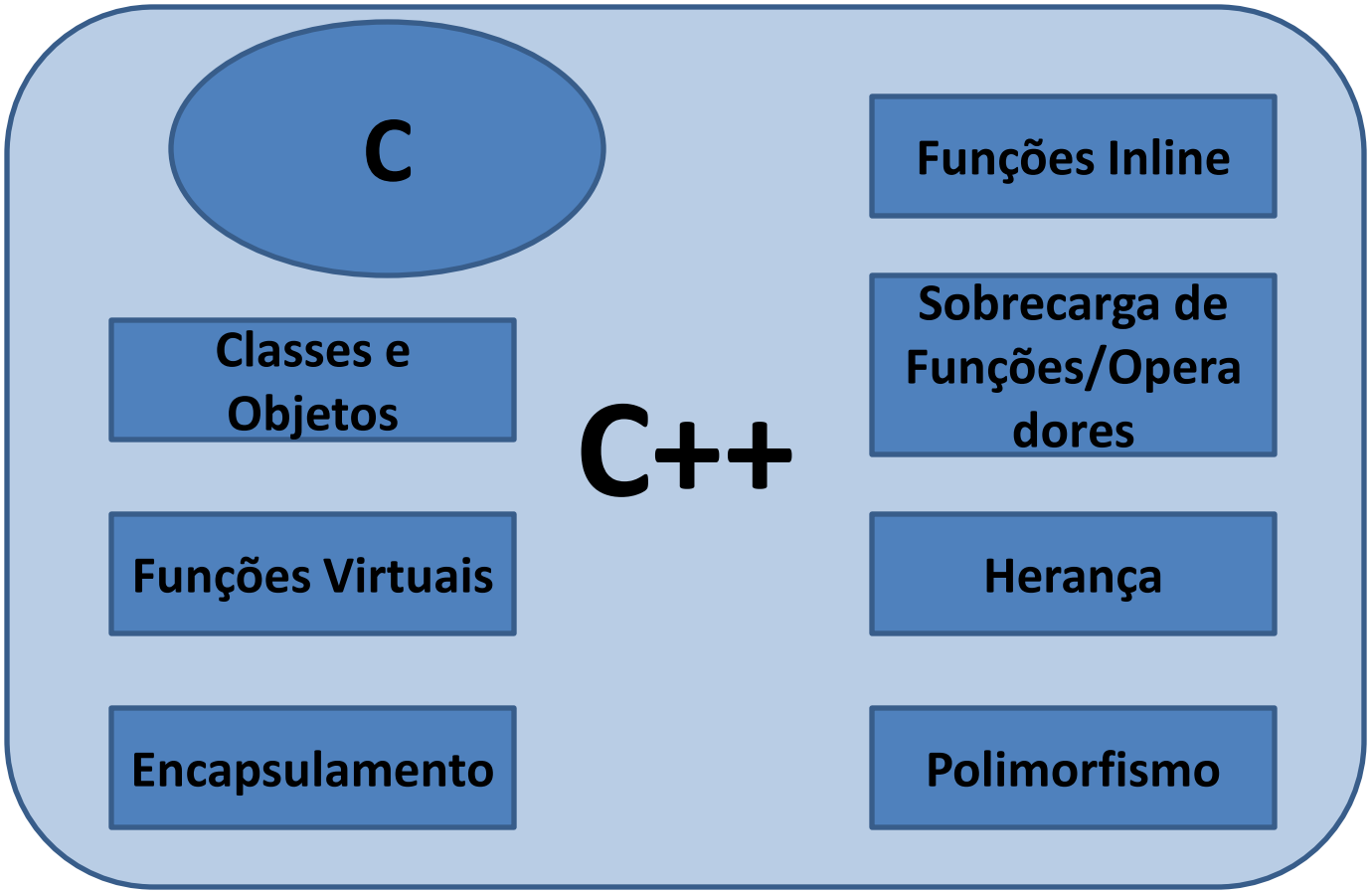
C++

Giselle Lopes Ferrari Ronque
ferrari@eletrica.ufpr.br

Contextualização



Programação Orientada a Objetos

- Características da POO:
  - Classes e Objetos
  - Encapsulamento
  - Sobrecarga
  - Herança
  - Funções Virtuais e Polimorfismo

Classes e Objetos

- POO: combinação, em um único registro, de dados e funções que vão operar nestes dados – CLASSES.
- “variáveis” de um determinado tipo de Classe – OBJETOS (instâncias).
- Campos das classes: MEMBROS ou ATRIBUTOS.
- Funções: FUNÇÕES-MEMBRO ou MÉTODOS.

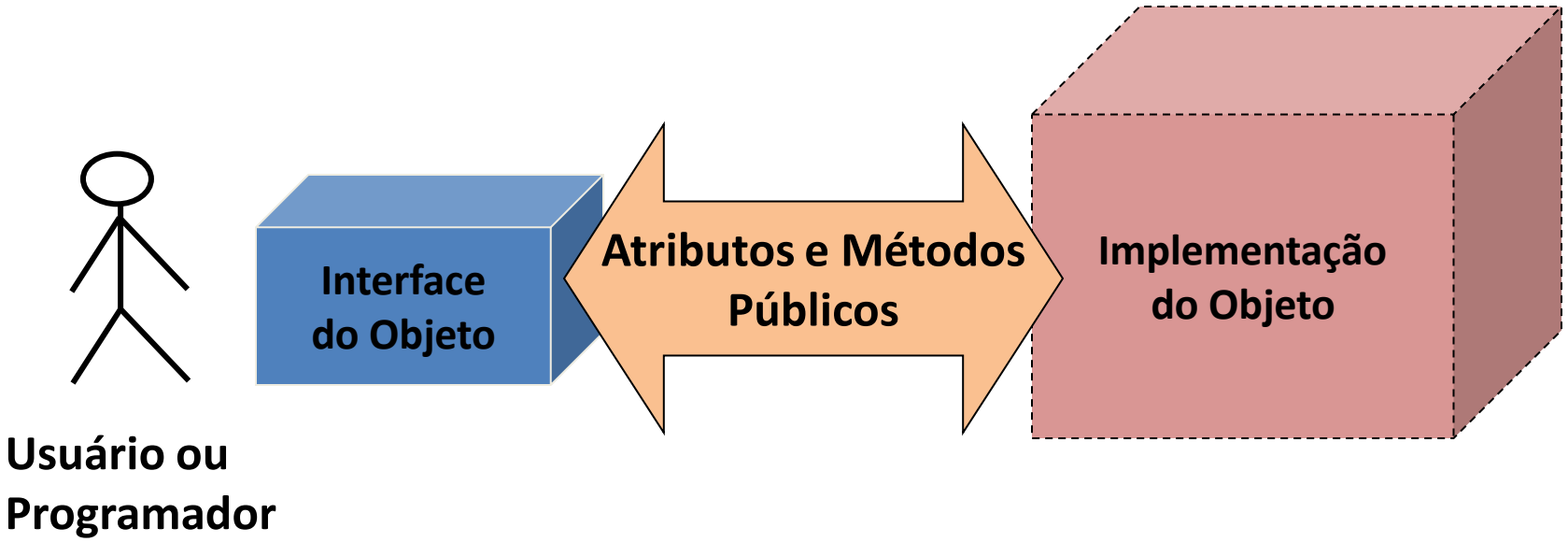
Encapsulamento

– As classes (que “formatam” os objetos) encapsulam tudo aquilo que é necessário para que este objeto exista, seja particularizado e interaja com outros objetos.

– Um determinado código não precisa conhecer a estrutura interna de um objeto (implementação), para relacionar-se com ele

basta conhecer a interface!

Encapsulamento



Vantagens

- Modularidade – alterações no código ou tipo de entrada não alterarão os demais objetos, pois todos são implementados de forma independente.
- Encapsulamento – implementa-se as classes de forma completa e proíbe-se o acesso ao código interno.
- Reutilização – os objetos podem ser utilizados livremente em outros sistemas.

Classes em C++

- Definição:

```
class Nome_Classe
{       private:
           Declarações MembrosPrivados;
           DeclaraçõesMetodosPrivados( );
        public:
           Declarações MembrosPúblicos;
           DeclaraçõesMetodosPúblicos( );
        protected:
           Declarações MembrosProtegidos;
           DeclaraçõesMetodosProtegidos( );
};
```

Classes em C++

- Não se faz inicialização explícita de atributos na declaração da Classe*. Contra-ex: public int x = 0;
- Especificadores de Acesso:
  - Membros e Métodos Privados – **private** : podem ser acessados APENAS por outros métodos do mesmo objeto ou a funções *friend* da classe;
  - Membros e Métodos Públicos – **public** : são acessíveis em todos os lugares onde a classe estiver acessível (escopo da classe);

Classes em C++

- Membros e Métodos Protegidos – protected : são acessíveis pelas classes derivadas (através de herança);
- Membros e Métodos Publicados (Borland C++ Builder) – published : têm acesso público e são publicados no Object Inspector, quando a classe/componente (necessariamente derivada de TObject) é adicionada ao projeto. Não admitem Construtor/Destrutor.

Exemplo

```
class Retangulo                    // declaração da classe
{ private:
      int base, altura;            // Dados membros privados
  public:
      void Inic(int b, int h) {    // método de inicialização
            base = b;  altura = h;
      }
      void ImprimeSaida( ) {       // método que imprime a área
            cout << "\nBase = " << base << " Altura=" << altura;
            cout << "\nÁrea = " << (base*altura);
      }
};
```

Definição de um Objeto

Nome_Classe Lista_Objetos ;

```
#include <iostream.h>
 /* declaração ou inclusão da classe  */
void main( ) // ou outra função
{  Retangulo  x, y;  // declara dois objetos
    x.Inic(5,3);
    y.Inic(10,6);
    x.ImprimeSaida();
    y.ImprimeSaida();
}
```

Observações

- Métodos definidos dentro da definição da classe – “Métodos Inline”.

- Quando optar-se pela definição externa: no corpo da classe usa-se o protótipo do método.

  void Inic(int b, int h);

- E no corpo do método:

  tipo NomeClasse :: *NomeMétodo*( )
  { ...
  }

Exemplo

```
class Retangulo // declaração da classe
{   private: int base, altura;
    public:
                void Inic(int b, int h);   // método de inicialização
                void ImprimeSaida( ); // método que imprime área
};

void Retangulo::Inic(int b, int h)
{ base = b;  altura = h;  }

void Retangulo::ImprimeSaida()
{  cout << "\nBase = " << base << " Altura=" << altura;
   cout << "\nÁrea = " << (base*altura);}
```

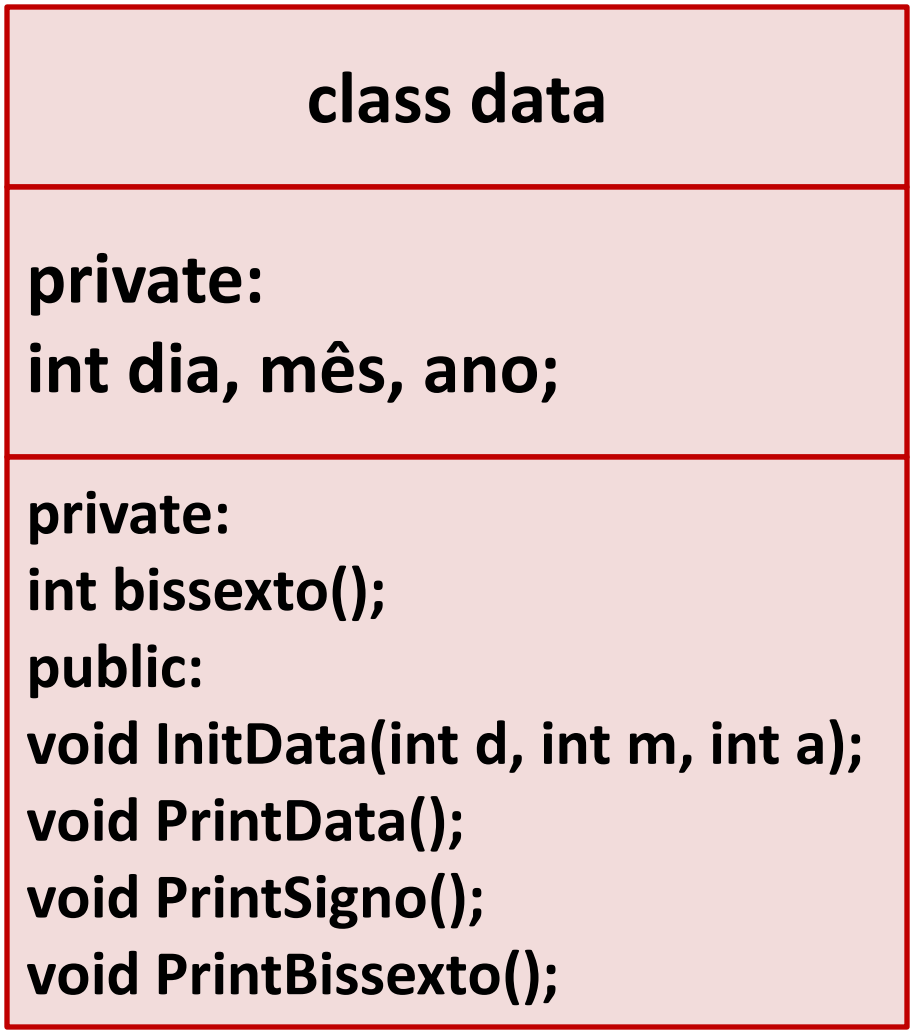
Os métodos de uma classe acessam DIRETAMENTE os atributos (não é preciso passá-los como argumentos)!!

Exercício 1

- Implementar um programa em C++, usando paradigma de orientação a objetos, que:
  - Receba a data de nascimento do usuário;

    01/05/1979
  - Faça a validação da data;
  - Imprima:
    - Data por extenso; 01 de maio de 1979
    - Signo; Touro
    - Bissexto. Não é ano bissexto.
  - Os dados da classe devem ser privados;

Exercício 1


class data
private:
int dia, mês, ano;
private:
int bissexto();
public:
void InitData(int d, int m, int a);
void PrintData();
void PrintSigno();
void PrintBissexto();

Exercício 2

- Estudar os comandos:
  - cout
  - cin