%i\n", i);
}
```

## A construção for: semântica

- Execução do comando for:

for ( **e1** ; **e2** ; **e3** )

**c**

1. A inicialização **e1** é executada;
2. A guarda **e2** é executada,
  - 2.1. se o resultado é 0, termina a execução do comando for
  - 2.2. se o resultado não é 0:
    - 2.2.1. o corpo **c** é executado
    - 2.2.2. o incremento **e3** é executado
    - 2.2.3. continua a execução a partir do ponto 2

# Exemplo 1

```
#include <stdio.h>

int main (void)
{
    int n;
    int i;
    scanf ("%i", &n);
    i = 1;
    for (i = 1; i <= n; i = i + 1)
        printf ("dentro> %i\n", i);
    printf ("fora> %i\n", i);
    return 0;
}
```

## A construção `for`: exercício

- Escreva um programa que lê um número inteiro positivo, digamos  $n$ , calcule–com o comando `for`–e imprime  $n!$  ( $n$  fatorial).

## Exemplo 2

```
#include <stdio.h>

int main (void)
{
    int n;
    int i;
    int f;
    scanf ("%i", &n);
    i = 1;
    f = 1;
    for (i = 2; i <= n; i = i + 1)
        f = f * i;
    printf ("%i\n", f);
    return 0;
}
```

## A construção `for`: exercício

- A sequência de Fibonacci inicia com os números 0 e 1, e prossegue com a soma dos últimos dois números:
  - 0, 1, 1, 2, 3, 5, 8,...
- Escreva um programa que lê um número, digamos  $n$ , tal que  $n \geq 2$ , calcule e imprime—usando o comando `for`— os  $n$  primeiros elementos da sequência de Fibonacci.

## Exemplo 3

```
#include <stdio.h>

int main (void)
{
    int n, i, p, u;
    scanf("%i", &n);
    i = 1;
    p = 0;
    u = 1;
    for (i = 2; i <= n; i = i + 1) {
        int novo;
        novo = p + u;
        p = u;
    }
    printf("%i\n", f);
    return 0;
}
```

## A construção do: exemplo

- A construção **do** é usada quando sabemos que o processamento deve ser executado uma vez pelo menos.

```
int n; // variável armazenará valor lido
do
    scanf("%i", &n);
while (n < 1);
```

## A construção do: exemplo

- A construção do é usada quando sabemos que o processamento deve ser executado uma vez pelo menos.

```
int n; // variável armazenará valor lido
do
    scanf("%i", &n);
while (n < 1);
```

Corpo

## A construção do: exemplo

- A construção do é usada quando sabemos que o processamento deve ser executado uma vez pelo menos.

```
int n; // variável armazenará valor lido
do
    scanf("%i", &n);
while (n < 1);
```

Corpo  
Guarda

## A construção do: sintaxe

- `<comando do> ::=`  
do  
`<comando>`  
while ( `<expressão >` ) ;

Exemplo:

```
int n; /* variável armazenará valor lido */
int tentativas = 0; /* quantidade de leituras */
do {
    tentativas = tentativas + 1;
    scanf("%i", &n);
} while (n < 1);
```

## A construção do: sintaxe

• <comando do> ::=

do

<comando>

while ( <expressão > ) ;

Corpo

Exemplo:

```
int n; /* variável armazenará valor lido */
int tentativas = 0; /* quantidade de leituras */
do {
    tentativas = tentativas + 1;
    scanf("%i", &n);
} while (n < 1);
```

## A construção do: sintaxe

• <comando do> ::=

do

<comando>

while ( <expressão > ) ;

Corpo  
Guarda

Exemplo:

```
int n; /* variável armazenará valor lido */
int tentativas = 0; /* quantidade de leituras */
do {
    tentativas = tentativas + 1;
    scanf("%i", &n);
} while (n < 1);
```

## A construção do: semântica

- Execução do comando for:

do

**c**

while (e);

- 1.O corpo **c** é executado;
- 2.A guarda **e** é executada,
  - 2.1.se o resultado é 0, termina a execução do comando do
  - 2.2.se o resultado não é 0, continua a execução a partir do ponto 1.

## Comando do: exercício

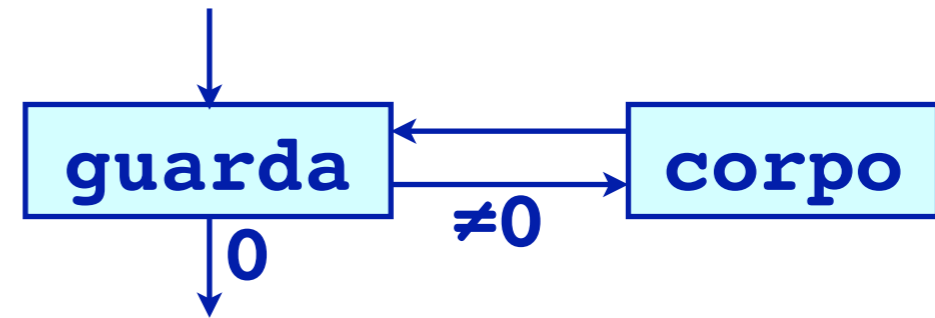
- A biblioteca padrão fornece um método que retorna um número aleatório, com a seguinte interface:
  - `#include <stdlib.h>`
  - `long random();`
- Escreva um joguinho que
  1. sorteia um número entre 1 e 100 - o número não é revelado ao jogador.
  2. o jogador deve adivinhar qual número foi sorteado.
  3. quando o jogador encontra o número segredo, é impresso o número de tentativas realizadas.

## Resumo sobre os comandos de repetição

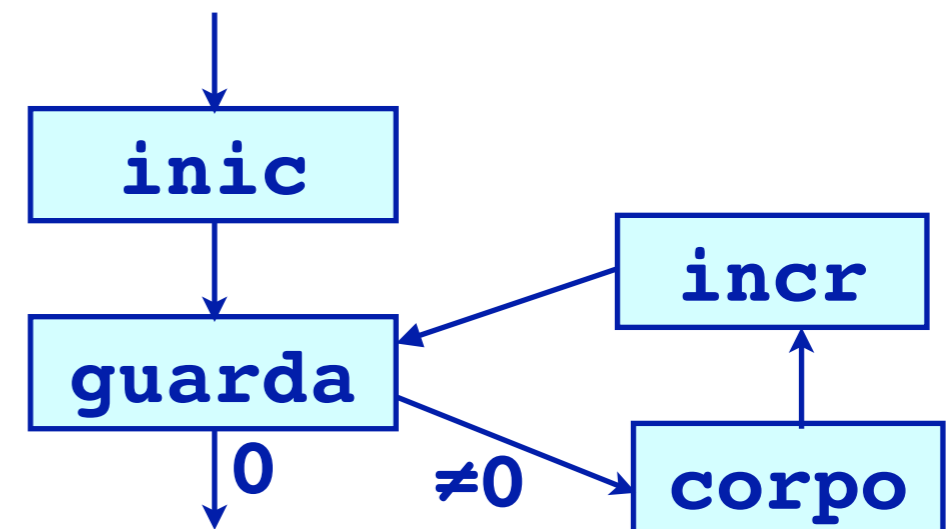
- C provê três comandos de repetição:
  - while
  - for
  - do
- Todos têm em comum:
  - guarda e corpo
  - a repetição continua enquanto a guarda for diferente de 0
  - os três comandos terminam quando a guarda for 0
- Difere
  - o momento que a guarda é avaliada
  - a sintaxe do comando inclui expressões para inicialização (antes da primeira avaliação da guarda) e incremento (depois de cada execução do corpo).

## Resumo sobre os comandos de repetição

**while (guarda) corpo**



**for (inic; guarda; incr) corpo**



**do corpo while (guarda);**

