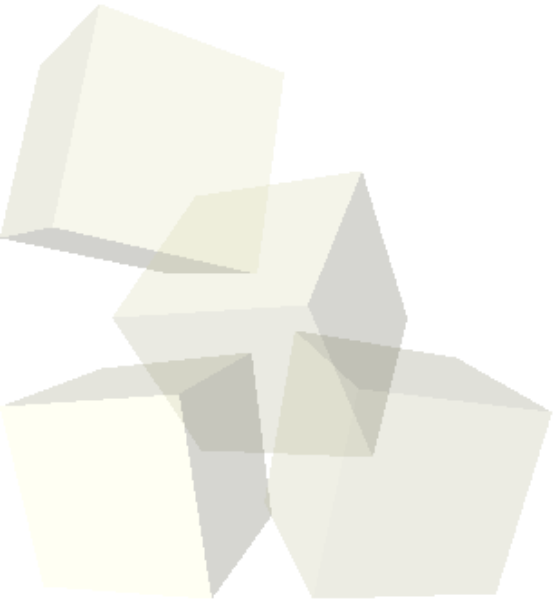
Programação Orientada a Objetos

Aula I – Declaração de classes, métodos construtores

Prof.: Bruno E. G. Gomes
IFRN

Introdução

- Na aula de hoje:
  - Declaração de classes
  - Métodos Construtores
  - Exercícios de criação de classes e objetos



Declaração de Classes

- A declaração de uma classe tem a forma:

```
class <nome> {
 <acesso>:
     <atributos ou métodos>;
 <acesso>:
     <atributos ou métodos>;
 ...
 };
```

- Acesso pode ser:
  - ***public*** – todo objeto da classe pode acessar diretamente o atributo ou método
  - ***private*** – atributo ou método pode ser acessado diretamente apenas dentro da classe
  - *sem nome* – acesso é privado se você não especificar

Exemplo de definição de classe

```
class Retangulo {
    int x, y;
  public:
    void inserir_lados (int a, int b) {
        x  = a;
        y  = b;
    }

    int area (void) {
        return x * y;
    }
};
```

```
class Retangulo {
  private:
    int x, y;
  public:
    void inserir_lados (int a, int b) {
        x  = a;
        y  = b;
    }

    int area (void) {
        return x * y;
    }
};
```

As duas formas são equivalentes, mas prefira a segunda, com o modificador de acesso explícito.

Classe Retangulo – outra forma de definir

```
class Retangulo {
  private:
      int x, y;
  public:
      void inserir_lados (int a, int b);
      int area (void);
};

void Retangulo::inserir_lados (int a, int b) {
        x  = a;
        y  = b;
}
int Retangulo::area (void) {
      return x * y;
}
```

- Nessa forma, apenas o protótipo dos métodos aparece na declaração da classe.
- Os métodos são declarados posteriormente, utilizando o nome da classe seguido de :: antes do nome do método.

Criando objetos de uma classe

- Um objeto é criado a partir da sua classe correspondente
- Exemplo:
  - ***Retangulo*** *ret1, ret2;* → cria dois objetos do tipo retângulo.
- Para chamar um membro público da classe (atributo ou método):
  - *<nome do objeto>*.*<nome do membro público>*
  - Ex.: *ret1*.*adicionar_lados* (2, 2);
- *Mensagem*: chamada de um método de um objeto.

Exemplo

```
int main() {
    cout << "Digite os valores dos lados do retângulo:"
            << endl;
    int lado1, lado2;
    cin >> lado1 >> lado2;

    Retangulo ret;
    ret.inserir_lados(lado1, lado2);
    cout << "Área do retângulo: " << ret.area();

    return 0;
}
```

Criando objetos de uma classe

- Um objeto é criado a partir da sua classe correspondente

- Exemplo:
  - ***Retangulo*** *ret1, ret2;* → cria dois objetos do tipo retângulo.

- Para chamar um membro público da classe (atributo ou método):
  - *<nome do objeto>*.*<nome do membro público>*
  - Ex.: *ret1*.*adicionar_lados* (2, 2);

- **Questão**: *E se o método **area** de algum objeto retângulo for chamado sem que o método **adicionar_lados** tenha sido chamado anteriormente, o que irá acontecer?*

- Nesse caso, os atributos x e y não serão inicializado, portanto o cálculo será feito corretamente.

- Uma solução para esse problema seria criar um método construtor

Método construtor

- Método que é chamado sempre que um novo objeto é criado
  - Permite inicializar os atributos de um objeto no momento da sua criação
  - Também pode realizar qualquer verificação que seja necessária antes ao criarmos um novo objeto

- Características do construtor:
  - Possui o mesmo nome da classe
  - Pode ser chamado apenas no momento da criação da classe
  - Não possuem retorno (não precisa colocar *void* na declaração)

Método Construtor

- É possível ter mais de um método construtor em uma classe, desde que:
  - A quantidade e/ou tipo dos parâmetros seja diferente
- Todo objeto, mesmo que você não declare, tem um construtor
  - O ambiente C++ instancia um construtor vazio (sem parâmetros)
- Caso você declare um construtor, não há mais o construtor padrão.
  - Você só poderá utilizar os construtores que você criar.

Exemplo de construtor

```
class Retangulo {
  private:
      int x, y;
  public:
      Retangulo(int a, int b);
      int area (void);
      void inserir_lados (int a, int b);
};

Retangulo::Retangulo(int a, int b) {
     inserir_lados (a, b);
}

int Retangulo::area (void) { return x * y; }
void Retangulo::inserir_lados (int a,int b) { x = a; y = b; }
```

Sobrecarga de Construtores

- Um construtor, como qualquer outra função, pode ser sobrecarregado
  - *Sobrecarga: mais de um construtor com o mesmo nome, mas com quantidade e/ou tipo de parâmetros diferentes*

- O compilador vai saber que construtor chamar a partir dos parâmetros que você passar
  - ou não passar, no caso do construtor não ter parâmetros

Exemplo

```
class Retangulo {
    private:
        int x, y;
    public:
       //construtor sem parâmetros
       Retangulo();
       //construtor com dois parâmetros
       Retangulo(int a, int b);
       int area (void);
       void inserir_lados (int a, int b);
};

Retangulo::Retangulo() {
    x = 0;
    y = 0;
}
```

```
int main() {
    Retangulo ret(6, 4);
    cout << ret.area() << endl;

    Retangulo ret2;
    cout << ret2.area();

    return 0;
}
```

Exercício

- No exercício da biblioteca (aula anterior) crie métodos construtores na classes Livro e Biblioteca