Instrução Switch em C++

Introdução

A instrução `switch` em C++ é uma estrutura de controle de fluxo que permite selecionar uma entre várias alternativas com base no valor de uma expressão. É útil quando há múltiplos caminhos de execução possíveis que dependem do valor de uma variável.

1. Definição e Sintaxe

- Definição: A instrução `switch` avalia uma expressão e executa o bloco de código associado ao valor correspondente.
- Sintaxe:

```
switch (expressão) {
    case valor1:
        // Bloco de código executado se expressão == valor1
        break;
    case valor2:
        // Bloco de código executado se expressão == valor2
        break;
    // ...
    default:
        // Bloco de código executado se nenhum valor corresponder
        break;
}
```

2. Exemplo Simples

- Exemplo:

```
int main() {
    int dia = 4;

    switch (dia) {
        case 1:
            std::cout << "Segunda-feira" << std::endl;
            break;
        case 2:
            std::cout << "Terça-feira" << std::endl;
            break;
        case 3:
            std::cout << "Quarta-feira" << std::endl;
            break;
        case 4:
            std::cout << "Quinta-feira" << std::endl;
            break;
        case 5:
            std::cout << "Sexta-feira" << std::endl;
            break;
        case 6:
            std::cout << "Sábado" << std::endl;
            break;
        case 7:
```

```
            std::cout << "Domingo" << std::endl;
            break;
        default:
            std::cout << "Dia inválido" << std::endl;
            break;
    }

    return 0;
}
```

3. Importância do `break`

- Definição: A instrução `break` é usada para sair do bloco `switch` após a execução do caso correspondente. Sem `break`, a execução continua nos casos subsequentes (comportamento de fall-through).
- Exemplo Sem `break`:

```
int main() {
    int dia = 4;

    switch (dia) {
        case 1:
            std::cout << "Segunda-feira" << std::endl;
        case 2:
            std::cout << "Terça-feira" << std::endl;
        case 3:
            std::cout << "Quarta-feira" << std::endl;
```

```
        case 4:
            std::cout << "Quinta-feira" << std::endl;
        case 5:
            std::cout << "Sexta-feira" << std::endl;
        case 6:
            std::cout << "Sábado" << std::endl;
        case 7:
            std::cout << "Domingo" << std::endl;
        default:
            std::cout << "Dia inválido" << std::endl;
    }

    return 0;
}
```

4. Uso do `default`

- Definição: O caso `default` é opcional, mas é uma boa prática usá-lo para lidar com valores não esperados.
- Exemplo:

```
int main() {
    int opcao = 3;

    switch (opcao) {
        case 1:
            std::cout << "Opção 1 selecionada" << std::endl;
```

```
            break;
        case 2:
            std::cout << "Opção 2 selecionada" << std::endl;
            break;
        default:
            std::cout << "Opção inválida" << std::endl;
            break;
    }

    return 0;
}
```

5. Casos com a Mesma Ação

- Definição: Múltiplos casos podem ser combinados para executar o mesmo bloco de código.
- Exemplo:

```
int main() {
    char letra = 'A';

    switch (letra) {
        case 'A':
        case 'a':
            std::cout << "A primeira letra do alfabeto" << std::endl;
            break;
        case 'B':
        case 'b':
```

```
            std::cout << "A segunda letra do alfabeto" << std::endl;
            break;
        default:
            std::cout << "Outra letra" << std::endl;
            break;
    }

    return 0;
}
```

6. Instrução `switch` com Inicialização (C++17)

- Definição: Desde o C++17, a instrução `switch` pode incluir uma inicialização, permitindo a definição de variáveis limitadas ao escopo do `switch`.
- Sintaxe:

```
switch (auto valor = expressão; valor) {
    case valor1:
        // Bloco de código
        break;
    // ...
}
```

- Exemplo:

```
int main() {
    switch (int x = 10; x) {
        case 10:
```

```
            std::cout << "x é 10" << std::endl;
            break;
        default:
            std::cout << "x não é 10" << std::endl;
            break;
    }

    return 0;
}
```

Dicas de Boas Práticas

- Consistência: Sempre use `break` para evitar comportamento de fall-through, a menos que seja intencional.
- Clareza: Use `default` para lidar com valores não esperados e melhorar a robustez do código.
- Legibilidade: Organize os casos de forma clara e lógica para facilitar a leitura e manutenção.

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