Classes Virtuais e Polimorfismo

Polimorfismo:

- Polimorfismo: Capacidade de um código de programação comportar-se de diversas formas dependendo do contexto.
- Em C++, suportado de três maneiras:
  - Polimorfismo paramétrico: permite que uma função ou um tipo de dados seja escrito genericamente, para que ele possa manipular valores de maneira uniforme sem depender de seu tipo. Implementado através dos templates em C++.
  - Polimorfismo de função (ou *ad hoc*): funções que podem ser usadas com argumentos de tipos diferentes, mas que se comportam de maneira diferente, dependendo do tipo de argumento ao qual elas são aplicadas. Conhecido como sobrecarga de função ou sobrecarga de operador em C++. Resolvido em tempo de compilação (*early binding*), com maior eficiência computacional.
  - Polimorfismo por subtipagem: quando o código de chamada de uma função é independente de qual classe na hierarquia de herança executará a função - a classe pai ou um de seus descendentes. Permite que uma função seja escrita para receber como parâmetro um objeto de um determinado tipo T, mas também funcione corretamente se for passado um objeto que pertence a um tipo S, que é um subtipo de T (herda de T). Em C++ é implementado através da herança com métodos virtuais. Resolvido em tempo de execução (*late binding*), com maior flexibilidade.

Métodos virtuais

- Método que é declarado como `virtual` em uma classe base e redefinida pela classe derivada, com a mesma assinatura (mesmo nome, mesmos parâmetros):
  - Exemplo:

```
class Base {
  virtual void metodo(int i);
};
class Deriv: public Base {
  void metodo(int i);
};
```

  - Não é necessária o `virtual` na declaração dos métodos nas classes derivadas.
- A redefinição do método na classe derivada se sobrepõe à definição na classe base.
- Quando acessados normalmente (através de um objeto da classe), métodos virtuais se comportam como qualquer outro tipo de função membro da classe, sem polimorfismo.
- Quando acessados através de um apontador (ponteiro) para a classe base, os métodos virtuais podem exibir comportamento polimórfico:
  - Um ponteiro para a classe base pode ser usado para apontar para objetos de qualquer classe derivada daquela classe base.
  - C ++ determina qual versão daquele método chamar baseada no tipo do objeto apontado pelo apontador.
  - Quando objetos de classes derivadas diferentes são apontados, versões diferentes do método virtual são executadas.

Condições necessárias para que ocorra polimorfismo através de métodos virtuais:

- O método tenha sido declarado como `virtual` em uma classe base.
- Uma classe derivada tenha sido definida como herdeira da classe base.
- O método virtual tenha sido redefinido pela classe derivada, com a mesma assinatura.
- O método virtual seja acessado através de um ponteiro para classe base, e não via um objeto de alguma das classes ou via ponteiro para a classe derivada.

- Exemplo:

```
Base B, *pB;
Derivada D, *pD;
B.metodo();      // NÃO polimorfismo: acesso via objeto
D.metodo();      // NÃO polimorfismo: acesso via objeto
pB = &D;
pB->metodo();    // OCORRE polimorfismo
pD = &D;
pD->método();    // NÃO polimorfismo: ptr classe derivada
```

Destrutores virtuais

- Classes base com métodos virtuais devem ter destrutores virtuais, para que, ao ser destruído um objeto de uma classe derivada que é armazenado através de um ponteiro para a classe base, e não como um objeto da classe derivada, o destrutor correto seja chamado.

Vtable

- A resolução dinâmica (*late binding*) é implementada por meio do uso de uma tabela (matriz de ponteiros) de métodos virtuais, chamada *v-table*.
- Em cada classe que contém um método virtual, o compilador acrescenta um membro de dado que armazena o endereço na tabela *v-table*. Portanto, objetos de classes com métodos virtuais têm um byte adicional (oculto).

Métodos virtuais puros e classes abstratas

- Métodos virtuais puros são métodos virtuais definidos em uma classe base sem nenhuma implementação, apenas com a definição de sua assinatura. São reconhecidos pelo acréscimo de "`= 0`" após a declaração:
  - Exemplo:

```
class Base {
  virtual void metodo(int i) = 0;
};
```

- Quando um método virtual é feita puro, qualquer classe derivada deve fornecer sua própria definição. Caso contrário, um erro de compilação será acusado.
- Classes que contenham ao menos um método virtual puro são chamadas de classes abstratas. Pode-se dizer que uma classe abstrata é a definição de um tipo incompleto, que serve como fundação para classes derivadas.
- Objetos não podem ser criados para classes abstratas, mas apontadores para classes abstratas são válidos. Isto permite que classes abstratas suportem polimorfismo.