Curso de Ciência da Computação

Algoritmos e Programação de Computadores 2per

Programação Orientada a Objetos POO

Profa. Fernanda dos Santos Cunha

Conversões Automáticas entre Tipos Primitivos

- Conversão implícita ou automática

  double dvar = 'A';

- Conversão explícita: via operadores de conversão de tipo

  double dvar = static_cast<double>('A');

  Ou

  double dvar = double('A');

Conversões de Objetos para Tipos Primitivos

```
class Radianos {
    double rad;
public:
    Radianos( ) { rad = 0.0 ; }
    Radianos(double r) { rad = r ;}
    operator double() { return rad; } // converte p/ double
    void print( ) {
        cout << setiosflags (ios :: fixed)
             << setiosflags (ios :: showpoint)
             << setprecision (2) << rad << "rad \n";
    }
};
```

Conversões entre Objetos de Classes Diferentes

Duas categorias:

- função-membro conversora da classe B
- construtor membro da classe A

Exemplo do 1ª categoria: função conversora classes graus e radianos - converter objetos da classe radianos em objetos da classe graus (grau = radiano).
Função conversora na classe Radianos.

Conversões entre Objetos de Classes Diferentes – 1ª categoria

```
#include <iostream>                    #include <iomanip>
#define PI 3.141592653
using namespace std;
class Graus {
    double g;
public :
    Graus ( ) { g = 0; }
    Graus (double x) { g = x; }
    void print( ) {
        cout << setiosflags (ios :: fixed)
             << setiosflags (ios :: showpoint)
             << setprecision (2) << g << " \xf8\n";
    }
} ;
```

Conversões entre Objetos de Classes Diferentes – 1ª categoria

```
class Radianos {
    double rad;
public:
    Radianos( ) {r ad = 0.0 ; }
    Radianos(double r) { rad = r ;}
    operator double() { return rad; } // converte p/ double
    operator Graus ( ) const {  //conversor de rad. p/graus
       return Graus(180.0*rad/PI);
    }
    void print( ) {
        cout << setiosflags (ios :: fixed)
             << setiosflags (ios :: showpoint)
             << setprecision (2) << rad << "rad \n";   }
};
```

Conversões entre Objetos de Classes Diferentes – 1ª categoria

```
int main( ) {
    Graus gr;
    Radianos rd(PI);
    gr = rd;    // converte radianos para graus
    gr.print();  // mostrara 180 graus
    return 1;
}
```

Conversões entre Objetos de Classes Diferentes

Duas categorias:
- função-membro conversora da classe B
- construtor membro da classe A

Exemplo do 2ª categoria: construtor classes graus e radianos - converter objetos da classe radianos em objetos da classe graus (grau = radiano).
Construtor na classe Radianos.

Conversões entre Objetos de Classes Diferentes – 2ª categoria

```
#include <iostream> #include <iomanip>
#define PI 3.141592653
using namespace std;
class Radianos {
    double rad;
public :
    Radianos( ) {rad = 0.0 ; }
    Radianos(double r) {rad = r ;}
    operator double() { return rad; } // converte p/ double
    void print( ) {
        cout << setiosflags (ios :: fixed)
             << setiosflags (ios :: showpoint)
             << setprecision (2) << rad << "rad \n";
    }
};
```

Conversões entre Objetos de Classes Diferentes – 2ª categoria

```
class Graus {
    double g;
public :
    Graus ( ) {g = 0; }
    Graus (double x) {g=x; }
    Graus (Radianos r ) { g=(static_cast<double>(r)*180.0)/PI; }
    // usando o operator double da classe anterior
    void print( ) {
        cout << setiosflags (ios :: fixed)
             << setiosflags (ios :: showpoint)
             << setprecision (2) << g << " \xf8\n";
    }
} ;
```

Conversões entre Objetos de Classes Diferentes – 2ª categoria

```
int main( ) {
    Graus gr;
    Radianos rd(PI);
    gr = rd;    // converte radianos para graus
    gr.print();  // mostrara 180 graus
    return 1;
}
```

Conversões entre Objetos de Classes Diferentes – de ambos os lados

Converter graus em radianos e radianos em graus:

```
class Radianos {
   double rad;
public :
   Radianos( ) {rad = 0.0 ; }
   Radianos(double r) {rad = r ;}
   operator double() { return rad; } // converte p/ double
   void print( ) {
      cout << setiosflags (ios :: fixed)
           << setiosflags (ios :: showpoint)
           << setprecision (2) << rad << "rad \n";
   }
};
```

Conversões entre Objetos de Classes Diferentes – de ambos os lados

```
class Graus {
   double g;
public :
   Graus ( ) { g = 0.0; }
   Graus (double x) { g=x; }
   Graus (Radianos r) { g=(static_cast<double>(r)*180.0)/PI; }
   // usando o operator double da classe anterior
   operator Radianos( ) const { return Radianos(g*PI/180.0;}
   void print( ) {
      cout << setiosflags (ios :: fixed)
           << setiosflags (ios :: showpoint)
           << setprecision (2) << g << " \xf8\n";
   }
} ;
```

Conversões entre Objetos de Classes Diferentes – de ambos os lados

```
int main( ) {
   Graus gr, gA(180);
   Radianos rd(PI), rA;
   gr = rd;    // converte radianos para graus
   gr.print();
   rA = gA;    // converte graus para radianos
   gr.print();  // mostrara PI
   return 1;
}
```

Conversões: quando usar o que?

Quando as duas classes são acessíveis, pode-se escolher entre o uso de função conversora ou construtora.
Porém se uma das classes pertencer a uma biblioteca, da qual não se tem acesso ao fonte, dois casos podem acontecer:

- Usando instruções como **meu_obj = lib_obj;**
  Deve-se implementar função construtora de um argumento.
- Já nas instruções **lib_obj = meu_obj;**
  É preciso implementar função conversora de tipo.

Exercício

- Redefina a classe Fração, incluindo um conversor de tipo que converte um objeto fração para um tipo float.