in4mation HUB

# Estruturas de Repetição

O que foi aprendido

in4mation HUB

# QUAIS SÃO OS TIPOS ESTRUTURAS DE REPETIÇÃO, AFINAL?

**01**

**Laço while:** Repete um bloco de código enquanto uma condição é verdadeira.

**02**

**Laço do-while:** Executa um bloco de código pelo menos uma vez, e então repete enquanto uma condição é verdadeira.

**03**

**Laço for:** Utilizado para repetir uma operação um número específico de vezes.

## while

```
while (condição) {
    // Corpo do loop
    // Código a ser repetido enquanto a condição for verdadeira
}
```

## do-while

```
do {
    // Corpo do loop
    // Código a ser repetido pelo menos uma vez
} while (condição);
```

## for

```
for (inicialização; condição; atualização) {
    // Corpo do loop
    // Código a ser repetido
}
```

Cada um possui o seu próprio papel e necessidade.

in4mation HUB

in4mation HUB

# QUAL É A APLICABILIDADE?

Os laços de repetição são aplicáveis sempre que precisamos executar uma ou mais instruções múltiplas vezes. Eles nos permitem automatizar tarefas repetitivas e processar conjuntos de dados de forma eficiente, economizando tempo e esforço.

# EXEMPLO E POSSÍVEIS SOLUÇÕES:

Escreva um programa em C que solicite ao usuário que insira um número inteiro positivo e calcule o produto de todos os dígitos desse número.

in4mation HUB

# 01

## Resolução utilizando while:

```c
#include <stdio.h>

int main() {
    int numero, produto = 1, digito;

    printf("Digite um numero inteiro positivo: ");
    scanf("%d", &numero);

    while (numero > 0) {
        digito = numero % 10; // Obtém o último dígito
        produto *= digito; // Multiplica o produto pelo dígito
        numero /= 10; // Remove o último dígito do número
    }

    printf("O produto dos digitos e: %d\n", produto);

    return 0;
}
```

in4mation HUB

# 02

## Resolução utilizando do-while:

```c
#include <stdio.h>

int main() {
    int numero, produto = 1, digito;

    printf("Digite um numero inteiro positivo: ");
    scanf("%d", &numero);

    do {
        digito = numero % 10; // Obtém o último dígito
        produto *= digito; // Multiplica o produto pelo dígito
        numero /= 10; // Remove o último dígito do número
    } while (numero > 0);

    printf("O produto dos digitos e: %d\n", produto);

    return 0;
}

```

in4mation HUB

03

# Resolução utilizando for:

```c
#include <stdio.h>

int main() {
    int numero, produto = 1, digito;

    printf("Digite um numero inteiro positivo: ");
    scanf("%d", &numero);

    for (; numero > 0; numero /= 10) {
        digito = numero % 10; // Obtém o último dígito
        produto *= digito; // Multiplica o produto pelo dígito
    }

    printf("O produto dos digitos e: %d\n", produto);

    return 0;
}
```

in4mation HUB

# NESTE CASO, QUAL A MELHOR OPÇÃO?

01 **while :** É adequado quando não temos garantia de que o bloco de código será executado pelo menos uma vez, como quando precisamos verificar se o número é positivo antes de calcular o produto.

02 **do-while :** Não traz vantagem aqui, pois não é necessário calcular o produto antes de verificar a validade do número.

03 **for :** É a opção mais natural, pois sabemos quantas vezes o bloco de código deve ser executado (uma vez para cada dígito). É mais conciso e expressivo para percorrer os dígitos do número.

in4mation HUB

# EXERCÍCIO EM CONJUNTO

Desenvolva um programa em C que exiba a tabuada de multiplicação não apenas do número fornecido pelo usuário, mas também do número anterior e do número posterior. O programa deve solicitar ao usuário que insira um número e, em seguida, mostrar a tabuada desses três números (número anterior, número fornecido e número posterior) de 1 a 10.

01 **Utilize laço for para calcular a tabuada do número anterior ao passado pelo usuário.**

02 **Utilize laço while para calcular a tabuada do número passado pelo usuári pa**

03 **Utilize laço do-while para calcular a tabuada do número posterior ao passado pelo usuário.**

in4mation HUB

# EXERCÍCIO EM CONJUNTO

01

```c
#include <stdio.h>

int main() {
    int numero, i;

    // Solicita ao usuário que insira um número
    printf("Digite um numero: ");
    scanf("%d", &numero);

    printf("\nTabuada do numero anterior (%d):\n", numero - 1);
    // Tabuada do número anterior usando o laço for
    for (i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", numero - 1, i, (numero - 1) * i);
    }
```

Primeiro deve-se declarar as variáveis, pedir o número da tabuada ao usuário e então usar do laço **for** para retornar a tabuada de seu anterior.

in4mation HUB

# EXERCÍCIO EM CONJUNTO

02

```c
printf("\nTabuada do numero digitado (%d):\n", numero);
// Tabuada do número digitado usando o laço while
i = 1;
while (i <= 10) {
    printf("%d x %d = %d\n", numero, i, numero * i);
    i++;
}
```

Agora use o laço **while** para retornar a tabuada do valor parrado pelo usuário.

in4mation HUB

# EXERCÍCIO EM CONJUNTO

**03**

```
    printf("\nTabuada do proximo numero (%d):\n", numero + 1);
    // Tabuada do próximo número usando o laço do-while
    i = 1;
    do {
        printf("%d x %d = %d\n", numero + 1, i, (numero + 1) * i);
        i++;
    } while (i <= 10);

    return 0;
}
```

Agora faça o mesmo para o numero posterior utilizando do laço **do-while**.

in4mation HUB

# CODIGO FINAL:

```c
#include <stdio.h>

int main() {
    int numero, i;

    // Solicita ao usuário que insira um número
    printf("Digite um numero: ");
    scanf("%d", &numero);

    printf("\nTabuada do numero anterior (%d):\n", numero - 1);
    // Tabuada do número anterior usando o laço for
    for (i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", numero - 1, i, (numero - 1) * i);
    }

    printf("\nTabuada do numero digitado (%d):\n", numero);
    // Tabuada do número digitado usando o laço while
    i = 1;
    while (i <= 10) {
        printf("%d x %d = %d\n", numero, i, numero * i);
        i++;
    }

    printf("\nTabuada do proximo numero (%d):\n", numero + 1);
    // Tabuada do próximo número usando o laço do-while
    i = 1;
    do {
        printf("%d x %d = %d\n", numero + 1, i, (numero + 1) * i);
        i++;
    } while (i <= 10);

    return 0;
}
```

in4mation HUB

# DESAFIO

Desenvolva um programa em C que solicite ao usuário um número inteiro positivo n e, em seguida, imprima um triângulo composto por asteriscos (*) de altura **n**. Por exemplo, se o usuário inserir 5, o programa deve exibir o seguinte padrão:

```
*
**
***
****
*****
```