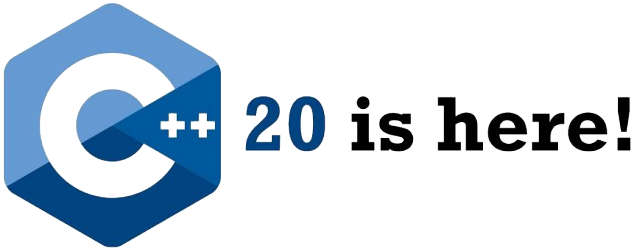
Sobrecarga de Operadores

Eng. Computação

Prof. Victor A P Oliveira



Fundamentos

Sobrecarga de operadores

Dar um novo significado a um operador para que eles **funcionem com objetos**

Contribui para a **extensibilidade** da C++

Fundamentos

Sobrecarga de operadores

Utilize somente **se tornar o programa mais claro**

Embora não obrigatório, os operadores sobrecarregados **devem manter seu significado original** (+ como +; - como - etc.)

Fundamentos

Sobrecarga de operadores

Restrições

Sobrecargas permitidas

Operadores que podem ser sobrecarregados							
+	-	*	/	%	^	&	|
~	!	=	<	>	+=	-=	*=
/=	%=	^=	&=	|=	<<	>>	>>=
<<=	==	!=	<=	>=	&&	||	++
--	->*	,	->	[]	()	new	delete
new[]	delete[]						

Fundamentos

Sobrecarga de operadores

Restrições

Sobrecargas não permitidas

Operadores que não podem ser sobrecarregados			
.	.*	::	?:

Fundamentos

Sobrecarga de operadores

Restrições

- Precedência não pode ser alterada
- Associatividade não pode ser alterada
- Número de operandos não pode ser alterado

Fundamentos

Sobrecarga de operadores

Para sobrecarregar um operador deve-se utilizar a palavra-chave **operator**, seguido do operador, como sendo o nome da **função**

Essa **função** pode ser:

função-membro (método)

função friend

função global (raramente)

Fundamentos

Sobrecarga de operadores

Essa função pode ser:

função-membro (método)

Ao sobrecarregar (), [], -> ou qualquer um dos operadores de atribuição. Os demais operadores podem ser métodos ou não

Regra geral: sempre que “seu” objeto invocar a operação

Ex.: res = SEU_OBJETO + OUTRO_OBJETO

Fundamentos

Sobrecarga de operadores

Essa função pode ser:

função friend

Regra geral: Quando o objeto invocador não for o “seu”

Ex.: res = OUTRO_OBJETO + SEU_OBJETO

Fundamentos

Sobrecarga de operadores

Essa função pode ser:

função global (menos comum)

Regra geral: Quando nenhum dos objetos for “seu”

Ex.: res = OUTRO_OBJETO + MAIS_OUTRO

Preparados???

Mas antes...

Uma palavrinha sobre **construtor de cópia**

É **invocado automaticamente** quando:

um objeto é passado por valor para uma função

ao retornar o objeto por valor

ao inicializar o objeto com a cópia de outro objeto de mesma Classe

Implementar quando você precisar implementar um comportamento diferente para cópia de objetos. Normalmente quando envolve ponteiros e alocação de memória

Mas antes...

Uma palavrinha sobre **construtor de cópia**

Exemplo problemático: “sem” o construtor de cópia

Exemplo com o construtor de cópia implementado

GitHub

Agora vai...

Estudo de caso: classe Array

Responsabilidades

Armazenar um array de tamanho dinâmico :)

Suportar operadores

==, !=, =, []

Estudo de caso: classe Array

Responsabilidades

Adicionar suporte aos operadores << e >> da entrada e saída padrão

Estudo de caso: bitwise em string

Responsabilidades

Fazer com que os operadores << e >> desloquem caracteres para esquerda e direita, respectivamente, no sentido parecido que funcionam a nível de bits.

Estudo de caso: classe String

Responsabilidades

Armazenar uma cadeia de caracteres de tamanho dinâmico

Suportar os operadores

=, +=, !, ==, !=, <, >, <=, >=, [], ()

Veja mais: classe Date

Responsabilidades

- Armazenar uma data
- Suportar os operadores
  - ++, +=

Outros tópicos

Conversão entre tipos

```
operator tipoDest() const;
```

Construtores explicit