

Tabela de Métodos Virtuais

POO e C++

Tabela de Métodos Virtuais

◆ Considere a seguinte hierarquia de classes:

```
class A {  
    int a;  
    public:  
        virtual void f();  
        virtual void g();  
};  
class B : public A {  
    int b;  
    public:  
        virtual void f(); // redefinição de A::f  
        virtual void h();  
};  
void chamaf(A *a) { a->f(); }
```

Tabela de Métodos Virtuais

- A função `chamaf` executará o método `f` do objeto passado como parâmetro
- Dependendo do tipo do objeto, a mesma linha de código executará `A::f()`; ou `B::f()`;
- Ou seja, é como se a função `chamaf` fosse implementada assim:

```
void chamaf(A *a)
{
    switch (tipo de a)
    {
        case A: a->A::f(); break;
        case B: a->B::f(); break;
    }
}
```

Tabela de Métodos Virtuais

- Isso significa que cada classe derivada de `A` precisa de um `case` nesse `switch`, o que tem várias desvantagens:
 - Essa função não poderia ser utilizada com novas classes que venham a ser criadas no futuro
 - quanto mais classes na hierarquia pior seria a eficiência e, acima de tudo, o compilador não tem como saber todas as classes derivadas de `A`
- Esta análise mostra que esta implementação é irreal, e outra estratégia deve ser usada
- Para resolver este problema, `C++` utiliza tabelas virtuais, que são descrições dos métodos de uma determinada classe.
- Estas tabelas são vetores de funções. O número de entradas da tabela é igual ao número de métodos virtuais da classe e cada posição da tabela tem o ponteiro para uma função virtual.
- Quando uma classe tem algum método virtual, todos os objetos desta classe terão uma referência para esta tabela, que é o descritor da classe

Tabela de Métodos Virtuais

- No exemplo acima, existem duas tabelas virtuais: uma para a classe A e outra para B.
- Quando um objeto é criado, ele leva uma referência para esta tabela. A tabela de A tem duas posições, uma para cada método virtual (f e g).
- A tabela de B tem uma posição a mais para o método h. A posição dos métodos na tabela é sempre a mesma, ou seja, se na tabela de A a primeira posição apontar para o método f, em todas as classes derivadas a primeira posição será de f.
- Na tabela de A, este ponteiro aponta para A::f, enquanto que em B ele aponta para B::f.
- Quando acontece alguma chamada no código, a função não é chamada pelo nome, e sim por indexação a esta tabela.
- Em qualquer tabela de classes derivadas de A o método f estará na mesma posição, no caso, a primeira.

Tabela de Métodos Virtuais

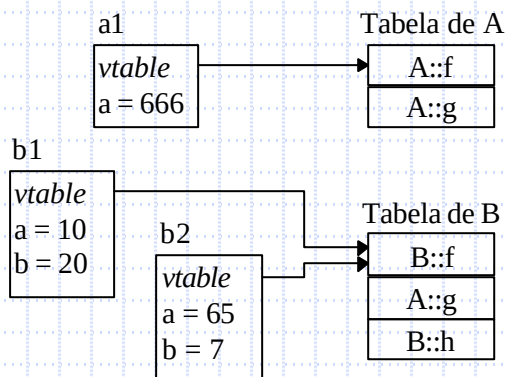


Tabela de Métodos Virtuais

- Considerando que a tabela virtual é um campo de todos objetos (com nome `vtable` por exemplo), a função `chamaf` pode ser implementada assim:

```
void chamaf(A *a)
{
    a->vtable[0](); // posição 0 corresponde ao método f
}
```

- Esta implementação funcionará com qualquer objeto derivado de A, até mesmo de classes futuras, pois não exige que o compilador saiba quais são as classes existentes.
- A eficiência é sempre a mesma independente de quantas classes existam ou qual a sua altura na árvore de hierarquia.
- Esta eficiência não é a mesma de uma chamada de função normal (com `early-binding`) pois envolve uma indireção.