Classes em C++

Classes Aninhadas em C++

Introdução

Classes aninhadas em C++ são classes definidas dentro do escopo de outra classe. Elas permitem organizar o código de maneira hierárquica e encapsular a lógica que é especificamente relevante para a classe externa. Classes aninhadas podem ser úteis para melhorar a modularidade e a legibilidade do código.

1. Definição e Sintaxe

- Definição: Uma classe aninhada é uma classe declarada dentro do escopo de outra classe.
- Sintaxe:

```
class Externa {
public:
    class Interna {
        // Definição da classe aninhada
    };
};
```

2. Acesso a Membros da Classe Externa

- Definição: A classe aninhada tem acesso direto aos membros da classe externa se estes forem públicos ou protegidos.

Classes em C++

- Exemplo:

```
class Externa {
private:
    int valor;

public:
    Externa(int v) : valor(v) {}

    class Interna {
    public:
        void mostrarValor(const Externa& e) {
            std::cout << "Valor: " << e.valor << std::endl;
        }
    };
};

int main() {
    Externa e(10);
    Externa::Interna i;
    i.mostrarValor(e);
    return 0;
}
```

3. Acesso a Membros da Classe Aninhada

Classes em C++

- Definição: A classe externa não tem acesso direto aos membros da classe aninhada.

- Exemplo:

```
class Externa {
public:
    class Interna {
    private:
        int valor;
    public:
        Interna(int v) : valor(v) {}
        int obterValor() const { return valor; }
    };

    void mostrarInternoValor(const Interna& i) {
        std::cout << "Valor Interno: " << i.obterValor() << std::endl;
    }
};

int main() {
    Externa::Interna i(20);
    Externa e;
    e.mostrarInternoValor(i);
    return 0;
}
```

4. Classes Aninhadas e Escopo

Classes em C++

- Definição: A classe aninhada está no escopo da classe externa, mas é um tipo completo separado.

- Exemplo:

```
class Externa {
public:
    int valor;

    class Interna {
    public:
        int valorInterno;
        Interna(int v) : valorInterno(v) {}
    };

    Externa(int v) : valor(v) {}
};

int main() {
    Externa e(30);
    Externa::Interna i(40);
    std::cout << "Valor Externo: " << e.valor << std::endl;
    std::cout << "Valor Interno: " << i.valorInterno << std::endl;
    return 0;
}
```

Classes em C++

5. Uso de Classes Aninhadas

- Organização: Classes aninhadas podem ser usadas para organizar o código de maneira lógica, agrupando funcionalidades relacionadas.
- Encapsulamento: Elas permitem encapsular detalhes de implementação que não precisam ser expostos fora da classe externa.
- Exemplo:

```
class Grafico {
public:
    class Ponto {
    private:
        int x, y;
    public:
        Ponto(int x, int y) : x(x), y(y) {}
        void mostrar() const {
            std::cout << "Ponto(" << x << ", " << y << ")" << std::endl;
        }
    };

    void desenharPonto(int x, int y) {
        Ponto p(x, y);
        p.mostrar();
    }
};
```

```
int main() {
    Grafico g;
    g.desenharPonto(10, 20);
    return 0;
}
```

Dicas de Boas Práticas

- Modularidade: Use classes aninhadas para manter o código modular e organizado.
- Encapsulamento: Aproveite classes aninhadas para encapsular detalhes de implementação que não precisam ser expostos externamente.
- Legibilidade: Certifique-se de que o uso de classes aninhadas melhora a legibilidade e a manutenção do código.

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