# POLIMORFISMO

C++

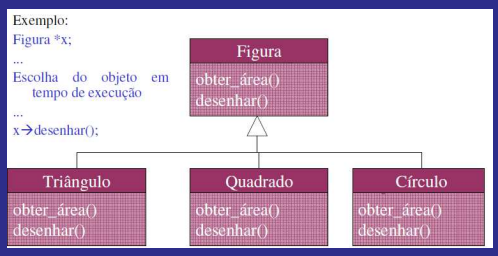
## POLIMORFISMO

- A decisão sobre qual o método que deve ser selecionado, de acordo com o tipo da classe derivada, é tomada em tempo de execução, através do mecanismo de ligação tardia.
- Em C++ isso ocorre utilizando-se ponteiros.

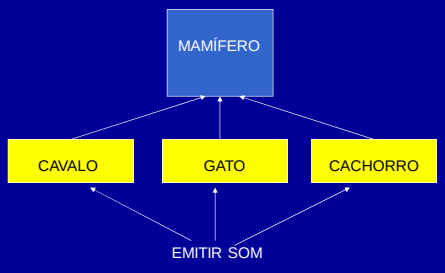
## DEFINIÇÃO

- Polimorfismo é a capacidade de assumir formas diferentes.
- Em termos de programação, polimorfismo representa a capacidade de uma única variável chamar métodos diferentes, dependendo do que a variável contém.
- Maneira de escrever programas de forma genérica que permite a manipulação de uma grande variedade de classes.

## POLIMORFISMO




## POLIMORFISMO




Extensível...

## POLIMORFISMO

Aspectos importantes

- Usa-se uma variável de um tipo único (do tipo da super-classe) para referenciar objetos variados do tipo das sub-classes.
- Envolve o uso automático do objeto armazenado na super-classe para selecionar um método de uma das sub-classes. O tipo do objeto armazenado não é conhecido até a execução do programa. A escolha do método a ser executado é feita dinamicamente.

## POLIMORFISMO - EXEMPLO

```
Class mamifero
{
Public:
   mamifero( );
   virtual ~mamifero( );
   virtual void emitir_som( );
   void comer( );
   virtual void andar( );
Protected:
   int idade;
};
```

## POLIMORFISMO

- As construções apresentadas anteriormente são válidas em c++, e fazem sentido na vida real, pois cavalo, gato e cachorro são mamíferos.
- Mais ainda, emitir_som é uma ação que todos eles fazem, mas cada um de forma diferente.

## POLIMORFISMO - EXEMPLO

```
Class cachorro: public mamifero
{
Public:
   cachorro( );
   ~cachorro( );
   void emitir_som( );
   void comer( );
Protected:
   int raça;
};
```

## POLIMORFISMO

Mamifero *p_mamifero = new cachorro;

Essa instrução cria um novo objeto cachorro e retorna um ponteiro para esse objeto, que é atribuído a um ponteiro para mamífero.

Como cachorro é um mamífero, OK !

## POLIMORFISMO - EXEMPLO

```
Main( )
{
   mamifero *p_mamifero = new cachorro;
   p_mamifero→comer( );
   p_mamifero→emitir_som( );
   p_mamifero→andar( );
}
```

## POLIMORFISMO

Com este artifício pode-se invocar qualquer método em mamífero.

Conceitualmente, a capacidade de atribuir o endereço de um objeto de uma classe derivada a um ponteiro de uma classe base é a essência do polimorfismo.

## POLIMORFISMO

VOLTANDO AO EXEMPLO

P: A chamada p_mamifero→comer( ) faz o que? Aciona o método de qual classe?

R: O método acionado será o da classe base, pois foi declarada como: void comer( );

## POLIMORFISMO

```
Class cachorro: public mamifero
{
Public:
   cachorro( );
   ~cachorro( );
   void emitir_som( );
   void comer( );
   void abanar_rabo( );
Protected:
   int raça;
};
```

## POLIMORFISMO

P: E a chamada p_mamifero→emitir_som( ) faz o que? Aciona o método de qual classe?

R: O método acionado será o da classe cachorro, pois foi declarada como: virtual emitir_som( );

## POLIMORFISMO

O método adicionado não pertence a mamífero. Assim, não existe uma maneira simples de você acessar o método abanar_rabo( ) tendo criado um ponteiro para mamífero.

## POLIMORFISMO

P: E a chamada p_mamifero→andar( ) faz o que? Aciona o método de qual classe?

R: O método acionado será o da classe base, pois apesar de declarado como virtual, não existe um "novo" método andar na classe derivada.

## POLIMORFISMO

- **virtual** faz com que o método a executar seja escolhido de acordo com o **conteúdo do** ponteiro (classe derivada)
  - não de acordo com o tipo do ponteiro (classe base)

A mágica da função virtual só opera em ponteiros e referências. A passagem de um objeto por cópia não permitirá que as funções membros virtuais sejam invocadas.

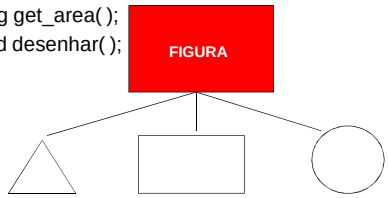
## POLIMORFISMO
## DESTRUTORES VIRTUAIS

**REGRA GERAL**

SE UM DOS MÉTODOS DA CLASSE FOR VIRTUAL, O DESTRUTOR TAMBÉM DEVERÁ SER.

## POLIMORFISMO
## TIPOS ABSTRATOS

```
Class figura
{
Public:
   figura( )
   virtual ~figura( );
   virtual long get_area( );
   virtual void desenhar( );
};
```

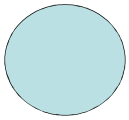



## Exercício

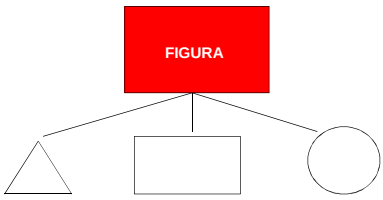
- Classe básica animal (Abstrata)
  - Nome (atributo)
  - Método mover()
  - Localização (atributo)
    - distância para um ponto - int
  - Classes Derivadas
    - peixe – nada 10 metros
      - Profundidade max (atributo)
    - pássaro – voa 20 metros
    - Tartaruga – anda 1 metro
      - Tipo (atributo)

## POLIMORFISMO
## TIPOS ABSTRATOS

```
Class circulo: public figura
{
Public:
   circulo( )
   ~circulo( );
   long get_area( );
   void desenhar( );
Private:
   int raio;
   int circunferencia;
};
```



## POLIMORFISMO
## TIPOS ABSTRATOS

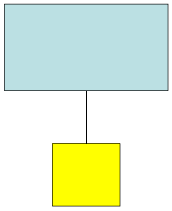



## POLIMORFISMO
## TIPOS ABSTRATOS

```
Class retangulo: public figura
{
Public:
   retangulo(int l, int c)
   virtual ~retangulo( );
   virtual long get_area( );
   virtual void desenhar( );
   virtual int get_largura( );
   virtual int get_comprimento( );
Private:
   int largura;
   int comprimento;
};
```

## POLIMORFISMO
## TIPOS ABSTRATOS

```
Class quadrado: public
   retangulo
{
Public:
   quadrado(int l)
   ~quadrado( );
   long get_area( );
   void desenhar( );
};
```



## POLIMORFISMO
## TIPOS ABSTRATOS

```
      if (! sair)
      {
            ptr_figura → desenhar( );
      }
   }
}
```

## POLIMORFISMO
## TIPOS ABSTRATOS

```
Main( )
{
    int opcao;
    circulo c(5);
    bool sair = false;
    figura  *ptr_figura;
    while (! sair)
    {
          cout << "Qual sua opção ?" << endl;
          cout << "(1) circulo" << endl;
          cout << "(2) retangulo" << endl;
          cout << "(3) quadrado" << endl;
          cout << "(4) sair " << endl;
```

## POLIMORFISMO
## FUNÇÕES VIRTUAIS PURAS

Uma função virtual é tornada pura quando é inicializada com 0:

virtual void desenho( ) = 0;

QUALQUER CLASSE COM UMA OU MAIS FUNÇÕES VIRTUAIS PURAS É UM TIPO ABSTRATO DE DADOS E NÃO SE PODE INSTANCIAR UM OBJETO DE TAL CLASSE.

## POLIMORFISMO
## TIPOS ABSTRATOS

```
cin >> opcao;

switch (opcao)
{
          case 1: ptr_figura = &c;
          break;
          case 2: ptr_figura = new retangulo(4,6);
          break;
          case 3: ptr_figura = new quadrado(5);
          break;
          default: sair = true;
          break;
}
```

## POLIMORFISMO
## FUNÇÕES VIRTUAIS PURAS

Quando você coloca uma função (ou método) virtual pura em sua classe está sinalizando duas coisas:

1. O programador não deverá instanciar um objeto desta classe;
2. A função deve ser "anulada" na classe derivada;

## POLIMORFISMO
## FUNÇÕES VIRTUAIS PURAS

Assim, se retângulo herda de figura e figura tem três funções virtuais puras, retângulo deve anular todas elas para não ser ele também um tipo abstrato de dados, e não se pode instanciar um objeto da classe retângulo