Instrução Continue em C++

Introdução

A instrução `continue` em C++ é usada dentro de laços (`for`, `while`, `do-while`) para pular o restante do código no bloco atual e iniciar a próxima iteração do laço. É útil quando você deseja ignorar certas partes da iteração e continuar o loop com a próxima iteração.

1. Definição e Sintaxe

- Definição: A instrução `continue` faz com que o laço pule para a próxima iteração, ignorando o restante do código dentro do bloco atual.
- Sintaxe:

```
continue;
```

2. Uso em Laço `for`

- Definição: No laço `for`, a instrução `continue` faz com que a execução salte para a expressão de incremento.
- Exemplo:

```
int main() {
    for (int i = 0; i < 10; ++i) {
        if (i % 2 == 0) {
            continue;  // Pula para a próxima iteração quando i é par
        }
        std::cout << "i = " << i << std::endl;
```

```
    }
    return 0;
}
```

3. Uso em Laço `while`

- Definição: No laço `while`, a instrução `continue` faz com que a execução salte de volta para a condição do laço.
- Exemplo:

```
int main() {
    int i = 0;

    while (i < 10) {
        ++i;
        if (i % 2 == 0) {
            continue;  // Pula para a próxima iteração quando i é par
        }
        std::cout << "i = " << i << std::endl;
    }
    return 0;
}
```

4. Uso em Laço `do-while`

- Definição: No laço `do-while`, a instrução `continue` faz com que a execução salte de volta para a condição do laço.

- Exemplo:

```
int main() {
    int i = 0;

    do {
        ++i;
        if (i % 2 == 0) {
            continue;  // Pula para a próxima iteração quando i é par
        }
        std::cout << "i = " << i << std::endl;
    } while (i < 10);

    return 0;
}
```

5. Instrução `continue` em Laços Aninhados

- Definição: Em laços aninhados, a instrução `continue` afeta apenas o laço no qual está inserida.
- Exemplo:

```
int main() {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (j == 1) {
                continue;  // Pula para a próxima iteração do laço interno quando j é 1
            }
            std::cout << "(" << i << ", " << j << ")" << std::endl;
```

```
        }
    }
    return 0;
}
```

Dicas de Boas Práticas

- Legibilidade: Use a instrução `continue` com moderação para evitar tornar o fluxo do programa confuso.
- Clareza: Certifique-se de que o uso de `continue` seja claramente intencional e facilite a leitura e manutenção do código.
- Alternativas: Considere reestruturar o código para evitar `continue` se isso melhorar a clareza do fluxo lógico.

Esta seção abrange os conceitos sobre a instrução `continue` em C++. Para mais detalhes, consulte a documentação oficial: https://en.cppreference.com/w/cpp/language/continue