Laços de Repetição em C

Os **laços de repetição** (ou **loops**) são estruturas de controle que permitem executar um bloco de código várias vezes, enquanto uma condição for verdadeira. Em C, temos três tipos principais de laços: `for`, `while`, e `do-while`. A seguir, veremos uma explicação detalhada de cada um, incluindo exemplos de uso.

1. Laço `for`

O laço `for` é usado quando você sabe de antemão o número exato de iterações que deseja realizar. Ele é particularmente útil quando você deseja percorrer um conjunto finito de elementos, como um array ou uma sequência numérica.

Sintaxe do `for`

```
for (inicialização; condição; incremento/decremento) {
    // Bloco de código a ser executado enquanto a condição for verdadeira
}
```

- **inicialização**: A variável de controle do laço é inicializada aqui. Ela é executada apenas uma vez, no início.
- **condição**: O laço continua enquanto a condição for verdadeira.
- **incremento/decremento**: Após cada iteração, a variável de controle é atualizada (incrementada ou decrementada).

Exemplo de `for`

```c
#include <stdio.h>

int main() {
    // Laço for que imprime os números de 1 a 5
    for(int i = 1; i <= 5; i++) {
        printf("%d\n", i); // Imprime o valor de i em cada iteração
    }
    return 0;
}
```

Saída esperada:

```
1
2
3
4
5
```

Explicação:

- A variável `i` começa em 1.
- A condição `i <= 5` é verificada a cada iteração. Se for verdadeira, o bloco de código dentro do `for` é executado.
- Após cada execução do bloco, o valor de `i` é incrementado em 1.

2. Laço `while`

O laço `while` é utilizado quando não se sabe o número exato de iterações, mas a execução do bloco de código depende de uma condição que é verificada antes de cada iteração.

Sintaxe do `while`

```
while (condição) {
    // Bloco de código a ser executado enquanto a condição for verdadeira
}
```

- A **condição** é verificada antes de cada execução do bloco de código.
- Se a condição for **verdadeira**, o bloco de código será executado.
- Se a condição for **falsa** logo no início, o código dentro do `while` não será executado.

Exemplo de `while`

```c
#include <stdio.h>

int main() {
    int i = 1;
    // Laço while que imprime os números de 1 a 5
    while(i <= 5) {
        printf("%d\n", i); // Imprime o valor de i
        i++; // Incrementa i
    }
    return 0;
}
```

Saída esperada:

```
1
2
3
4
5
```

Explicação:

- O valor inicial de `i` é 1.
- A condição `i <= 5` é verificada antes de cada iteração.
- Se a condição for verdadeira, o código dentro do `while` é executado e `i` é incrementado.
- Quando `i` chega a 6, a condição se torna falsa, e o laço é interrompido.

3. Laço `do-while`

O laço `do-while` é semelhante ao `while`, mas a diferença principal é que a **condição** é verificada **após** a execução do bloco de código. Isso significa que o bloco de código será executado pelo menos uma vez, independentemente de a condição ser verdadeira ou falsa.

Sintaxe do `do-while`

```
do {
    // Bloco de código a ser executado
} while (condição);
```

- O bloco de código dentro do `do` será executado **pelo menos uma vez**, mesmo que a condição seja falsa logo no início.
- A **condição** é verificada após a execução do bloco de código.
- Se a condição for verdadeira, o bloco de código será executado novamente. Caso contrário, o laço é interrompido.

Exemplo de `do-while`

```
#include <stdio.h>

int main() {
    int i = 1;
    // Laço do-while que imprime os números de 1 a 5
    do {
        printf("%d\n", i); // Imprime o valor de i
        i++; // Incrementa i
    } while(i <= 5);
    return 0;
}
```

Saída esperada:

```
1
2
3
```

```
4
5
```

Explicação:

- O valor inicial de `i` é 1.
- O código dentro do `do` é executado uma vez antes de verificar a condição.
- A condição `i <= 5` é verificada após cada execução, e o laço continua enquanto for verdadeira.
- Quando `i` chega a 6, a condição se torna falsa e o laço é interrompido.

Diferenças entre os tipos de laços

- **`for`**: Usado quando você sabe o número de iterações antecipadamente. Ideal para contar ou percorrer estruturas de dados com índice.
- **`while`**: Usado quando você não sabe o número exato de iterações, mas deseja repetir enquanto uma condição for verdadeira.
- **`do-while`**: Semelhante ao `while`, mas garante que o código seja executado pelo menos uma vez.

Laços Aninhados

Em C, também é possível **anidar** laços, ou seja, colocar um laço dentro de outro. Isso é útil para percorrer estruturas mais complexas, como matrizes bidimensionais.

Exemplo de laços aninhados

```c
#include <stdio.h>

int main() {
    // Laços aninhados para imprimir uma matriz 3x3
    for(int i = 1; i <= 3; i++) {
        for(int j = 1; j <= 3; j++) {
            printf("M[%d][%d] = %d\n", i, j, i * j);
        }
    }
    return 0;
}
```

Saída esperada:

```
M[1][1] = 1
M[1][2] = 2
M[1][3] = 3
M[2][1] = 2
M[2][2] = 4
M[2][3] = 6
M[3][1] = 3
```

```
M[3][2] = 6
M[3][3] = 9
```

Explicação:

- O primeiro laço percorre as linhas da matriz.
- O segundo laço percorre as colunas para cada linha.
- O valor exibido é o produto de `i` e `j`, formando a tabela de multiplicação.

Conclusão

Laços de repetição são fundamentais para realizar tarefas repetitivas de forma eficiente e organizada. Com os laços `for`, `while`, e `do-while`, você pode controlar o fluxo de execução do seu código de acordo com condições específicas, melhorando a performance e legibilidade do programa.