Curso de C/C++ Avançado

Aula 6 – this, Membros Estáticos, Friends, Métodos Virtuais, Overload de Operadores

Allan Lima – http://allanlima.wordpress.com

creative commons
COMMONS DEED

SOME RIGHTS RESERVED

- **Você pode:**
  - *copiar, distribuir, exibir e executar a obra*
  - *criar obras derivadas*
  - *fazer uso comercial da obra*
- **Sob as seguintes condições:**
  - *Atribuição. Você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante.*
  - *Compartilhamento pela mesma Licença. Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.*
  - *Para cada novo uso ou distribuição, você deve deixar claro para outros os termos da licença desta obra.*
  - *Qualquer uma destas condições podem ser renunciadas, desde que Você obtenha permissão do autor.*
- Veja aqui a licença completa

O ponteiro this

- **É um atributo que toda classe possui**
- **Ele é um ponteiro para o objeto no qual um membro esta sendo executado.**
- **Exemplo:**

```
Ponto::Ponto(float x, float y) {
  this->x = x;
  this->y = y;
}
```

Membros Estáticos

- São membros que possuem apenas uma cópia compartilhada por todos os objetos de um tipo
- Podem ser métodos ou atributos:
  - *Atributos estáticos devem ser definidos dentro da própria classe e inicializados fora dela*
  - *Métodos estáticos só podem fazer referência a atributos estáticos.*

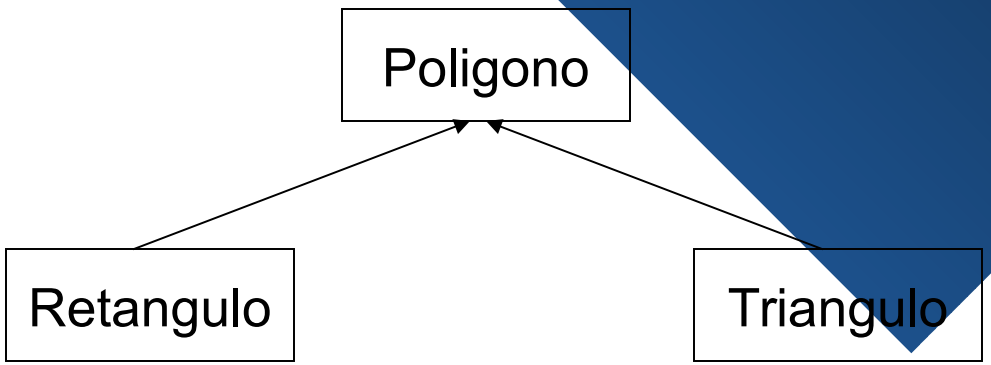
Exemplo

- ***exemploStatic.cpp***

Friends

- Permitem a quebra dos níveis de acesso pelas palavras reservadas private e protected
- Podemos ter funções, métodos ou classes friends
- As declarações são feitas através da palavra reservada friend
- Toda classe é friend dela mesma
- Exemplo:
  - *exemploFriend.cpp*

Métodos Virtuais



- **Problema:**
  - ***Método area()***
  - ***Queremos que as classes Retangulo e Triangulo tenham o método area() com implementações diferentes***

Métodos Virtuais

- **Por padrão, C++ não "ativa" dynamic binding**
  - ***Com isto ganhamos muita performance***
  - ***Mas temos que dizer exatamente quais são os métodos que devem possui o dynamic binding***

Métodos Virtuais

- **Solução:**
  - *Declarar o método area() virtual*
  - *virtual int area();*
  - *O modificador virtual permite que um ponteiro para uma classe básica possa chamar o método correto*
  - *Exemplo:*

```
Triangulo t(45, 10);
Poligono *pp = &t;
cout << "Area de pp: " << pp->area() << endl;
```

Exemplo

- ***exemploMetodosVirtuais.cpp***

Métodos Virtuais

- **Dica importante:**
  - ***Declare os destrutures das suas classes como virtuais***

Classes Abstratas

- São classes que não podem ter objetos instanciados
- Uma classe é dita abstrata quando possui pelo menos um método abstrato
- Um método abstrato é aquele que não é implementado na classe
  - *Usamos "= 0" após o protótipo do método para torná-lo abstrato*
  - *Exemplo: virtual int area() = 0;*

Classes Abstratas

- Toda classe que herda de uma classe abstrata deve implementar os métodos abstratos
  - *Exceção: Quando a sub-classe também é abstrata*
- Métodos abstratos podem ser utilizados normalmente em classes abstratas

Classes Abstratas

- ***exemploClassesAbstratas.cpp***

Interfaces

- C++ não possui o conceito de interface
- Mas podemos simular uma interface com uma classe abstrata em que todos os métodos são abstratos
- Usando herança múltipla podemos “implementar” várias interfaces

Exemplo

- ***exemploInterfaces.cpp***

Overload de Operadores

- C++ permite a aplicação de alguns operadores aos tipos que nós criamos
- Operadores como o “=” são criados automaticamente
- Porém outros devem ser implementados pelo programador
- Sintaxe:
  - *tipo operator símbolo(parâmetros);*

Overload de Operadores

- Operadores podem ser sobrecarregados de diversas formas
- Podemos implementar os operadores como funções ou membros de classes
- Quando o primeiro parâmetro não é do tipo da própria classe o operador deve ser implementado como uma função

Overload de Operadores

- **Podemos implementar várias vezes o mesmo operador para diversos tipos:**
  - ***Ponto operator + (Ponto &v);***
  - ***Ponto operator + (float v);***
- **Agora podemos escrever:**
  - ***Ponto p3 = p1 + p2;***
  - ***Ponto p4 = p1 + 15.0f***

Exemplo

exemploOperadores1.cpp

exemploOperadores2.cpp

Overload de Operadores

- **C++ permite a sobrecarga dos seguintes operadores:**

Unários:	+	-	*	!	~	&	++	--	()	->	->*
	new	delete									
Binários:	+	-	*	/	%	&	|	^	<<	>>	
	=	+=	-=	/=	%=	&=	|=	^=	<<=	>>=	
	==	!=	<	>	<=	>=	&&	||	[]	()	,

Overload de Operadores

- **Operadores que não podem ser sobrecarregados:**
  - ***. (Ponto)***
  - ***, **, *** ...***
- **Não podemos criar novos operadores**
- **Não podemos modificar a tabela de precedência dos operadores**

Onde declarar os operadores?

- **Murray, sugere os seguintes guidelines:**

Operador	Implementação
Unários	Método
= () [] -> ->*	Deve ser um método
+= –= /= *= ^= &= |= %= >>= <<=	Método
Demais operadores binários	Função

Rob Murray, C++ Strategies & Tactics, Addison-Wesley, 1993, page 47.

Overload de Operadores

- Overload de operadores pode ser muito bom para tornar o código mais legível
- Para isto temos que tomar cuidado com a semântica de cada operador sobrecarregado

Exercícios

1) Crie uma classe ElementoGeometrico com os seguintes métodos públicos virtuais:
   - *float virtual perimetro() const = 0;*
   - *float virtual area() const = 0;*
   - *Crie as classes Circulo e Quadrado que herdam de ElementoGeometrico e implementam os seus métodos*
   - *Sinta-se a vontade para definir os atributos e construtores necessários para cada classe*

Exercícios

2) Crie uma classe Ponto e implemente os seguintes operadores:

==

!=

+ (para dois pontos e um ponto + um float)

- (para dois pontos e um ponto - um float)

Referências

- Stroustrup, Bjarne. The C++ Programming Language, Special Edition.
- Eckel, Bruce. Thinking in C++, 2nd ed. Volume 1.
- Slides de Gustavo (ghpc@cin.ufpe.br)do curso de C/C++