# Templates em C++

## Template de Função em C++

Introdução

Templates de função em C++ permitem a criação de funções genéricas que podem operar com qualquer tipo de dado. Isso é extremamente útil para escrever código reutilizável e flexível.

1. Definição e Sintaxe

- Definição: Um template de função é um modelo de função que pode ser instanciado com diferentes tipos de dados.
- Sintaxe:

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

2. Exemplo Básico de Template de Função

- Definição: Um template de função básico que retorna o maior de dois valores.
- Exemplo:

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

```
}

int main() {
    int x = 10, y = 20;
    double a = 10.5, b = 20.5;
    std::cout << max(x, y) << std::endl; // Uso do template de função com int
    std::cout << max(a, b) << std::endl; // Uso do template de função com double
    return 0;
}
```

3. Especificação Explícita de Argumentos de Template

- Definição: Argumentos de template podem ser especificados explicitamente ao chamar uma função 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;
}
```

**Templates em C++**

4. Dedução Automática de Argumentos de Template

- Definição: O compilador pode deduzir automaticamente os argumentos de template com base nos argumentos passados para a função.
- 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;
}
```

5. Template de Função com Múltiplos Parâmetros

- Definição: Templates de função podem ter múltiplos parâmetros de template.
- Exemplo:

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

int main() {
```

```
    int x = 10;
    double y = 20.5;
    std::cout << max(x, y) << std::endl; // Uso do template de função com múltiplos parâmetros
    return 0;
}
```

6. Template de Função com Parâmetros Não-Tipo

- Definição: Templates de função podem ter parâmetros de template que não são tipos.
- Exemplo:

```
template <typename T, int N>
T arraySum(T (&arr)[N]) {
    T sum = 0;
    for (int i = 0; i < N; ++i) {
        sum += arr[i];
    }
    return sum;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    std::cout << arraySum(arr) << std::endl; // Uso do template de função com parâmetro não-tipo
    return 0;
}
```

**Templates em C++**

7. Funções Template Especiais

- Definição: É possível especializar funções template para tipos específicos.
- Exemplo:

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

template <>
const char* max<const char*>(const char* a, const char* b) {
    return (std::strcmp(a, b) > 0) ? a : b;
}

int main() {
    const char* x = "apple";
    const char* y = "banana";
    std::cout << max(x, y) << std::endl; // Uso da função template especializada
    return 0;
}
```

8. Sobrecarga de Funções Template

- Definição: Funções template podem ser sobrecarregadas como funções regulares.
- Exemplo:

**Templates em C++**

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

template <typename T>
T max(T a, T b, T c) {
    return max(max(a, b), c);
}

int main() {
    std::cout << max(10, 20) << std::endl; // Uso da função template com dois parâmetros
     std::cout << max(10, 20, 15) << std::endl; // Uso da função template sobrecarregada com três parâmetros
    return 0;
}
```

Dicas de Boas Práticas

- Reutilização de Código: Use templates de função para criar funções genéricas e reutilizáveis.
- Especialização: Especialize funções template quando necessário para lidar com tipos específicos.
- Clareza e Manutenção: Mantenha os templates de função claros e bem documentados para facilitar a manutenção do código.

Esta seção abrange os conceitos sobre templates de função em C++. Para mais detalhes, consulte

# Templates em C++

a documentação oficial: https://en.cppreference.com/w/cpp/language/function_template