Classes em C++

Membros de Dados Não Estáticos em C++

Introdução

Membros de dados não estáticos são variáveis que pertencem a instâncias de uma classe. Cada instância da classe tem suas próprias cópias desses membros. Eles são usados para armazenar o estado de um objeto e são uma parte fundamental da programação orientada a objetos em C++.

1. Definição e Sintaxe

- Definição: Membros de dados não estáticos são variáveis declaradas dentro de uma classe sem a palavra-chave `static`.
- Sintaxe:

```
class NomeClasse {
public:
    tipo membro;
};
```

2. Declaração e Inicialização

- Declaração: Membros de dados não estáticos são declarados dentro da definição da classe.
- Inicialização: Eles podem ser inicializados usando inicializadores de membro ou dentro do construtor da classe.
- Exemplo:

Classes em C++

```
class Pessoa {
public:
    std::string nome;
    int idade;

    Pessoa(const std::string& nome, int idade) : nome(nome), idade(idade) {}
};

int main() {
    Pessoa p("Marcos", 20);
    std::cout << "Nome: " << p.nome << ", Idade: " << p.idade << std::endl;
    return 0;
}
```

3. Acesso aos Membros de Dados

- Definição: Membros de dados não estáticos são acessados através de objetos da classe usando o operador de acesso a membro (`.`) ou o operador de seta (`->`) se o objeto for um ponteiro.
- Exemplo:

```
Pessoa p("Marcos", 20);
std::cout << p.nome << std::endl;
Pessoa* ptr = &p;
std::cout << ptr->idade << std::endl;
```

4. Modificadores de Acesso

Classes em C++

- Definição: Modificadores de acesso (`public`, `protected`, `private`) controlam a visibilidade dos membros de dados não estáticos.

- Exemplo:

```
class ContaBancaria {
private:
    double saldo;

public:
    ContaBancaria(double saldoInicial) : saldo(saldoInicial) {}

    double obterSaldo() const {
        return saldo;
    }
};

int main() {
    ContaBancaria conta(1000.0);
    std::cout << "Saldo: " << conta.obterSaldo() << std::endl;
    return 0;
}
```

5. Inicialização de Membros de Dados

- Inicializadores de Membro: Membros de dados não estáticos podem ser inicializados diretamente

na declaração da classe a partir do C++11.

- Exemplo:

```
class Produto {
public:
    std::string nome = "Desconhecido";
    double preco = 0.0;

    Produto() = default;
    Produto(const std::string& nome, double preco) : nome(nome), preco(preco) {}
};

int main() {
    Produto p;
    std::cout << "Nome: " << p.nome << ", Preço: " << p.preco << std::endl;
    return 0;
}
```

6. Constantes e Referências

- Constantes: Membros de dados não estáticos podem ser constantes (`const`), indicando que seu valor não pode ser modificado após a inicialização.
- Referências: Membros de dados não estáticos podem ser referências, que devem ser inicializadas na lista de inicialização do construtor.
- Exemplo:

```
class Exemplo {
```

```
  public:
      const int constante;
      int& referencia;

      Exemplo(int valor, int& ref) : constante(valor), referencia(ref) {}
  };

  int main() {
      int x = 10;
      Exemplo e(5, x);
      std::cout << "Constante: " << e.constante << ", Referência: " << e.referencia << std::endl;
      return 0;
  }
```

7. Membros de Dados Mutáveis

- Definição: O especificador `mutable` permite que um membro de dados não estático seja modificado mesmo se ele fizer parte de um objeto constante.
- Exemplo:

```
  class Exemplo {
  public:
      mutable int contador;

      Exemplo() : contador(0) {}
```

```
    void incrementar() const {
        contador++;
    }
};

int main() {
    const Exemplo e;
    e.incrementar();
    std::cout << "Contador: " << e.contador << std::endl;
    return 0;
}
```

Dicas de Boas Práticas

- Encapsulamento: Mantenha os membros de dados não estáticos privados e forneça métodos públicos para acesso e modificação.
- Inicialização: Sempre inicialize membros de dados não estáticos para evitar valores indeterminados.
- Consistência: Use inicializadores de membro para inicializar membros de dados de forma consistente e clara.

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