# Construtores e Destrutores

Prof. Valdir Pedrinho de Tomin Junior, Eng.

Universidade Federal de Santa Catarina
Centro Tecnológico de Joinville
EMB5631 – Programação III

e-mail: valdir.pedrinho@ufsc.br

27 de agosto de 2018

# Construtores e Destrutores

## Introdução

Em uma linguagem de programação qualquer, quando uma variável de um tipo existente, como um inteiro, é declarada, o compilador cria espaço para aquela variável. Além disso, a linguagem C/C++ permite que uma variável seja declarada e inicializada simultaneamente, evitando problemas de utilização de variáveis não inicializadas. Porém, quando uma variável de um tipo existente sai fora de escopo, o compilador faz com que o espaço para aquela variável seja liberado. Na verdade, o compilador "constrói" e "destrói" a variável.

# Construtores

## Características de Métodos Construtores

A linguagem C++ é concebida para fazer tipos definidos pelo usuário (classes). Para permitir uma inicialização direta dos objetos o compilador necessita de uma função para chamar quando a variável é criada, isto é, o compilador chama um **construtor**. Os construtores são muito úteis em tarefas ligadas à inicialização de classes. Nelas pode-se encontrar a inicialização direta de membros, alocação dinâmica de memória, recuperação de dados em arquivos etc.

## Funcionamento

Construtores podem receber parâmetros e permitem sobrecarga, isto é, mais de um construtor para uma mesma classe, mas não possuem tipos de dados de retorno (nem mesmo *void*); são opcionais; são **chamados automaticamente** para inicializarem a variável de classe.

# Destrutores

## Características de Métodos Destrutores

Quando um objeto sai fora do seu escopo, um **destrutor** é chamado. Embora a liberação de memória (alocada dinamicamente) seja uma aplicação bastante frequente para funções destrutoras, esta não é sua única utilidade. Uma segunda utilidade para funções destrutoras pode ser a finalização de dispositivos ou subsistemas que tenham sido ativados como classes. Por exemplo, ao terminar a impressão pode-se finalizar a impressora desativando eventuais modos de impressão ajustados pelo sistema. Outra utilidade para a função destrutora é gravar dados quando um objeto sai fora do seu escopo (ou antes de liber a memória).

# Destrutores

## Funcionamento

Destrutores não possuem parâmetros e não possuem tipos de dados de retorno (nem mesmo *void*); são opcionais; são **chamados automaticamente** para encerrarem a variável de classe; programas raramente chamam destrutores diretamente.

# Construtores e Destrutores

## Construtores e Destrutores Padrões

Se o programador não proporcionar construtores (pode haver mais de um) e um destrutor (só pode haver um) para uma classe, o compilador assume as ações mais simples. Em outras palavras, caso não seja definido um construtor e um destrutor para a classe, o compilador cria um construtor e um destrutor *default*, sem código e sem parâmetros, que são chamados, respectivamente, a cada declaração de instância, e cada vez que a instância sai fora do escopo.

## Ordem de Criação / Destruição

Deve-se observar que a ordem que o processador segue ao chamar o destrutor implicitamente para cada objeto é inversa à ordem de chamada das funções construtoras, ou seja, o primeiro objeto a ser criado é o último a ser destruído.

## Exemplo de Chamada de Métodos Construtor e Destrutor

Programa 9.11-13 do livro *DEITEL. C++, Como programar. 5ª ed.*

# Inicialização de Variáveis

## Antes do construtor...

É possível inicializar atributos antes do compilador chamar o método construtor. Para tanto é necessário utilizar o formato:

```
NomeDaClasse::NomeDaClasse : _variável1(valor), ...,
    _variávelN(valor)
{
  instruções
}
```

## Exemplo

Classe Lampada inicializando atributos antes do construtor.

# Alocação Dinâmica e o Método Construtor

## Alocação dinâmica de variável simples

É possível utilizar o método construtor para alocação de uma variável simples.

## Exemplo

Classe Lampada alocado dinamicamente.

# Alocação Dinâmica e o Método Construtor

## Alocação dinâmica de um *array*

Não é possível alocar dinamicamente um *array* de uma classe com um construtor que espera parâmetros, exceto se esses parâmetros possuírem um valor padrão.

## Exemplo

*Array* de Objetos do tipo Lampada.

# Alocação Dinâmica e o Método Construtor

## Alocação dinâmica de um *array* com o C++ 11

As versões mais novas da linguagem permitem a passagem de parâmetros para o método construtor na criação de um *array*. É necessária a utilização de uma *flag* de compilação.

## Exemplo

*Array* de Objetos do tipo Lampada passando valores na instanciação.