

Linguagens de Programação

Prof. Miguel Elias Mitre Campista

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

Parte II

Introdução à Programação em C++
(Continuação)

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Relembrando da Última Aula...

- Ponteiros e strings
- Construtores e destrutores
- Mais exemplos de programação orientada a objetos...

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Objetos const e Funções-membro const

- Declarar um objeto como **const** ajuda a impor o princípio do menor privilégio
 - As tentativas de modificar o objeto são capturadas em tempo de compilação e não provocam erros em tempo de execução
 - Utilizar **const** adequadamente é fundamental para o projeto da classe, o projeto do programa e a codificação adequada

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Objetos const e Funções-membro const

- Declarar variáveis e objetos **const** pode melhorar o desempenho
 - Compiladores podem realizar otimizações em constantes que não podem ser executadas em variáveis
 - Constantes podem ser substituídas pelos seus valores nas instruções compiladas
 - Parte do código pode ser eliminado se o valor da constante for utilizado como um teste que nunca é verdadeiro

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Primeiro 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 ();

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

#endif
```

Primeiro 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
}
```

Primeiro 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");
}
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Primeiro 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 (18); // Objeto - Função Membro
    noon.setHour (12); // não-const - não-const
    wakeup.getHour (); // não-const - const
    noon.getMinute (); // const - const
    noon.printUniversal (); // const - const
    noon.printStandard (); // const - não-const

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Primeiro 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 (18); // Objeto - Função Membro
    noon.setHour (12); // não-const - não-const X
    wakeup.getHour (); // não-const - const
    noon.getMinute (); // const - const
    noon.printUniversal (); // const - const
    noon.printStandard (); // const - não-const X

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Primeiro 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 (18); // Objeto - Função Membro
    noon.setHour (12); // não-const - não-const X
    wakeup.getHour (); // não-const - const
    noon.getMinute (); // const - const
    noon.printUniversal (); // const - const
    noon.printStandard (); // const - não-const X

    return 0;
}
```

Line	File	Message
15	C:\Users\Miguel\Documents\UFRJ\...	In function 'int main()': passing 'const Time' as 'this' argument of 'void Time::setHour()' discards qualifiers
22	C:\Users\Miguel\Documents\UFRJ\...	passing 'const Time' as 'this' argument of 'void Time::printStandard()' discards qualifiers
	C:\Users\Miguel\Documents\UFRJ\...	[Build Error] see ""(aula10\ex1_0)Error 1

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Segundo 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
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Segundo 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;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Segundo 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;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Segundo Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 2
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "IncrementCap10Ex2.h"
int main () {
    cout << "Antes de incrementar: count = 10, increment = 5\n";
    cout << "Depois do incremento 0: count = 15, increment = 5\n";
    cout << "Depois do incremento 1: count = 20, increment = 5\n";
    cout << "Depois do incremento 2: count = 25, increment = 5\n";
    cout << "Depois do incremento 3: count = 30, increment = 5\n";
    cout << "Pressione qualquer tecla para continuar. . .\n";
    cin.get();
    for (int i = 0; i <= 3; i++) {
        value.addIncrement ();
        cout << "Depois do incremento " << i << ": ";
        value.print ();
    }
    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Objetos const e Funções-membro const

- Declare como `const` todas as funções-membro de uma classe que não modificam o objeto em que elas operam
 - Ocasionalmente, isso pode parecer inadequado, caso não haja intenção de criar objetos `const` dessa classe ou de acessar objetos dessa classe por meio de referências `const` ou ponteiros para `const`
 - Apesar disso, declarar essas funções-membro `const` oferece benefício
 - Se a função-membro for inadvertidamente escrita para modificar o objeto, o compilador emitirá uma mensagem de erro

Terceiro 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
    
```

Terceiro Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 3
 * Arquivo incrementCap10Ex3.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;
}
    
```

Terceiro 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;
}
    
```

Terceiro 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)',
8	C:\Users\Miguel\Documents\UFRJ\...	uninitialized member 'Increment::increment' with 'const type' 'const int'
10	C:\Users\Miguel\Documents\UFRJ\...	assignment of read-only data member 'Increment::increment'
10	C:\Users\Miguel\Documents\UFRJ\...	[Build Error] exe: "" [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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quarto 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
    
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quarto 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 << "Destructor da classe Date ";
    print ();
    cout << endl;
}
    
```

Quarto 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
    
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quarto 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 valido dentro do mês especifico
    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";
}
    
```

Linguagens de Programação – DEL-Poli/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 << "Destructor do objeto Employee: ";
    << lastName << ", " << firstName << endl;
}
    
```

```

//
* 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 &dateOfHire, const Date &dateOfRetire) :
    birthDate (dateOfHire), hireDate (dateOfRetire) {
    // Copia para first name
    int length = strlen (first);
    length = (length < 25 ? length : 24);
    strcpy (firstName, first, length);
    firstName [length] = '\0';
    // 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: ";
    hireDate.print ();
    cout << " BirthDay: ";
    birthDate.print ();
    cout << endl;
}

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

```

Construtores de cópia default

Quarto 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 (13, 12, 1988);
    Employee manager ("Bob", "Blue", birth, hire);

    cout << endl;
    manager.print ();

    cout << "\nTeste de construtor de Dados com valores invalidos:\n";
    Date lastDayOfR (14, 33, 1991); // mês e dia inválidos
    cout << endl;

    return 0;
}

```

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista

Quarto Exemplo Usando Classes em C++

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista

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

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista

Quarto Exemplo Usando Classes em C++

```

//
* Aula 10 - Exemplo 4
* Arquivo dateCap10Ex4.h
* Autor: Miguel Campista
//
#ifdef DATE_H
#define DATE_H

#include <iostream>

using namespace std;

class Date {
public:
    Date (int m = 1, int d = 1, int y = 1999);
    void print () const;
    ~Date ();
private:
    int month, day, year;
    int checkDay (int) const;
};

#endif

```

Eliminar o construtor-padrão...

Date (int, int, int)

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista

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

```

Mantém igual...

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista

Quarto Exemplo Usando Classes em C++

```
int Date::checkDay (int testDay) const {
    static const int daysPerMonth [12] =
        {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 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 inválido (" << day << ") colocado em 1.\n";
}
```

Mantém igual...

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quarto 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...

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```
/*
 * 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);
        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 << "Destructor do objeto Employee: "
            << lastName << ", " << firstName << endl;
    }
}
```

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

Quarto 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 (9, 12, 1989);
    Employee manager ("Bob", "Blue", birth, hire);

    cout << endl;
    manager.print ();

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

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quarto Exemplo Usando Classes em C++

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

C:\Users\Miguel\Documents\UFRJ\... In constructor '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, 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, 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;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

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...

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções friend e Classes friend

- Função `friend` de uma classe
 - É definida fora do escopo da classe
 - Não é uma função-membro dessa classe
 - Ainda assim, tem o direito de acessar os membros não-`public` (e `public`) dessa classe
 - Funções independentes ou classes inteiras podem ser declaradas como amigas de uma outra classe
 - Em geral, é apropriada quando uma função-membro não pode ser usada por determinadas operações

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções friend e Classes friend

- Para declarar uma função como `friend` de uma classe:
 - Forneça o protótipo de função na definição de classe precedido pela palavra-chave `friend`
- Para declarar uma classe como amiga de uma classe:
 - Coloque a declaração da forma
 - `friend class ClassTwo;` na definição de classe `ClassOne`
 - Todas as funções-membro da classe `ClassTwo` são `friends` da classe `ClassOne`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções friend e Classes friend

- A amizade é concedida, não obtida
 - Para a classe `B` ser amiga da classe `A`, a classe `A` deve declarar explicitamente que a classe `B` é sua amiga
- A relação de amizade não é simétrica nem transitiva
 - Se a classe `A` for amiga da classe `B` e a classe `B` for amiga da classe `C`, não significa que a classe `B` é amiga da classe `A`, que a `C` é amiga da `B` ou que a `A` é amiga da `C`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções friend e Classes friend

- É possível especificar funções sobrecarregadas como `friends` de uma classe
 - Cada função sobrecarregada que se quer tornar `friend` deve ser declarada explicitamente como `friend` da classe
- Mesmo que os protótipos para funções `friend` apareçam na definição de classe...
 - As funções `friend` não são funções-membro

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções friend e Classes friend

- As especificações de acesso `private`, `protected` e `public` não são relevantes às declarações `friend`
 - Portanto, as declarações `friend` podem ser colocadas em qualquer lugar de uma definição de classe
 - Entretanto, o código se torna mais claro se todas as declarações de amizade forem colocadas em primeiro lugar no corpo da definição de classe e não inseridas depois de algum especificador de acesso

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quinto Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 8
 * Arquivo countCap10Ex8.h
 * Autor: Miguel Campista
 */
#ifndef COUNT_H
#define COUNT_H

#include <iostream>

using namespace std;

class Count {
    friend void setX (Count &, int);

public:
    Count ();
    void print () const;

private:
    int x;
};

#endif
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quinto Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 5
 * Arquivo countCap10Ex5.cpp
 * Autor: Miguel Campista
 */
#include "countCap10Ex5.h"

// A função setX pode alterar algum atributo private de classe Count
// Já que foi declarada como um método friend.
void setX (Count &c, int val) {
    c.x = val; // permitido pois setX Count é uma friend de Count
}

Count::Count () : x (0) {}

void Count::print () const {
    cout << x << endl;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quinto Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 5
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "countCap10Ex5.h"

int main() {
    Count counter;

    cout << "counter.x depois da instanciação: ";
    counter.print ();

    setX(counter, 8); // Dependo da função friend
    cout << "counter.x depois da chamada da função friend: ";
    counter.print ();

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quinto Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 5
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "countCap10Ex5.h"
```

```
int main() {
    Count counter;

    counter.x depois da instanciação: 0
    counter.x depois da chamada da função friend: 8
    Pressione qualquer tecla para continuar. - - -

    counter.print ();

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sexto Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 6
 * Arquivo countCap10Ex6.h
 * Autor: Miguel Campista
 */
#ifndef COUNT_H
#define COUNT_H

#include <iostream>

using namespace std;

class Count {
public:
    Count ();
    void print () const;
private:
    int x;
};

#endif
```

➡ Não tem mais o método friend!

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sexto Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 6
 * Arquivo countCap10Ex6.cpp
 * Autor: Miguel Campista
 */
#include "countCap10Ex6.h"

// A função setX não pode mais alterar atributo private de classe Count
// Já que não foi declarada como um método friend.
void setX (Count &c, int val) {
    c.x = val; // Erro!
}

Count::Count () : x (0) {}

void Count::print () const {
    cout << x << endl;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sexto Exemplo Usando Classes em C++

```
/*
 * Aula 10 - Exemplo 6
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "countCap10Ex6.h"

int main() {
    Count counter;

    cout << "counter.x depois da instanciação: ";
    counter.print ();

    setX(counter, 8); // Dependo da função friend
    cout << "counter.x depois da chamada da função friend: ";
    counter.print ();

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sexto Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 6
 * Arquivo_Principal.cpp
 * Autor: Miguel Campista
 */
#include "countCap10Ex6.h"

int main() {
    C:Users\Miguel\Documents\UFRJ\... In function 'void set0(Count&, int)':
20 C:Users\Miguel\Documents\UFRJ\... int Count: x is private
11 C:Users\Miguel\Documents\UFRJ\... when this context:
C:Users\Miguel\Documents\UFRJ\... [Build Error] exe: "" [countCap10Ex6.o] Error 1

    cout << "counter: x depois da chamada da funcao friends: ";
    counter.print ();

    return 0;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

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 do 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sétimo 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

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sétimo 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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sétimo Exemplo Usando Classes em C++

```

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

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

    return 0;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sétimo 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;
}

```

```

C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projeto\aula10-ex7.exe
this->x = 12
this->y = 12
Pressione qualquer tecla para continuar. . .

```

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 &obj;
Classe &refObj = obj;
refObj.x;

// Ponteiro
Classe obj;
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);

Oitavo 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 minute; // 0 - 59
    int second; // 0 - 59
};
#endif

```

Oitavo 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 iniciam 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 encadeamento
}

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

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

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

```

Oitavo 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 () const {
    cout << ((hour == 0 || hour == 12) ? 12 : hour & 12) << ":" <<
        << setfill('0') << setw(2) << minute << ":" << setw(2) <<
        << second << (hour < 12 ? " AM" : " PM");
}

```

Oitavo 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.setTime(10, 20, 20).setMinute(30).setSecond(22);

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

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

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Oitavo 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.setTime(10, 20, 20).setMinute(30).setSecond(22);

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

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

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Gerenciamento de Memória Dinâmica

- 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Gerenciamento de Memória Dinâmica

- 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Gerenciamento de Memória Dinâmica

- 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Gerenciamento de Memória Dinâmica

- Operador **delete**
 - Destroi 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

Linguagens de Programação – DEL-Poli/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);`

Linguagens de Programação – DEL-Poli/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 Dinâmico

- 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Gerenciamento de Memória Dinâmico

- 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Gerenciamento de Memória Dinâmico

- Utilizar `delete` em vez de `delete []` para arrays de objetos pode provocar erros de lógica em tempo de execução
 - Para que cada objeto no array receba uma chamada de destrutor, sempre exclua a memória alocada como array com o operador `delete []`
 - De modo semelhante, sempre exclua a memória alocada como um elemento individual com o operador `delete`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Membros de Classe `static`

- Membros de dados `static`
 - Apenas uma cópia de uma variável deve ser compartilhada por todos os objetos de uma classe
 - Representa informações no "nível da classe"
 - Uma propriedade da classe compartilhada por todas as instâncias, não uma propriedade de um objeto específico da classe
 - A declaração começa com a palavra-chave `static`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Membros de Classe static

- Membros de dados **static**
 - Ex.: Videogame com `Martian` [`martianCount`]
 - Todo `Martian` precisa conhecer a `martianCount`
 - `martianCount` deve ser um dado **static** em nível de classe
 - Todo `Martian` pode acessar `martianCount` como se fosse membro de dados desse `Martian`
 - Há somente uma cópia de `martianCount`
 - Embora pareçam variáveis globais, têm escopo de classe
 - Podem ser declarados `public`, `private` ou `protected`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Membros de Classe static

- Membros de dados **static**
 - Membros de dados **static** do tipo fundamental
 - São inicializados por padrão em 0
 - Se desejar um valor inicial diferente, um membro de dados **static** pode ser inicializado uma vez (e apenas uma única vez)
 - Membros de dados **static const**
 - Podem ser inicializados na sua declaração na definição da classe
 - Todos os outros membros de dados **static**
 - Devem ser definidos no escopo de arquivo, ou seja, fora do corpo da definição de classe

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Membros de Classe static

- Membros de dados **static** do tipo objeto
 - Não precisam ser inicializados porque o respectivo construtor-padrão é chamado

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Membros de Classe static

- Membros de dados **static**
 - Existem mesmo quando não há nenhum objeto da classe
 - Para acessar um membro de classe **public static** quando não existe **nenhum** objeto da classe
 - Prefixe o nome da classe e o operador binário de resolução de escopo (`::`) ao nome do membro de dados
 - » Ex.: `Martian::martianCount`
 - Podem ser acessadas também por meio de qualquer objeto dessa classe
 - Use o nome do objeto, o operador `.` e o nome do membro
 - Ex.: `myMartian.martianCount`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Membros de Classe static

- Função-membro **static**
 - É um serviço da classe, não um objeto específico da classe
- Palavra-chave **static** é aplicada a um item no escopo de arquivo
 - Esse item torna-se conhecido apenas nesse arquivo
 - Os membros **static** da classe precisam estar disponíveis em qualquer código-cliente que use a classe
 - De modo que não podemos declará-los **static** no arquivo `.cpp`
 - São declarados **static** apenas no arquivo `.h`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Membros de Classe static

- Membros de dados **static** e funções-membro **static** de uma classe
 - Existem e podem ser utilizados mesmo quando nenhum objeto dessa classe tiver sido instanciado
- É um erro de compilação incluir a palavra-chave **static** na definição de membros de dados **static** no escopo de arquivo
 - Uma variável só pode ser declarada **static** na definição da classe, já que membros de dados e as funções membro **static** podem ser usadas sem ter um objeto relacionado

```
static int Employee::count = 0;
```

Nono Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 9
 * Arquivo employeeCap10Ex9.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);
    ~Employee ();
    const char *getFirstName () const;
    const char *getLastName () const;
    // Função-membro static
    static int getCount ();
private:
    char *firstName;
    char *lastName;
    // Dados static
    static int count;
};
#endif

```

Nono Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 9
 * Arquivo employeeCap10Ex9.cpp
 * Autor: Miguel Campista
 */
#include "employeeCap10Ex9.h"

int Employee::count = 0;

// Define a função-membro static que retorna o número de
// objetos Employee instanciados (static declarado no .h)
int Employee::getCount () { return count; }

Employee::Employee (const char * const first, const char * const last) {
    firstName = new char [strlen (first) + 1];
    strcpy (firstName, first);

    lastName = new char [strlen (last) + 1];
    strcpy (lastName, last);

    count ++; // Construtor pode mudar membros static
}

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

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

delete [] firstName; // Libera memória
delete [] lastName; // Libera memória

count --;
}

```

Nono Exemplo Usando Classes em C++

```

const char *Employee::getFirstName () const {
    // const antes do tipo de retorno impede que o cliente modifique
    // dados private; o cliente pode copiar o string retornada antes de
    // o destrutor excluir o armazenamento para impedir um ponteiro
    // indefinido
    return firstName;
}

const char *Employee::getLastName () const {
    // const antes do tipo de retorno impede que o cliente modifique
    // dados private; o cliente pode copiar o string retornada antes de
    // o destrutor excluir o armazenamento para impedir um ponteiro
    // indefinido
    return lastName;
}

```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Nono Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 9
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "employeeCap10Ex9.h"

using namespace std;

int main () {
    cout << "Número de empregados antes da instanciação de qq objeto eh: "
        << Employee::getCount () << endl;

    // Utiliza new para criação de objetos
    Employee *e1Ptr = new Employee ("Susan", "Baker");
    Employee *e2Ptr = new Employee ("Robert", "Jones");

    cout << "Número de empregados depois da instanciação de objetos eh: "
        << e1Ptr->getCount () << endl;

    cout << "\n\nEmpregado 1: " << e1Ptr->getFirstName () << " "
        << e1Ptr->getLastName ()
        << "\n\nEmpregado 2: " << e2Ptr->getFirstName () << " "
        << e2Ptr->getLastName () << "\n\n";

    delete e1Ptr; // desaloca memória
    e1Ptr = 0; // desconecta o ponteiro do espaço de armazenamento livre
    delete e2Ptr; // desaloca memória
    e2Ptr = 0; // desconecta o ponteiro do espaço de armazenamento livre

    cout << "Número de empregados depois dos objetos serem deletados eh: "
        << Employee::getCount () << endl;

    return 0;
}

```

Nono Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 9
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "employeeCap10Ex9.h"

```

```

C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projeto\aula10-ex9.exe
Número de empregados antes da instanciação de qq objeto eh: 0
Construtor do objeto Employee para Susan Baker
Construtor do objeto Employee para Robert Jones
Número de empregados depois da instanciação de objetos eh: 2

Empregado 1: Susan Baker
Empregado 2: Robert Jones

Destrutor do objeto Employee para Susan Baker
Destrutor do objeto Employee para Robert Jones
Número de empregados depois dos objetos serem deletados eh: 0
Pressione qualquer tecla para continuar.

```

Nono Exemplo Usando Classes em C++

```

/*
 * Aula 10 - Exemplo 9
 * Arquivo Principal
 * Autor: Miguel Campista
 */
#include "employeeCap10Ex9.h"

using namespace std;

int main () {
    cout << "Número de empregados antes da instanciação de qq objeto eh: "
        << Employee::getCount () << endl;

    // Utiliza new para criação de objetos
    Employee *e1Ptr = new Employee ("Susan", "Baker");
    Employee *e2Ptr = new Employee ("Robert", "Jones");

    e1Ptr->setCount (1);

    cout << "Número de empregados depois da instanciação de objetos eh: "
        << e1Ptr->getCount () << endl;

    cout << "\n\nEmpregado 1: " << e1Ptr->getFirstName () << " "
        << e1Ptr->getLastName ()
        << "\n\nEmpregado 2: " << e2Ptr->getFirstName () << " "
        << e2Ptr->getLastName () << "\n\n";

    delete e1Ptr; // desaloca memória
    e1Ptr = 0; // desconecta o ponteiro do espaço de armazenamento livre
    delete e2Ptr; // desaloca memória
    e2Ptr = 0; // desconecta o ponteiro do espaço de armazenamento livre

    cout << "Número de empregados depois dos objetos serem deletados eh: "
        << Employee::getCount () << endl;

    return 0;
}

```

E se incluir a função membro setCount para incrementar o atributo count?



Nono Exemplo Usando Classes em C++

```

//
// Aula 10 - Exemplo 9
// Arquivo employeeCapitola.h
// Autor: Miguel Campista
//
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

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

using namespace std;

class Employee {
public:
    Employee(const char * const, const char * const):
        ~Employee():
        const char *getFirstName() const:
        const char *getLastName() const:
        void setCount():
        // Função membro static
        static int getCount():
private:
    char *firstName;
    char *lastName;
    // Dados static
    static int count;
};
#endif
    
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Nono Exemplo Usando Classes em C++

```

const char *Employee::getFirstName() const {
    // const antes do tipo de retorno impede que o cliente modifique
    // dados private; o cliente pode copiar a string retornada antes de
    // o destrutor excluir o armazenamento para impedir um ponteiro
    // indefinido
    return firstName;
}

const char *Employee::getLastName() const {
    // const antes do tipo de retorno impede que o cliente modifique
    // dados private; o cliente pode copiar a string retornada antes de
    // o destrutor excluir o armazenamento para impedir um ponteiro
    // indefinido
    return lastName;
}

void Employee::setCount() { count++; }
    
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Nono Exemplo Usando Classes em C++

```

C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projetos\aula10-ex9.exe
Numero de empregados antes da instanciação de qq objeto eh: 0
Construtor do objeto Employee para Susan Baker
Construtor do objeto Employee para Robert Jones
Numero de empregados depois da instanciação de objetos eh: 4
Empregado 1: Susan Baker
Empregado 2: Robert Jones
Destruir do objeto Employee para Susan Baker
Destruir do objeto Employee para Robert Jones
Numero de empregados depois dos objetos serem deletados eh: 2
Pressione qualquer tecla para continuar. . .
1
void Employee::setCount() { count++; }
    
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Nono Exemplo Usando Classes em C++

```

//
// Aula 10 - Exemplo 9
// Arquivo Principal
// Autor: Miguel Campista
//
#include "employeeCapitola.h"

using namespace std;

int main() {
    cout << "Numero de empregados antes da instanciação de objetos eh: "
    << Employee::getCount() << endl;

    // Utiliza new para criação de objetos
    Employee *e1Ptr = new Employee("Susan", "Baker");
    Employee *e2Ptr = new Employee("Robert", "Jones");

    Employee::setCount(); Employee::setCount();

    cout << "Numero de empregados depois da instanciação de objetos eh: "
    << Employee::getCount() << endl;

    cout << "\nEmpregado 1: " << e1Ptr->getFirstName() << " "
    << e1Ptr->getLastName()
    << "\nEmpregado 2: " << e2Ptr->getFirstName() << " "
    << e2Ptr->getLastName() << "\n\n";

    delete e1Ptr; // desaloja memória
    e1Ptr = 0; // desconecta o ponteiro do espaço de armazenamento livre
    delete e2Ptr; // desaloja memória
    e2Ptr = 0; // desconecta o ponteiro do espaço de armazenamento livre

    cout << "Numero de empregados depois dos objetos serem deletados eh: "
    << Employee::getCount() << endl;

    return 0;
}
    
```

Qual deve ser a modificação na definição da classe Employee ?



Membros de Classe static

- Declaração de uma função-membro static
 - Não pode acessar membros de dados não-static ou funções-membro não-static da classe

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Membros de Classe static

- Utilizar o ponteiro this em uma função-membro static é um erro de compilação
 - Uma função static opera independente da existência de um objeto associado
- Declarar uma função-membro static como const é um erro de compilação
 - O qualificador const indica que uma função não pode modificar o conteúdo do objeto em que opera, mas as funções-membro static existem e operam independentemente de qualquer objeto da classe

static int getCount() const;

Membros de Classe `static`

- Depois de excluir a memória alocada dinamicamente, configure como 0 o ponteiro que referenciava essa memória
 - Isso desconecta o ponteiro do espaço anteriormente alocado no armazenamento livre
 - Esse espaço na memória ainda poderia conter informações, apesar de ter sido excluído
 - Configurando o ponteiro como 0, o programa perde qualquer acesso a esse espaço de armazenamento livre, o qual, de fato, já poderia ter sido realocado para um propósito diferente

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Abstração de Dados e Ocultação de Informações

- Ocultação de informações
 - Uma classe, em geral, oculta dos clientes detalhes da sua implementação
- Abstração de dados
 - O cliente se preocupa com **qual** funcionalidade a classe oferece, e não com **como** essa funcionalidade é implementada
 - Ex.: Um cliente de uma classe pilha não precisa conhecer a implementação da pilha (lista encadeada ou não)
 - Os programadores não devem escrever códigos que dependam de detalhes de implementação da classe

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Abstração de Dados e Ocultação de Informações

- Importância dos dados
 - É elevada no C++ e na comunidade orientada a objetos
 - As principais atividades da programação orientada a objetos no C++
 - Criação de tipos (isto é, classes)
 - Expressão das interações entre os objetos desses tipos
 - Tipos de dados abstratos (ADTs)
 - Melhoram o processo de desenvolvimento dos programas

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Abstração de Dados e Ocultação de Informações

- Tipos abstratos de dados (ADTs)
 - São formas de representar noções do mundo real em um nível até certo ponto satisfatório, em um sistema de computador
 - Tipos como `int`, `double`, `char` são todos ADTs
 - Por exemplo, `int` é uma representação abstrata de um inteiro
 - Capturam duas noções:
 - Representação de dados
 - Operações que podem ser executadas nos dados
 - As classes C++ implementam ADTs e seus serviços

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Abstração de Dados e Ocultação de Informações

- O programador é capaz de criar novos tipos pelo mecanismo de classe
 - Esses novos tipos podem ser projetados para ser utilizados tão convenientemente quanto os tipos predefinidos
 - Portanto, o C++ é uma linguagem extensível
 - Entretanto, embora a linguagem seja fácil de estender com esses novos tipos
 - Linguagem básica em si não pode ser alterada

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo: Tipo de Dados Abstrato `vector`

- Várias operações de array não estão predefinidas no C++
 - Por exemplo, verificação de intervalo de subscrito
- Os programadores podem desenvolver um ADT de array como uma classe, que é preferível aos arrays "brutos"
 - A classe de array pode fornecer muitas capacidades novas e úteis
- Template da classe `vector` na C++ *Standard Template Library*

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo: Tipo de Dados Abstrato string

- Não existe nenhum tipo de dados string entre os tipos de dados predefinidos da linguagem C++
 - C++ é uma linguagem intencionalmente esparsa
 - Oferece aos programadores apenas capacidades elementares necessárias para a criação de uma ampla gama de sistemas
 - Projetada para minimizar o fardo do desempenho
 - Projetada para incluir mecanismos para a criação e implementação de tipo de dados abstrato string por meio de classes
 - Classe string da C++ Standard Library

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo: Tipo de Dados Abstrato queue

- Itens na ordem primeiro a entrar, primeiro a sair (FIFO)
- Oculta a representação interna de dados que de alguma forma monitora os itens que atualmente estão aguardando na fila
- Bom exemplo de tipo de dados abstrato
 - Os clientes invocam uma operação de *enfileiramento* para colocar uma coisa por vez na fila
 - Os clientes invocam uma operação de *desenfileiramento* para recuperar essas coisas individualmente por demanda
- Classe queue da C++ Standard Library

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Classes Contêineres e Iteradores

- Classes contêineres (também chamadas de classes de coleção)
 - São classes projetadas para abrigar coleções de objetos
 - Em geral, fornecem serviços como inserção, exclusão, pesquisa, classificação e teste de um item para determinar se ele é um membro da coleção
 - Ex.:
 - Arrays
 - Pilhas
 - Filas
 - Árvores
 - Listas vinculadas

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Classes Contêineres e Iteradores

- Objetos iteradores — ou, resumidamente, iteradores
 - Em geral, estão associados com as classes contêineres
 - Objeto que "percorre" uma coleção, retornando o item seguinte (ou executando alguma ação no item seguinte)
 - Uma classe contêiner pode ter simultaneamente vários iteradores em operação
 - Cada iterador mantém as respectivas informações sobre sua "posição"

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo 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
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo 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;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo 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
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo 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;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo 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;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo 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;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo 1

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



Exemplo 1

```
/*
 * Aula 10 - Exemplo 11
 * Programa polygonCap10Ex11.h
 * Autor: Miguel Campista
 */
#ifdef POLYGON_H
#define POLYGON_H
#include <iostream>

using namespace std;

class Point;

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

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

private:
    Point * points;
};

#endif
```

Exemplo 1

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

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

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

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

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

Exemplo 1

```
/*
 * Aula 10 - Exemplo 11
 * Programa pointCap10Ex11.h
 * Autor: Miguel Campista
 */
#ifdef POINT_H
#define POINT_H
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <cmath>
#include <iomanip>

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
```

Exemplo 1

```
/*
 * 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 << " = " << 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;
}
```

Exemplo 1

```
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;
    }
}
```

Exemplo 1

```
/*  
 * Aula 10 - Exemplo 11  
 * Programa Principal  
 * Autor: Miguel Campista  
 */  
#include "polygonCap10Ex11.h"  
  
using namespace std;  
  
int main ()  
{  
    Polygon p (4);  
  
    p.getPoints ();  
    p.getSides ();  
  
    return 0;  
}
```

Exemplo 1

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

```
C:\Windows\system32\cmd.exe  
C:\Users\Miguel\Documents\UFRJ\disciplinas\Linguagens\projetos>main10-ex11.exe  
No construtor da classe poligono...  
Ponto 0: 10  
Ponto 1: 2  
Ponto 2: 2  
Ponto 3: 7  
Ponto 4: 11  
Ponto 1: 7  
Ponto 2: 10  
Ponto 3: 6  
No destrutor da classe poligono...  
10 10  
2 2  
2 10  
7 6  
lado 0: 12,04  
lado 1: 17,06  
lado 2: 13,27  
lado 3: 9,43  
Por favor, qualquer tecla para continuar. . .  
No destrutor da classe poligono...  
No destrutor da classe poligono...  
C:\Users\Miguel\Documents\UFRJ\disciplinas\Linguagens\projetos>
```

Leitura Recomendada

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