

# Fundamentos de Programação

## *Loops* determinados com o `for`

Dainf - UTFPR

Profa. Leyza B. Dorini  
Prof. Bogdan T. Nassu

# Estruturas de Repetição: caso 01 (com for)

Este material discute o uso do comando `for` para fazer laços de repetição (*loops*) **determinados**, ou seja, quando a quantidade de vezes que um comando (ou bloco de comandos) deve ser executado é conhecida antes do início do *loop*.

## Lembrando: *loops* determinados com o `while`

Como vimos anteriormente, ao usar o `while` em *loops* determinados, em geral temos os seguintes pontos principais:

Inicialização  
do contador.

```
1 i = 0;
```

A condição para  
continuação  
do loop depende  
do contador.

```
2 while (i < 10)
```

```
3 {
```

```
4     scanf ("%d", &num);
```

```
5     printf ("Digitou %d\n", num);
```

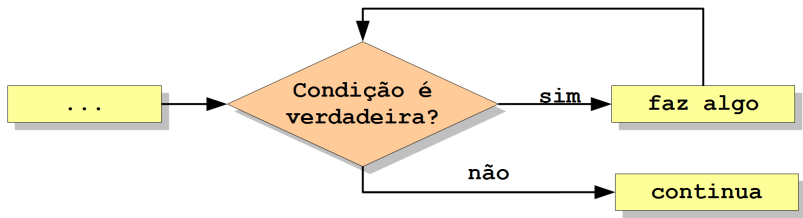
```
6     i++;
```

```
7 }
```

Atualização (incremento ou  
decremento) do contador.

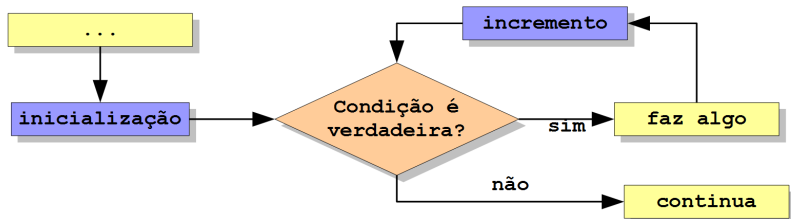
Bloco de comandos  
que é executado  
se a condição for  
verdadeira.

## Lembrando: fluxo do while



# Fluxo do for

O for é uma especialização do while, cuja sintaxe já inclui a inicialização e o incremento!



# for: sintaxe

*Inicialização do contador.*

*A condição para continuação do loop depende do contador.*

*Atualização do contador.*

*Não tem ; aqui!!!*

```
1
2 for (i = 0; i < 10; i++)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6 }
```

*Bloco de comandos que é executado enquanto a condição é verdadeira.*

## Loops determinados com o for: inicialização (0)

A primeira etapa consiste em inicializar o contador, que é a variável que vai nos ajudar a controlar quantas vezes o *loop* será executado.

*Inicialização  
do contador.*

*Vai ser executado apenas  
a primeira vez que  
"chega" no for.*

```
1  for (i = 0; i < 10; i++)  
2  {  
3      scanf ("%d", &num);  
4      printf ("Digitou %d\n", num);  
5  }
```

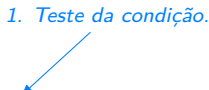
É equivalente à inicialização que fazíamos **antes** do `while`!

# Loops determinados com o for: *loop*

A partir de agora, o processo é o mesmo do while:

- 1 Teste da condição.

1. *Teste da condição.*



A blue arrow points from the text '1. Teste da condição.' to the condition 'i < 10' in the code snippet below. The code snippet is a C for loop that increments 'i' from 0 to 9 and prints the value of 'num' entered by the user.

```
1 for (i = 0; i < 10; i++)  
2 {  
3     scanf ("%d", &num);  
4     printf ("Digitou %d\n", num);  
5 }
```



# Loops determinados com o for: *loop*

A partir de agora, o processo é o mesmo do while:

- 1 Teste da condição.
- 2 Se condição for verdadeira, execução do bloco de comandos.

*1. Teste da condição.*

```
1 for (i = 0; i < 10; i++)  
2 {  
3     scanf ("%d", &num);  
4     printf ("Digitou %d\n", num);  
5 }
```

*2. Executa bloco de comandos.*

# Loops determinados com o for: *loop*

A partir de agora, o processo é o mesmo do while:

- 1 Teste da condição.
- 2 Se condição for verdadeira, execução do bloco de comandos.
- 3 Atualização do contador.

1. Teste da condição.

3. Atualiza  
o contador.

```
1 for (i = 0; i < 10; i++)  
2 {  
3     scanf ("%d", &num);  
4     printf ("Digitou %d\n", num);  
5 }
```

2. Executa bloco de comandos.

# Loops determinados com o for

O bloco é repetido **até que a condição se torne falsa!**

1. *Teste da condição.*

3. *Atualiza o contador.*

```
1 for (i = 0; i < 10; i++)  
2 {  
3     scanf ("%d", &num);  
4     printf ("Digitou %d\n", num);  
5 }
```

2. *Executa bloco de comandos.*

## Atenção!

Observe que a condição se torna falsa quando contador atinge o limite! O bloco acima **não será** executado quando  $i$  valer 10.

Você pode ler o for acima como: “para cada  $i \in [0, 10)$ , faça:”.

# Uso das chaves

Assim como para o while:

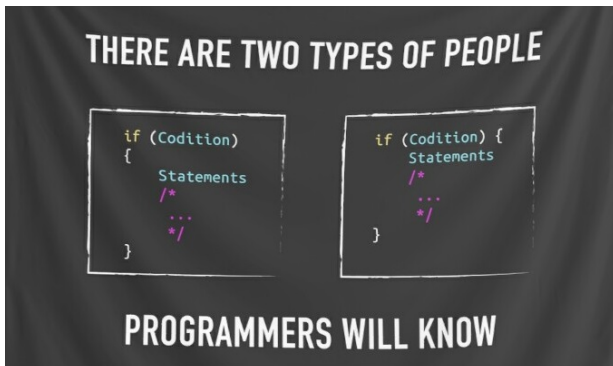
```
1 for (i = 0; i < 10; i++)
```

```
2 {
```

```
3     printf("oi. ");
```

```
4 }
```

chaves são opcionais se o bloco  
tiver apenas um comando



@redbubble

# Posso criar um contador dentro do for?

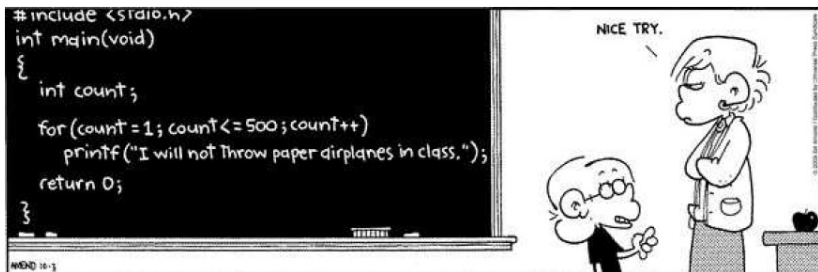
As versões mais recentes da linguagem C, assim como C++, permitem criar uma variável contadora para o for dentro da própria estrutura. Isto **não é** permitido em C ANSI!

```
1 for (int i = 0; i < 10; i++)
```



**Não faça isto!!!!**

# Nice try!



No código acima, o contador se chama `count` e a contagem iniciou em 1. Tirinha nota 7/10.

# Checkpoint

Reescreva o seguinte programa usando for ao invés de while.

```
1  int main () {
2      int i, idade;
3
4      i = 0;
5      while (i <= 4) {
6          printf("Digite a idade: ");
7          scanf("%d", &idade);
8
9          printf("A idade digitada foi: %d \n", idade);
10
11         i++;
12     }
13     return 0;
14 }
```

## Checkpoint

Resposta:

```
1  int main () {  
2      int i, idade;  
3  
4      for (i = 0; i <= 4; i++) {  
5          printf("Digite a idade: ");  
6          scanf("%d", &idade);  
7  
8          printf("A idade digitada foi: %d \n", idade);  
9      }  
10     return 0;  
11 }
```



# for e while



# Atualização do contador no for

Assim como no while, podemos ter um contador que é **decrementado**.

```
1 printf ("Imprimindo contagem regressiva ... ");
2 for (i = 10; i >= 1; i--)
3     printf("%d \n", i);
```

Assim como no while, podemos ter um contador que não é incrementado de 1 em 1.

```
1 printf ("Os pares entre 10 e 20 são: ");
2 for (i = 10; i <= 20; i += 2)
3     printf("%d \n", i);
```

## Exemplos - for

---

São os mesmos exemplos dos slides sobre a estrutura `while`! Fica como exercício comparar as soluções!

## Ex. 01 - Imprimindo os números inteiros de 1 a n

```
1  int main () {  
2  
3      int i, n;  
4  
5      scanf ("%d",&n);  
6  
7      for (i=1; i<=n; i++) {  
8          printf ("%d ", i);  
9      }  
10  
11     return 0;  
12 }
```

## Ex. 02 - Ler 10 números e contar quantos são pares

```
1  int main () {  
2  
3      int i,  
4          aux,  
5          contPares = 0;  
6  
7      for (i=1; i<=10; i++) {  
8          scanf ("%d", &aux);  
9  
10         if (aux%2 == 0)  
11             contPares++;  
12     }  
13     printf ("Qtde de pares: %d", contPares);  
14  
15     return 0;  
16 }
```