Templates em C++

Manipulação de Exceções em C++

Introdução

A manipulação de exceções em C++ é feita utilizando blocos catch, que capturam e tratam exceções lançadas em blocos try. Isso permite que você trate erros de forma controlada, separando o código de tratamento de erros do código principal.

1. Definição e Sintaxe

- Definição: Blocos catch são usados para capturar e tratar exceções lançadas por um bloco try. Cada bloco catch pode capturar um tipo específico de exceção.
- Sintaxe:

```
try {
    // Código que pode lançar uma exceção
} catch (const std::exception& e) {
    // Código para tratar a exceção
}
```

2. Uso Básico do Bloco catch

- Definição: Envolva o código que pode lançar exceções em um bloco try e capture as exceções com blocos catch.
- Exemplo:

```
int main() {
    try {
        throw std::runtime_error("Erro ocorrido");
    } catch (const std::runtime_error& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

3. Múltiplos Blocos catch

- Definição: Você pode usar múltiplos blocos catch para capturar diferentes tipos de exceções.
- Exemplo:

```
int main() {
    try {
        throw std::runtime_error("Erro de tempo de execução");
    } catch (const std::logic_error& e) {
        std::cout << "Exceção lógica capturada: " << e.what() << std::endl;
    } catch (const std::runtime_error& e) {
        std::cout << "Exceção de tempo de execução capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

4. Bloco catch Genérico

- Definição: Você pode usar um bloco catch genérico para capturar qualquer exceção.
- Exemplo:

```
int main() {
    try {
        throw std::runtime_error("Erro ocorrido");
    } catch (const std::exception& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    } catch (...) {
        std::cout << "Exceção desconhecida capturada" << std::endl;
    }
    return 0;
}
```

5. Re-lançamento de Exceções

- Definição: Você pode re-lançar uma exceção capturada usando a palavra-chave throw sem argumentos dentro de um bloco catch.
- Exemplo:

```
int main() {
    try {
        try {
            throw std::runtime_error("Erro original");
        } catch (const std::runtime_error& e) {
            std::cout << "Tratando e re-lançando: " << e.what() << std::endl;
```

```
            throw;  // Re-lança a exceção
        }
    } catch (const std::runtime_error& e) {
        std::cout << "Exceção re-lançada capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

6. Captura de Exceções por Referência

- Definição: É uma boa prática capturar exceções por referência para evitar cópias desnecessárias e preservar o polimorfismo.
- Exemplo:

```
int main() {
    try {
        throw std::runtime_error("Erro de tempo de execução");
    } catch (const std::runtime_error& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

7. Captura de Exceções por Valor

- Definição: Embora seja menos comum, você também pode capturar exceções por valor. No

entanto, isso pode resultar em cópias desnecessárias.

- Exemplo:

```
int main() {
    try {
        throw std::runtime_error("Erro de tempo de execução");
    } catch (std::runtime_error e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

8. Captura de Ponteiros de Exceção

- Definição: Em alguns casos, pode ser necessário capturar exceções lançadas como ponteiros.

- Exemplo:

```
int main() {
    try {
        throw new std::runtime_error("Erro de tempo de execução");
    } catch (std::runtime_error* e) {
        std::cout << "Exceção capturada: " << e->what() << std::endl;
        delete e;  // Lembre-se de liberar a memória
    }
    return 0;
}
```

Templates em C++

9. Captura de Exceções Especificadas pelo Usuário

- Definição: Você pode capturar exceções definidas pelo usuário da mesma forma que captura exceções padrão.
- Exemplo:

```
class MinhaExcecao : public std::exception {
public:
    const char* what() const noexcept override {
        return "Minha exceção personalizada";
    }
};

int main() {
    try {
        throw MinhaExcecao();
    } catch (const MinhaExcecao& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

Dicas de Boas Práticas

- Uso Apropriado de Blocos catch: Utilize blocos catch para capturar e tratar exceções de maneira adequada, garantindo a robustez do programa.

- Clareza e Manutenção: Mantenha o código de tratamento de exceções claro e bem documentado para facilitar a leitura e a manutenção do código.
- Evite Blocos catch Vazios: Certifique-se de tratar adequadamente as exceções capturadas e não simplesmente ignorá-las.

Esta seção abrange os conceitos sobre a manipulação de exceções em C++. Para mais detalhes, consulte a documentação oficial: https://en.cppreference.com/w/cpp/language/catch