

Linguagem C - Estruturas de Repetição

PARTE 3

Lógica de Programação

Professor: Vinícius T. Guimarães
viniciusguimaraes@ifsul.edu.br



Tecnólogo em Sistema para Internet
Campus Charqueadas

Introdução

Até o presente momento, estudamos três estruturas de repetição fundamentais:

- **for()**
- **while()**
- **do ... while().**

Hoje vamos dar prosseguimento nos nossos estudos, aprofundando o que conseguimos fazer com os comandos que já conhecemos! 😊

Primeiro exemplo

<https://replit.com/@vicoguim/exemplo-repeticao-3-1>

```
1  #include <stdio.h>
2  int main(void) {
3      for(int i = 1; i <= 3; i++){
4          printf("Valor de i = %i\n",i);
5          for(int j = 1; j <= 3; j++){
6              printf("\tValor de j = %i\n",j);
7          }
8      }
9      return 0;
10 }
```

O que esse código faz?



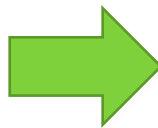
Primeiro exemplo

<https://replit.com/@vicoguim/exemplo-repeticao-3-1>

```
1  #include <stdio.h>
2  int main(void) {
3      for(int i = 1; i <= 3; i++){
4          printf("Valor de i = %i\n",i);
5          for(int j = 1; j <= 3; j++){
6              printf("\tValor de j = %i\n",j);
7          }
8      }
9      return 0;
10 }
```

```
➤ ./main
Valor de i = 1
    Valor de j = 1
    Valor de j = 2
    Valor de j = 3
Valor de i = 2
    Valor de j = 1
    Valor de j = 2
    Valor de j = 3
Valor de i = 3
    Valor de j = 1
    Valor de j = 2
    Valor de j = 3
```

Esse será o
resultado que será
impresso. **E como
isso funciona?**



Primeiro exemplo

```
1  #include <stdio.h>
2  int main(void)
3  {
4      for(int i = 1; i <= 3; i++){
5          for(int j = 1; j <= 3; j++){
6              printf("\tValor de j = %i\n", j);
7          }
8      }
9      return 0;
10 }
```

Laço controlado pela variável **i**.

Laço controlado pela variável **j**, o qual está dentro do bloco do laço mais externo.

Neste primeiro exemplo, temos o que chamamos de **estruturas de repetição aninhadas** ou, simplesmente, repetição dentro de repetição. Observem que temos um **laço for externo** (controlado pela variável **i**) e outro laço de **repetição dentro dele** (controlado pela variável **j**).

Primeiro exemplo

```
1  #include <stdio.h>
2  int main(void) {
3      for(int i = 1; i <= 3; i++){
4          printf("Valor de i = %i\n",i);
5          for(int j = 1; j <= 3; j++){
6              printf("\tValor de j = %i\n",j);
7          }
8      }
9      return 0;
10 }
```

Vamos fazer um teste de mesa para entender melhor como esse código funciona?



Primeiro exemplo

```
3 for(int i = 1; i <= 3; i++){
4     printf("Valor de i = %i\n",i);
5     for(int j = 1; j <= 3; j++){
6         printf("\tValor de j = %i\n",j);
7     }
8 }
```

PASSO 1

- Variável `i` recebe 1
- Verifica se `i <= 3`
- Entra na repetição

Lógica de Programação

Resultado do código

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){  
4      printf("Valor de i = %i\n",i);  
5      for(int j = 1; j <= 3; j++){  
6          printf("\tValor de j = %i\n",j);  
7      }  
8  }
```

PASSO 2

- Executa o printf
- Observe o resultado na tela

Resultado do código

Valor de i = 1

Memória e nossas variáveis

i	j
1	

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5  for(int j = 1; j <= 3; j++){
6      printf("\tValor de j = %i\n",j);
7  }
8  }
```

Resultado do co

PASSO 3

- Chegamos no outro comando for
- Variável j recebe 1
- Verifica se $j \leq 3$
- Entra na repetição

Lógica de Programação

Resultado do código

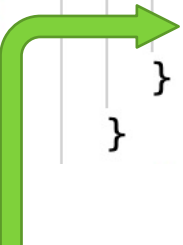
Valor de $i = 1$

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){  
4      printf("Valor de i = %i\n",i);  
5  for(int j = 1; j <= 3; j++){  
6      printf("\tValor de j = %i\n",j);  
7  }  
8  }
```



PASSO 4

- Executa o printf.
- Após executar o printf o código vai para o incremento da variável j

Resultado do código

```
Valor de i = 1  
    Valor de j = 1
```

Memória e nossas variáveis

i	j
1	
1	1

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5      for(int j = 1; j <= 3; j++){
6          printf("\tValor de j = %i\n",j);
7      }
8  }
```

PASSO 5

- Adiciona 1 ao valor de j
- Verifica se $j \leq 3$
- Entra na repetição

Lógica de Programação

Resultado do código

Valor de i = 1
Valor de j = 1

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5      for(int j = 1; j <= 3; j++){
6          printf("\tValor de j = %i\n",j);
7      }
8  }
```

PASSO 6

- Executa o printf
- Após executar o printf o código vai para o incremento da variável j

Lógica de Programação

Resultado do código

```
Valor de i = 1
  Valor de j = 1
  Valor de j = 2
```

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5      for(int j = 1; j <= 3; j++){
6          printf("\tValor de j = %i\n",j);
7      }
8  }
```

PASSO 7

- Adiciona 1 ao valor de j
- Verifica se $j \leq 3$
- Entra na repetição

Lógica de Programação

Resultado do código


```
Valor de i = 1
  Valor de j = 1
  Valor de j = 2
```

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  □ for(int i = 1; i <= 3; i++){  
4      printf("Valor de i = %i\n",i);  
5  □ for(int j = 1; j <= 3; j++){  
6      printf("\tValor de j = %i\n",j);  
7  }  
8  }
```



PASSO 8

- Executa o printf
- Após executar o printf o código vai para o incremento da variável j

Resultado do código

```
Valor de i = 1  
  Valor de j = 1  
  Valor de j = 2  
  Valor de j = 3
```

Memória e nossas variáveis

i	j
1	
1	1
1	2
1	3

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5      for(int j = 1; j <= 3; j++){
6          printf("\tValor de j = %i\n",j);
7      }
8  }
```

PASSO 10

- Adiciona 1 ao valor de j
- Verifica se $j \leq 3$
- **RESULTADO FALSO! Sai da repetição controlada pela variável j!**

Lógica de Programação

Resultado do código

```
Valor de i = 1
  Valor de j = 1
  Valor de j = 2
  Valor de j = 3
```

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5      for(int j = 1; j <= 3; j++){
6          printf("\tValor de j = %i\n",j);
7      }
8  }
```



Observe que o valor de **j** é 4. Além disso, estamos dentro do bloco **for()** controlado pela variável **i**. Logo, temos continuar a execução dele.

Segue o fio ...

Lógica de Programação

Resultado do código

```
Valor de i = 1
  Valor de j = 1
  Valor de j = 2
  Valor de j = 3
```

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5      for(int j = 1; j <= 3; j++){
6          printf("\tValor de j = %i\n",j);
7      }
8  }
```

PASSO 11

- Adiciona 1 ao valor de i
- Verifica se $i \leq 3$
- Entra na repetição

Lógica de Programação

Resultado do código

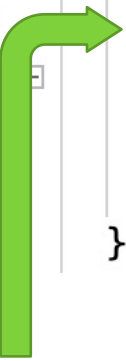
```
Valor de i = 1
  Valor de j = 1
  Valor de j = 2
  Valor de j = 3
```

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){  
4      printf("Valor de i = %i\n",i);  
5      for(int j = 1; j <= 3; j++){  
6          printf("\tValor de j = %i\n",j);  
7      }  
8  }
```



PASSO 12

- Executa o printf
- Observe o resultado na tela

Resultado do código

```
Valor de i = 1  
  Valor de j = 1  
  Valor de j = 2  
  Valor de j = 3  
Valor de i = 2
```

Memória e nossas variáveis

i	j
1	
1	1
1	2
1	3
1	4
2	4

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5      for(int j = 1; j <= 3; j++){
6          printf("\tValor de j = %i\n",j);
7      }
8  }
```

PASSO 13

- Chegamos no outro comando for
- Variável j recebe 1
- Verifica se $j \leq 3$
- Entra na repetição

Lógica de Programação

Resultado do código

```
Valor de i = 1
  Valor de j = 1
  Valor de j = 2
  Valor de j = 3
Valor de i = 2
```

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){
4      printf("Valor de i = %i\n",i);
5      for(int j = 1; j <= 3; j++){
6          printf("\tValor de j = %i\n",j);
7      }
8  }
```



Observe que o valor de **j** agora é **1**.
Portanto, vamos iniciar um novo ciclo controlado pela variável **j**, o qual irá acontecer por 3 vezes.

Lógica de Programação

Resultado do código

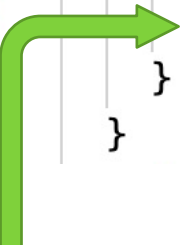
```
Valor de i = 1
  Valor de j = 1
  Valor de j = 2
  Valor de j = 3
Valor de i = 2
```

Memória e nossas variáveis

[illegible]

Primeiro exemplo

```
3  for(int i = 1; i <= 3; i++){  
4      printf("Valor de i = %i\n",i);  
5  for(int j = 1; j <= 3; j++){  
6      printf("\tValor de j = %i\n",j);  
7  }  
8  }
```



PASSO 14

- Executa o printf
- Após executar o printf o código vai para o incremento da variável j

Resultado do código

```
Valor de i = 1  
  Valor de j = 1  
  Valor de j = 2  
  Valor de j = 3  
Valor de i = 2  
  Valor de j = 1
```

Memória e nossas variáveis

i	j
1	
1	1
1	2
1	3
1	4
2	4
2	1

Primeiro exemplo

```
3  ▢  for(int i = 1; i <= 3; i++){  
4      printf("Valor de i = %i\n",i);  
5  ▢  for(int j = 1; j <= 3; j++){  
6      printf("\tValor de j = %i\n",j);  
7      }  
8  }
```



- Observe que o laço controlado pela variável **j** será repetido mais **3 vezes**, até que a variável **j** chegue no **valor 4**. Quando isso acontece, o resultado da condição retorna **false** e, conseqüentemente, a repetição controlada pela variável **j** é encerrada.
- Porém, ainda estamos no bloco de comando do laço controlado pela variável **i**. Logo, a variável **i** deve ser atualizada (acrescenta +1 nesse caso), e verifica a condição se **i <= 3**. Se a condição retornar **true**, executa mais uma vez todo o bloco de código, incluindo o laço controlado pela variável **j**.

Primeiro exemplo

```
3  ▢ for(int i = 1; i <= 3; i++){  
4      printf("Valor de i = %i\n",i);  
5  ▢ for(int j = 1; j <= 3; j++){  
6      printf("\tValor de j = %i\n",j);  
7  }  
8  }
```



Veja como ficou
o resultado na
tela e os
valores das
variáveis na
memória.

Lógica de Programação

Resultado do código

```
Valor de i = 1  
  Valor de j = 1  
  Valor de j = 2  
  Valor de j = 3  
Valor de i = 2  
  Valor de j = 1  
  Valor de j = 2  
  Valor de j = 3  
Valor de i = 3  
  Valor de j = 1  
  Valor de j = 2  
  Valor de j = 3
```

Memória e nossas variáveis

i	j
1	
1	1
1	2
1	3
1	4
2	4
2	1
2	2
2	3
2	4
3	4
3	1
3	2
3	3
3	4
4	4

E na prática?

E na prática, como posso utilizar essas estruturas de repetição aninhadas?

Já pensaram no tabuleiro do xadrez ou dama ou mesmo na estrutura do jogo da velha? Vamos tentar fazer um tabuleiro com código?



Aplicando na prática ...

```
1  #include <stdio.h>
2
3  int main(void) {
4      for(int i = 1; i <= 3; i++){
5          for(int j = 1; j <= 3; j++){
6              if(j % 2 == 0)
7                  printf("| 0 ");
8              else
9                  printf("| X ");
10         }
11         printf("\n");
12     }
13     return 0;
14 }
```

Observem que fazendo algumas pequenas modificações no código do exemplo, conseguimos mostrar na tela uma simples representação das casas do jogo da velha!



```
> ./main
| X | 0 | X |
| X | 0 | X |
| X | 0 | X |
```

<https://replit.com/@vicoguim/exemplo-repeticao-3-2>

Aplicando na prática ...

```
1  #include <stdio.h>
2
3  int main(void) {
4      printf(" _ _ _ _ _ _ _ _ \n");
5      for(int i = 1; i <= 8; i++){
6          for(int j = 1; j <= 8; j++){
7              printf("|_");
8          }
9          printf("|\\n");
10     }
11     return 0;
12 }
```

Com mais algumas modificações conseguimos montar um tabuleiro de 64 casas, semelhante ao que utilizamos para jogar dama e xadrez!

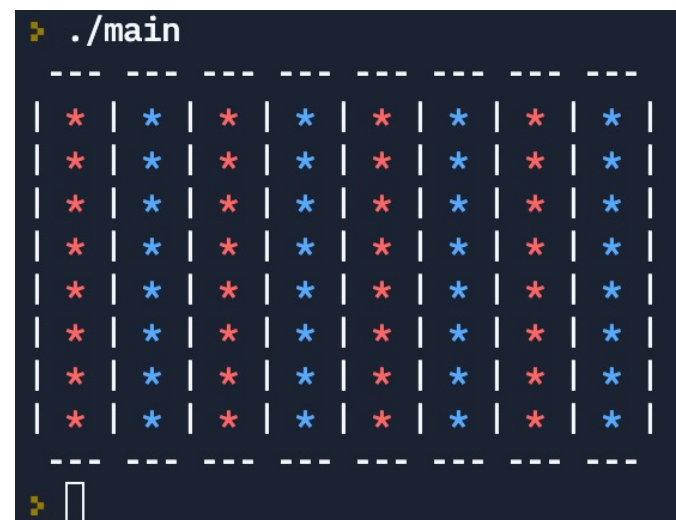


<https://replit.com/@vicoguim/exemplo-repeticao-3-3>

Aplicando na prática ...

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void) {
5      int azul = 4, vermelho = 1;
6      printf(" --- --- --- --- ---\n");
7      for(int i = 1; i <= 8; i++){
8          for(int j = 1; j <= 8; j++){
9              printf("\033[0;37m");
10             printf("|");
11             if(j % 2 == 0)
12                 printf("\033[0;37m", azul);
13             else
14                 printf("\033[0;37m", vermelho);
15             printf(" * ");
16         }
17         printf("\033[0;37m");
18         printf("\n");
19     }
20     printf(" --- --- --- --- ---\n");
21     return 0;
22 }
```

Agora preenchendo o tabuleiro com * e colorindo o * de acordo com a coluna.



<https://replit.com/@vicoguim/exemplo-repeticao-3-4>

Aplicando na prática ...

```
➤ ./main
- - - - -
| * | * | * | * | * | * | * | * |
| * | * | * | * | * | * | * | * |
| * | * | * | * | * | * | * | * |
| * | * | * | * | * | * | * | * |
| * | * | * | * | * | * | * | * |
| * | * | * | * | * | * | * | * |
| * | * | * | * | * | * | * | * |
| * | * | * | * | * | * | * | * |
- - - - -
➤
```

Agora preenchendo o tabuleiro com *, porém fazendo alternância das casas no estilo que funciona o tabuleiro de xadrez e dama.

<https://replit.com/@vicoguim/exemplo-repeticao-3-5>

Resumo ...

- O uso de estruturas de repetição nos possibilita um conjunto bastante vasto de possibilidades.
- Mostramos alguns exemplos bem simples para montagem, por exemplo, de tabuleiros. Porém, esse é apenas um ponto de partida.
- O uso de estruturas de repetição aninhadas será muito importante **no decorrer dessa disciplina e do curso** para diversas aplicações, tais como:
 - **Percorrer matrizes e vetores;**
 - **Varrer resultados de consultas a um banco de dados;**
 - **Organizar interfaces;**
 - **Desenvolver jogos;**
 - **Enfim, o céu é o limite ...**

**NÃO TRABALHAMOS
COM LIMITE.**