Classes em C++

Classes Abstratas em C++

Introdução

Classes abstratas em C++ são classes que não podem ser instanciadas diretamente. Elas servem como uma base para outras classes e contêm pelo menos uma função virtual pura. Classes abstratas são usadas para definir interfaces e forçar a implementação de certas funções nas classes derivadas.

1. Definição e Sintaxe

- Definição: Uma classe abstrata é uma classe que contém pelo menos uma função virtual pura. Ela não pode ser instanciada diretamente.
- Sintaxe:

```
class BaseAbstrata {
public:
    virtual void funcaoPura() = 0;  // Função virtual pura
};
```

2. Exemplo de Classe Abstrata

- Exemplo: Uma classe abstrata define funções que devem ser implementadas por classes derivadas.

```
#include <iostream>
```

Classes em C++

```
class Forma {
public:
    virtual void desenhar() = 0;  // Função virtual pura
};

class Circulo : public Forma {
public:
    void desenhar() override {
        std::cout << "Desenhar círculo" << std::endl;
    }
};

class Quadrado : public Forma {
public:
    void desenhar() override {
        std::cout << "Desenhar quadrado" << std::endl;
    }
};

int main() {
    Forma* f1 = new Circulo();
    Forma* f2 = new Quadrado();
    f1->desenhar();
    f2->desenhar();
```

```
    delete f1;
    delete f2;
    return 0;
}
```

3. Funções Virtuais Puras

- Definição: Funções virtuais puras são funções declaradas com `= 0` na classe base. Elas não têm implementação na classe base e devem ser implementadas nas classes derivadas.
- Exemplo:

```
class Animal {
public:
    virtual void fazerSom() = 0;  // Função virtual pura
};

class Cachorro : public Animal {
public:
    void fazerSom() override {
        std::cout << "Latido" << std::endl;
    }
};

class Gato : public Animal {
public:
    void fazerSom() override {
```

```
        std::cout << "Miau" << std::endl;
    }
};

int main() {
    Animal* a1 = new Cachorro();
    Animal* a2 = new Gato();
    a1->fazerSom();
    a2->fazerSom();
    delete a1;
    delete a2;
    return 0;
}
```

4. Classes Abstratas como Interfaces

- Definição: Classes abstratas podem ser usadas para definir interfaces que especificam um conjunto de métodos que as classes derivadas devem implementar.
- Exemplo:

```
class ICalculadora {
public:
    virtual int somar(int a, int b) = 0;
    virtual int subtrair(int a, int b) = 0;
};
```

```
class CalculadoraSimples : public ICalculadora {
public:
    int somar(int a, int b) override {
        return a + b;
    }

    int subtrair(int a, int b) override {
        return a - b;
    }
};

int main() {
    ICalculadora* calc = new CalculadoraSimples();
    std::cout << "Soma: " << calc->somar(3, 4) << std::endl;
    std::cout << "Subtração: " << calc->subtrair(7, 2) << std::endl;
    delete calc;
    return 0;
}
```

5. Herança Múltipla e Classes Abstratas

- Definição: Classes abstratas podem ser usadas com herança múltipla para fornecer várias interfaces que uma classe derivada deve implementar.
- Exemplo:

```
class IDesenhavel {
```

```
public:
    virtual void desenhar() = 0;
};

class IMovivel {
public:
    virtual void mover(int x, int y) = 0;
};

class Personagem : public IDesenhavel, public IMovivel {
public:
    void desenhar() override {
        std::cout << "Desenhar personagem" << std::endl;
    }

    void mover(int x, int y) override {
        std::cout << "Mover para (" << x << ", " << y << ")" << std::endl;
    }
};

int main() {
    Personagem p;
    p.desenhar();
    p.mover(10, 20);
    return 0;
```

```
}
```

Dicas de Boas Práticas

- Interfaces: Use classes abstratas para definir interfaces claras que devem ser implementadas por classes derivadas.
- Polimorfismo: Aproveite o polimorfismo para escrever código mais flexível e reutilizável.
- Documentação: Documente claramente as funções virtuais puras para indicar que elas devem ser implementadas nas classes derivadas.

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