Templates em C++

Especificador noexcept em C++

Introdução

O especificador noexcept em C++ é utilizado para indicar que uma função não lança exceções. Isso pode melhorar a otimização do código e a segurança, pois o compilador pode fazer suposições sobre o comportamento da função.

1. Definição e Sintaxe

- Definição: O especificador noexcept pode ser usado para declarar que uma função não lança exceções.
- Sintaxe:

```
void func() noexcept;
```

2. Uso Básico do noexcept

- Definição: O especificador noexcept pode ser aplicado a funções para indicar que elas não lançarão exceções.
- Exemplo:

```
void func() noexcept {
    // Código que não lança exceções
}
```

```
int main() {
    func();
    return 0;
}
```

3. Uso Condicional do noexcept

- Definição: Você pode usar uma expressão condicional com noexcept para indicar que uma função lança exceções apenas sob certas condições.
- Exemplo:

```
void func() noexcept(true) {
    // Código que não lança exceções
}

void outraFunc() noexcept(sizeof(int) == 4) {
    // Código que pode lançar exceções dependendo da condição
}

int main() {
    func();
    outraFunc();
    return 0;
}
```

4. Uso com Funções Membro

Templates em C++

- Definição: O especificador noexcept pode ser aplicado a funções membro, incluindo construtores e destrutores.

- Exemplo:

```
class Exemplo {
public:
    Exemplo() noexcept {
        // Construtor que não lança exceções
    }

    ~Exemplo() noexcept {
        // Destrutor que não lança exceções
    }

    void metodo() noexcept {
        // Método que não lança exceções
    }
};

int main() {
    Exemplo e;
    e.metodo();
    return 0;
}
```

Templates em C++

5. Funções noexcept e Otimização

- Definição: O uso de noexcept permite ao compilador otimizar o código melhor, já que ele pode fazer suposições sobre a ausência de exceções.

- Exemplo:

```
void func() noexcept {
    // Código otimizado pelo compilador
}

int main() {
    func();
    return 0;
}
```

6. Verificação de noexcept

- Definição: A expressão noexcept pode ser usada para verificar se uma função é noexcept.

- Exemplo:

```
void func() noexcept {}

int main() {
    std::cout << std::boolalpha << noexcept(func()) << std::endl;  // Saída: true
    return 0;
}
```

Templates em C++

7. Lançamento de Exceções em Funções noexcept

- Definição: Lançar uma exceção em uma função marcada como noexcept resulta na chamada de std::terminate.
- Exemplo:

```
void func() noexcept {
    throw std::runtime_error("Erro");  // Chama std::terminate
}

int main() {
    try {
        func();
    } catch (...) {
        std::cout << "Isso nunca será impresso" << std::endl;
    }
    return 0;
}
```

8. Especificador noexcept e Sobrecarga

- Definição: O especificador noexcept faz parte da assinatura da função e pode ser usado para sobrecarregar funções.
- Exemplo:

```
void func() noexcept {
    // Função que não lança exceções
```

```
}

void func() {
    // Função que pode lançar exceções
}

int main() {
    func();  // Chama a versão que pode lançar exceções
    return 0;
}
```

Dicas de Boas Práticas

- Uso Apropriado de noexcept: Utilize noexcept para funções que você sabe que não lançarão exceções, melhorando a otimização e a segurança do código.
- Clareza e Manutenção: Mantenha o uso de noexcept claro e bem documentado para facilitar a leitura e a manutenção do código.
- Verificação de noexcept: Use a expressão noexcept para verificar se uma função é realmente noexcept, garantindo a correção do programa.

Esta seção abrange os conceitos sobre o especificador noexcept em C++. Para mais detalhes, consulte a documentação oficial: https://en.cppreference.com/w/cpp/language/noexcept_spec