

# Programação Orientada a Objetos para Redes de Computadores

Prof. Miguel Elias Mitre Campista

<http://www.gta.ufrj.br/~miguel>

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## PARTE 2

### Programação em C++ - Classes e Objetos (Continuação)

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Empacotador de Pré-processor

- Evita que o código seja incluído mais de uma vez
  - #ifndef - "se não definido"
    - Pula esse código se já tiver sido incluído
  - #define
    - Define um nome para que esse código não seja incluído novamente
  - #endif
  - Se o cabeçalho tiver sido incluído previamente
    - O nome estará definido e o arquivo .h não será incluído novamente
  - Evita erros de múltiplas definições

```
#ifndef TIME_H
#define TIME_H
... // code
#endif
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Empacotador de Pré-processor

- Utilize diretivas de pré-processor #ifndef, #define e #endif para formar um empacotador de pré-processor
  - O empacotador impede que os arquivos de cabeçalho sejam incluídos mais de uma vez em um programa
- Utilize o nome do arquivo do cabeçalho em caixa alta
  - Substitua o ponto por um sublinhado nas diretivas de pré-processor #ifndef e #define de um arquivo de cabeçalho

arquivo.h → ARQUIVO\_H

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 1
 * Arquivo Header
 * Autor: Miguel Campista
 */
#ifndef TIME_H
#define TIME_H

class Time {
public:
    Time ();
    void setTime (int, int, int);
    void printUniversal ();
    void printStandard ();

private:
    int hour; // 0 - 23
    int minute; // 0 - 59
    int second; // 0 - 59
};

#endif
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 1
 * Arquivo timeCap09x1.cpp
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include "timeCap09x1.h"

Time::Time () {
    hour = minute = second = 0;
}

void Time::setTime (int h, int m, int s) {
    hour = (h >= 0 && h < 24) ? h : 0; // valida horas
    minute = (m >= 0 && m < 60) ? m : 0; // valida minutos
    second = (s >= 0 && s < 24) ? s : 0; // valida segundos
}

void Time::printUniversal () {
    cout << setw(10) << setw(2) << hour << ":" <<
    << setw(2) << minute << ":" << setw(2) << second;
}

void Time::printStandard () {
    cout << ((hour == 0 || hour == 12) ? 12 : hour % 12) << ":" <<
    << setw(10) << setw(2) << minute << ":" << setw(2)
    << second << ((hour < 12) ? " AM" : " PM");
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo Usando Classes em C++

```
1 //  
2 * Aula 9 - Exemplo 1  
3 * Arquivo Principal  
4 * Autor: Miguel Campista  
5 */  
6 #include <iostream>  
7 #include "time.cppEx1.h"  
8  
9 int main () {  
10     Time t;  
11  
12     cout << "\n hora Universal inicial eh: ";  
13     t.printUniversal();  
14     cout << "\n\n hora Padrao inicial eh: ";  
15     t.printStandard();  
16  
17     t.setTime(13, 27, 6); // Altera a hora  
18     cout << "\n\n hora Universal alterada eh: ";  
19     t.printUniversal();  
20     cout << "\n\n hora Padrao alterada eh: ";  
21     t.printStandard();  
22  
23     t.setTime(99, 99, 99);  
24     cout << "\n\n hora depois de tentar valores invalidos:"  
25     cout << "\n\n hora Universal: ";  
26     t.printUniversal();  
27     cout << "\n\n hora Padrao: ";  
28     t.printStandard();  
29     cout << endl;  
30  
31     return 0;  
32 }
```

## Primeiro Exemplo Usando Classes em C++

```
1 //  
2 * Aula 9 - Exemplo 1  
3 * Arquivo Principal  
4 * Autor: Miguel Campista  
5 */  
6 #include <iostream>  
7 #include "time.cppEx1.h"  
8  
9 int main () {  
10     Time t;  
11  
12     cout << "\n hora Universal inicial eh: 00:00:00  
13     hora Padrao inicial eh: 12:00:00 AM  
14  
15     hora Universal alterada eh: 13:27:06  
16     hora Padrao alterada eh: 1:27:06 PM  
17  
18     hora depois de tentar valores invalidos:  
19     hora Universal: 00:00:00  
20     hora Padrao: 12:00:00 AM  
21     Pressione qualquer tecla para continuar. . .  
22  
23     t.setTime(99, 99, 99);  
24     cout << "\n\n hora depois de tentar valores invalidos:"  
25     cout << "\n\n hora Universal: ";  
26     t.printUniversal();  
27     cout << "\n\n hora Padrao: ";  
28     t.printStandard();  
29     cout << endl;  
30  
31     return 0;  
32 }
```

## Escopo de Classe e Acesso a Membros de Classe

- O escopo de classe contém:
  - Membros de dados
    - Variáveis declaradas na definição de classe
  - Funções-membro
    - Funções declaradas na definição de classe
- As funções não-membro são definidas no escopo de arquivo

## Escopo de Classe e Acesso a Membros de Classe

- Dentro do escopo de classe
  - Os membros de classe podem ser acessados por todas as funções-membro
- Fora do escopo de classe
  - Os membros de classe public são referenciados por meio de um *handle*
    - Um nome de objeto
    - Uma referência a um objeto
    - Um ponteiro para um objeto

## Escopo de Classe e Acesso a Membros de Classe

- Variáveis declaradas em uma função-membro
  - Têm escopo de bloco
    - Variáveis locais
  - São conhecidas apenas por essa função
- Ocultando uma variável de escopo de classe
  - Em uma função-membro, defina uma variável com o mesmo nome de uma variável com escopo de classe
  - Essa variável oculta pode ser acessada colocando o nome da classe seguido pelo operador de resolução de escopo (::) antes do nome da variável

## Escopo de Classe e Acesso a Membros de Classe

- Operador de seleção de membro ponto (.)
  - Acessa os membros do objeto
  - Usado com o nome de um objeto ou com uma referência a um objeto
- Operador de seleção de membro seta (->)
  - Acessa os membros do objeto
  - Usado com um ponteiro para um objeto

## Segundo Exemplo Usando Classes em C++

```

/*
 * Aula 9 - Exemplo 2
 * Arquivo countCap9Ex2.h
 * Autor: Miguel Campista
 */
#ifndef COUNT_H
#define COUNT_H
#include <iostream>

using namespace std;

class Count {
public:
    // Configura o valor do membro de dados private x
    void setX (int value) {
        x = value;
    }
    // Imprime o valor do membro de dados private x
    void getX () {
        cout << x << endl;
    }
private:
    int x;
};
#endif

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Segundo Exemplo Usando Classes em C++

```

/*
 * Aula 9 - Exemplo 2
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include "countCap9Ex2.h"

int main () {
    Count counter; // Objeto counter
    Count *counterPtr = &counter; // Ponteiro para counter
    Count &counterRef = counter; // Referência para counter

    cout << "Atribui 1 a x e imprime usando o nome do objeto: ";
    counter.setX(1);
    counter.getX();

    cout << "Atribui 2 a x e imprime usando a referência para o objeto: ";
    counterRef.setX(2);
    counterRef.getX();

    cout << "Atribui 3 a x e imprime usando o ponteiro para o objeto: ";
    counterPtr->setX(3);
    counterPtr->getX();

    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

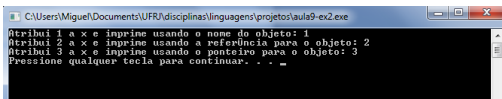
Prof. Miguel Campista

## Segundo Exemplo Usando Classes em C++

```

/*
 * Aula 9 - Exemplo 2
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include "countCap9Ex2.h"

```



```

    counter.getX();

    cout << "Atribui 2 a x e imprime usando a referência para o objeto: ";
    counterRef.setX(2);
    counterRef.getX();

    cout << "Atribui 3 a x e imprime usando o ponteiro para o objeto: ";
    counterPtr->setX(3);
    counterPtr->getX();

    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Destrutores

- Uma função-membro especial
- O nome é o caractere til (~) seguido pelo nome da classe
  - Por exemplo, ~Time
- É chamado implicitamente quando um objeto é destruído
  - Por exemplo, isso ocorre quando um objeto automático é destruído porque a execução do programa deixou o escopo no qual esse objeto estava instanciado

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Destrutores

- Não liberam a memória do objeto
  - Realizam uma "faxina de terminação"
  - Em seguida, o sistema reivindica a memória do objeto
    - Memória pode ser reutilizada para abrigar novos objetos

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Destrutores

- Não recebem nenhum parâmetro e não retornam nenhum valor
  - Não especificam tipo de retorno (nem mesmo void)
    - É um erro de sintaxe:
      - Passar argumentos para um destrutor
      - Especificar um tipo de retorno (mesmo void não pode ser especificado)
      - Retornar valores de um destrutor
      - Sobrecarregar um destrutor

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Destrutores

- Uma classe pode ter um único destrutor
  - A sobrecarga de destrutores não é permitida
- Se o programador não fornecer um destrutor explicitamente...
  - O compilador criará um destrutor "vazio"

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quando Construtores e Destrutores são Chamados?

- São chamados implicitamente pelo compilador
  - A ordem dessas chamadas de função depende da ordem segundo a qual a execução entra e sai dos escopos em que os objetos estão instanciados
- Geralmente,
  - As chamadas de destrutor são feitas na ordem inversa às chamadas de construtor correspondentes
    - **Último objeto construído é o primeiro a ser destruído**
- Entretanto,
  - As classes de armazenamento de objetos podem alterar a ordem segundo a qual os destrutores são chamados

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quando Construtores e Destrutores são Chamados?

- Para os objetos definidos no escopo global
  - Os construtores são chamados antes que qualquer outra função (incluindo main) nesse arquivo inicie a execução
  - Os destrutores correspondentes são chamados quando main termina

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quando Construtores e Destrutores são Chamados?

- Função `exit`
  - Força um programa a terminar imediatamente
    - **Não executa os destrutores de objetos automáticos, mas executa os destrutores de objetos globais e estáticos**
  - Em geral, é usada para terminar um programa quando é detectado um erro

```
A a;
void test() {
    static A b;
    A c;
    exit(0); // Não executa o destrutor de c
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quando Construtores e Destrutores são Chamados?

- Função `abort`
  - É semelhante à função `exit`
    - **Mas força o programa a terminar imediatamente sem permitir que os destrutores de qualquer objeto sejam chamados**
  - Normalmente, é usada para indicar uma terminação anormal do programa

```
A a;
void test() {
    static A b;
    A c;
    abort(); // Não executa o destrutor de a, b, c
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quando Construtores e Destrutores são Chamados?

- Para um objeto local automático
  - O construtor é chamado quando esse objeto é definido
  - O destrutor correspondente é chamado quando a execução sai do escopo do objeto
- Resumindo...
  - Os construtores e destrutores são chamados toda vez que a execução entra e sai do escopo do objeto
  - Os destrutores de objeto automático não serão chamados se o programa terminar com uma função `exit` ou `abort`

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quando Construtores e Destrutores são Chamados?

- Para um objeto local `static`
  - O construtor é chamado uma única vez
    - Quando a execução atinge pela primeira vez o local em que o objeto é definido
  - O destrutor é chamado quando `main` termina ou o programa chama a função `exit`
    - O destrutor não será chamado se o programa terminar com uma chamada para a função `abort`

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quando Construtores e Destrutores são Chamados?

- Os objetos global e `static` são destruídos na ordem inversa à que foram criados
  - Primeiro cria-se os objetos globais e depois os `static`
  - Inversamente, primeiro destrói-se os objetos `static` e depois os globais

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 8
 * Arquivo createdestroy.h
 * Autor: Miguel Campista
 */
#ifndef CREATE_H
#define CREATE_H

#include <string>
#include <iostream>

using namespace std;

class CreateAndDestroy {
public:
    CreateAndDestroy (int, string);
    ~CreateAndDestroy ();
private:
    int objID;
    string message;
};

#endif
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 8
 * Arquivo createdestroy.cpp
 * Autor: Miguel Campista
 */
#include "createdestroy.h"

CreateAndDestroy::CreateAndDestroy (int ID, string messageString) {
    objID = ID;
    message = messageString;

    cout << "Objeto " << objID << " construtor executa "
         << message << endl;
}

CreateAndDestroy::~CreateAndDestroy () {
    cout << (objID == 1 || objID == 6 ? "\n" : "");
    cout << "Objeto " << objID << " destrutor executa "
         << message << endl;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 8
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include "createdestroy.h"

void create () {
    CreateAndDestroy first (1, "(global before main)"); // objeto global
}

int main () {
    cout << "\nFUNCAO PRINCIPAL COMEÇOU EXECUCAO:" << endl;
    CreateAndDestroy second (2, "(local automatic in main)");
    static CreateAndDestroy third (3, "(local static in main)");

    create (); // Chama função para criar objetos

    cout << "\nFUNCAO PRINCIPAL REINICIA EXECUCAO:" << endl;
    CreateAndDestroy fourth (4, "(local automatic in main)");
    cout << "\nFUNCAO PRINCIPAL TERMINA EXECUCAO:" << endl;

    return 0;
}

void create () {
    cout << "\nFUNCAO CREATE COMEÇOU EXECUCAO:" << endl;
    CreateAndDestroy fifth (5, "(local automatic in create)");
    static CreateAndDestroy sixth (6, "(local static in create)");
    CreateAndDestroy seventh (7, "(local automatic in create)");
    cout << "\nFUNCAO CREATE TERMINA EXECUCAO:" << endl;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 8
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "createdestroy.h"

void create () {
    CreateAndDestroy first (1, "(global before main)"); // objeto global
}

int main () {
    cout << "\nFUNCAO PRINCIPAL COMEÇOU EXECUCAO:" << endl;
    CreateAndDestroy second (2, "(local automatic in main)");
    static CreateAndDestroy third (3, "(local static in main)");

    create (); // Chama função para criar objetos

    cout << "\nFUNCAO PRINCIPAL REINICIA EXECUCAO:" << endl;
    CreateAndDestroy fourth (4, "(local automatic in main)");
    cout << "\nFUNCAO PRINCIPAL TERMINA EXECUCAO:" << endl;

    return 0;
}

void create () {
    cout << "\nFUNCAO CREATE COMEÇOU EXECUCAO:" << endl;
    CreateAndDestroy fifth (5, "(local automatic in create)");
    static CreateAndDestroy sixth (6, "(local static in create)");
    CreateAndDestroy seventh (7, "(local automatic in create)");
    cout << "\nFUNCAO CREATE TERMINA EXECUCAO:" << endl;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Estudo de Caso da Classe Time

- Retornando uma referência a um objeto
  - Alias para o nome de um objeto
  - Um *lvalue* aceitável que pode receber um valor
    - Pode ser usado no lado esquerdo de uma instrução de atribuição
  - Se uma função retornar uma referência *const*
    - Essa referência não poderá ser usada como um *lvalue* modificável

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Estudo de Caso da Classe Time

- Retorno de uma referência a um objeto
  - Uma forma "arriscada" de usar essa capacidade
    - Uma função-membro *public* de uma classe retorna uma referência a um membro de dados *private* dessa classe
      - O código-cliente poderia alterar os dados *private*
      - O mesmo problema ocorreria se retornasse um ponteiro para dados *private*
  - Retornar uma referência ou um ponteiro para um membro de dados *private* quebra o encapsulamento da classe

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Classes em C++

```

/*
 * Aula 9 - Exemplo 6
 * Arquivo timeCap08x6.h
 * Autor: Miguel Campista
 */
#ifndef TIME_H
#define TIME_H

using namespace std;

class Time {
public:
    Time (int = 0, int = 0, int = 0); // Construtor padrão

    void setTime (int, int, int);
    void setHour (int);
    void setMinute (int);
    void setSecond (int);
    int getHour ();
    int getMinute ();
    int getSecond ();

    void printUniversal ();
    void printStandard ();

    int &addSetHour (int); // retorno de referência "PERIGOSO"

private:
    int hour; // 0 - 23
    int minute; // 0 - 59
    int second; // 0 - 59
};
#endif
    
```

## Quarto Exemplo Usando Classes em C++

```

/*
 * Aula 9 - Exemplo 6
 * Arquivo timeCap08x6.h
 * Autor: Miguel Campista
 */
#ifndef TIME_H
#define TIME_H

using namespace std;

class Time {
public:
    Time (int = 0, int = 0, int = 0); // Construtor padrão

    void setTime (int, int, int);
    void setHour (int);
    void setMinute (int);
    void setSecond (int);
    int getHour ();
    int getMinute ();
    int getSecond ();

    void printUniversal ();
    void printStandard ();

    int &addSetHour (int); // retorno de referência "PERIGOSO"

private:
    int hour; // 0 - 23
    int minute; // 0 - 59
    int second; // 0 - 59
};
#endif
    
```

Protótipo de uma função que retorna uma referência

## Quarto Exemplo Usando Classes em C++

```

/*
 * Aula 9 - Exemplo 6
 * Arquivo timeCap08x6.cpp
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include "timeCap08x6.h"

// Construtor padrão inicializa cada membro de dados com zero;
// logo, eis assegura que os objetos Time iniciem em um estado consistente
Time::Time (int hr, int min, int sec) {
    setTime (hr, min, sec);
}

void Time::setTime (int h, int m, int s) {
    setHour (h);
    setMinute (m);
    setSecond (s);
}

void Time::setHour (int h) {
    hour = (h >= 0 && h < 24) ? h : 0; // valida horas
}

void Time::setMinute (int m) {
    minute = (m >= 0 && m < 60) ? m : 0; // valida minutos
}

void Time::setSecond (int s) {
    second = (s >= 0 && s < 60) ? s : 0; // valida segundos
}
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Classes em C++

```

int Time::getHour () { return hour; }

int Time::getMinute () { return minute; }

int Time::getSecond () { return second; }

void Time::printUniversal () {
    cout << setfill('0') << setw(2) << hour << ":" <<
        << setw(2) << minute << ":" << setw(2) << second;
}

void Time::printStandard () {
    cout << ((hour == 0 || hour == 12) ? 12 : hour % 12) << ":" <<
        << setfill('0') << setw(2) << minute << ":" << setw(2) <<
        << second << (hour < 12 ? " AM" : " PM");
}

// Próxima de programação não recomendada
int &Time::addSetHour (int hh) {
    hour = (hh >= 0 && hh < 24) ? hh : 0;
    return hour; // retorno de referência
}
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Classes em C++

```
int Time::getHour () { return hour; }
int Time::getMinute () { return minute; }
int Time::getSecond () { return second; }
void Time::printUniversal () {
    cout << setfill('0') << setw(2) << setw(2) << " " << endl;
}
void Time::printStandard () {
    cout << ((hour == 0) ? "hour == 12" : hour < 12 ? "hour < 12" : "PM") << endl;
}
// Pratica de programação não recomendada
int &Time::badSetHour (int hh) {
    hour = (hh >= 0 && hh < 24) ? hh : 0;
    return hour; // retorno de referência
}
```

Função que retorna uma referência para um atributo privado

## Quarto Exemplo Usando Classes em C++

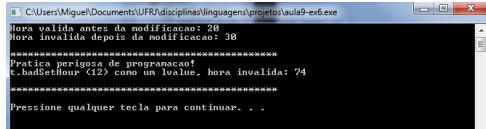
```
/*
 * Aula 9 - Exemplo 6
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include "timeCap9Ex6.h"

int main () {
    Time t;
    // Inicializa hourRef com a referência retornada por badSetHour
    int &hourRef = t.badSetHour (20); // Hora válida
    cout << "Hora válida antes da modificação: " << hourRef;
    hourRef = 30; // Usa hourRef para atribuir um valor inválido
    cout << "Inhora inválida depois da modificação: " << t.getHour();
    // Perigo: Chamada de função que retorna uma referência
    // pode ser usado como um lvalue!
    t.badSetHour (12) = 74;
    cout << "\n\n-----\n"
        << "Pratica perigosa de programação!\n"
        << "t.badSetHour (12) como um lvalue, hora inválida: "
        << t.getHour ()
        << "\n\n-----\n"
        << endl;
    return 0;
}
```

## Quarto Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 6
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include "timeCap9Ex6.h"

int main () {
    Time t;
    // Inicializa hourRef com a referência retornada por badSetHour
    int &hourRef = t.badSetHour (20); // Hora válida
    cout << "Hora válida antes da modificação: " << hourRef;
    hourRef = 30; // Usa hourRef para atribuir um valor inválido
    cout << "Inhora inválida depois da modificação: " << t.getHour();
    // Perigo: Chamada de função que retorna uma referência
    // pode ser usado como um lvalue!
    t.badSetHour (12) = 74;
    cout << "\n\n-----\n"
        << "Pratica perigosa de programação!\n"
        << "t.badSetHour (12) como um lvalue, hora inválida: "
        << t.getHour ()
        << "\n\n-----\n"
        << endl;
    return 0;
}
```



## Atribuição-padrão Membro a Membro

- Operador de atribuição (=)
  - Pode ser usado para atribuir um objeto a outro objeto do mesmo tipo
    - Cada membro de dados do objeto à direita é atribuído ao mesmo membro de dados do objeto à esquerda
  - Isso pode provocar sérios problemas quando os membros de dados contêm ponteiros para memória alocada dinamicamente
    - Essa memória poderia ser desalocada...

## Quinto Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 7
 * Arquivo dateCap9Ex7.h
 * Autor: Miguel Campista
 */
#ifndef DATE_H
#define DATE_H

#include <iostream>
using namespace std;

class Date {
public:
    Date (int m = 1, int d = 1, int y = 2000);
    void print ();
private:
    int month, day, year;
};

#endif
```

## Quinto Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 7
 * Arquivo dateCap9Ex7.cpp
 * Autor: Miguel Campista
 */
#include "dateCap9Ex7.h"

Date::Date (int m, int d, int y) {
    month = m;
    day = d;
    year = y;
}

void Date::print () {
    cout << month << '/' << day << '/' << year;
}
```

## Quinto Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 7
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "dateCap9Ex7.h"

int main () {
    Date date1 (7, 4, 2010);
    Date date2; // Assume padrão 1/1/2000

    cout << "date1 = ";
    date1.print ();
    cout << "\ndate2 = ";
    date2.print ();

    date2 = date1; // Atribuição padrão de membro a membro

    cout << "\n\nDepois atribuição padrão membro a membro, date2 = ";
    date2.print ();
    cout << endl;

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quinto Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo 7
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "dateCap9Ex7.h"


int main () {
    Date date1 (7, 4, 2010);
    Date date2; // Assume padrão 1/1/2000

    cout << "date1 = ";
    date1.print ();
    cout << "\ndate2 = ";
    date2.print ();

    date2 = date1; // Atribuição padrão de membro a membro

    cout << "\n\nDepois atribuição padrão membro a membro, date2 = ";
    date2.print ();
    cout << endl;

    return 0;
}
```



POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Atribuição-padrão Membro a Membro

- Construtor de cópia
  - Permite que os objetos sejam passados por valor
    - É usado para copiar valores originais do objeto em um novo objeto passado a uma função ou que retornou de uma função
  - O compilador fornece um construtor-padrão de cópia
    - Copia cada membro do objeto original no membro correspondente do novo objeto (ou seja, é uma atribuição de membro a membro)
  - Também pode provocar sérios problemas quando os membros de dados contêm ponteiros para memória alocada dinamicamente

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Atribuição-padrão Membro a Membro

- A passagem de um objeto por valor é adequada do ponto de vista de segurança
  - A função chamada não tem acesso ao objeto original no chamador, mas pode diminuir o desempenho ao fazer uma cópia de um objeto grande
- É possível passar um objeto por referência passando um ponteiro ou uma referência ao objeto
  - A passagem por referência oferece bom desempenho, mas menor segurança porque a função chamada recebe acesso ao objeto original

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Atribuição-padrão Membro a Membro

- A passagem por referência `const` é uma alternativa segura de bom desempenho
  - Pode ser implementada com um parâmetro de referência `const` ou com um parâmetro de ponteiro para dados `const`

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sexto Exemplo Usando Classes em C++

```
/*
 * Aula 9 - Exemplo de Construtor de Cópia
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include <iostream>
using namespace std;

class C {
public:
    C (int s = 10) : size (s), data (new int [s]) {
        cout << "Construtor aqui!" << endl;
    }
    ~C () { delete [] data; }
    void getSize () { cout << size << endl; }
    void getData () { cout << data [0] << endl; }

private:
    int size;
    int *data;
};

int main () {
    C obj (10);
    {
        C copiaObj (obj);
        copiaObj.getSize ();
    }
    obj.getData ();
    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista



## Sexto Exemplo Usando Classes em C++

```

/* Aula 9 - Exemplo de Construtor de Cópia
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include <iostream>
using namespace std;

class C {
public:
    C (int s = 10) : size (s), data (new int [s]) {
        cout << "Construtor Padrão!" << endl;
    }
    C (const C &copiar) : size (copiar.getSize (), data (new int [copiar.getSize ()]) {
        cout << "Construtor Cópia!" << endl;
    }
    ~C () { delete [] data; }
    void getSize () { cout << size << endl; }
    void getData () { cout << data [0] << endl; }

private:
    int size;
    int *data;
};

int main () {
    C obj (10);
    C copiaObj (obj);
    obj.getData ();
    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sexto Exemplo Usando Classes em C++

```

/* Aula 9 - Exemplo de Construtor de Cópia
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include <iostream>
using namespace std;

class C {
public:
    C (int s = 10) : size (s), data (new int [s]) {
        cout << "Construtor Padrão!" << endl;
    }
    C (const C &copiar) : size (copiar.getSize (), data (new int [copiar.getSize ()]) {
        cout << "Construtor Cópia!" << endl;
    }
    ~C () { delete [] data; }
    void getSize () { cout << size << endl; }
    void getData () { cout << data [0] << endl; }

private:
    int size;
    int *data;
};

int main () {
    C obj (10);
    C copiaObj (obj);
    obj.getData ();
    return 0;
}

```

## Sexto Exemplo Usando Classes em C++

```

/* Aula 9 - Exemplo de Construtor de Cópia
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include <iostream>
using namespace std;

class C {
public:
    C (int s = 10) : size (s), data (new int [s]) {
        cout << "Construtor Padrão!" << endl;
    }
    C (const C &copiar) : size (copiar.getSize (), data (new int [copiar.getSize ()]) {
        cout << "Construtor Cópia!" << endl;
    }
    ~C () { delete [] data; }
    void getSize () { cout << size << endl; }
    void getData () { cout << data [0] << endl; }

private:
    int size;
    int *data;
};

int main () {
    C obj (10);
    C copiaObj (obj);
    copiaObj.getSize ();
    obj.getData ();
    return 0;
}

```

Construtor de Cópia

## Sexto Exemplo Usando Classes em C++

```

/* Aula 9 - Exemplo de Construtor de Cópia
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include <iostream>
using namespace std;

class C {
public:
    C (int s = 10) : size (s), data (new int [s]) {
        cout << "Construtor Padrão!" << endl;
    }
};

int main () {
    C obj (10);
    C copiaObj (obj);
    copiaObj.getSize ();
    obj.getData ();
    return 0;
}

```

## Objetos const e Funções-membro const

- Princípio do menor privilégio
  - Um dos princípios mais fundamentais da boa engenharia de software
  - Aplica-se também a objetos
- Objetos const
  - Palavra-chave const
  - Especifica que um objeto não é modificável
  - Tentativas de modificar o objeto provocarão erros de compilação

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Objetos const e Funções-membro const

- Funções-membro const
  - Somente funções-membro const podem ser chamadas para objetos const
  - Até mesmo funções do tipo "get"
  - Funções-membro declaradas const não podem modificar o objeto

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Objetos const e Funções-membro const

- Funções-membro const
  - Uma função é especificada como const tanto em seu protótipo quanto em sua definição
  - Declarações const não são permitidas a construtores e destrutores
    - Construtores inicializam o objeto e o destrutores fazem a "faxina" em memória do objeto

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Objetos const e Funções-membro const

- Erro de compilação
  - Definir função-membro const que modifica um membro de dados de um objeto
  - Definir função-membro const que chama uma função-membro não-const da mesma classe
  - Invocar uma função-membro não-const em um objeto const
  - Declarar um construtor ou um destrutor const é um erro de compilação

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Objetos const e Funções-membro const

- Uma função-membro const pode ser sobrecarregada com uma versão não-const
  - O compilador escolhe qual deve utilizar com base no objeto em que a função é invocada
    - Se o objeto for const → o compilador utiliza a const
    - Se o objeto não for const → o compilador utiliza a não-const

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sétimo Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 1
 * Arquivo timeCap10Ex1.h
 * Autor: Miguel Campista
 */
#ifndef TIME_H
#define TIME_H

using namespace std;

class Time {
public:
    Time (int = 0, int = 0, int = 0); // Construtor padrão

    void setTime (int, int, int);
    void setHour (int);
    void setMinute (int);
    void setSecond (int);
    int getHour () const;
    int getMinute () const;
    int getSecond () const;

    void printUniversal () const;
    void printStandard () const;

private:
    int hour; // 0 - 23
    int minute; // 0 - 59
    int second; // 0 - 59
};

#endif
```

## Sétimo Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 1
 * Arquivo timeCap10Ex1.cpp
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include "timeCap10Ex1.h"

// Construtor padrão inicializa cada membro de dados com zero;
// Logo, ele assegura que os objetos Time iniciem em um estado consistente
Time::Time (int hr, int min, int sec) {
    setTime (hr, min, sec);
}

void Time::setTime (int h, int m, int s) {
    setHour (h);
    setMinute (m);
    setSecond (s);
}

void Time::setHour (int h) {
    hour = (h >= 0 && h < 24) ? h : 0; // valida horas
}

void Time::setMinute (int m) {
    minute = (m >= 0 && m < 60) ? m : 0; // valida minutos
}

void Time::setSecond (int s) {
    second = (s >= 0 && s < 60) ? s : 0; // valida segundos
}
```

## Sétimo Exemplo Usando Classes em C++

```
int Time::getHour () const { return hour; }

int Time::getMinute () const { return minute; }

int Time::getSecond () const { return second; }

void Time::printUniversal () const {
    cout << setfill('0') << setw(2) << hour << ":" <<
        setw(2) << minute << ":" << setw(2) << second;
}

void Time::printStandard () {
    cout << ((hour == 0 || hour == 12) ? 12 : hour % 12) << ":" <<
        setfill('0') << setw(2) << minute << ":" << setw(2) <<
        second << (hour < 12 ? " AM" : " PM");
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sétimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 1
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include "timeCap10Ex1.h"

int main () {
    Time wakeup (6, 45, 0);
    const Time noon (12, 0, 0);

    wakeup.setHour (10); // Objeto - Função Membro
                          // não-const - não-const
    noon.setHour (12); // const - não-const
    //
    wakeup.getHour (); // não-const - const
    //
    noon.getMinsute (); // const - const
    noon.printUniversal (); // const - const
    //
    noon.printStandard (); // const - não-const
    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sétimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 1
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include "timeCap10Ex1.h"

int main () {
    Time wakeup (6, 45, 0);
    const Time noon (12, 0, 0);

    wakeup.setHour (10); // Objeto - Função Membro
                          // não-const - não-const
    noon.setHour (12); // const - não-const X
    //
    wakeup.getHour (); // não-const - const
    //
    noon.getMinsute (); // const - const
    noon.printUniversal (); // const - const
    //
    noon.printStandard (); // const - não-const X
    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sétimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 1
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include "timeCap10Ex1.h"

int main () {
    Time wakeup (6, 45, 0);

```

Line	File	Message
15	C:\Users\Miguel\Documents\UFRJ\...	In function 'int main()':
22	C:\Users\Miguel\Documents\UFRJ\...	passing 'const Time' as 'this' argument of 'void Time::setHour(int)' discards qualifiers
	C:\Users\Miguel\Documents\UFRJ\...	passing 'const Time' as 'this' argument of 'void Time::printStandard()' discards qualifiers
	C:\Users\Miguel\Documents\UFRJ\...	[Build Error] exe: "" [x64] Error 1

```

    wakeup.getHour (); // não-const - const
    //
    noon.getMinsute (); // const - const
    noon.printUniversal (); // const - const
    //
    noon.printStandard (); // const - não-const
    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Objetos const e Funções-membro const

- Inicializadores de membro de dados
  - São necessários à inicialização
    - Membros de dados const
    - Membros de dados que são referências

**Ambos devem ser inicializados ao serem declarados!**

- Podem ser utilizados para qualquer membro de dados

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Objetos const e Funções-membro const

- Lista de inicializadores de membro
  - Aparece entre uma lista de parâmetros do construtor e a chave esquerda que inicia o corpo do construtor
  - É separada da lista de parâmetros por dois-pontos (:)

**construtor (lista de parâmetros) : lista de inicializadores de membro**

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Objetos const e Funções-membro const

- Lista de inicializadores de membro
  - Cada inicializador de membro consiste do nome do membro de dados (atributo) seguido do valor inicial do membro entre parênteses
  - Múltiplos inicializadores de membro são separados por vírgulas
  - Executa antes do corpo do construtor executar

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Oitavo Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 2
 * Arquivo incrementCap10Ex2.h
 * Autor: Miguel Campista
 */
#ifndef INCREMENT_H
#define INCREMENT_H

#include <iostream>

using namespace std;

class Increment {
public:
    Increment (int = 0, int = 1);
    void addIncrement ();
    void print () const;
private:
    int count;
    const int increment;
};

#endif
```

## Oitavo Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 2
 * Arquivo incrementCap10Ex2.cpp
 * Autor: Miguel Campista
 */
#include "incrementCap10Ex2.h"

Increment::Increment (int c, int i): count (c), increment (i) {}

void Increment::addIncrement () {
    count += increment;
}

void Increment::print () const {
    cout << "count = " << count << ", increment = " << increment << endl;
}
```

## Oitavo Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 2
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "incrementCap10Ex2.h"

int main () {
    Increment value (10, 5);

    cout << "Antes de incrementar: ";
    value.print ();

    for (int i = 0; i <= 3; i++) {
        value.addIncrement ();
        cout << "Depois do incremento " << i << ": ";
        value.print ();
    }

    return 0;
}
```

## Oitavo Exemplo Usando Classes em C++

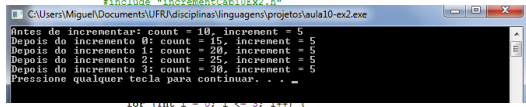
```
/*
 * Aula 10 - Exemplo 2
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "incrementCap10Ex2.h"

int main () {
    Increment value (10, 5);

    cout << "Antes de incrementar: count = 10, increment = 5\n";
    value.print ();

    for (int i = 0; i <= 3; i++) {
        value.addIncrement ();
        cout << "Depois do incremento " << i << ": ";
        value.print ();
    }

    return 0;
}
```



## Objetos const e Funções-membro const

- Um objeto `const` não pode ser modificado por atribuição
  - Logo, deve ser inicializado
    - Quando um membro de dados de uma classe é declarado `const`, um inicializador de membro deve ser utilizado para fornecer ao construtor o valor inicial do membro de dados para um objeto da classe
    - O mesmo é verdadeiro para referências
- Não fornecer um inicializador de membro para um membro de dados `const` é um erro de compilação

## Nono Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 3
 * Arquivo incrementCap10Ex3.h
 * Autor: Miguel Campista
 */
#ifndef INCREMENT_H
#define INCREMENT_H

#include <iostream>

using namespace std;

class Increment {
public:
    Increment (int = 0, int = 1);
    void addIncrement ();
    void print () const;
private:
    int count;
    const int increment;
};

#endif
```

## Nono Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 3
 * Arquivo IncrementCap10Ex2.cpp
 * Autor: Miguel Campista
 */
#include "incrementCap10Ex3.h"

Increment::Increment (int c, int i) {
    count = c;
    increment = i; // Erro!
}

void Increment::addIncrement () {
    count += increment;
}

void Increment::print () const {
    cout << "count = " << count << ", increment = " << increment << endl;
}
    
```

## Nono Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 3
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "incrementCap10Ex3.h"

int main () {
    Increment value (10, 5);

    cout << "Antes de incrementar: ";
    value.print ();

    for (int i = 0; i <= 3; i++) {
        value.addIncrement ();
        cout << "Depois do incremento " << i << ": ";
        value.print ();
    }

    return 0;
}
    
```

## Nono Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 3
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "incrementCap10Ex3.h"
    
```

Line	File	Message
8	C:\Users\Miguel\Documents\UFRJ\...	In constructor 'Increment::Increment(int, int)', uninitialized member 'Increment::increment' with 'const' type 'const int'
10	C:\Users\Miguel\Documents\UFRJ\...	assignment of read-only data-member 'Increment::increment' [Build Error] see: "" [IncrementCap10Ex3.o] Error 1

```

        value.addIncrement ();
        cout << "Depois do incremento " << i << ": ";
        value.print ();
    }

    return 0;
}
    
```

## Composição: Objetos como Membros de Classes

- Composição
  - É às vezes referida como relacionamento "tem-um"
  - Uma classe pode ter objetos de outras classes como membros
    - Ex.: Objeto **AlarmClock** com um objeto **Time** como membro

## Composição: Objetos como Membros de Classes

- Inicializando objetos-membro
  - Inicializadores de membro passam argumentos do construtor do objeto para os construtores do objeto-membro
  - Os objetos-membro são construídos na ordem em que são declarados na definição de classe
    - Não na ordem em que são relacionados na lista de inicializadores de membro do construtor
    - Antes do objeto da classe contêiner ser construído
  - Se não for fornecido um inicializador de membro...
    - O construtor-padrão do objeto-membro será chamado implicitamente

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo DateCap10Ex4.h
 * Autor: Miguel Campista
 */
#ifndef DATE_H
#define DATE_H

#include <iostream>

using namespace std;

class Date {
public:
    Date (int = 1, int = 1, int = 1900);
    void print () const;
    ~Date ();
private:
    int month, day, year;
    int checkDay (int) const;
};

#endif
    
```

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo dateCap10Ex4.cpp
 * Autor: Miguel Campista
 */
#include "dateCap10Ex4.h"

Date::Date (int m, int d, int y) {
    if (m > 0 && m <= 12) {
        month = m;
    } else {
        month = 1;
        cout << "Mes invalido (" << m << ") colocado em 1.\n";
    }
    year = y;
    day = checkDay (d);

    cout << "Construtor da classe Date ";
    print ();
    cout << endl;
}

void Date::print () const {
    cout << month << '/' << day << '/' << year;
}

Date::~Date() {
    cout << "Destruitor da classe Date ";
    print ();
    cout << endl;
}

```

## Décimo Exemplo Usando Classes em C++

```

int Date::checkDay (int testDay) const {
    static const int daysPerMonth [13] =
        {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    // Determina se o day é válido dentro do mês específico
    if (testDay > 0 && testDay <= daysPerMonth [month])
        return testDay;

    // Verifica se o ano é bissexto
    if (month == 2 && testDay == 29 && (year % 400 == 0 ||
        (year % 4 == 0 && year % 100 != 0)))
        return testDay;

    cout << "Dia invalido (" << day << ") colocado em 1.\n";
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo employeeCap10Ex4.h
 * Autor: Miguel Campista
 */
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <iostream>
#include <string>
#include "dateCap10Ex4.h"

using namespace std;

class Employee {
public:
    Employee (const char * const, const char * const,
              const Date &, const Date &);
    void print () const;
    ~Employee ();

private:
    char firstName [25];
    char lastName [25];
    const Date birthDate;
    const Date hireDate;
};

#endif

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo employeeCap10Ex4.cpp
 * Autor: Miguel Campista
 */
#include "employeeCap10Ex4.h"

Employee::Employee (const char * const first, const char * const last,
                    const Date &dateOfBirth, const Date &dateOfHire) :
    birthDate (dateOfBirth), hireDate (dateOfHire) {

    // Copia para first name
    int length = strlen (first);
    length = (length < 25 ? length : 24);
    strncpy (firstName, first, length);
    firstName [length] = '\0';

    // Copia para last name
    length = strlen (last);
    length = (length < 25 ? length : 24);
    strncpy (lastName, last, length);
    lastName [length] = '\0';

    cout << "Construtor do objeto Employee: "
        << lastName << ", " << firstName << endl;
}

void Employee::print () const {
    cout << lastName << ", " << firstName << " Hired: ";
    hireDate.print ();
    cout << " BirthDay: ";
    birthDate.print ();
    cout << endl;
}

Employee::~Employee() {
    cout << "Destruitor do objeto Employee: "
        << lastName << ", " << firstName << endl;
}

```

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo employeeCap10Ex4.cpp
 * Autor: Miguel Campista
 */
#include "employeeCap10Ex4.h"

Employee::Employee (const char * const first, const char * const last,
                    const Date &dateOfBirth, const Date &dateOfHire) :
    birthDate (dateOfBirth), hireDate (dateOfHire) {

    // Copia para first name
    int length = strlen (first);
    length = (length < 25 ? length : 24);
    strncpy (firstName, first, length);
    firstName [length] = '\0';

    // Copia para last name
    length = strlen (last);
    length = (length < 25 ? length : 24);
    strncpy (lastName, last, length);
    lastName [length] = '\0';

    cout << "Construtor do objeto Employee: "
        << lastName << ", " << firstName << endl;
}

void Employee::print () const {
    cout << lastName << ", " << firstName << " Hired: ";
    hireDate.print ();
    cout << " BirthDay: ";
    birthDate.print ();
    cout << endl;
}

Employee::~Employee() {
    cout << "Destruitor do objeto Employee: "
        << lastName << ", " << firstName << endl;
}

```

Construtores de cópia default

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "employeeCap10Ex4.h"

int main () {
    Date birth (7, 24, 1949);
    Date hire (3, 12, 1988);
    Employee manager ("Bob", "Blue", birth, hire);

    cout << endl;
    manager.print ();

    cout << "\nTeste de construtor de Dados com valores invalidos:\n";
    Date lastDateOf (14, 35, 1994); // mês e dia inválidos
    cout << endl;

    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 */
C:\Users\Miguel\Documents\UFPE\Disciplinas\linguagens\projetos\aula10-ex4.exe
Construtor da classe Date 7/24/1949
Construtor da classe Date 3/12/1988
Construtor do objeto Employee: Bob
Bob - Bob hired: 3/12/1988 BirthDay: 7/24/1949
teste do construtor de Date com valores invalidos:
Mes invalido (18) colocado em 1.
Dia invalido (33) colocado em 1.
Construtor da classe Date 1/8/1994
Pressione qualquer tecla para continuar. . .
destrutor da classe Date 1/8/1994
destrutor do objeto Employee: Bob
destrutor da classe Date 3/12/1988
destrutor da classe Date 7/24/1949
destrutor da classe Date 3/12/1988
destrutor da classe Date 7/24/1949
C:\Users\Miguel\Documents\UFPE\Disciplinas\linguagens\projetos\
return 0;
}

```

## Composição: Objetos como Membros de Classes

- Se a classe do objeto-membro não fornecer um construtor padrão...
  - Isto é, a classe do objeto-membro define um ou mais construtores, mas nenhum deles é um construtor-padrão
- Ocorre um erro de compilação se um objeto-membro não for inicializado com um inicializador de membro

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo dateCap10Ex4.h
 * Autor: Miguel Campista
 */
#ifndef DATE_H
#define DATE_H
#include <iostream>
using namespace std;
class Date {
public:
    Date(int m, int d, int y) : m(m), d(d), y(y) {}
    void print() const;
    ~Date();
private:
    int month, day, year;
    int checkDay(int) const;
};
#endif

```

Eliminar o construtor-padrão...

Date (int, int, int)

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo dateCap10Ex4.cpp
 * Autor: Miguel Campista
 */
#include "dateCap10Ex4.h"
Date::Date (int m, int d, int y) {
    if (m > 0 && m <= 12) {
        month = m;
    } else {
        month = 1;
        cout << "Mes invalido (" << m << ") colocado em 1.\n";
    }
    year = y;
    day = checkDay (d);
    cout << "Construtor da classe Date ";
    print ();
    cout << endl;
}
void Date::print () const {
    cout << month << '/' << day << '/' << year;
}
Date::~Date() {
    cout << "Destruitor da classe Date ";
    print ();
    cout << endl;
}

```

Mantém igual...

## Décimo Exemplo Usando Classes em C++

```

int Date::checkDay (int testDay) const {
    static const int daysPerMonth [18] =
        {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    // Determina se o day é válido dentro do mês específico
    if (testDay > 0 && testDay <= daysPerMonth [month])
        return testDay;
    // Verifica se o ano é bissexto
    if (month == 2 && testDay == 29 && (year % 400 == 0 ||
        (year % 4 == 0 && year % 100 != 0)))
        return testDay;
    cout << "Dia invalido (" << day << ") colocado em 1.\n";
}

```

Mantém igual...

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo employeeCap10Ex4.h
 * Autor: Miguel Campista
 */
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include <iostream>
#include <string>
#include "dateCap10Ex4.h"
using namespace std;
class Employee {
public:
    Employee (const char * const, const char * const,
        const Date &, const Date &);
    void print () const;
    ~Employee ();
private:
    char firstName [25];
    char lastName [25];
    const Date birthDate;
    const Date hireDate;
};
#endif

```

Mantém igual...

```

/*
 * Aula 10 - Exemplo 4
Employee::Employee (const char * const first, const char * const last,
                    const Date &dateOfBirth, const Date &dateOfHire) {
#include "employeeCap10Ex4.h"
Employee::Employee (const char * const first, const char * const last,
                    const Date &dateOfBirth, const Date &dateOfHire) :
    birthDate (dateOfBirth), hireDate (dateOfHire) {
// copia para first name
int length = strlen (first);
length = (length < 25 ? length : 24);
strcpy (firstName, first, length);
// copia para last name
length = strlen (last);
length = (length < 25 ? length : 24);
strcpy (lastName, last, length);
lastName [length] = '\0';

cout << "Construtor do objeto Employee: "
    << lastName << ", " << firstName << endl;

void Employee::print () const {
    cout << lastName << ", " << firstName << " Hired: ";
    birthDate.print ();
    cout << " BirthDay: ";
    birthDate.print ();
    cout << endl;
}

Employee::~Employee () {
    cout << "Destructor do objeto Employee: "
        << lastName << ", " << firstName << endl;
}
}

```

Não inicializar os objetos-membro na lista de inicialização de membro...

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo Principal
 * Autor: Miguel Campista
#include "employeeCap10Ex4.h"
int main () {
    Date birth (7, 24, 1949);
    Date hire (15, 12, 1988);
    Employee manager ("Bob", "Blue", birth, hire);

    cout << endl;
    manager.print ();

    cout << "\nTeste de construtor de Dados com valores invalidos:\n";
    Date lastDayOff (14, 35, 1994); // mês e dia inválidos
    cout << endl;

    return 0;
}

```

## Décimo Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 4
 * Arquivo Principal
 * Autor: Miguel Campista
#include "employeeCap10Ex4.h"
C:\Users\Miguel\Documents\UFRJ\... In construtor 'Employee:Employee(const char*, const char*, const Date&, const Date&)'
C:\Users\Miguel\Documents\UFRJ\... no matching function for call to 'Date:Date()'
C:\Users\Miguel\Documents\UFRJ\... candidates are: Date:Date(const Date&)
C:\Users\Miguel\Documents\UFRJ\... Date:Date(int, int)
C:\Users\Miguel\Documents\UFRJ\... no matching function for call to 'Date:Date()'
C:\Users\Miguel\Documents\UFRJ\... candidates are: Date:Date(const Date&)
C:\Users\Miguel\Documents\UFRJ\... Date:Date(int, int)
C:\Users\Miguel\Documents\UFRJ\... [Build Error] exe: "" [employeeCap10Ex4.o] Error 1
Date lastDayOff (14, 35, 1994); // mês e dia inválidos
cout << endl;

return 0;
}

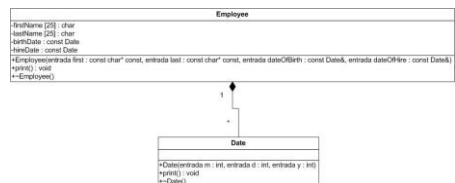
```

## Composição: Objetos como Membros de Classes

- Inicialize explicitamente objetos-membro por meio de inicializadores de membro
- Isso elimina o overhead de "inicializar duplamente" objetos-membro
- Uma vez quando o construtor-padrão do objeto-membro for chamado e outra quando as funções set forem chamadas no corpo do construtor (ou posteriormente) para inicializar o objeto-membro

Mesma coisa acontece quando se inicializa um objeto e depois se inicializa os atributos do objeto...

## Composição: Objetos como Membros de Classes



## Utilizando o Ponteiro this

- As funções-membro sabem que membros de dados do objeto devem manipular
- Todo objeto tem acesso a seu próprio endereço por meio do ponteiro chamado this
- Palavra-chave de C++
- O ponteiro this do objeto não faz parte do objeto em si
- O ponteiro this é passado (pelo compilador) como um argumento implícito para cada uma das funções-membro não-static do objeto



## Utilizando o Ponteiro this

- Os objetos usam o ponteiro `this` implicitamente ou explicitamente
  - Implicitamente, quando acessa membros de maneira direta
  - Explicitamente, quando usa a palavra-chave `this`
- O tipo do ponteiro `this` depende do tipo de objeto e se a função-membro que está executando está declarada como `const`
  - Se a função-membro for não-const → ponteiro `this` é `const` e os dados são não-const
  - Se a função-membro for `const` → ponteiro `this` é `const` e os dados são `const`

## Décimo Primeiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 7
 * Arquivo testCap10Ex7.h
 * Autor: Miguel Campista
 */
#ifndef TEST_H
#define TEST_H

#include <iostream>

using namespace std;

class Test {
public:
    Test (int = 0);
    void print () const;
private:
    int x;
};

#endif
    
```

## Décimo Primeiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 7
 * Arquivo testCap10Ex7.cpp
 * Autor: Miguel Campista
 */
#include "testCap10Ex7.h"

Test::Test (int value) : x (value) {}

void Test::print () const {
    // Utiliza implicitamente o ponteiro this para acessar x
    cout << "    x = " << x << endl;

    // Utiliza explicitamente o ponteiro this para acessar x
    cout << "    this->x = " << this->x;

    // Utiliza explicitamente o ponteiro this desreferenciado
    // e o operador ponto para acessar o membro x
    cout << "    !n(*this).x = " << (*this).x << endl;
}
    
```

## Décimo Primeiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 7
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "testCap10Ex7.h"

int main() {
    Test testObj (12);
    testObj.print ();

    return 0;
}
    
```

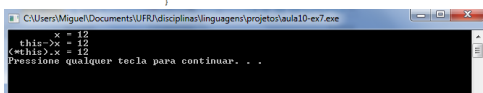
## Décimo Primeiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 7
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "testCap10Ex7.h"

int main() {
    Test testObj (12);
    testObj.print ();

    return 0;
}
    
```



## Utilizando o Ponteiro this

- Tentar utilizar o operador de seleção de membro (`.`) com um ponteiro para um objeto é um erro de compilação
  - O operador ponto de seleção de membro pode ser utilizado apenas com um *lvalue* como o nome de um objeto, uma referência para um objeto ou um ponteiro desreferenciado para um objeto

**// Nome**  
**Classe obj;**  
**obj.x;**

**// Referência**  
**Classe &refObj = obj;**  
**refObj.x;**

**// Ponteiro**  
**Classe \*ptrObj = &obj;**  
**(\*ptrObj).x;**

## Utilizando o Ponteiro this

- Chamadas de funções-membro em cascata
  - Múltiplas funções são invocadas na mesma instrução
  - São habilitadas pelas funções-membro que retornam o ponteiro this desreferenciado
- Ex.: `t.setMinute( 30 ).setSecond( 22 );`  
 Chamadas `t.setMinute( 30 );`  
 Em seguida, chamadas `t.setSecond( 22 );`

## Décimo Segundo Exemplo Usando Classes em C++

```

* Aula 10 - Exemplo 8
* Arquivo timeCap10Ex8.h
* Autor: Miguel Campista
*/

#ifndef TIME_H
#define TIME_H

#include <iostream>

using namespace std;

class Time {
public:
    Time (int = 0, int = 0, int = 0); // Construtor padrão

    Time &setTime (int, int, int);
    Time &setHour (int);
    Time &setMinute (int);
    Time &setSecond (int);

    int getHour () const;
    int getMinute () const;
    int getSecond () const;

    void printUniversal () const;
    void printStandard () const;

private:
    int hour; // 0 - 23
    int minutes; // 0 - 59
    int second; // 0 - 59
};

#endif
    
```

## Décimo Segundo Exemplo Usando Classes em C++

```

* Aula 10 - Exemplo 8
* Arquivo timeCap10Ex8.cpp
* Autor: Miguel Campista
*/

#include <iostream>
#include <iomanip>
#include "timeCap10Ex8.h"

// Construtor padrão inicializa cada membro de dados com zero;
// Logo, ele assegura que os objetos Time iniciem em um estado consistente
Time::Time (int hr, int min, int sec) {
    setTime (hr, min, sec);
}

Time &Time::setTime (int h, int m, int s) {
    setHour (h);
    setMinute (m);
    setSecond (s);
    return *this; // permite cascateamento
}

Time &Time::setHour (int h) {
    hour = (h >= 0 && h < 24) ? h : 0; // valida horas
    return *this; // permite cascateamento
}

Time &Time::setMinute (int m) {
    minutes = (m >= 0 && m < 60) ? m : 0; // valida minutos
    return *this; // permite cascateamento
}

Time &Time::setSecond (int s) {
    second = (s >= 0 && s < 60) ? s : 0; // valida segundos
    return *this; // permite cascateamento
}
    
```

## Décimo Segundo Exemplo Usando Classes em C++

```

int Time::getHour () const { return hour; }

int Time::getMinute () const { return minutes; }

int Time::getSecond () const { return second; }

void Time::printUniversal () const {
    cout << setfill('0') << setw(2) << hour << ":" <<
        << setw(2) << minutes << ":" << setw(2) << second;
}

void Time::printStandard () const {
    cout << ((hour == 0 || hour == 12) ? 12 : hour % 12) << ":" <<
        << setfill('0') << setw(2) << minutes << ":" << setw(2) <<
        << second << ((hour < 12 ? " AM" : " PM"));
}
    
```

## Décimo Segundo Exemplo Usando Classes em C++

```

/*
* Aula 10 - Exemplo 8
* Arquivo Principal
* Autor: Miguel Campista
*/
#include "timeCap10Ex8.h"

int main() {
    Time t;

    // Chamada em cascata de funções
    t.setHour(18).setMinute(30).setSecond(22);

    // sera saída em formato universal e padrão
    cout << "Formato universal: ";
    t.printUniversal ();
    cout << "\n\nFormato padrao: ";
    t.printStandard ();

    cout << "\n\nNova hora padrao: ";
    // chamada em cascata
    t.setTime(20, 20, 20).printStandard();
    cout << endl;

    return 0;
}
    
```

## Décimo Segundo Exemplo Usando Classes em C++

```

/*
* Aula 10 - Exemplo 8
* Arquivo Principal
* Autor: Miguel Campista
*/
#include "timeCap10Ex8.h"

int main() {
    Time t;

    // Chamada em cascata de funções
    t.setHour(18).setMinute(30).setSecond(22);

    // sera saída em formato universal e padrão
    cout << "Formato universal: ";
    t.printUniversal ();
    cout << "\n\nFormato padrao: ";
    t.printStandard ();

    cout << "\n\nNova hora padrao: ";
    // chamada em cascata
    t.setTime(20, 20, 20).printStandard();
    cout << endl;

    return 0;
}
    
```

## Gerenciamento de Memória Dinâmico

- Permite que os programadores aloquem e desaloquem memória para qualquer tipo predefinido ou definido pelo usuário
- É realizado pelos operadores **new** e **delete**
- Por exemplo, alocar memória dinamicamente para um array, em vez de usar um array de tamanho fixo

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Gerenciamento de Memória Dinâmico

- Operador **new**
  - Aloca (isto é, reserva) armazenamento de tamanho apropriado para um objeto em tempo de execução
  - Chama o construtor para inicializar o objeto
  - Retorna um ponteiro do tipo especificado à direita de **new**
  - Pode ser usado para alocar dinamicamente qualquer tipo fundamental (como **int** ou **double**) ou qualquer tipo de objeto de classe

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Gerenciamento de Memória Dinâmico

- Armazenamento livre
  - É também chamado de **heap**
    - Área de memória alocada para variáveis alocadas dinamicamente
  - Região da memória atribuída a cada programa para armazenar variáveis (objetos) criadas em tempo de execução

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Gerenciamento de Memória Dinâmico

- Operador **delete**
  - Destrói um objeto alocado dinamicamente
  - Chama o destrutor do objeto
  - Desaloca (isto é, libera) memória do armazenamento livre
  - A memória pode então ser reutilizada pelo sistema para alocar outros objetos

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Gerenciamento de Memória Dinâmico

- Inicialização de um objeto alocado por **new**
  - Inicializador para uma variável do tipo fundamental recém-criada
    - Exemplo
      - `double *ptr = new double (3.14159);`
  - Especifique uma lista de argumentos separada por vírgula ao construtor de um objeto
    - Exemplo
      - `Time *timePtr = new Time (12, 45, 0);`

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Gerenciamento de Memória Dinâmico

- Inicialização de um objeto alocado por **new**
  - Inicializador para uma variável do tipo fundamental recém-criada
    - Exemplo
      - `double *ptr = new double (3.14159);`
  - Especifique uma lista de argumentos separada por vírgula ao construtor de um objeto
    - Exemplo
      - `Time *timePtr = new Time (12, 45, 0);`

Não liberar memória alocada dinamicamente quando não for mais necessária pode fazer com que o sistema fique sem memória prematuramente. Isso às vezes é chamado de "vazamento de memória"

## Gerenciamento de Memória Dinâmica

- O operador `new` pode ser usado para alocar arrays dinamicamente
  - Aloque dinamicamente um array de inteiros de 10 elementos:
    - Exemplo:
      - `int *gradesArray = new int [10];`
  - O tamanho do array alocado dinamicamente
    - É especificado por meio de qualquer expressão integral que possa ser avaliada em tempo de execução

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Gerenciamento de Memória Dinâmica

- Exclua um array alocado dinamicamente:
  - `delete [] gradesArray;`
  - Isso desaloca o array para o qual `gradesArray` aponta
  - Se o ponteiro apontar para um array de objetos
    - Primeiro chame o destrutor para cada objeto no array
    - Em seguida, desaloque a memória
  - Se a instrução não incluir os colchetes `[]` e `gradesArray` apontar para um array de objetos
    - Apenas o primeiro objeto no array terá a chamada de destrutor

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Classes Proxy

- Os arquivos de cabeçalho contêm parte da implementação de uma classe e dicas sobre outras
  - Por exemplo, os membros de uma classe `private` estão relacionados na definição de classe em um arquivo de cabeçalho
  - Existe a possibilidade de exporem informações proprietárias aos clientes da classe

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Classes Proxy

- Classe proxy
  - Oculta dos clientes até mesmo os dados `private` de uma classe
  - Conhece apenas a interface `public` de sua classe
  - Permite que os clientes usem os serviços de sua classe sem lhe conceder acesso aos detalhes de implementação de sua classe
  - Isola o código-cliente das alterações na implementação

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Décimo Terceiro Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 10
 * Arquivo implementationCap10Ex10.h
 * Autor: Miguel Campista
 */
#ifndef IMPLEMENTATION_H
#define IMPLEMENTATION_H

class Implementation {
public:
    Implementation (int);
    void setValue (int);
    int getValue () const;
private:
    int value;
};

#endif
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Décimo Terceiro Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 10
 * Arquivo implementationCap10Ex10.cpp
 * Autor: Miguel Campista
 */
#include "implementationCap10Ex10.h"

Implementation::Implementation (int v): value (v) {}

void Implementation::setValue (int v) {
    value = v;
}

int Implementation::getValue () const {
    return value;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Décimo Terceiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 10
 * Arquivo InterfaceCap10Ex10.h
 * Autor: Miguel Campista
 */
#ifndef INTERFACE_H
#define INTERFACE_H

#include <iostream>

using namespace std;

class Implementation; // Declaração de classe antecipada

class Interface {
public:
    Interface (int);
    void setValue (int);
    int getValue () const;
    ~Interface ();
private:
    Implementation *ptr;
};

#endif

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Décimo Terceiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 10
 * Arquivo InterfaceCap10Ex10.cpp
 * Autor: Miguel Campista
 */
#include "ImplementationCap10Ex10.h"
#include "InterfaceCap10Ex10.h"

Interface::Interface (int v) : ptr (new Implementation (v)) {}

void Interface::setValue (int v) {
    ptr->setValue (v);
}

int Interface::getValue () const {
    ptr->getValue ();
}

Interface::~Interface () {
    delete ptr;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Décimo Terceiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 10
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "InterfaceCap10Ex10.h"

int main() {
    Interface i (5); // Cria objeto Interface

    cout << "Interface contem: " << i.getValue ()
         << " antes do setValue" << endl;

    i.setValue (10);

    cout << "\nInterface contem: " << i.getValue ()
         << " depois do setValue\n" << endl;

    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Décimo Terceiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 10
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "InterfaceCap10Ex10.h"

int main() {
    Interface i (5); // Cria objeto Interface

    cout << "Interface contem: 5 antes do setValue
Interface contem: 10 depois do setValue
Pressione qualquer tecla para continuar. . .

    i.setValue (10);

    cout << "\nInterface contem: " << i.getValue ()
         << " depois do setValue\n" << endl;

    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 1

- Escreva um programa que receba a largura e o comprimento de um retângulo e imprima as coordenadas cartesianas desse quadrado. Assuma que uma das coordenadas é o ponto (0, 0). O retângulo deve possuir uma largura e um comprimento default que devem ser alterados caso o usuário deseje.

?

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

```

/*
 * Aula 9 - Exemplo 8
 * Programa rectangleCap9Ex8.h
 * Autor: Miguel Campista
 */
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include <iostream>
#include "pointCap9Ex8.h"

using namespace std;

class Rectangle {
public:
    Rectangle (float = 1, float = 1);
    ~Rectangle ();

    float getLength ();
    float getWidth ();
    void getCoord ();
    void setLength ();
    void setWidth ();
    void setFourPoints ();

private:
    float length, width;
    static const float max_length = 20, max_width = 20;
    Point points [4];

    bool checkLength (float);
    bool checkWidth (float);
};

#endif

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

```

/*
 * Aula 9 - Exemplo 8
 * Programa rectangleCap9Ex8.cpp
 * Autor: Miguel Campista
 */
#include "rectangleCap9Ex8.h"

Rectangle::Rectangle(float l, float w) {
    cout << "No construtor...\n";
    length = l;
    width = w;
    setFourPoints ();
}

Rectangle::~Rectangle () {
    cout << "No destrutor...\n";
}

float Rectangle::getLength () { return length; }

float Rectangle::getWidth () { return width; }

void Rectangle::getCoord () {
    for (int i = 0; i < 4; i++)
        points [i].getCoord ();
}

void Rectangle::setLength () {
    float l;
    cout << "Entre com o comprimento do quadrado: ";
    do {
        cin >> l;
    } while (checkLength (l));
    length = l;
}

```

## Exemplo 1

```

void Rectangle::setWidth () {
    float w;
    cout << "Entre com a largura do quadrado: ";
    do {
        cin >> w;
    } while (checkWidth (w));
    width = w;
}

void Rectangle::setFourPoints () {
    for (int i = 0; i < 2; i++) {
        for (int w = 0; w < 2; w++)
            points[2*i + w].setCoord (l*length, w*width);
    }
}

bool Rectangle::checkLength (float x) {
    if ((x < 0) || (x > max_length)) {
        cout << "O comprimento deve estar entre 0 e 20.\n";
        cout << "Entre novamente com um novo comprimento.\n";
        return true;
    }
    return false;
}

bool Rectangle::checkWidth (float y) {
    if ((y < 0) || (y > max_width)) {
        cout << "O comprimento deve estar entre 0 e 20.\n";
        cout << "Entre novamente com um novo comprimento.\n";
        return true;
    }
    return false;
}

```

## Exemplo 1

```

/*
 * Aula 9 - Exemplo 8
 * Programa pointCap9Ex8.cpp
 * Autor: Miguel Campista
 */
#include "pointCap9Ex8.h"

void Point::setCoord (float x, float y) {
    coord_x = x;
    coord_y = y;
}

void Point::getCoord () {
    cout << "(" << coord_x << ", " << coord_y << ")\n";
}

```

## Exemplo 1

```

/*
 * Aula 9 - Exemplo 8
 * Programa pointCap9Ex8.h
 * Autor: Miguel Campista
 */
#ifndef POINT_H
#define POINT_H

#include <iostream>
using namespace std;

class Point {
public:
    void setCoord (float, float);
    void getCoord ();
private:
    float coord_x, coord_y;
};

#endif

```

## Exemplo 1

```

/*
 * Aula 9 - Exemplo 8
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "rectangleCap9Ex8.h"

using namespace std;

int main () {
    Rectangle r;
    string op;

    r.getCoord ();

    cout << "Mudar o valor padrao de comprimento e largura? (S/N) ";
    getline (cin, op);

    if ((!op.compare("S")) || !op.compare("s")) {
        r.setLength (l);
        r.setWidth (w);
        r.setFourPoints ();
        r.getCoord ();
    } else
        cout << "Tchau" << endl;

    return 0;
}

```

## Exemplo 2

- Escreva um programa que receba o número de lados de um polígono e escolha os pontos aleatoriamente. Use o conceito de classes proxy.



## Exemplo 2

```
/*
 * Aula 10 - Exemplo 11
 * Programa polygonCap10Ex11.h
 * Autor: Miguel Campista
 */
#ifndef POLYGON_H
#define POLYGON_H

#include <iostream>

using namespace std;

class Point {
public:
    Polygon (int);
    ~Polygon ();

    void getPoints () const;
    void getSides () const;

private:
    Point * pointer;
};

#endif
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 2

```
/*
 * Aula 10 - Exemplo 11
 * Programa polygonCap10Ex11.cpp
 * Autor: Miguel Campista
 */
#include "pointCap10Ex11.h"
#include "polygonCap10Ex11.h"

Polygon::Polygon (int s) : pointer (new Point (s)) {
    cout << "No construtor da classe poligono...\n";
}

Polygon::~Polygon () {
    cout << "No destrutor da classe poligono...\n";
    pointer->cleanPoints ();
    delete pointer;
}

void Polygon::getPoints () const {
    pointer->getPoints ();
}

void Polygon::getSides () const {
    pointer->getSides ();
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 2

```
/*
 * Aula 10 - Exemplo 11
 * Programa pointCap10Ex11.h
 * Autor: Miguel Campista
 */
#ifndef POINT_H
#define POINT_H

#include <iostream>
#include <ctime>
#include <cstdlib>
#include <cmath>
#include <omanip>

using namespace std;

class Point {
public:
    Point (int = 3);
    ~Point ();

    void getPoints () const;
    void getSides () const;
    void cleanPoints () const;

private:
    const int sides;
    int *x, *y;

    void setPoints ();
    void randomPoint (int *);
};

#endif
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 2

```
/*
 * Aula 10 - Exemplo 11
 * Programa pointCap10Ex11.cpp
 * Autor: Miguel Campista
 */
#include "pointCap10Ex11.h"

Point::Point (int s) : sides (s) {
    cout << "No construtor da classe ponto...\n";
    srand (time (0));
    setPoints ();
}

Point::~Point () {
    cout << "No destrutor da classe ponto...\n";
}

void Point::cleanPoints () const {
    delete [] x;
    delete [] y;
}

void Point::getPoints () const {
    for (int i = 0; i < sides; i++)
        cout << "x [" << i << "] << ", " << y [i] << "\n";
}

void Point::getSides () const {
    for (int i = 0; i < sides; i++)
        cout << "lado " << i << ": " << setprecision (2) << fixed
        << sqrt (pow (static_cast <double> (x [i % sides] - x [(i + 1) % sides]), 2) +
        pow (static_cast <double> (y [i % sides] - y [(i + 1) % sides]), 2))
        << endl;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 2

```
void Point::setPoints () {
    x = new int [sides];
    y = new int [sides];
    randomPoint (x);
    randomPoint (y);
}

void Point::randomPoint (int *p) {
    for (int i = 0; i < sides; i++) {
        p [i] = rand () % 20;
        cout << "Ponto " << i << ": " << p [i] << endl;
    }
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 2

```
/*
 * Aula 10 - Exemplo 11
 * Programa Principel
 * Autor: Miguel Campista
 */
#include "polygonCap10Ex11.h"

using namespace std;

int main () {
    Polygon p (4);

    p.getPoints ();
    p.getSides ();

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 2

```
/*
 * Aula 10 - Exemplo 11
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "polygonCap10Ex11.h"

C:\Windows\system32\cmd.exe
C:\Users\Miguel\Documents\IFRR\disciplinas\linguagens\projetos\aula10-ex11.exe
No construtor da classe ponto...
Ponto 0: 10
Ponto 1: 5
Ponto 2: 7
Ponto 3: 3
Ponto 4: 11
Ponto 5: 2
Ponto 6: 8
Ponto 7: 10
Ponto 8: 6
No construtor da classe poligono...
(18, 11)
(2, 2)
(7, 10)
(2, 8)
lado 0: 12,04
lado 1: 12,04
lado 2: 13,07
lado 3: 9,85
Pressione qualquer tecla para continuar. . .
No destrutor da classe poligono...
No destrutor da classe ponto...
C:\Users\Miguel\Documents\IFRR\disciplinas\linguagens\projetos>
```

## Leitura Recomendada

- Capítulos 9 e 10 do livro
  - Deitel, "C++ How to Program", 5th edition, Editora Prentice Hall, 2005