INSTITUTO FEDERAL
Catarinense
Campus Blumenau

Bacharelado em
ENGENHARIA
ELÉTRICA

# Algoritmos - Estrutura de Repetição - For - Básico

## Tarefa - Algoritmo - Estrutura de Repetição - For - Básico

## 1 Estrutura de Repetição - For - Básico

Uma estrutura de repetição é um elemento que aparece em diversas linguagem de programação, ela permite que uma ação seja executada repetidamente. Esta estrutura, na linguagem de programação C, é representada pelo comando `for`. Essa repetição pode ser finita ou infinita, nesse caso ela é conhecida como *loop infinito*. Erros de lógica geralmente conduzem a execução de um *loop infinito*, fique atento ao especificar valores de inicialização, paradas e condição de parada de sua estrutura de repetição.

Para controlar o comportamento da repetição são utilizados três parâmetros, eles controlarão em que valor a repetição deve iniciar, se ela deve ou não parar, qual valor ela deve parar, como a atualização das variáveis de controle deve ocorrer e muitas outras variantes são possíveis. O `for` na linguagem C é muito versátil e pode ser configurado para diversos tipos de repetição. Além dos parâmetros é necessário informar a ação ou o bloco de ações a serem executados repetidamente. Os parâmetros são informados dentro de parentêses e separados por ponto e virgulas, a ação ou o bloco de ações deve vir imediatamente ao final do símbolo de fechamento de parênteses.

O primeiro parâmetro deste comando é utilizando somente uma única vez quando a linha contendo o `for` é alcançada, ele é utilizado para inicializar a variável ou as variáveis de controle. Essas variáveis podem ser utilizadas para controlar o número de repetição.

O segundo parâmetro é executado antes de cada execução das ações a serem repetidas, este parâmetro controla quando a repetição deve ocorrer. Se esse parâmetro for uma expressão booleana então ela será avaliada e a execução da ação de repetição ocorrerá enquanto a expressão for verdadeira. Caso esse parametro seja omitido então a repetição será permanente, configurando um *loop* infinito.

O último parâmetro ocorre após a execução da ação, ele é utilizado para atualizar os valores de variáveis.

```c
#include <stdio.h>

int main()
{
    int i = 0;
    for (i = 0; i < 10; i++)
        printf("%d\n");
```

```
8       return 0;
9   }
```

Programa 1: Exemplo de programa utilizando a estrutura de repetição `for`

Não existem restrições sobre o uso de instruções que representam ações dentro de uma estrutura de repetição, inclusive é possível utilizar estruturas de repetição dentro de estruturas de repetição. Segue um exemplo:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Pares ordenados de inteiros entre (0,0) e (5,5)\n");
6      for (int i = 0; i <= 5; i++)
7      {
8          for (int j = 0; j <= 5; j++)
9          {
10             printf("(%d,%d) ", i, j);
11         }
12         printf("\n");
13     }
14     return 0;
15 }
```

Programa 2: Exemplo de programa utilizando estruturas de repetição aninhadas

Repare também neste último exemplo que o parâmetro de inicialização foi utilizado para declarar variáveis. Isso é possível, porém dependerá do padrão ANSI C de se seu compilador. Para testar um padrão de compilação, nos compiladores GCC ou clang, que não permite a declaração de variáveis dentro do for utilize `-std=c90`. O padrão `-std=c99`, por exemplo, compilará o código normalmente. Isso influenciará seu trabalho caso utilize compiladores mais simples que utilizem padrões antigos. Existem outras restrições que compiladores podem fazer, um exemplo é somente permitir declaração de variáveis no início de uma função, fique atento ao utilizar compiladores antigos ou compiladores simplificados.

Os livros [Paul Deitel, 2022] e [Brian W. Kernighan, 1988] podem ser utilizados para obter mais informações sobre a estrutura de repetição `for`.

## 2 Exercícios

1. Escreva uma função chamada `imprimir_pares` que aceitará como parâmetros dois valores $i$ e $f$. Sua função deverá imprimir os números pares entre $i$ e $f$ (inclusive $i$ e $f$ caso sejam pares).

2. Escreva uma função chamada `somar_n_primeiros` que aceitará como parâmetro um número $n$ e deverá retornar a soma dos primeiros $n$ inteiros positivos.

3. Escreva um programa que aceite como entrada uma taxa de juros mensal, uma quantidade de valor monetário inicial e uma quantidade de tempo, imprima os valores dos juros simples mês após mês.

4. Escreva um programa que aceite como entrada uma taxa de juros mensal, uma quantidade de valor monetário inicial e uma quantidade de tempo, imprima os valores dos juros compostos mês após mês.

5. Escreva um programa que exibe os primeiros $n$ termos de uma progressão geométrica.

6. Escreva um programa que calcula a soma dos $n$ primeiros termos de uma progressão geométrica.

7. Escreva uma função chamada `iniciar_array_rand` que deverá aceitar como parâmetros um array $a$, um número um número $n$, um valor $max$ e valor $min$. O parâmetro $n$ informará o número de posições disponíveis no array. Os valores do array deverão ser inicializados com valores aleatórios entre $min$ e $max$

8. Escreva uma função chamada `fatorial` para calcular o fatorial de um número $n$. Teste sua função com o número 80. Sua função calcula o valor corretamente? Justifique.

9. Escreva uma função chamada `soma_riemann` para calcular o resultado numérico da integral $\int_a^b f(x)dx$. A função `soma_riemann` deverá aceitar como parâmetro um número $n$ que descreverá a quantidade de subintervalos da soma de riemann e os números $a$ e $b$ do intervalo de integração.

   (Observação 1: Soma de Riemann pode ser estudada em [Stewart et al., 2020], na biblioteca é possível encontrar uma versão mais antiga desse livro, procure no volume 1)

   (Observação 2: caso queira passar uma função arbitrária para sua função é possível utilizar ponteiros para função, caso contrário declare uma função $f$ antes da função `soma_riemann`).

10. Escreva uma `struct` para representar um vetor matemático. Uma sugestão é ter um número $n$ que descreve a dimensão do vetor e um array de double para armazenar as componentes do vetor. Faça funções para inicializar, somar, subtrair, calcular produto interno, calcular produto externo (para dimensão $n = 3$)

11. Escreva uma `struct` para representar uma matriz matemática. Faça funções para inicializar, somar, subtrair e multiplicar matrizes. Lembre-se de respeitar as regras de operações com matrizes.

12. Escreva uma função chamada `imprimir_quadrado` que aceitará um número $n$ que deverá ser um número impar positivo. A função deverá imprimir o seguinte padrão conforme $n$

    Se $n = 1$ então a saída deverá ser

    ```
    X
    ```

    Se $n = 3$ então a saída deverá ser

    ```
    000
    ```

```
0X0
000
```

Se $n = 5$ então a saída deverá ser

```
00000
00000
00X00
00000
00000
```

13. Escreva uma função chamada `imprimir_quadrado2` que aceitará um número $n$ que deverá ser um número impar positivo. A função deverá imprimir o seguinte padrão conforme $n$

Se $n = 1$ então a saída deverá ser

```
X
```

Se $n = 3$ então a saída deverá ser

```
\-/
|X|
/-\
```

Se $n = 5$ então a saída deverá ser

```
\---/
|\./|
|.X.|
|/.\|
/---\
```

Se $n = 7$ então a saída deverá ser

```
\-----/
|\.../|
|.\./.|
|..X..|
|./.\.|
|/...\|
/-----\
```

14. Faça um programa que exibe as 10 primeiras tabuadas (tabuada do 1 até tabuada do 10).

# Referências

[Brian W. Kernighan, 1988] Brian W. Kernighan, D. M. R. (1988). *C Programming Language*. Prentice Hall, 2 edition.

[Paul Deitel, 2022] Paul Deitel, H. D. (2022). *C How to Program*. Pearson, 9 edition.

[Stewart et al., 2020] Stewart, J., Clegg, D., and Watson, S. (2020). *Calculus: Early Transcendentals*. Cengage Learning.