Templates em C++

Template de Membros em C++

Introdução

Templates de membros permitem que membros individuais de uma classe, como funções ou classes aninhadas, sejam templates. Isso proporciona maior flexibilidade e reutilização de código dentro de uma classe.

1. Definição e Sintaxe

- Definição: Templates de membros são membros de uma classe que podem ser templates.
- Sintaxe:

```
class NomeClasse {
    template <typename T>
    void nomeFuncao(T argumento);
};
```

2. Exemplo Básico de Template de Membro

- Definição: Um template de membro básico que define uma função de membro template.
- Exemplo:

```
class Exemplo {
public:
    template <typename T>
```

```
    void funcaoTemplate(T argumento) {
        std::cout << argumento << std::endl;
    }
};

int main() {
    Exemplo obj;
    obj.funcaoTemplate(10);     // Uso do template de membro com int
    obj.funcaoTemplate(3.14);   // Uso do template de membro com double
    obj.funcaoTemplate("texto"); // Uso do template de membro com string
    return 0;
}
```

3. Template de Função Membro com Múltiplos Parâmetros

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

```
class Exemplo {
public:
    template <typename T, typename U>
    void funcaoTemplate(T primeiro, U segundo) {
        std::cout << primeiro << " " << segundo << std::endl;
    }
};
```

```
int main() {
    Exemplo obj;
    obj.funcaoTemplate(10, 20.5);     // Uso do template de membro com int e double
    obj.funcaoTemplate("texto", 42);  // Uso do template de membro com string e int
    return 0;
}
```

4. Template de Classe Membro

- Definição: Classes aninhadas dentro de uma classe podem ser templates.
- Exemplo:

```
class Exemplo {
public:
    template <typename T>
    class ClasseAninhada {
    private:
        T valor;

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

int main() {
```

```
    Exemplo::ClasseAninhada<int> objInt(10);
    Exemplo::ClasseAninhada<double> objDouble(10.5);
    std::cout << objInt.getValor() << std::endl;    // Uso do template de classe membro com int
        std::cout << objDouble.getValor() << std::endl; // Uso do template de classe membro com double
    return 0;
 }
```

5. Template de Função Membro Especializada

- Definição: Funções membro template podem ser especializadas para tipos específicos.
- Exemplo:

```
 class Exemplo {
 public:
    template <typename T>
    void funcaoTemplate(T argumento) {
        std::cout << "Genérico: " << argumento << std::endl;
    }

    template <>
    void funcaoTemplate(int argumento) {
        std::cout << "Especializado para int: " << argumento << std::endl;
    }
 };
```

```
int main() {
    Exemplo obj;
    obj.funcaoTemplate(10);      // Chama a versão especializada para int
    obj.funcaoTemplate(3.14);    // Chama a versão genérica
    obj.funcaoTemplate("texto"); // Chama a versão genérica
    return 0;
}
```

6. Uso de Templates de Membro em Classes Template

- Definição: Templates de membro podem ser usados dentro de classes que também são templates.
- Exemplo:

```
template <typename T>
class Exemplo {
public:
    template <typename U>
    void funcaoTemplate(U argumento) {
        std::cout << argumento << std::endl;
    }
};

int main() {
    Exemplo<int> obj;
    obj.funcaoTemplate(10);     // Uso do template de membro com int
```

```
    obj.funcaoTemplate(3.14);   // Uso do template de membro com double
    obj.funcaoTemplate("texto"); // Uso do template de membro com string
    return 0;
}
```

Dicas de Boas Práticas

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

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