Módulo 5 - Laços de repetição While e For

While

Em linguagem C (e em muitas outras linguagens de programação), `while` e `continue` são duas estruturas de controle de fluxo que permitem criar loops e controlar o comportamento dentro deles. Aqui está uma explicação de como elas funcionam:

1. `while`:
    - O `while` é uma estrutura de controle de loop em C.
    - Ele permite que você execute um bloco de código repetidamente enquanto uma condição especificada for verdadeira.
    - A condição é verificada antes de cada iteração do loop, e se a condição for falsa no início, o bloco de código dentro do `while` nunca será executado.

Exemplo de uso do `while`:

```
int contador = 0;
while (contador < 5) {
    printf("Contagem: %d\n", contador);
    contador++;
}
```

Neste exemplo, o bloco de código dentro do `while` será executado enquanto a variável `contador` for menor que 5.

1. `continue`:
    - A instrução `continue` é usada dentro de um loop (como `while`, `for` ou `do-while`) para pular a iteração atual e passar para a próxima iteração.

- Quando o `continue` é encontrado, o controle do programa volta ao teste da condição do loop (no caso de `while` ou `for`) ou ao início do loop (no caso de `do-while`), ignorando qualquer código que esteja após o `continue` na iteração atual.

Exemplo de uso do `continue`:

```
for (int i = 0; i < 5; i++) {
    if (i == 2) {
        printf("Ignorando a iteração %d usando continue\n", i);
        continue; // Pula a iteração quando i é igual a 2.
    }
    printf("Iteração %d\n", i);
}
```

Neste exemplo, quando `i` é igual a 2, a instrução `continue` é executada, pulando a impressão da mensagem "Iteração 2" e passando para a próxima iteração.

Em resumo, o `while` é usado para criar loops que são executados enquanto uma condição é verdadeira, e o `continue` é usado para pular a iteração atual de um loop e continuar com a próxima iteração. Eles são ferramentas úteis para controlar o fluxo do seu programa em cenários de repetição.

For

A instrução `for` é uma das estruturas de controle de fluxo em linguagem C (e em muitas outras linguagens de programação) usadas para criar loops. Ela é especialmente útil quando você sabe antecipadamente quantas vezes deseja repetir um bloco de código. A estrutura geral de um loop `for` em C é a seguinte:

```
for (inicialização; condição; incremento/decremento) {
    // Corpo do loop
}
```

Aqui está uma explicação de cada parte da estrutura `for`:

- `inicialização`: É a parte onde você inicializa uma variável de controle do loop. Essa parte é executada apenas uma vez, no início do loop.

- `condição`: É uma expressão booleana que é verificada antes de cada iteração do loop. Se a condição for verdadeira, o loop continua a ser executado; caso contrário, o loop é encerrado.
- `incremento/decremento`: É a parte onde você atualiza a variável de controle do loop após cada iteração. Isso permite que você controle como a variável muda a cada passagem pelo loop.

Aqui está um exemplo de um loop `for` simples que imprime os números de 1 a 5:

```
for (int i = 1; i <= 5; i++) {
    printf("%d\n", i);
}
```

Neste exemplo:

- A `inicialização` cria a variável `i` e a define como 1.
- A `condição` verifica se `i` é menor ou igual a 5.
- O `incremento` aumenta `i` em 1 após cada iteração.

O loop será executado cinco vezes, imprimindo os números de 1 a 5.

Você pode usar a estrutura `for` em uma variedade de cenários, desde contagens simples até iterações mais complexas. Ela é uma ferramenta poderosa para criar loops com um controle preciso sobre a inicialização, a condição e o incremento ou decremento da variável de controle.