Templates em C++

Parâmetros e Argumentos de Template em C++

Introdução

Templates em C++ permitem a criação de código genérico que pode operar com diferentes tipos de dados. Os parâmetros e argumentos de template são essenciais para definir e utilizar templates de forma flexível e eficiente.

1. Definição e Sintaxe

- Definição: Os parâmetros de template são especificados na declaração de um template e determinam os tipos ou valores que podem ser passados para o template.
- Sintaxe:

```
template <typename T>
class NomeClasse {
    // Definição da classe
};

template <typename T>
T nomeFuncao(T argumento) {
    // Definição da função
}
```

2. Parâmetros de Tipo

Templates em C++

- Definição: Parâmetros de tipo são utilizados para especificar tipos genéricos em templates de classes e funções.

- Exemplo:

```
template <typename T>
class Caixa {
private:
    T valor;

public:
    Caixa(T v) : valor(v) {}
    T getValor() { return valor; }
};

int main() {
    Caixa<int> caixaInt(10);
    Caixa<double> caixaDouble(10.5);
    std::cout << caixaInt.getValor() << std::endl; // Uso do template de classe com int
    std::cout << caixaDouble.getValor() << std::endl; // Uso do template de classe com double
    return 0;
}
```

3. Parâmetros Não-Tipo

- Definição: Parâmetros não-tipo permitem a passagem de valores constantes, como inteiros ou

ponteiros, para um template.

- Exemplo:

```
template <typename T, int N>
class Array {
private:
    T arr[N];

public:
    T& operator[](int index) { return arr[index]; }
};

int main() {
    Array<int, 10> arrayInt;
    arrayInt[0] = 1;
    std::cout << arrayInt[0] << std::endl; // Uso do template de classe com parâmetro não-tipo
    return 0;
}
```

4. Parâmetros de Template Template

- Definição: Permitem que templates aceitem outros templates como parâmetros.
- Exemplo:

```
template <template <typename> class Container, typename T>
class CaixaDeCaixas {
private:
```

```
    Container<T> caixa;

public:
    CaixaDeCaixas(T v) : caixa(v) {}
    T getValor() { return caixa.getValor(); }
};

template <typename T>
class Caixa {
private:
    T valor;

public:
    Caixa(T v) : valor(v) {}
    T getValor() { return valor; }
};

int main() {
    CaixaDeCaixas<Caixa, int> caixaInt(10);
    std::cout << caixaInt.getValor() << std::endl; // Uso de template template parameter
    return 0;
}
```

5. Aliases de Templates

Templates em C++

- Definição: Introduzido no C++11, permite a criação de aliases para templates, tornando o código mais legível.

- Exemplo:

```
template <typename T>
using Vec = std::vector<T>;


int main() {
    Vec<int> vetorInt;
    vetorInt.push_back(10);
    std::cout << vetorInt[0] << std::endl; // Uso do alias de template
    return 0;
}
```

6. Especificação Explícita de Argumentos de Template

- Definição: Argumentos de template podem ser especificados explicitamente ao chamar uma função ou instanciar uma classe template.

- Exemplo:

```
template <typename T>
T max(T a, T b) {
    return (a > b) ? a : b;
}


int main() {
    std::cout << max<int>(10, 20) << std::endl; // Especificação explícita do argumento de template
```

```
    return 0;
}
```

7. Dedução Automática de Argumentos de Template

- Definição: O compilador pode deduzir automaticamente os argumentos de template com base nos argumentos passados para a função ou classe.
- Exemplo:

```
template <typename T>
T max(T a, T b) {
    return (a > b) ? a : b;
}

int main() {
    std::cout << max(10, 20) << std::endl; // Dedução automática do argumento de template
    return 0;
}
```

Dicas de Boas Práticas

- Reutilização de Código: Use parâmetros e argumentos de template para criar código genérico e reutilizável.
- Especificação Clara: Seja explícito ao especificar argumentos de template quando necessário para evitar ambiguidades.
- Evitar Complexidade: Mantenha os templates simples e claros para facilitar a leitura e a

manutenção do código.

Esta seção abrange os conceitos sobre parâmetros e argumentos de template em C++. Para mais detalhes, consulte a documentação oficial: https://en.cppreference.com/w/cpp/language/template_parameters