Classes em C++

Operador de Atribuição de Cópia em C++

Introdução

O operador de atribuição de cópia em C++ é usado para copiar os valores de um objeto para outro objeto existente do mesmo tipo. Ele é essencial para garantir que a cópia de objetos seja realizada corretamente, especialmente quando o objeto contém ponteiros ou outros recursos que requerem cópia profunda.

1. Definição e Sintaxe

- Definição: O operador de atribuição de cópia é um operador especial que atribui os valores de um objeto a outro objeto existente do mesmo tipo.
- Sintaxe:

```
class NomeClasse {
public:
    NomeClasse& operator=(const NomeClasse& outro);  // Declaração do operador de atribuição de cópia
};
```

2. Operador de Atribuição de Cópia Implicitamente Definido

- Definição: Se nenhum operador de atribuição de cópia for explicitamente definido, o compilador gera automaticamente um operador de atribuição de cópia padrão que realiza a cópia membro a

membro.

- Exemplo:

```
class Exemplo {
public:
    int valor;
};

int main() {
    Exemplo obj1;
    obj1.valor = 42;
    Exemplo obj2;
    obj2 = obj1;  // Operador de atribuição de cópia implicitamente definido
    return 0;
}
```

3. Operador de Atribuição de Cópia Explicitamente Definido

- Definição: Um operador de atribuição de cópia pode ser explicitamente definido pelo programador para realizar uma cópia profunda ou outras operações especiais durante a cópia.

- Exemplo:

```
class Exemplo {
public:
    int* ptr;

    Exemplo(int valor) {
```

```
        ptr = new int(valor);
    }

    // Operador de atribuição de cópia explicitamente definido
    Exemplo& operator=(const Exemplo& outro) {
        if (this == &outro) return *this;  // Auto-atribuição
        delete ptr;
        ptr = new int(*outro.ptr);  // Cópia profunda
        return *this;
    }

    ~Exemplo() {
        delete ptr;
    }
};

int main() {
    Exemplo obj1(42);
    Exemplo obj2(0);
    obj2 = obj1;  // Chama o operador de atribuição de cópia
    return 0;
}
```

4. Operadores de Atribuição e Herança

Classes em C++

- Definição: Quando se utiliza herança, é importante garantir que os operadores de atribuição de cópia das classes base e derivadas funcionem corretamente.

- Exemplo:

```
class Base {
public:
    int valorBase;

    Base(int valor) : valorBase(valor) {}

    Base& operator=(const Base& outro) {
        if (this != &outro) {
            valorBase = outro.valorBase;
        }
        return *this;
    }
};

class Derivada : public Base {
public:
    int valorDerivado;

    Derivada(int valorBase, int valorDerivado) : Base(valorBase), valorDerivado(valorDerivado) {}

    Derivada& operator=(const Derivada& outro) {
        if (this != &outro) {
```

```
            Base::operator=(outro);
            valorDerivado = outro.valorDerivado;
        }
        return *this;
    }
};

int main() {
    Derivada obj1(1, 2);
    Derivada obj2(3, 4);
    obj2 = obj1;  // Chama o operador de atribuição de cópia
    return 0;
}
```

5. Operador de Atribuição de Cópia `default`

- Definição: Um operador de atribuição de cópia pode ser explicitamente declarado como `default` para indicar que o compilador deve gerar a implementação padrão.
- Exemplo:

```
class Exemplo {
public:
    int valor;

    Exemplo(int v) : valor(v) {}
```

```
    Exemplo& operator=(const Exemplo& outro) = default;  // Solicita ao compilador que gere o operador de atribuição de cópia padrão
};

int main() {
    Exemplo obj1(42);
    Exemplo obj2(0);
    obj2 = obj1;  // Chama o operador de atribuição de cópia padrão
    return 0;
}
```

6. Desabilitando o Operador de Atribuição de Cópia

- Definição: O operador de atribuição de cópia pode ser desabilitado explicitamente para impedir a cópia de objetos.
- Exemplo:

```
class Exemplo {
public:
    Exemplo(int valor) {
        // Construtor
    }

    Exemplo& operator=(const Exemplo& outro) = delete;  // Desabilita o operador de atribuição de cópia
};
```

```
int main() {
    Exemplo obj1(42);
    Exemplo obj2(0);
    // obj2 = obj1;  // Erro: operador de atribuição de cópia está desabilitado
    return 0;
}
```

Dicas de Boas Práticas

- Cópia Profunda vs. Cópia Superficial: Use cópia profunda para evitar problemas com ponteiros e recursos dinâmicos.
- Desempenho: Avalie o impacto de desempenho ao definir operadores de atribuição de cópia, especialmente em classes com muitos membros ou recursos dinâmicos.
- Uso de `default`: Use `default` para indicar claramente que a implementação padrão do operador de atribuição de cópia deve ser gerada pelo compilador.

Esta seção abrange os conceitos sobre o operador de atribuição de cópia em C++. Para mais detalhes, consulte a documentação oficial: https://en.cppreference.com/w/cpp/language/copy_assignment