Classes em C++

Especificadores de Acesso em C++

Introdução

Especificadores de acesso em C++ são usados para definir a acessibilidade dos membros de uma classe. Eles determinam quais partes do programa podem acessar ou modificar os membros de dados e funções membros de uma classe. Os três especificadores de acesso principais são: `public`, `protected` e `private`.

1. Definição e Sintaxe

- Definição: Especificadores de acesso são palavras-chave que definem o nível de acesso aos membros de uma classe.
- Sintaxe:

```
class NomeClasse {
public:
    // Membros públicos
protected:
    // Membros protegidos
private:
    // Membros privados
};
```

2. Acesso Público (`public`)

Classes em C++

- Definição: Membros públicos são acessíveis de qualquer parte do programa.
- Exemplo:

```
class ExemploPublico {
public:
    int valorPublico;
    void funcaoPublica() {
        std::cout << "Funcao Publica" << std::endl;
    }
};

int main() {
    ExemploPublico obj;
    obj.valorPublico = 10;
    obj.funcaoPublica();
    return 0;
}
```

3. Acesso Protegido (`protected`)

- Definição: Membros protegidos são acessíveis dentro da própria classe e em classes derivadas, mas não fora dessas classes.
- Exemplo:

```
class Base {
protected:
```

```
    int valorProtegido;
};

class Derivada : public Base {
public:
    void acessarProtegido() {
        valorProtegido = 20;
    }
};

int main() {
    Derivada obj;
    obj.acessarProtegido();
    return 0;
}
```

4. Acesso Privado (`private`)

- Definição: Membros privados são acessíveis apenas dentro da própria classe. Eles não são acessíveis em classes derivadas ou em outras partes do programa.
- Exemplo:

```
class ExemploPrivado {
private:
    int valorPrivado;
public:
```

```
    void definirValor(int v) {
        valorPrivado = v;
    }
    int obterValor() {
        return valorPrivado;
    }
};

int main() {
    ExemploPrivado obj;
    obj.definirValor(30);
    std::cout << "Valor Privado: " << obj.obterValor() << std::endl;
    return 0;
}
```

5. Especificadores de Acesso em Herança

- Definição: Os especificadores de acesso também afetam a visibilidade dos membros de uma classe base na classe derivada.
- Tipos de Herança:
  - Herança Pública:

```
class Base {
public:
    int valorPublico;
protected:
```

```
    int valorProtegido;
private:
    int valorPrivado;
};

class Derivada : public Base {
    // valorPublico é público na Derivada
    // valorProtegido é protegido na Derivada
    // valorPrivado não é acessível na Derivada
};
```

- Herança Protegida:

```
class Derivada : protected Base {
    // valorPublico é protegido na Derivada
    // valorProtegido é protegido na Derivada
    // valorPrivado não é acessível na Derivada
};
```

- Herança Privada:

```
class Derivada : private Base {
    // valorPublico é privado na Derivada
    // valorProtegido é privado na Derivada
    // valorPrivado não é acessível na Derivada
};
```

6. Melhores Práticas

Classes em C++

- Encapsulamento: Use especificadores de acesso para encapsular os detalhes de implementação e proteger a integridade dos dados.
- Público para Interfaces: Torne públicos apenas os membros que fazem parte da interface da classe e que devem ser acessíveis fora da classe.
- Protegido para Herança: Use membros protegidos para informações que devem ser acessíveis em classes derivadas.
- Privado para Implementação: Use membros privados para detalhes de implementação que não devem ser expostos.

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