

Estruturas de Repetição

Conceito e Sintaxe

A IMPORTÂNCIA DE ESTRUTURAS DE REPETIÇÃO



01 Capaz de Repetir trechos de códigos múltiplas vezes

Enquanto. (**while**)

Fazer Enquanto (**do-while**)

Para (**for**)

02 Reaproveitamento de código

Evitando redundância e aumentando a eficiência.

03 Modularidade e Legibilidade

Permite que tarefas sejam descritas de forma mais clara e concisa.

NECESSIDADE DE ESTRUTURAS DE REPETIÇÃO



Avaliação de condições para determinar se, o que e como algo deve ou não ser feito.

01 Processamento de Dados em Bancos

Bancos utilizam estruturas de repetição para processar grandes volumes de transações financeiras, como pagamentos, transferências e saques.

02 Controle de Estoque em Supermercados

Estruturas de repetição para atualizar o estoque de produtos, realizar inventários e monitorar a disponibilidade de itens nas prateleiras.

03 Análise de Dados

Pesquisadores utilizam estruturas de repetição para analisar grandes conjuntos de dados em pesquisas computacionais.

04 Processamento de Texto

Editores de texto utilizam estruturas de repetição para percorrer cada caractere ou linha de um documento, permitindo diferentes operações.

QUAL O PAPEL PRÁTICO DESSAS ESTRUTURAS?



01

Automação de Tarefas Repetitivas: As estruturas de repetição automatizam tarefas que precisam ser realizadas várias vezes, evitando a redundância e simplificando o código.

02

Processamento de Dados: Permitem o processamento de grandes volumes de dados de forma eficiente, permitindo a iteração sobre listas, arrays e outros tipos de estruturas de dados.

03

Iteração e Controle de Fluxo: Permitem a iteração sobre uma sequência de elementos, como caracteres em uma string, itens em uma lista ou linhas em um arquivo, proporcionando um controle detalhado sobre o fluxo de execução do programa.

EXEMPLOS TEÓRICOS

01

Validação de Usuário: (Testa correspondência de senha)

senha correta = x
senha digitada = y

senha digitada **!=** senha correta



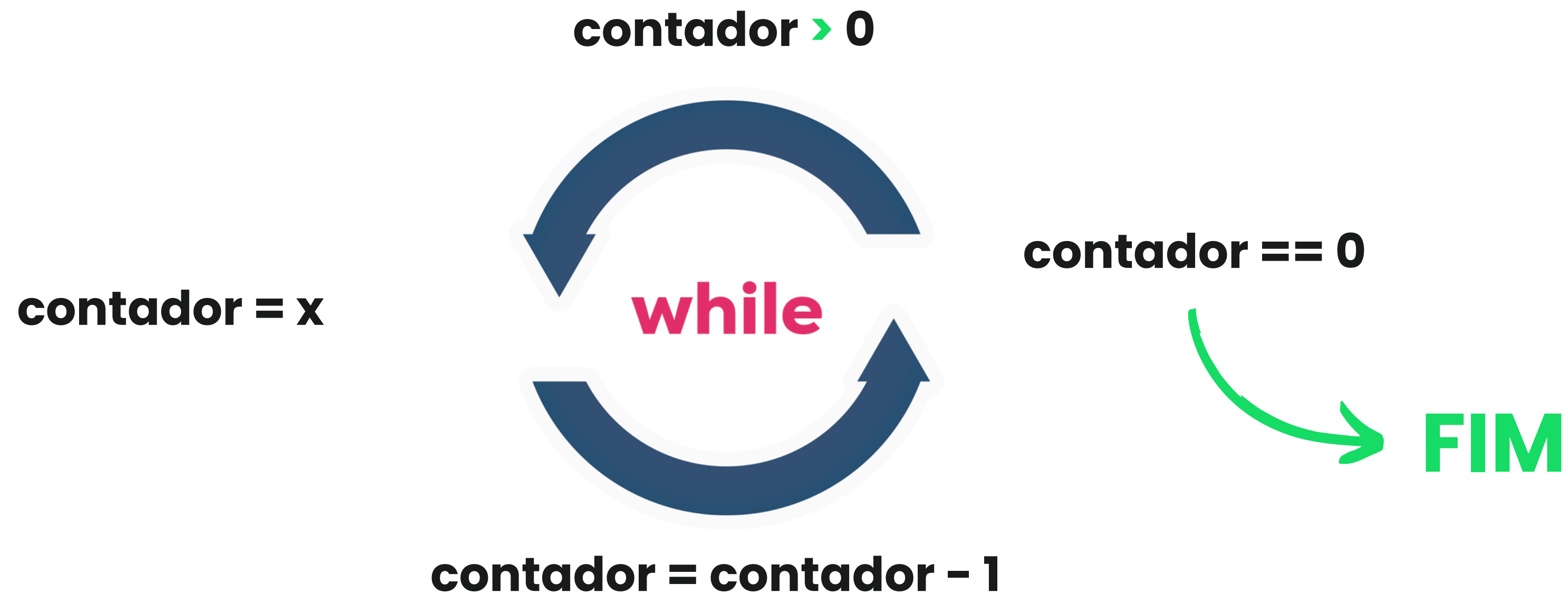
senha digitada = ?

senha digitada **==** senha correta

FIM

02

Contador: (Conta do numero digitado até 0)



03

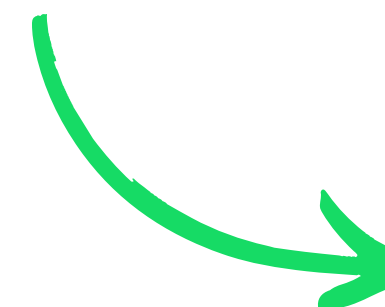
Calculo de horas de estudo: (Conta horas de estudo ao longo de uma semana)

dia = 1
horas_estudo = 0
total_horas = 0



horas_estudo = x
total_horas = total_horas + horas_estudo
dia = dia + 1

dia == 7



FIM
total_horas

SINTAXE DE ESTRUTURA DE REPETIÇÃO WHILE



```
1  while (condição) {  
2      // Corpo do loop  
3      // Código a ser repetido enquanto a condição for verdadeira  
4  }  
5
```


EXEMPLOS PRÁTICOS:

01

Validação de Usuário: (Testa correspondência de senha)

```
1  #include <stdio.h>
2
3  int main() {
4      char senha_correta[] = "senha123";
5      char senha_digitada[20];
6
7      printf("Digite sua senha: ");
8      scanf("%s", senha_digitada);
9
10     while (strcmp(senha_digitada, senha_correta) != 0) {
11         printf("Senha incorreta. Tente novamente: ");
12         scanf("%s", senha_digitada);
13     }
14
15     printf("Senha correta! Bem-vindo!\n");
16
17     return 0;
18 }
```

Contador: (Conta do numero digitado até 0)



```
1  #include <stdio.h>
2
3  int main() {
4      int contador = 10;
5
6      printf("Contagem regressiva:\n");
7      while (contador >= 0) {
8          printf("%d\n", contador);
9          contador--;
10     }
11
12     printf("Fogo!\n");
13
14     return 0;
15 }
```

Calculo de horas de estudo: (Conta horas de estudo ao longo de uma semana)



```
1  #include <stdio.h>
2
3  int main() {
4      int horas_estudo, dia = 1, total_horas = 0;
5
6      printf("Contador de Horas de Estudo\n\n");
7
8      // Loop para registrar as horas de estudo por dia
9      while (dia <= 7) {
10         printf("Digite o número de horas de estudo no dia %d: ", dia);
11         scanf("%d", &horas_estudo);
12
13         total_horas += horas_estudo; // Adiciona as horas de estudo ao total
14         dia++; // Incrementa o contador de dias
15     }
16
17     printf("\nTotal de horas de estudo na semana: %d\n", total_horas);
18
19     return 0;
20 }
21
```

DESAFIO

Crie um programa simples que conte e exiba a quantidade de números pares dentro de um intervalo fornecido pelo usuário.

- a. O programa deve solicitar ao usuário que insira dois números inteiros que representem o início e o fim do intervalo.
- b. O programa deve contar e exibir a quantidade de números pares dentro desse intervalo, incluindo os próprios limites do intervalo, se aplicável.

