Templates em C++

Requer Expressão (Requires Expression) em C++

Introdução

A cláusula requires em C++20 introduz um mecanismo para especificar requisitos complexos para parâmetros de template, melhorando a legibilidade e a segurança do código. A requires expression é uma maneira de definir condições que os parâmetros de template devem satisfazer.

1. Definição e Sintaxe

- Definição: A requires expression é usada para definir requisitos em um contexto de template, garantindo que os parâmetros atendam a certas condições antes que o código seja instanciado.
- Sintaxe:

```
template <typename T>
concept Conceito = requires(T a) {
    { a + a } -> std::convertible_to<int>;
};
```

2. Exemplo Básico de requires

- Definição: Um exemplo básico que demonstra como usar a cláusula requires para impor requisitos a um parâmetro de template.
- Exemplo:

```
template <typename T>
```

```
concept Incrementavel = requires(T a) {
    { a++ } -> std::same_as<T&>;
};

template <Incrementavel T>
void incrementar(T& valor) {
    ++valor;
}

int main() {
    int x = 5;
    incrementar(x);  // Válido
    return 0;
}
```

3. Expressões requires com Múltiplos Requisitos

- Definição: A requires expression pode incluir múltiplos requisitos para verificar diversas condições.
- Exemplo:

```
template <typename T>
concept SomavelMultiplicavel = requires(T a, T b) {
    { a + b } -> std::convertible_to<T>;
    { a * b } -> std::convertible_to<T>;
};
```

```
template <SomavelMultiplicavel T>
T somarMultiplicar(T a, T b) {
    return (a + b) * (a + b);
}

int main() {
    std::cout << somarMultiplicar(2, 3) << std::endl;  // Válido
    return 0;
}
```

4. Uso de requires com std::integral e std::floating_point

- Definição: Conceitos pré-definidos como std::integral e std::floating_point podem ser usados em expressões requires para verificar tipos inteiros e de ponto flutuante.
- Exemplo:

```
#include <concepts>

template <std::integral T>
T dobro(T valor) {
    return valor * 2;
}

template <std::floating_point T>
T metade(T valor) {
    return valor / 2.0;
```

```
}

int main() {
    std::cout << dobro(5) << std::endl;        // Válido
    std::cout << metade(5.0) << std::endl;     // Válido
    return 0;
}
```

5. Expressões requires em Funções Membros

- Definição: A requires expression pode ser usada dentro de funções membros para verificar condições específicas.
- Exemplo:

```
template <typename T>
concept TemTamanho = requires(T a) {
    { a.size() } -> std::convertible_to<std::size_t>;
};

template <TemTamanho T>
std::size_t tamanho(const T& container) {
    return container.size();
}

int main() {
    std::vector<int> v = {1, 2, 3};
```

```
    std::cout << tamanho(v) << std::endl;  // Válido
    return 0;
}
```

6. Expressões requires em Classes Template

- Definição: A requires expression pode ser usada para definir requisitos em classes template, garantindo que os membros atendam às condições necessárias.
- Exemplo:

```
template <typename T>
concept Imprimivel = requires(T a) {
    { std::cout << a } -> std::convertible_to<std::ostream&>;
};

template <Imprimivel T>
class Impressora {
public:
    void imprimir(const T& valor) {
        std::cout << valor << std::endl;
    }
};

int main() {
    Impressora<int> imp;
    imp.imprimir(10);  // Válido
```

```
    return 0;
}
```

7. Uso de requires com Funções Sobrecargas

- Definição: A requires expression pode ser usada para selecionar a sobrecarga de função apropriada com base nos requisitos.
- Exemplo:

```
template <typename T>
concept Addable = requires(T a, T b) {
    { a + b } -> std::convertible_to<T>;
};

template <typename T>
concept Multipliable = requires(T a, T b) {
    { a * b } -> std::convertible_to<T>;
};

template <Addable T>
T operar(T a, T b) {
    return a + b;
}

template <Multipliable T>
T operar(T a, T b) {
```

```
    return a * b;
}

int main() {
    std::cout << operar(3, 4) << std::endl;  // Seleciona a sobrecarga de soma
    std::cout << operar(3.0, 4.0) << std::endl;  // Seleciona a sobrecarga de multiplicação
    return 0;
}
```

Dicas de Boas Práticas

- Uso Consistente de requires: Utilize a cláusula requires para especificar claramente os requisitos dos parâmetros de template.
- Clareza e Manutenção: Mantenha as expressões requires bem documentadas para facilitar a leitura e a manutenção do código.
- Verificação de Requisitos: Use expressões requires para garantir que os parâmetros de template atendam aos requisitos necessários, evitando erros de compilação e execução.

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