

IMD0030 LINGUAGEM DE PROGRAMAÇÃO I

Aula 06 – Sobrecarga de Operadores

Objetivos desta aula

- Introduzir o mecanismo de sobrecarga de operadores em C++
- Para isso, estudaremos:
 - Como realizar a sobrecarga dos operadores já existentes na linguagem C++
 - Como utilizar tais operadores com classes criadas pelo usuário
- Ao final da aula, espera-se que o aluno seja capaz de:
 - Compreender como sobrecarregar operadores da linguagem C++
 - Implementar métodos para sobrecarregar operadores, habilitando-os para utilização com classes por ele criadas

Nas cenas dos capítulos anteriores...



Nas cenas dos capítulos anteriores...

- Vimos como **criar classes e instanciar objetos** utilizando a linguagem C++
 - Classes possuem atributos e métodos como membros
 - A instanciação de um objeto de uma classe é feita de forma similar à declaração de uma variável em C++
- Vimos como implementar e utilizar **construtores para instanciar objetos** de uma classe
 - **Construtor padrão:** forma padrão de instanciar um objeto
 - **Construtor parametrizado:** recebe como parâmetros valores a serem utilizados para inicializar o objeto
 - **Construtor cópia:** cria um objeto a partir de um outro previamente instanciado, copiando membro por membro de um objeto para outro

Um problema...

- Suponhamos que temos uma classe chamada `Tempo`, que representa um instante de tempo definido em termos de horas, minutos e segundos
- Como somar dois instantes de tempo (ou seja, dois objetos da classe `Tempo`)?

```
class Tempo {  
private:  
    short horas;  
    short minutos;  
    short segundos;  
  
public:  
    Tempo(short h, short m, short s);  
};
```

```
#include <iostream>  
#include "tempo.h"  
  
int main() {  
    Tempo r(12, 30, 0); // 12h 30min 0s  
    Tempo t(1, 20, 0); // 1h 20min 0s  
  
    std::cout << r + t;  
  
    return 0;  
}
```

Qual o resultado da
compilação e execução
deste programa?

Um problema...

- Erro:

In function 'int main()':
error: no match for 'operator+' (operand types are 'Tempo' and 'Tempo')

O compilador não sabe como somar dois objetos da classe Tempo

- Solução inicial: implementar um método que realize a operação de soma

- Parâmetro: objeto da classe Tempo, cujos valores de horas, minutos e segundos serão somados aos do objeto que invoca o método para realizar a soma

- Retorno: novo objeto da classe Tempo

```
Tempo somar(Tempo t) {  
    short h = horas + t.getHoras();  
    short m = minutos + t.getMinutos();  
    short s = segundos + t.getSegundos();  
    return Tempo(h, m, s);  
}
```

Com isso, será possível
invocar o método:
`r.somar(t)`

Um problema...



Solução definitiva (e intuitiva): **sobrecarregar o operador de adição (+)**

Sobrecarga de operadores

- A linguagem C++ **não permite criar novos operadores**, mas permite alterar o comportamento de operadores já existentes na própria linguagem, ou seja, realizar uma **sobrecarga de operadores**
- A sobrecarga de operadores é um **recurso muito útil em C++** para permitir realizar certas operações sobre objetos de classes criadas pelo usuário
 - O objetivo é fornecer, para os tipos definidos pelo usuário, **as mesmas expressões** que a linguagem oferece no conjunto de operadores padrão para tipos primitivos, **de forma intuitiva**
 - Apesar de ser um recurso útil, **a sobrecarga deve ser utilizada com atenção**, evitando, por exemplo, usar o operador / (geralmente utilizado para divisão) para realizar uma operação de adição
 - **Boa prática de programação:** deve-se sobrecarregar operadores para executar a mesma função ou função semelhante à sua definição original, ou seja, sobrecarregar o operador + para adição/concatenação, o operador = para atribuição, e assim por diante

Sobrecarga de operadores

Como se sobrecarrega um operador?

É necessário definir um método, com as seguintes características:

- O **nome** na assinatura deve ser formado pela palavra-chave **operator** seguida do símbolo do operador a ser sobre carregado
- A **quantidade de parâmetros** deve ser igual ao número de operandos do operador menos um
 - Com isso, um método para sobre carregar operadores unários não recebe parâmetro
 - Já um método para sobre carregar operador binário recebe apenas um parâmetro, um objeto da classe em questão
- O **corpo** deve especificar como o operador deve operar sobre os parâmetros e o objeto da classe em questão
- O **retorno** deve ser o resultado esperado para aquele operador

Sobrecarga de operadores

Exemplo: sobrecarga do **operador +** para somar dois objetos da classe **Tempo**

```
class Tempo {  
    private:  
        short horas;  
        short minutos;  
        short segundos;  
  
    public:  
        Tempo(short h, short m, short s);  
        Tempo(Tempo &t);  
        Tempo operator+(Tempo &t);  
};  
  
Tempo Tempo::operator+(Tempo &t) {  
    short h = horas + t.horas;  
    short m = minutos + t.minutos;  
    short s = segundos + t.segundos;  
    return Tempo(h, m, s);  
}
```

```
#include <iostream>  
#include "tempo.h"  
  
int main() {  
    Tempo r(12, 30, 0); // 12h 30min 0s  
    Tempo t(1, 20, 0); // 1h 20min 0s  
  
    Tempo s(r + t); // 13h 50min 0s  
  
    return 0;  
}
```

a soma de dois objetos da classe
Tempo retorna um novo objeto
dessa mesma classe

o método **operator+** é invocado
como r.operator+(t) e retorna um
novo objeto da classe Tempo

Sobrecarga de operadores

Exemplo: sobrecarga do **operador** = para atribuir um objeto da classe **Tempo** a outro

```
class Tempo {  
    private:  
        short horas;  
        short minutos;  
        short segundos;  
  
    public:  
        Tempo(short h, short m, short s);  
        Tempo& operator= (Tempo const &t);  
};  
  
Tempo& Tempo::operator= (Tempo const &t) {  
    horas = t.horas;  
    minutos = t.minutos;  
    segundos = t.segundos;  
    return *this;  
}
```

é retornado um
ponteiro para o próprio
objeto que invocou o
método

```
#include <iostream>  
#include "tempo.h"  
  
int main() {  
    Tempo r(12, 30, 0); // 12h 30min 0s  
    Tempo t(1, 20, 0); // 1h 20min 0s  
  
    Tempo s = r + t; // 13h 50min 0s  
  
    return 0;  
}
```

o método **operator=** é invocado
como `s.operator=(r + t)` e retorna
uma referência para o próprio objeto
`s`

Sobrecarga de operadores

- A maioria dos operadores em C++ pode ser sobre carregada...

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

- ...mas nem todos os operadores podem

. :: ?: sizeof

Sobrecarga de operadores

Através da sobrecarga de operadores, **não é possível**

- Sobrecarregar um operador que não seja sobrecarregável
- Alterar a associatividade ou a precedência de um operador
 - Exemplo: a precedência da multiplicação continua sendo superior a da adição em uma expressão
- Alterar o número de operandos aceitos por um operador
 - Operadores unários sobrecarregados continuam sendo unários, tendo um único operando
 - Operadores binários sobrecarregados continuam sendo binários, tendo dois operandos
 - O único operador ternário de C++ (?:) não pode ser sobrecarregado
- Criar novos operadores, ou seja, somente os operadores existentes podem ser sobrecarregados

Sobrecarga dos operadores de inserção e extração

A sobrecarga dos operadores de inserção (`<<`) e extração (`>>`) de dados em *streams* é feita de forma um pouco diferente dos operadores convencionais

- O operador de inserção de dados em um *stream* de saída (saída padrão ou arquivo) pode ser sobreescarregado para determinar como um objeto de uma classe pode ser impresso
- O método para sobreescarregar o operador de inserção (`<<`)
 - Deve ser um método ***friend*** da classe, isto é, um método que não é membro da classe porém tem acesso aos seus membros privados
 - Recebe um objeto *ostream*, passado por referência
 - Recebe um objeto da classe em questão
 - Retorna uma referência para um objeto *ostream*

Objetos *ostream*:

- Saída padrão: `cout` (biblioteca `<iostream>`)
- Saída em arquivo: `ofstream` (biblioteca `<fstream>`)
- String: `stringstream` (biblioteca `<sstream>`)

Sobrecarga dos operadores de inserção e extração

Exemplo: impressão de um instante de tempo (objeto da classe `Tempo`) no formato `hh:mm:ss`

```
#include <iostream>

class Tempo {
private:
    short horas;
    short minutos;
    short segundos;

public:
    friend std::ostream& operator<< (std::ostream &o, Tempo const t);
};

std::ostream& operator<< (std::ostream &o, Tempo const t) {
    o << t.horas << ":" << t.minutos << ":" << t.segundos;
    return o;
}
```

O método é declarado como **friend** da classe `Tempo` para ter acesso aos seus atributos privados

O número de parâmetros deve ser igual ao número de operandos que o operador exige

Note que o método que sobrecarrega o operador `<<` não é membro da classe e, portanto, sua implementação não demanda o operador de resolução de escopo (`::`)

Sobrecarga dos operadores de inserção e extração

Exemplo: impressão de um instante de tempo (objeto da classe Tempo) no formato hh:mm:ss

```
#include <iostream>
#include "tempo.h"

int main() {
    Tempo r(12, 30, 0);
    std::cout << r;

    return 0;
}
```

É exibido o valor
"12:30:0" na
saída padrão

Sobrecarga dos operadores de inserção e extração

Relembrando: a sobrecarga dos operadores de inserção (`<<`) e extração (`>>`) de dados em *streams* é feita de forma um pouco diferente dos operadores convencionais

- O operador de extração de dados em um *stream* de entrada (entrada padrão ou arquivo) pode ser sobreescrito para determinar como um objeto de uma classe pode ser construído a partir dos dados lidos da entrada
- O método para sobreescrugar o operador de extração (`>>`)
 - Deve ser um método **friend** da classe, isto é, um método que não é membro da classe porém tem acesso aos seus membros privados
 - Recebe um objeto `istream`, passado por referência
 - Recebe um objeto da classe em questão, por referência
 - Retorna uma referência para um objeto `istream`

Objetos `istream`:

- Entrada padrão: `cin` (biblioteca `<iostream>`)
- Entrada por arquivo: `ifstream` (biblioteca `<fstream>`)
- String: `stringstream` (biblioteca `<sstream>`)

Sobrecarga dos operadores de inserção e extração

Exemplo: construção de um instante de tempo (objeto da classe `Tempo`) a partir de valores horas, minutos e segundos fornecidos como entrada

```
#include <iostream>
```

```
class Tempo {  
private:  
    short horas;  
    short minutos;  
    short segundos;  
  
public:  
    friend std::istream& operator>> (std::istream &i, Tempo &t);  
};  
  
std::istream& operator>> (std::istream &i, Tempo &t) {  
    i >> t.horas >> t.minutos >> t.segundos;  
    return i;  
}
```

O método é declarado como **friend** da classe `Tempo` para ter acesso aos seus atributos privados

A quantidade de parâmetros de um método para sobrecarga de operador deve ser igual ao número de operandos que o operador exige

Note que o método que sobrecarrega o operador `>>` não é membro da classe e, portanto, sua implementação não demanda o operador de resolução de escopo (`::`)

Sobrecarga dos operadores de inserção e extração

Exemplo: construção de um instante de tempo (objeto da classe Tempo) a partir de valores horas, minutos e segundos fornecidos como entrada

```
#include <iostream>
#include "tempo.h"
```

```
int main() {
    Tempo t;
    std::cin >> t;
    return 0;
}
```

O usuário digita respectivamente os valores referentes às horas, minutos e segundos, que serão atribuídos aos atributos do objeto t

Aqui presume-se que o formato de entrada seja de valores inteiros separados por espaço, por exemplo **12 30 15** (para representar 12h 30 min 15 s)

Sobrecarga dos operadores de inserção e extração

Por que os métodos utilizados para sobrecarregar os operadores de inserção e extração de *stream* foram declarados como *friends* (não-membros) da classe *Tempo*?

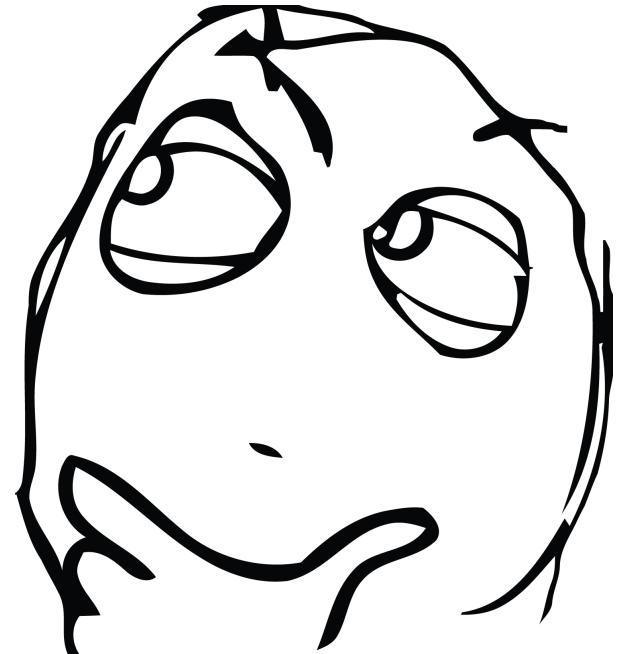
Relembrando...

Quando utilizamos a sobre carga do operador + para fazer a soma de dois objetos da classe *Tempo* (*t* e *s*), temos internamente:

$\text{Tempo } r = t + s$



$\text{Tempo } r = t.\text{operator}+(s)$



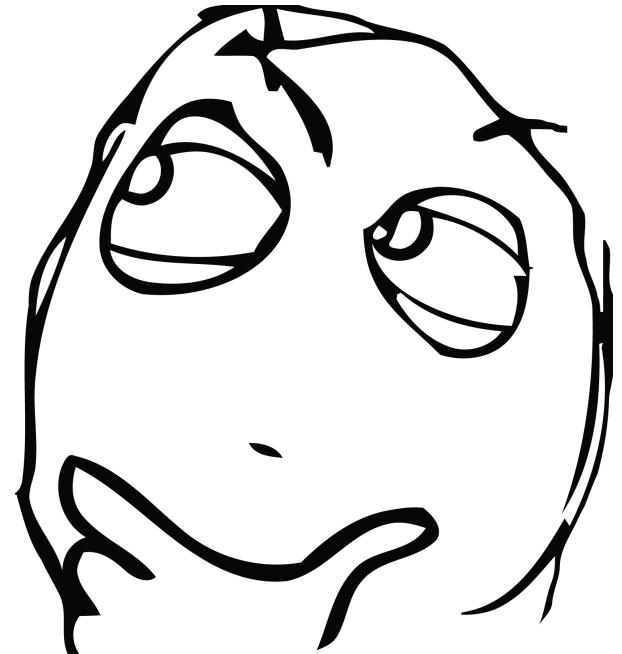
Sobrecarga dos operadores de inserção e extração

Por que os métodos utilizados para sobrecarregar os operadores de inserção e extração de *stream* foram declarados como *friends* (não-membros) da classe *Tempo*?

Seguindo a mesma lógica...

Para sobrecarregar o operador << para imprimir um objeto da classe *Tempo* (*t*) usando um *stream* de saída, teríamos internamente:

cout << t  cout.operator<<(t)

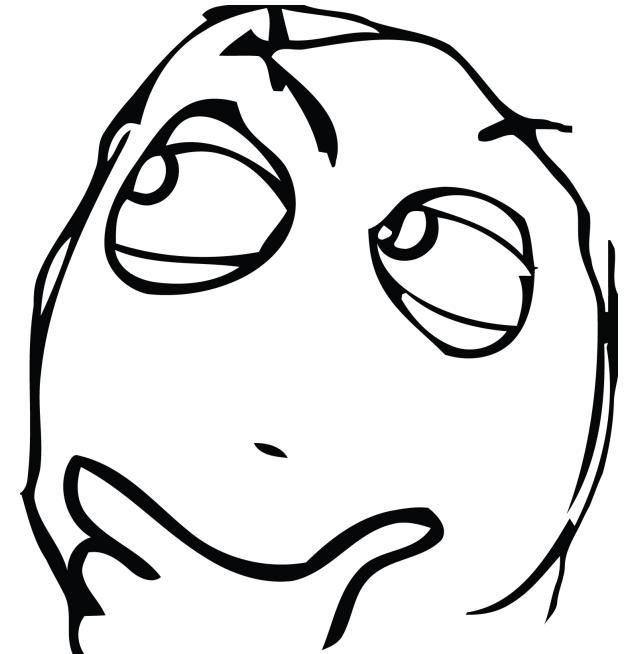


Isto não é possível pois o *stream* cout não é objeto da classe *Tempo*

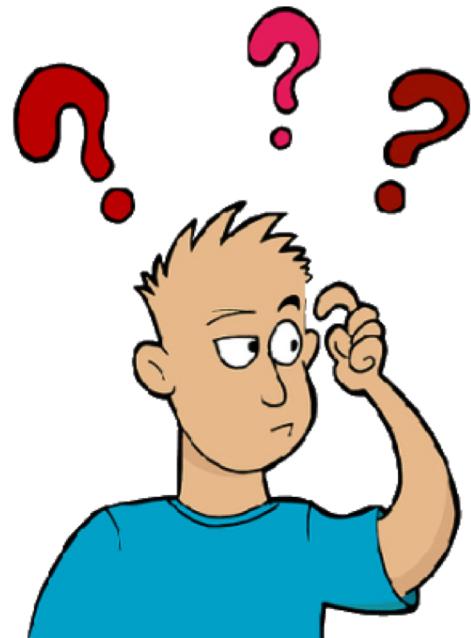
Sobrecarga dos operadores de inserção e extração

Por que os métodos utilizados para sobrecarregar os operadores de inserção e extração de *stream* foram declarados como *friends* (não-membros) da classe *Tempo*?

A solução é justamente fazer com que o método que sobrecarrega o operador << seja *friend* da classe *Tempo* para que ele consiga fazer acesso aos atributos privados da classe. O mesmo se aplica ao método que sobrecarrega o operador >>.



Alguma Questão?



Exercício de Aprendizagem

