

# 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++ - Arrays e Ponteiros

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Arrays

- Estruturas de dados que contêm itens de dados relacionados do mesmo tipo
- Tamanho constante desde o momento em que são criados
  - Entidades "estáticas"
- Arrays de caracteres podem também representar strings
- Arrays podem ser representados como em C
  - Entretanto, podem também ser objetos vetores como implementado na STL (*Standard Template Library*)
    - Os vetores são mais seguros e versáteis

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Arrays

- Grupo consecutivo de posições da memória
  - Todas são do mesmo tipo
- Índice
  - Número da posição usado para indicar a localização/elemento específico
  - Deve ser um inteiro positivo ou uma expressão do tipo inteiro
  - O primeiro elemento tem índice zero
    - Ex.: Suponha  $a = 5$  e  $b = 6$
    - $c[a + b] += 2$ ;
      - » Adiciona 2 ao elemento do array  $c[11]$

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 3
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

int main() {
    const int numElems = 10;
    int n [numElems]; // Array n de 10 inteiros

    // Cálculo dos elementos do array
    for (int i = 0; i < numElems; i++)
        n [i] = 2 + 2 * i;

    cout << "Elemento" << setw(13) << "valor" << endl;

    // Saída dos elementos do array
    for (int j = 0; j < 10; j++)
        cout << setw(7) << j << setw(13) << n [j] << endl;

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 3
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

int main() {
    const int numElems = 10;
    int n [numElems]; // Array n de 10 inteiros

    // Cálculo dos elementos do array
    for (int i = 0; i < numElems; i++)
        n [i] = 2 + 2 * i;

    cout << "Elemento" << setw(13) << "valor" << endl;

    // Saída dos elementos do array
    for (int j = 0; j < 10; j++)
        cout << setw(7) << j << setw(13) << n [j] << endl;

    return 0;
}
```

Declaração do número de elementos do array utilizando uma variável const

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo Usando Array em C++

```

/*
 * Aula 7 - Exemplo 3
 * Autor: Miguel Campista
 */
shell>g++ exemplo.cpp -o ex3
shell>$ ./ex3
Elemento      valor
0              2
1              4
2              6
3              8
4             10
5             12
6             14
7             16
8             18
9             20
shell>$

for (int j = 0; j < 10; j++)
    cout << setw(7) << j << setw(13) << n [j] << endl;

return 0;
}

```

## Array

### • Variáveis constantes

- Não atribuir um valor a uma variável constante quando ela é declarada é um erro de compilação

`const int x;` ❌ Erro!

- Atribuir um valor a uma variável constante em uma instrução executável é um erro de compilação

`const int x = 1;`  
`x = 2;` ❌ Erro!

## Passagem de Array para Função

### • Parâmetros de array const

- Qualificador const
- Evita que valores do array sejam alterados no chamador por códigos na função chamada
- Os elementos no array são constantes na função
- Permite que o programador evite alterações acidentais nos dados



Como os arrays são passados por referência, é comum utilizar o qualificador `const` para evitar alterações

## Segundo Exemplo Usando Array em C++

```

/*
 * Aula 7 - Exemplo 10
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void tryToModify(const int []):

int main() {
    int a[] = {10, 20, 30};

    tryToModify(a);

    cout << a[0] << ' ' << a[1] << ' ' << a[2] << endl;

    return 0;
}

void tryToModify(const int a[]) {
    a[0] /= 2;
    a[1] /= 2;
    a[2] /= 2;
}

```

## Segundo Exemplo Usando Array em C++

```

/*
 * Aula 7 - Exemplo 10
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void tryToModify(const int []):

int main() {
    int a[] = {10, 20, 30};

    tryToModify(a);

    cout << a[0] << ' ' << a[1] << ' ' << a[2] << endl;

    return 0;
}

void tryToModify(const int a[]) {
    a[0] /= 2;
    a[1] /= 2;
    a[2] /= 2;
}

```

Uso do const evita que a função altere o array

O array só é const dentro da função

O array não pode ser modificado dentro do corpo da função

## Segundo Exemplo Usando Array em C++

```

/*
 * Aula 7 - Exemplo 10
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void tryToModify(const int []):

int main() {
    int a[] = {10, 20, 30};

    tryToModify(a);

    cout << a[0] << ' ' << a[1] << ' ' << a[2] << endl;

    return 0;
}

void tryToModify(const int a[]) {
    a[0] /= 2;
    a[1] /= 2;
    a[2] /= 2;
}

```

```

shell>$ g++ exemplo.cpp -o ex10
Erro!
shell>$

```

## Array

- Arrays locais **static** e arrays locais automáticos
  - Uma variável local **static** em uma função
    - Existe durante a execução do programa
    - Mas é visível apenas no corpo da função
  - Um array local **static**
    - Existe durante a execução do programa
    - É inicializado quando sua declaração é encontrada pela primeira vez
      - Todos os elementos são inicializados em zero, se não forem inicializados explicitamente
        - » Isso não ocorre com os arrays locais automáticos

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 8
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

// Protótipo de funções
void staticArrayInit();
void automaticArrayInit();

int main() {
    cout << " ** Primeira chamada para cada funcao:\n\n";
    staticArrayInit();
    cout << endl;
    automaticArrayInit();

    cout << "\n\n ** Segunda chamada para cada funcao:\n\n";
    staticArrayInit();
    cout << endl;
    automaticArrayInit();
    cout << "\n" << endl;

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Array em C++

```
void staticArrayInit() {
    // Inicialize elementos como 0 na primeira vez que é chamada
    static int array1 [3];

    // Saída do array
    for (int i = 0; i < 3; i++)
        cout << "array1[" << i << "] = " << array1[i] << " ";

    cout << "\nValores no array estatico\n";

    for (int i = 0; i < 3; i++)
        cout << "array1[" << i << "] = " << (array1[i] + 5) << " ";

    cout << "\nValores no array estatico ao sair da funcao\n";
}

void automaticArrayInit() {
    // Inicialize elementos toda vez que a função é chamada
    int array2 [3] = {1, 2, 3};

    for (int i = 0; i < 3; i++)
        cout << "array2[" << i << "] = " << array2[i] << " ";

    cout << "\nValores atribuidos ao array automatico\n";

    for (int i = 0; i < 3; i++)
        cout << "array2[" << i << "] = " << (array2[i] + 5) << " ";

    cout << "\nValores no array automatico ao sair da funcao\n";
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Array em C++

```
void staticArrayInit() {
    // Inicialize elementos como 0 na primeira vez que é chamada
    static int array1 [3];

    // Saída do array
    for (int i = 0; i < 3; i++)
        cout << "array1[" << i << "] = " << array1[i] << " ";

    cout << "\nValores no array estatico\n";

    for (int i = 0; i < 3; i++)
        cout << "array1[" << i << "] = " << (array1[i] + 5) << " ";

    cout << "\nValores no array estatico ao sair da funcao\n";
}

void automaticArrayInit() {
    // Inicialize elementos toda vez que a função é chamada
    int array2 [3] = {1, 2, 3};

    for (int i = 0; i < 3; i++)
        cout << "array2[" << i << "] = " << array2[i] << " ";

    cout << "\nValores atribuidos ao array automatico\n";

    for (int i = 0; i < 3; i++)
        cout << "array2[" << i << "] = " << (array2[i] + 5) << " ";

    cout << "\nValores no array automatico ao sair da funcao\n";
}
```

Cria um array static

Cria um array automático

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Array em C++

```
void staticArrayInit() {
    // Inicialize elementos como 0 na primeira vez que é chamada
    static int array1 [3];

    // Saída do array
    for (int i = 0; i < 3; i++)
        cout << "array1[" << i << "] = " << array1[i] << " ";

    cout << "\nValores no array estatico\n";

    for (int i = 0; i < 3; i++)
        cout << "array1[" << i << "] = " << (array1[i] + 5) << " ";

    cout << "\nValores no array estatico ao sair da funcao\n";
}

void automaticArrayInit() {
    // Inicialize elementos toda vez que a função é chamada
    int array2 [3] = {1, 2, 3};

    for (int i = 0; i < 3; i++)
        cout << "array2[" << i << "] = " << array2[i] << " ";

    cout << "\nValores atribuidos ao array automatico\n";

    for (int i = 0; i < 3; i++)
        cout << "array2[" << i << "] = " << (array2[i] + 5) << " ";

    cout << "\nValores no array automatico ao sair da funcao\n";
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Estudo de Caso: Classe GradeBook

- Classe **GradeBook**
  - Representa um livro que armazena e analisa notas
  - Agora pode armazenar notas em um array
- Membros de dados **static**
  - Variáveis das quais os objetos de uma classe não têm uma cópia separada
    - Uma única cópia é compartilhada por todos os objetos da classe
  - Podem ser acessados mesmo sem objetos da classe
    - Nome da classe seguido do operador binário de resolução de escopo e o nome dos membros de dados **static**

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 11
 * Arquivo: GradeBookCap7Ex11.h
 * Autor: Miguel Campista
 */
#include <string>
#include <iomanip>

using namespace std;

// Definição da classe GradeBook
class GradeBook {

public:
    // Constante - Número de alunos que fizeram o teste
    const static int students = 10;
    // Construtor inicializa courseName com a string-argumento
    GradeBook(string, const int {});
    // Função que modifica o nome do curso
    void setCourseName(string);
    // Função que obtém o nome do curso
    string getCourseName();
    // Função para dar entrada nos conceitos dos alunos
    double getAverage();
    // Função para exibir os conceitos
    void displayGrades();
    // Função para retornar a menor nota dos alunos
    int getMinimum();
    // Função para retornar a maior nota dos alunos
    int getMaximum();
};
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 11
 * Arquivo: GradeBookCap7Ex11.h
 * Autor: Miguel Campista
 */
#include <string>
#include <iomanip>

using namespace std;

// Definição da classe GradeBook
class GradeBook {

public:
    // Constante - Número de alunos que fizeram o teste
    const static int students = 10;
    // Construtor inicializa courseName com a string-argumento
    GradeBook(string, const int {});
    // Função que modifica o nome do curso
    void setCourseName(string);
    // Função que obtém o nome do curso
    string getCourseName();
    // Função para dar entrada nos conceitos dos alunos
    double getAverage();
    // Função para exibir os conceitos
    void displayGrades();
    // Função para retornar a menor nota dos alunos
    int getMinimum();
    // Função para retornar a maior nota dos alunos
    int getMaximum();
};
```

students é uma variável static da classe

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
    // Função para realizar funções nas notas
    void processGrades();
    // Função para exibir a distribuição de notas de turma
    void displayBarChart();
    void displayMessage();

private:
    string courseName;
    int grades [students];
};
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
    // Função para realizar funções nas notas
    void processGrades();
    // Função para exibir a distribuição de notas de turma
    void displayBarChart();
    void displayMessage();

private:
    string courseName;
    int grades [students];
};
```

array grades para armazenar as notas

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 11
 * Arquivo: GradeBookCap7Ex11.cpp
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>
#include "GradeBookCap7Ex11.h"

// Construtor inicializa courseName com a string-argumento
GradeBook::GradeBook(string name, const int gradesArray[]) {
    setCourseName(name); // Chama a função set para inicialização

    for (int i = 0; i < students; i++)
        grades[i] = gradesArray[i];
}

// Função para retornar a menor nota dos alunos
int GradeBook::getMinimum() {
    int minimum = 100;
    for (int i = 0; i < students; i++) {
        if (grades[i] < minimum)
            minimum = grades[i];
    }
    return minimum;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 11
 * Arquivo: GradeBookCap7Ex11.cpp
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>
#include "GradeBookCap7Ex11.h"

// Construtor inicializa courseName com a string-argumento
GradeBook::GradeBook(string name, const int gradesArray[]) {
    setCourseName(name); // Chama a função set para inicialização

    for (int i = 0; i < students; i++)
        grades[i] = gradesArray[i];
}

// Função para retornar a menor nota dos alunos
int GradeBook::getMinimum() {
    int minimum = 100;
    for (int i = 0; i < students; i++) {
        if (grades[i] < minimum)
            minimum = grades[i];
    }
    return minimum;
}
```

Copia elementos de gradesArray para o atributo grades

Loop em grades para encontrar a nota mínima

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
// Função para retornar a menor nota dos alunos
int GradeBook::getMaximum() {
    int maximum = 0;
    for (int i = 0; i < students; i++) {
        if (grades[i] > maximum)
            maximum = grades[i];
    }
    return maximum;
}

// Função que calcula a média das notas da turma
double GradeBook::getAverage() {
    int total = 0;

    for (int i = 0; i < students; i++)
        total += grades[i];

    return static_cast<double>(total)/students;
}

// Função que configura o nome do curso
void GradeBook::setCourseName(string name) {
    if (name.length() <= 25) {
        courseName = name;
    } else {
        courseName = name.substr(0, 25);
        cout << "Warning: Nome \'" << name << "\'
        << "excede o limite máximo de 25 caracteres..." << endl <<
        "Nome limitado aos primeiros 25 caracteres: " << courseName <<
        endl;
    }
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
// Função para retornar a menor nota dos alunos
int GradeBook::getMaximum() {
    int maximum = 0;
    for (int i = 0; i < students; i++) {
        if (grades[i] > maximum)
            maximum = grades[i];
    }
    return maximum;
}

// Função que calcula a média das notas da turma
double GradeBook::getAverage() {
    int total = 0;

    for (int i = 0; i < students; i++)
        total += grades[i];

    return static_cast<double>(total)/students;
}

// Função que configura o nome do curso
void GradeBook::setCourseName(string name) {
    if (name.length() <= 25) {
        courseName = name;
    } else {
        courseName = name.substr(0, 25);
        cout << "Warning: Nome \'" << name << "\'
        << "excede o limite máximo de 25 caracteres..." << endl <<
        "Nome limitado aos primeiros 25 caracteres: " << courseName <<
        endl;
    }
}
```

Loop em grades para o encontrar a nota máxima

Loop para a soma e posterior divisão

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
// Função que retorna o nome do curso
string GradeBook::getCourseName() {
    return courseName;
}

void GradeBook::displayMessage() {
    cout << "Bem-vindo ao seu primeiro programa com classes em "
    << getCourseName() << "!" << endl;
}

// Função para realizar funções nas notas
void GradeBook::processGrades() {
    displayGrades();

    cout << "\nMédia da turma eh: " << setprecision(2) << fixed
    << getAverage() << endl;
    cout << "Menor nota foi: " << getMinimum()
    << "\nMaior nota foi: " << getMaximum() << endl;
    cout << endl;
    displayBarChart();
}

// Função para exibir os conceitos
void GradeBook::displayGrades() {
    for (int i = 0; i < students; i++) {
        cout << "Student " << setw(2) << i + 1
        << ": " << setw(3) << grades[i] << endl;
    }
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
// Função para exibir a distribuição de notas da turma
void GradeBook::displayBarChart() {
    cout << "Distribuição de notas: " << endl;

    const int frequencySize = 11;
    int frequency [frequencySize] = {};

    for (int i = 0; i < students; i++)
        frequency [grades[i]/10]++;

    for (int count = 0; count < frequencySize; count++) {
        if (count == 0)
            cout << " 0-9:";
        else if (count == 10)
            cout << " 100:";
        else
            cout << count * 10 << "-" << (count * 10) + 9 << ":";

        for (int stars = 0; stars < frequency [count]; stars++)
            cout << " * ";

        cout << endl;
    }
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
// Função para exibir a distribuição de notas da turma
void GradeBook::displayBarChart() {
    cout << "Distribuição de notas: " << endl;

    const int frequencySize = 11;
    int frequency [frequencySize] = {};

    for (int i = 0; i < students; i++)
        frequency [grades[i]/10]++;

    for (int count = 0; count < frequencySize; count++) {
        if (count == 0)
            cout << " 0-9:";
        else if (count == 10)
            cout << " 100:";
        else
            cout << count * 10 << "-" << (count * 10) + 9 << ":";

        for (int stars = 0; stars < frequency [count]; stars++)
            cout << " * ";

        cout << endl;
    }
}
```

Loop em grades para encontrar a frequência

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 11
 * Arquivo principal
 * Autor: Miguel Campista
 */

#include <iostream>
#include <string>
#include "GradeBookCap7Ex11.h" // Inclui a definição da classe

using namespace std;

int main() {
    // Array de notas dos alunos
    int gradesArray [GradeBook::students] = {87, 68, 94, 100, 83, 78, 85, 91, 76, 87};

    // Cria dois objetos GradeBook
    GradeBook gradeBook("Linguagens de Programacao", gradesArray);

    // Exibe o valor inicial de courseName
    gradeBook.displayMessage();
    gradeBook.processGrades();

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Array em C++

```

/*
 * Aula 7 -- Exemplo 11
 * Arquivo principal
 * Autor: Miguel Campista
 */

#include <iostream>
#include <string>
#include "GradeBookCap7Ex11.h" // Inclui a definição da classe

using namespace std;

int main() {
    // Array de notas
    int gradesArray [GradeBook::tstudent] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87};

    // Cria dois objetos GradeBook
    GradeBook gradeBook("Linguagem de Programacao", gradesArray);

    // Exibe o valor inicial de courseName
    gradeBook.displayMessage();
    gradeBook.processGrades();

    return 0;
}

```

Usa students declarado como static na classe

## Quarto Exemplo Usando Array em C++

```

/*
 * Aula 7 -- Exemplo 11
 * Arquivo principal
 * Autor: Miguel Campista
 */

#include <iostream>
#include <string>
#include "GradeBookCap7Ex11.h" // Inclui a definição da classe

using namespace std;

int main() {
    // Array de notas
    int gradesArray [GradeBook::tstudent] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87};

    // Cria dois objetos GradeBook
    GradeBook gradeBook("Linguagem de Programacao", gradesArray);

    // Exibe o valor inicial de courseName
    gradeBook.displayMessage();
    gradeBook.processGrades();

    return 0;
}

```

## Template vector da C++ Standard Library

- Arrays baseados em ponteiro ao estilo do C
  - Apresentam alta probabilidade de erros e várias deficiências
    - O C++ não verifica se os subscritos são colocados fora do intervalo do array
    - Dois arrays não podem ser comparados de modo significativo com operadores de igualdade ou relacionais
    - Um array não pode ser atribuído a outro que esteja usando os operadores de atribuição

```
int a[10], b[10];
if (a == b) {
    ...
}
```

X Erro!

```
int a[10], b[10];
int b[10] = a; X Erro!
```

## Template vector da C++ Standard Library

- Template de classe **vector**
  - Disponível para construção de aplicativos com o C++
  - Pode ser definido para armazenar qualquer tipo de dados
    - Especificado entre colchetes angulares em `vector<type>`
    - Todos os elementos em um `vector` são configurados em 0 por padrão
  - A função-membro `size` obtém o tamanho do array
    - Número de elementos como um valor do tipo `size_t` (unsigned integer)
  - Os objetos `vector` podem ser comparados por meio dos operadores de igualdade e relacionais
  - O operador de atribuição pode ser usado em `vectors`

## Template vector da C++ Standard Library

- Elementos `vector` podem ser obtidos como um *lvalue* (valor à esquerda) não modificável ou um *lvalue* modificável
  - *rvalue* não modificável
    - Expressão que identifica um objeto na memória, mas não pode ser usada para modificar esse objeto
      - Ex.: `cout << array [2]; // Operador é o []`
  - *lvalue* modificável
    - Expressão que identifica um objeto na memória, mas pode ser usada para modificar o objeto
      - Ex.: `array [2] = 5; // Operador é o []`

## Template vector da C++ Standard Library

- Função `at` de `vector`
  - Oferece acesso a elementos individuais
  - Verifica limites
    - Lança uma exceção quando um índice especificado é inválido
    - O acesso com colchetes não executa a verificação de limites

## Quinto Exemplo Usando Array em C++

```

/*
 * Aula 7 - Exemplo 16
 * Autor: Miguel Campista
 */
#include <iostream>
#include <omanip>
#include <vector>

using namespace std;

void outputVector(const vector<int> v) // Exibe o vetor
void inputVector(vector<int> v) // Exibe o vetor

int main() {
    vector<int> integer1(7); // vetor de inteiros de 7 elementos
    vector<int> integer2(10); // vetor de inteiros de 10 elementos

    // Imprime o tamanho de integer1 e conteúdo
    cout << "Tamanho do vetor integer1 eh " << integer1.size()
    << "\nvetor depois da inicializacao" << endl;
    outputVector(integer1);

    // Imprime o tamanho de integer2 e conteúdo
    cout << "Tamanho do vetor integer2 eh " << integer2.size()
    << "\nvetor depois da inicializacao" << endl;
    outputVector(integer2);

    // Insere e imprime integer1 e integer2
    cout << "Informe 17 inteiros" << endl;
    inputVector(integer1);
    inputVector(integer2);
}
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```

/*
 * Aula 7 - Exemplo 16
 * Autor: Miguel Campista
 */
#include <iostream>
#include <omanip>
#include <vector>

using namespace std;

void outputVector(const vector<int> v) // Exibe o vetor
void inputVector(vector<int> v) // Exibe o vetor

int main() {
    vector<int> integer1(7); // vetor de inteiros de 7 elementos
    vector<int> integer2(10); // vetor de inteiros de 10 elementos

    // Imprime o tamanho de integer1 e conteúdo
    cout << "Tamanho do vetor integer1 eh " << integer1.size()
    << "\nvetor depois da inicializacao" << endl;
    outputVector(integer1);

    // Imprime o tamanho de integer2 e conteúdo
    cout << "Tamanho do vetor integer2 eh " << integer2.size()
    << "\nvetor depois da inicializacao" << endl;
    outputVector(integer2);

    // Insere e imprime integer1 e integer2
    cout << "Informe 17 inteiros" << endl;
    inputVector(integer1);
    inputVector(integer2);
}
    
```

Uso do const evita que o array recebido seja alterado

Vectors que armazenam ints

POO para Redes de Computadores - COPPE-PEE/UFRJ Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```

/*
 * Aula 7 - Exemplo 16
 * Autor: Miguel Campista
 */
#include <iostream>
#include <omanip>
#include <vector>

using namespace std;

void outputVector(const vector<int> v) // Exibe o vetor
void inputVector(vector<int> v) // Exibe o vetor

int main() {
    vector<int> integer1(7); // vetor de inteiros de 7 elementos
    vector<int> integer2(10); // vetor de inteiros de 10 elementos

    // Imprime o tamanho de integer1 e conteúdo
    cout << "Tamanho do vetor integer1 eh " << integer1.size()
    << "\nvetor depois da inicializacao" << endl;
    outputVector(integer1);

    // Imprime o tamanho de integer2 e conteúdo
    cout << "Tamanho do vetor integer2 eh " << integer2.size()
    << "\nvetor depois da inicializacao" << endl;
    outputVector(integer2);

    // Insere e imprime integer1 e integer2
    cout << "Informe 17 inteiros" << endl;
    inputVector(integer1);
    inputVector(integer2);
}
    
```

Função size retorna o tamanho dos vectors

POO para Redes de Computadores - COPPE-PEE/UFRJ Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```

// Use operador de diferença (!=) com objetos vector
cout << "Analisando integer1 != integer2" << endl;

if (integer1 != integer2)
    cout << "integer1 e integer2 nao sao iguais" << endl;

// Criando vetor integer3 usando integer1 como um
// inicializador: imprime tamanho e conteúdo
vector<int> integer3(integer1); // copia construtor

cout << "Tamanho do vetor integer3 eh " << integer3.size()
<< "\nvetor depois da inicializacao" << endl;
outputVector(integer3);

// Use operador de atribuição (=) com objetos vector
cout << "Analisando integer2 aos integer1" << endl;
integer1 = integer2; // integer1 é maior que integer2

cout << "integer1" << endl;
outputVector(integer1);
cout << "integer2" << endl;
outputVector(integer2);
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```

cout << "InDepois de inputVector, os vetores contem:\n"
<< "integer1:" << endl;
outputVector(integer1);
cout << "integer2:" << endl;
outputVector(integer2);

// Use operador de diferença (!=) com objetos vector
cout << "Analisando integer1 != integer2" << endl;

if (integer1 != integer2)
    cout << "integer1 e integer2 nao sao iguais" << endl;

// Criando vetor integer3 usando integer1 como um
// inicializador: imprime tamanho e conteúdo
vector<int> integer3(integer1); // copia construtor

cout << "Tamanho do vetor integer3 eh " << integer3.size()
<< "\nvetor depois da inicializacao" << endl;
outputVector(integer3);

// Use operador de atribuição (=) com objetos vector
cout << "Analisando integer2 aos integer1" << endl;
integer1 = integer2; // integer1 é maior que integer2

cout << "integer1:" << endl;
outputVector(integer1);
cout << "integer2:" << endl;
outputVector(integer2);
    
```

Comparação dos vectors com "!="

Inicialização de um vector com outro

POO para Redes de Computadores - COPPE-PEE/UFRJ Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```

// Use operador de diferença (!=) com objetos vector
cout << "Analisando integer1 != integer2" << endl;

if (integer1 != integer2)
    cout << "integer1 e integer2 nao sao iguais" << endl;

// Criando vetor integer3 usando integer1 como um
// inicializador: imprime tamanho e conteúdo
vector<int> integer3(integer1); // copia construtor

cout << "Tamanho do vetor integer3 eh " << integer3.size()
<< "\nvetor depois da inicializacao" << endl;
outputVector(integer3);

// Use operador de atribuição (=) com objetos vector
cout << "Analisando integer2 aos integer1" << endl;
integer1 = integer2; // integer1 é maior que integer2

cout << "integer1:" << endl;
outputVector(integer1);
cout << "integer2:" << endl;
outputVector(integer2);
    
```

Atribuição dos valores de um vector para outro

POO para Redes de Computadores - COPPE-PEE/UFRJ Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```
// Use operador de equação (==) com objetos vetor
cout << "\nValidando: integers1 == integers2" << endl;

if (integers1 == integers2)
    cout << "integers1 e integers2 sao iguais" << endl;

// Use colchetes para criar evalue
cout << "\nintegers1[5] is " << integers1[5];

// Use colchetes para criar evalue
cout << "\nInserindo 1000 ao integers1[5]" << endl;
integers1[5] = 1000;
cout << "integers1:" << endl;
outputVector( integers1 );

system("PAUSE");
// Tentativa de usar indice fora do intervalo
cout << "\nTentativa de atribuir 1000 ao integers.at(15)" << endl;
integers.at(15) = 1000; // ERROR: fora do intervalo

return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```
// Use operador de equação (==) com objetos vetor
cout << "\nValidando: integers1 == integers2" << endl;

if (integers1 == integers2)
    cout << "integers1 e integers2 sao iguais" << endl;

cout << "\nintegers1[5] is " << integers1[5];

// Use colchetes para criar evalue
cout << "\nInserindo 1000 ao integers1[5]" << endl;
integers1[5] = 1000;
cout << "integers1:" << endl;
outputVector( integers1 );

system("PAUSE");
// Tentativa de usar indice fora do intervalo
cout << "\nTentativa de atribuir 1000 ao integers.at(15)" << endl;
integers.at(15) = 1000; // ERROR: fora do intervalo

return 0;
}
```

Comparação dos  
vectors com  
"=="

Exibindo um  
elemento de um  
vector

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```
// Use operador de equação (==) com objetos vetor
cout << "\nValidando: integers1 == integers2" << endl;

if (integers1 == integers2)
    cout << "integers1 e integers2 sao iguais" << endl;

// Use colchetes para criar evalue
cout << "\nintegers1[5] is " << integers1[5];

// Use colchetes para criar evalue
cout << "\nInserindo 1000 ao integers1[5]" << endl;
integers1[5] = 1000;
cout << "integers1:" << endl;
outputVector( integers1 );

system("PAUSE");
// Tentativa de usar indice fora do intervalo
cout << "\nTentativa de atribuir 1000 ao integers.at(15)" << endl;
integers.at(15) = 1000; // ERROR: fora do intervalo

return 0;
}
```

Atualizando o  
valor

Tentativa de  
atualizar um valor  
fora do intervalo

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```
void outputVector(const vector<int> &array) {
    size_t i; // declaração de variável de controle
    for (i = 0; i < array.size(); i++) {
        cout << setw( 12 ) << array[i];

        if ((i + 1) % 4 == 0) // 4 número por linha de saída
            cout << endl;
    }

    if (i % 4 != 0)
        cout << endl;
}

void inputVector(vector<int> &array) {
    for (size_t i = 0; i < array.size(); i++)
        cin >> array[i];
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quinto Exemplo Usando Array em C++

```
void outputVector(const vector<int> &array) {
    size_t i; // declaração de variável de controle
    for (i = 0; i < array.size(); i++) {
        cout << setw( 12 ) << array[i];

        if ((i + 1) % 4 == 0) // 4 número por linha de saída
            cout << endl;
    }

    if (i % 4 != 0)
        cout << endl;
}

void inputVector(vector<int> &array) {
    for (size_t i = 0; i < array.size(); i++)
        cin >> array[i];
}
```

Exibe os  
elementos do  
array

Inserção de  
elementos com o  
cin

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

```
C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projeto\aula7-ex16.exe
Tamanho do vector integers1 eh 7
vector depois da inicializacao:
0 0 0
Tamanho do vector integers2 eh 10
vector depois da inicializacao:
0 0 0 0 0 0 0 0 0 0
Entre 17 inteiros:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Depois de inputVector, os vectors contem:
integers1:
1 2 3 4
integers2:
8 9 10 11
12 13 14 15
16 17
Validando: integers1 != integers2
integers1 e integers2 nao sao iguais
Tamanho do vector integers3 eh 7
vector depois da inicializacao:
1 2 3 4
5 6
Inserindo integers2 aos integers1:
integers1:
0 9 10 11
12 13 14 15
16 17
integers2:
8 9 10 11
12 13 14 15
16 17
Validando: integers1 == integers2
integers1 e integers2 sao iguais
integers1[5] is 13
Inserindo 1000 ao integers1[5]
integers1:
0 9 10 11
12 1000 14 15
16 17
```



## Exemplo: Ordenamento de Vetores

- Escreva um programa em C++ para ordenar uma sequência de inteiros utilizando o método do "insertion sort" e utilizando "vectors"



## Exemplo: Ordenamento de Vetores

```

/*
 * Aula 7 - Exemplo 17 c/ vectors
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include <vector>

using namespace std;

void insertionSort(vector<int> &);

int main() {
    const int arraySize = 10;
    int a [] = {84, 56, 4, 10, 77, 51, 93, 30, 5, 52};

    vector<int> unsorted(arraySize);
    vector<int> sorted(arraySize);

    for (int i = 0; i < arraySize; i++)
        unsorted[i] = a[i];

    cout << "Unsorted array:" << endl;

    for (int i = 0; i < arraySize; i++)
        cout << setw(4) << unsorted[i];
    cout << endl;
}

```

## Exemplo: Ordenamento de Vetores

```

insertionSort(unsorted);
sorted = unsorted;

cout << "UnSorted array:" << endl;

for (int i = 0; i < arraySize; i++)
    cout << setw(4) << sorted[i];
cout << endl;

return 0;
}

void insertionSort(vector<int> &array) {
    int insert;
    for (int next = 1; next < array.size(); next++) {
        insert = array[next]; // Armazena o valor no elemento atual
        int moveItem = next; // Inicializa a localização para colocar o elemento

        while ((moveItem > 0) && (array[moveItem - 1] > insert)) {
            array[moveItem] = array[moveItem - 1];
            moveItem--;
        }
        array[moveItem] = insert;
    }
}

```

## Introdução à classe STL array do C++11

- classe STL **array**
  - Disponível a partir do C++11
  - Oferece métodos para interação com a estrutura de dados
    - Assim como a classe **vector**
  - Porém, a memória é alocada com tamanho fixo
    - Não é possível aumentar ou diminuir o tamanho da memória alocada para o **array** após a sua criação
      - Diferente da classe **vector**

## Sexto Exemplo Usando Array em C++11

```

#include <iostream>
#include <iomanip>
#include <array>

using namespace std;

int main () {
    array<int, 5> n;

    for (size_t i{0}; i < n.size (); i++) {
        n[i] = 0;
    }

    cout << "Element" << setw (10) << "Value" << endl;

    for (size_t i{0}; i < n.size (); i++) {
        cout << setw (7) << i << setw(10) << n [i] << endl;
    }

    return 0;
}

```

## Sexto Exemplo Usando Array em C++11

```

itaguas~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex27.cpp -o a
itaguas~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> ./a

```

```

Element      Value
0            0
1            0
2            0
3            0
4            0

```

```

using namespace std;

int main () {
    array<int, 5> n;

    for (size_t i{0}; i < n.size (); i++) {
        n[i] = 0;
    }

    cout << "Element" << setw (10) << "Value" << endl;

    for (size_t i{0}; i < n.size (); i++) {
        cout << setw (7) << i << setw(10) << n [i] << endl;
    }

    return 0;
}

```

## Sétimo Exemplo Usando Array em C++11

```
#include <iostream>
#include <iomanip>
#include <array>

using namespace std;

int main () {
    array<int, 5> n {{32, 27, 64, 18, 95}};

    cout << "Element" << setw(10) << "Value" << endl;

    for (size_t i{0}; i < n.size (); i++) {
        cout << setw(7) << i << setw(10) << n [i] << endl;
    }

    return 0;
}
```

Linguagens de Programação - DEL-Poli/UFRJ

Prof. Miguel

## Sétimo Exemplo Usando Array em C++11

Inicialização agregada (múltiplos valores para inicialização de uma única estrutura) requer parênteses dentro de parênteses.

```
#include <iostream>
#include <iomanip>
#include <array>

using namespace std;

int main () {
    array<int, 5> n {{32, 27, 64, 18, 95}};

    cout << "Element" << setw(10) << "Value" << endl;

    for (size_t i{0}; i < n.size (); i++) {
        cout << setw(7) << i << setw(10) << n [i] << endl;
    }

    return 0;
}
```

Linguagens de Programação - DEL-Poli/UFRJ

Prof. Miguel

## Sétimo Exemplo Usando Array em C++11

```
itagu:~/disciplinas/Linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex18.cpp -o a
itagu:~/disciplinas/Linguagens/aulas/2017/programasC++11-C++14> ./a
Element Value
0       32
1       27
2       64
3       18
4       95
```

```
#include <iostream>
#include <iomanip>
#include <array>

using namespace std;

int main () {
    array<int, 5> n {{32, 27, 64, 18, 95}};

    cout << "Element" << setw(10) << "Value" << endl;

    for (size_t i{0}; i < n.size (); i++) {
        cout << setw(7) << i << setw(10) << n [i] << endl;
    }

    return 0;
}
```

Linguagens de Programação - DEL-Poli/UFRJ

Prof. Miguel

## Range-based for no C++11

- Evita o uso de um contador para acessar os elementos do array...
  - Evita acesso a um elemento fora do intervalo
- Sintaxe:

```
//item recebe um elemento do array
for (tipo item : array)
//item recebe uma referência
for (tipo &item : array)
```
- Caso o índice seja necessário...
  - O *range-based* for não pode ser usado

Linguagens de Programação - DEL-Poli/UFRJ

Prof. Miguel

## Oitavo Exemplo Usando Array em C++11

```
#include <iostream>
#include <array>

using namespace std;

int main () {
    array<int, 3> itens {{1, 2, 3}};

    cout << "itens antes da modificação: ";
    for (int item : itens) {
        cout << item << endl;
    }

    cout << "multiplicação dos itens por 2..." << endl;
    for (int &itemRef : itens) {
        itemRef *= 2;
    }

    cout << "itens depois da modificação: ";
    for (int item : itens) {
        cout << item << endl;
    }

    return 0;
}
```

Linguagens de Programação - DEL-Poli/UFRJ

Prof. Miguel

## Oitavo Exemplo Usando Array em C++11

```
itagu:~/disciplinas/Linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex20.cpp -o a
itagu:~/disciplinas/Linguagens/aulas/2017/programasC++11-C++14> ./a
itens antes da modificação: 1
2
3
multiplicação dos itens por 2...
itens depois da modificação: 2
4
6
```

```
cout << "itens antes da modificação: ";
for (int item : itens) {
    cout << item << endl;
}

cout << "multiplicação dos itens por 2..." << endl;
for (int &itemRef : itens) {
    itemRef *= 2;
}

cout << "itens depois da modificação: ";
for (int item : itens) {
    cout << item << endl;
}

return 0;
}
```

Linguagens de Programação - DEL-Poli/UFRJ

Prof. Miguel

## Oitavo Exemplo Usando Array em C++11

E se fosse assim? Sem a referência...  
O que seria impresso na tela?

```
int main () {
    array<int, 3> itens {{1, 2, 3}};
    cout << "itens antes da modificação: ";
    for (int item : itens) {
        cout << item << endl;
    }

    cout << "multiplicação dos itens por 2..." << endl;
    for (int item : itens) {
        item *= 2;
    }

    cout << "itens depois da modificação: ";
    for (int item : itens) {
        cout << item << endl;
    }

    return 0;
}
```

## Oitavo Exemplo Usando Array em C++11

```
itagu:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex20.cpp -o a
itagu:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> ./a
itens antes da modificação: 1
2
3
multiplicação dos itens por 2...
itens depois da modificação: 1
2
3
```

```
cout << "itens antes da modificação: ";
for (int item : itens) {
    cout << item << endl;
}

cout << "multiplicação dos itens por 2..." << endl;
for (int item : itens) {
    item *= 2;
}

cout << "itens depois da modificação: ";
for (int item : itens) {
    cout << item << endl;
}

return 0;
}
```

## Range-based for no C++11 usando auto

- Palavra-chave: **auto**
  - Requer que o compilador determine por inferência o tipo da variável
    - Baseado no valor usado para inicializar a variável
- Sintaxe:
 

```
for (auto item : array)
```

## Oitavo Exemplo Usando Array em C++11

```
#include <iostream>
#include <array>

using namespace std;

int main () {
    array<int, 3> itens {{1, 2, 3}};
    cout << "itens antes da modificação: ";
    for (auto item : itens) {
        cout << item << endl;
    }

    cout << "multiplicação dos itens por 2..." << endl;
    for (auto &itemRef : itens) {
        itemRef *= 2;
    }

    cout << "itens depois da modificação: ";
    for (auto item : itens) {
        cout << item << endl;
    }

    return 0;
}
```

## Oitavo Exemplo Usando Array em C++11

```
itagu:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex20.cpp -o a
itagu:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> ./a
itens antes da modificação: 1
2
3
multiplicação dos itens por 2...
itens depois da modificação: 2
4
6
```

```
cout << "itens antes da modificação: ";
for (auto item : itens) {
    cout << item << endl;
}

cout << "multiplicação dos itens por 2..." << endl;
for (auto &itemRef : itens) {
    itemRef *= 2;
}

cout << "itens depois da modificação: ";
for (auto item : itens) {
    cout << item << endl;
}

return 0;
}
```

## Ponteiros

- Poderosos, mas difíceis de utilizar
- Podem ser usados para fazer passagem de parâmetro por referência
  - Podem ser utilizadas para gerenciar estruturas de dados dinâmicas
    - Aumentam e diminuem
- Aproximam o relacionamento entre arrays e strings
- Contêm endereços de memória como valores

## Declaração e Inicialização de Variáveis Ponteiros

- Indireção
  - Referência de valor por ponteiro
- Declaração de ponteiro
  - \* indica que a variável é um ponteiro
  - `int *myPtr; (declara ponteiro para int, ponteiro do tipo int *)`
  - Múltiplos ponteiros requerem múltiplos asteriscos
  - `int *myPtr1, *myPtr2; int **myPtrtoPtr1;`



POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Declaração e Inicialização de Variáveis Ponteiros

- Pode declarar ponteiros para qualquer tipo de dados
- Inicialização de ponteiro
  - Inicializado com 0, NULL, ou endereço
    - 0 ou NULL aponta para nada

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Operadores Ponteiros

- & (endereço do operador)
  - Retorna endereço de memória do operando
  - Ex: `int y = 5; int *yPtr = &y;`
  - `yPtr` "aponta para" `y`
- \* (operador de desreferenciação)
  - `*yPtr` retorna `y`, porque `yPtr` aponta para `y`
  - Ponteiro de desreferenciação é lvalue (valor à esquerda)
  - `*yPtr = 9; // atribui 9 para y`
- \* e & são opostos entre si

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 1
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int a; // É um inteiro
    int *aPtr; // É um ponteiro para um inteiro

    a = 7;
    aPtr = &a;

    cout << "O endereço de a eh " << &a
         << "\nO valor de aPtr eh " << aPtr;

    cout << "\n\nO valor de a eh " << a
         << "\nO valor de *aPtr eh " << *aPtr;

    cout << "\n\nMostrando que * e & são opostos entre si."
         << "\n&aPtr = " << &aPtr
         << "\n&aPtr = " << &aPtr << endl;

    return 0;
}
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 1
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;
    
```

```

shell>$ g++ -Wall exemplo.cpp -o ex1
shell>$ ./ex1
O endereço de a eh 0x28ff44
O valor de aPtr eh 0x28ff44

O valor de a eh 7
O valor de *aPtr eh 7

Mostrando que * e & são opostos entre si
&*aPtr = 0x28ff44
*&aPtr = 0x28ff44
shell>$
    
```

```

        return 0;
    }
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Chamada de Funções por Referência

- Três maneiras de passar argumentos para funções
  - Passagem por valor
  - Passagem por referência com ponteiros como argumentos
  - Passagem por referência com referências como argumentos
- return pode retornar um valor da função
- Argumentos passados para a função usando referências como argumentos
  - Modificam os valores originais dos argumentos
  - Mais de um valor pode ser "retornado"

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Chamada de Funções por Referência

- Passagem por referência com ponteiros como argumentos
  - Passagem por referência
    - Usa ponteiros e operador de desreferenciação
  - Passagem de endereço do argumento usando o operador &
  - Arrays não são passados com & porque o nome do array já é um ponteiro
  - \* operador usado como alias/apelido da variável dentro da função

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Segundo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 2
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int cubeByValue(int);

int main() {
    int number = 5;

    cout << "O valor original de number eh " << number;

    // Passagem de number por valor
    number = cubeByValue(number);

    cout << "\n\nO novo valor de number eh " << number << endl;

    system("PAUSE");
    return 0;
}

int cubeByValue(int n) { return n * n * n; }
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Segundo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 2
 * Autor: Miguel Campista
 */
#include <iostream>
    
```

```

shell>$ g++ -Wall exemplo.cpp -o ex2
shell>$ ./ex2
O valor original de number eh 5
O novo valor de number eh 125
shell>$
    
```

```

// Passagem de number por valor
number = cubeByValue(number);

cout << "\n\nO novo valor de number eh " << number << endl;

system("PAUSE");
return 0;
}

int cubeByValue(int n) { return n * n * n; }
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Segundo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 2
 * Autor: Miguel Campista
 */
#include <iostream>
    
```

```

shell>$ g++ -Wall exemplo.cpp -o ex2
shell>$ ./ex2
O valor original de number eh 5
O novo valor de number eh 125
shell>$
    
```

```

// Passagem de number por valor
number = cubeByValue(number);

cout << "\n\nO novo valor de number eh " << number << endl;

int cubeByValue(int n) { return n * n * n; }
    
```



Como ficaria se a passagem de parâmetro fosse por referência?

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 3
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void cubeByReference(int *);

int main() {
    int number = 5;

    cout << "O valor original de number eh " << number;

    // Passagem de number por valor
    cubeByReference(&number);

    cout << "\n\nO novo valor de number eh " << number << endl;

    return 0;
}

void cubeByReference(int *nPtr) { *nPtr = (*nPtr) * (*nPtr) * (*nPtr); }
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 3
 * Autor: Miguel Campista
 */
#include <iostream>
    
```

```

shell>$ g++ -Wall exemplo.cpp -o ex3
shell>$ ./ex3
O valor original de number eh 5
O novo valor de number eh 125
shell>$
    
```

```

// Passagem de number por valor
cubeByReference(&number);

cout << "\n\nO novo valor de number eh " << number << endl;

return 0;
}

void cubeByReference(int *nPtr) { *nPtr = (*nPtr) * (*nPtr) * (*nPtr); }
    
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Usando const com Ponteiros

- Qualificador `const`
  - Valor da variável não deve ser modificado
  - `const` usado quando a função não precisa mudar a variável
- Princípio do menor privilégio
  - Garante a função acesso suficiente para realizar a tarefa, mas nada além disso

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Usando const com Ponteiros

- Quatro maneiras para passar o ponteiro para a função
  - Pontoeiro não constante para dado não constante
    - Quantidade maior de acesso
  - Pontoeiro não constante para dado constante
  - Pontoeiro constante para dado não constante
  - Pontoeiro constante para dado constante
    - Quantidade menor de acesso

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 4
 * Autor: Miguel Campista
 */
#include <iostream>
#include <ctype>

void convertToUpperCase(char *);

using namespace std;

int main() {
    char phrase [] = "Caracteres e $32,99";
    cout << "A frase antes da conversao eh: " << phrase;
    convertToUpperCase(phrase);
    cout << "\n\nA frase depois da conversao eh: " << phrase << endl;

    return 0;
}

void convertToUpperCase(char *sPtr) {
    while (*sPtr != '\0') {
        if (islower(*sPtr))
            *sPtr = toupper(*sPtr);
        sPtr++;
    }
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Ponteiros em C++

Pontoeiro não constante para dado não constante

```
/*
 * Aula 8 - Exemplo 4
 * Autor: Miguel Campista
 */
#include <iostream>
#include <ctype>

void convertToUpperCase(char *);

using namespace std;

int main() {
    char phrase [] = "Caracteres e $32,99";
    cout << "A frase antes da conversao eh: " << phrase;
    convertToUpperCase(phrase);
    cout << "\n\nA frase depois da conversao eh: " << phrase << endl;

    return 0;
}

void convertToUpperCase(char *sPtr) {
    while (*sPtr != '\0') {
        if (islower(*sPtr))
            *sPtr = toupper(*sPtr);
        sPtr++;
    }
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quarto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 4
 * Autor: Miguel Campista
 */
#include <iostream>
#include <ctype>

shell>$ g++ -Wall exemplo.cpp -o ex4
shell>$ ./ex4
A frase antes da conversao eh: caracteres e $32,99
A frase depois da conversao eh: CARACTERES E $32,99
shell>$

    cout << "\n\nA frase depois da conversao eh: " << phrase << endl;

    return 0;
}

void convertToUpperCase(char *sPtr) {
    while (*sPtr != '\0') {
        if (islower(*sPtr))
            *sPtr = toupper(*sPtr);
        sPtr++;
    }
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quinto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 5
 * Autor: Miguel Campista
 */
#include <iostream>

void printCaracteres(const char *);

using namespace std;

int main() {
    char phrase [] = "Caracteres e $32,99";

    cout << "A frase antes da conversao eh: " << phrase << endl;
    printCaracteres(phrase);
    cout << endl;

    return 0;
}

void printCaracteres(const char *sPtr) {
    for (; *sPtr != '\0'; sPtr++)
        cout << *sPtr;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quinto Exemplo Usando Ponteiros em C++

```
/*
Ponteiro não constante para dado constante
#include <iostream>

void printCaracteres(const char *);

using namespace std;

int main() {
    char phrase [] = "caracteres e $32,99";

    cout << "A frase antes da conversao eh: " << phrase << endl;
    printCaracteres(phrase);
    cout << endl;

    return 0;
}

void printCaracteres(const char *sPtr) {
    for (; *sPtr != '\0'; sPtr++)
        cout << *sPtr;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Quinto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 5
 * Autor: Miguel Campista
 */
#include <iostream>

shell>$ g++ -Wall exemplo.cpp -o ex5
shell>$ ./ex5
A frase antes da conversao eh: caracteres e $32,99
caracteres e $32,99
shell>$

    cout << "A frase antes da conversao eh: " << phrase << endl;
    printCaracteres(phrase);
    cout << endl;

    return 0;
}

void printCaracteres(const char *sPtr) {
    for (; *sPtr != '\0'; sPtr++)
        cout << *sPtr;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sexto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 6
 * Autor: Miguel Campista
 */
#include <iostream>

void f(const int *);

using namespace std;

int main() {
    int y;

    f(&y); // Tenta modificação ilegal

    return 0;
}

void f(const int *xPtr) {
    *xPtr = 100;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sexto Exemplo Usando Ponteiros em C++

**Como é feita a passagem de parâmetro?  
O programa está correto?**

```
#include <iostream>

void f(const int *);

using namespace std;

int main() {
    int y;

    f(&y); // Tenta modificação ilegal

    return 0;
}

void f(const int *xPtr) {
    *xPtr = 100;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sexto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 6
 * Autor: Miguel Campista
 */
#include <iostream>

void f(const int *);

21 C:\Users\Miguel\Documents\UFRJ\... In function 'void f(const int*)':
C:\Users\Miguel\Documents\UFRJ\... assignment of read-only location
C:\Users\Miguel\Documents\UFRJ\... [Build Error] [aula8-ex6.o] Error 1

    int y;

    f(&y); // Tenta modificação ilegal

    return 0;
}

void f(const int *xPtr) {
    *xPtr = 100;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Usando const com Ponteiros

- Ponteiros **const**
  - Sempre aponta para o mesmo local de memória
    - O próprio nome do array
  - Deve ser inicializado quando declarado

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sétimo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 7
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x, y;

    /*
     * ptr é um ponteiro constante para um inteiro que pode
     * ser modificado através de ptr, mas ptr aponta sempre para
     * mesma posição de memória
     */
    int * const ptr = &x;

    *ptr = 7; // Permitted: *ptr não é constante
    ptr = &y; // Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sétimo Exemplo Usando Ponteiros em C++

Ponteiro constante para dado não constante

```
#include <iostream>

using namespace std;

int main() {
    int x, y;

    /*
     * ptr é um ponteiro constante para um inteiro que pode
     * ser modificado através de ptr, mas ptr aponta sempre para
     * mesma posição de memória
     */
    int * const ptr = &x;

    *ptr = 7; // Permitted: *ptr não é constante
    ptr = &y; // Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Sétimo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 7
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x = 5, y;

    /*
     * ptr é um ponteiro constante para um inteiro que pode
     * ser modificado através de ptr, mas ptr aponta sempre para
     * mesma posição de memória
     */
    int * const ptr = &x;

    *ptr = 7; // Permitted: *ptr não é constante
    ptr = &y; // Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Oitavo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 8
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x = 5, y;

    /*
     * ptr é um ponteiro constante para um inteiro constante;
     * ptr sempre aponta para a mesma posição de memória; o
     * inteiro naquela posição não pode ser modificado
     */
    const int *const ptr = &x;

    cout << *ptr << endl;

    *ptr = 7; // Erro: *ptr é constante, não pode receber um novo valor
    ptr = &y; // Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Oitavo Exemplo Usando Ponteiros em C++

Ponteiro constante para dado constante

```
/*
 * Aula 8 - Exemplo 8
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x = 5, y;

    /*
     * ptr é um ponteiro constante para um inteiro constante;
     * ptr sempre aponta para a mesma posição de memória; o
     * inteiro naquela posição não pode ser modificado
     */
    const int *const ptr = &x;

    cout << *ptr << endl;

    *ptr = 7; // Erro: *ptr é constante, não pode receber um novo valor
    ptr = &y; // Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Oitavo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 8
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x = 5, y;

    /*
     * ptr é um ponteiro constante para um inteiro constante;
     * ptr sempre aponta para a mesma posição de memória; o
     * inteiro naquela posição não pode ser modificado
     */
    const int *const ptr = &x;

    cout << *ptr << endl;

    *ptr = 7; // Erro: *ptr é constante, não pode receber um novo valor
    ptr = &y; // Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista



## Expressões com Ponteiros e Aritmética com Ponteiros

- Aritmética com ponteiro
  - Incremento/decremento de ponteiro (++ ou --)
  - Adição/subtração de inteiro para/de um ponteiro (+ ou +=, - ou -=)
  - Ponteiros podem ser subtraídos entre si
  - Aritmética de ponteiro sem significado exceto se realizado sobre ponteiro para array

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

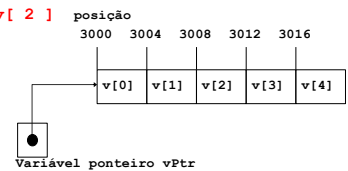
## Expressões com Ponteiros e Aritmética com Ponteiros

- Array de 5 elementos `int` em uma máquina usando inteiros de 4 bytes
  - `vPtr` aponta para o primeiro elemento `v[ 0 ]`, que está na posição 3000

```
cout << vPtr; // Imprime 3000
```

- `vPtr += 2`; atribui 3008 a `vPtr`

```
vPtr aponta para v[ 2 ]
```



## Expressões com Ponteiros e Aritmética com Ponteiros

- Subtração de ponteiros
    - Retorna número de elementos entre dois endereços
- ```
vPtr2 = &v[ 2 ]; vPtr = &v[ 0 ];  
cout << vPtr2 - vPtr; // Imprime 2
```
- Atribuição de ponteiro
    - Ponteiro pode ser atribuído para outro ponteiro se ambos forem do mesmo tipo
      - Se não forem, operador `cast` deve ser usado
    - Exceção: ponteiro para `void` (tipo `void *`)
      - Ponteiro genérico, representa qualquer tipo
      - Casting não é necessário para converter ponteiro para ponteiro `void`
      - Ponteiros `void` não podem ser desreferenciados

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Expressões com Ponteiros e Aritmética com Ponteiros

- Comparação de ponteiros
  - Uso de sinal de igualdade ou operadores relacionais
  - Comparações não fazem sentido exceto quando ponteiros apontam para algum membro do mesmo array
  - Comparações de endereços armazenados em ponteiros
  - Uso comum para determinar se um ponteiro é zero
    - O que significa que ele aponta para nada

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Ponteiros para Funções

- Ponteiros para funções
  - Contêm endereço da função
  - Parecido com o motivo pelo qual o nome do array é o endereço do primeiro elemento
  - Nome da função inicia endereço de código que define a função
- Ponteiros para funções podem ser
  - Passados para funções
  - Retornados das funções
  - Armazenados em arrays
  - Atribuídos a outros ponteiros para funções

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Ponteiros para Funções

- Funções que chamam funções através de ponteiros
    - Assumir parâmetro:
      - `bool ( *compare ) ( int, int )`
    - Executar a função com os dois inteiros
      - `( *compare ) ( int1, int2 )`
        - Referência indireta a um ponteiro para função executar
- OU
- `compare( int1, int2 )`
    - Poderia ser confuso
      - » Usuário pode pensar em comparar nome atual da função no programa e não usar o ponteiro

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

```

/*
 * Aula 8 - Exemplo 16
 * Autor: Miguel Campista
 */
#include <iostream>
#include <vector>

void bubble(int [], const int, bool (*)(int, int));
void swap(int *, const int, int *, const int);
bool ascending(int, int);
bool descending(int, int);

using namespace std;

int main() {
    const int arraySize = 10;
    int order, counter;
    int a [arraySize] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};

    cout << "Entre com 1 para ordenar em ordem ascendente,\n"
         << "Entre 2 para ordenar em ordem descendente: ";
    cin >> order;
    cout << "Dados na ordem original:\n";

    // Array original
    for (counter = 0; counter < arraySize; counter++)
        cout << setw(4) << a[counter] << " ";

    if (order == 1) {
        bubble(a, arraySize, ascending);
        cout << "Dados em ordem ascendente:\n";
    } else {
        bubble(a, arraySize, descending);
        cout << "Dados em ordem descendente:\n";
    }

    // Array ordenado
    for (counter = 0; counter < arraySize; counter++)
        cout << setw(4) << a[counter] << " ";

    cout << endl;

    return 0;
}

```

## Nono Exemplo Usando Ponteiros em C++

```

void bubble(int work[], const int size, bool (*compare)(int, int)) {
    for (int pass = 1; pass < size; pass++) {
        for (int count = 0; count < size - 1; count++) {
            if ((*compare)(work[count], work[count + 1]))
                swap(&work[count], &work[count + 1]);
        }
    }
}

void swap(int * const element1Ptr, int * const element2Ptr) {
    int hold = *element1Ptr;
    *element1Ptr = *element2Ptr;
    *element2Ptr = hold;
}

bool ascending(int a, int b) {
    return b < a; // swap se b for maior que a
}

bool descending(int a, int b) {
    return b > a; // swap se b for maior que a
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Nono Exemplo Usando Ponteiros em C++

```

// bubble(int work[], const int size, bool (*compare)(int, int)) {
shell>$ g++ -Wall exemplo.cpp -o ex16
shell>$ ./ex16
Entre com 1 para ordenar em ordem ascendente,
Entre com 2 para ordenar em ordem descendente: 1

Dados na ordem original
 2  6  4  8 10 12 89 68 45 37
Dados na ordem ascendente
 2  4  6  8 10 12 37 45 68 89
shell>$

bool descending(int a, int b) {
    return b > a; // swap se b for maior que a
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 1

- Escreva um programa que calcule o valor mínimo e máximo de um vetor. Para isso, utilize a classe vector e utilize ponteiro para funções.



POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 1

```

/*
 * Aula 8 - Exemplo 23
 * Autor: Miguel Campista
 */
#include <iostream>
#include <vector>

using namespace std;

void processData(int a, vector<int> &v, int (*)(vector<int> &));
int maximum(vector<int> &v);
int minimum(vector<int> &v);

int main() {
    const int arraySize = 10;
    vector<int> v(10);
    int resultado;

    int array [] = {3, 7, 23, 9, 10, 45, 65, 21, 18, 32};

    for (int i = 0; i < arraySize; i++)
        v[i] = array[i];

    processData(resultado, v, maximum);
    cout << "O valor máximo do vetor eh: " << resultado << endl;

    processData(resultado, v, minimum);
    cout << "O valor mínimo do vetor eh: " << resultado << endl;

    return 0;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 1

```

void processData(int a, vector<int> &vec, int (*)(vector<int> &)) {
    a = (*op)(vec);
}

int maximum(vector<int> &a) {
    int max = 0;
    int i;
    for (i = 0; i < a.size(); i++) {
        if (a[i] > max)
            max = a[i];
    }
    return max;
}

int minimum(vector<int> &a) {
    int min = 100;
    int i;
    for (i = 0; i < a.size(); i++) {
        if (a[i] < min)
            min = a[i];
    }
    return min;
}

```

POO para Redes de Computadores - COPPE-PEE/UFRJ

Prof. Miguel Campista

## Exemplo 2

- Escreva um programa que receba um cadastro <nome, idade> e escreva em um arquivo. O programa deve ainda ser capaz de exibir todos os cadastros do arquivo e de excluir o arquivo.



## Exemplo 2

```
/*
 * Aula 8 - Exemplo 25
 * Arquivo Cap8Ex25.h
 * Autor: Miguel Campista
 */
#ifndef CAP8EX25_H
#define CAP8EX25_H

#include <iostream>
#include <string>
#include <fstream>
#include <string>
#include <iomanip>
#include <string>
#include <string>

using namespace std;

class Cadastro {
public:
    Cadastro (string);
    ~Cadastro ();

    void writeLinesToFile (string, string);
    void readLinesFromFile();
    void clearFile();

private:
    string fileName;
    fstream fileStr;

    void checkIfFileOpen ();
};

#endif //CAP8EX25_H_*/
```

## Exemplo 2

```
/*
 * Aula 8 - Exemplo 25
 * Arquivo Cap8Ex25.cpp
 * Autor: Miguel Campista
 */
#include "Cap8Ex25.h"

Cadastro::Cadastro (string file) {
    fileName = file + ".txt";
    cout << "File: " << fileName << "\n\n" << endl;
}

Cadastro::~Cadastro () {
    cout << "Fechando o arquivo..." << endl;
}

void Cadastro::checkIfFileOpen () {
    if (!fileStr.is_open ()) {
        cout << "Problemas ao abrir o arquivo...\nSaindo" << endl;
        exit (3);
    }
}

void Cadastro::writeLinesToFile (string name, string age) {
    // Abrindo arquivo para escrita
    fileStr.open (fileName.c_str(), fstream::in | fstream::out | fstream::app);
    checkIfFileOpen ();

    // Escrevendo no arquivo uma linha
    fileStr << left << setw(25) << name << setw(3) << age << endl;
    fileStr.close ();
}
```

## Exemplo 2

```
void Cadastro::readLinesFromFile () {
    string lines;
    fileStr.open (fileName.c_str(), fstream::in | fstream::out | fstream::app);
    checkIfFileOpen ();

    while (!fileStr.eof ()) {
        getline (fileStr, lines);
        cout << line << endl;
    }
    fileStr.close ();
}

void Cadastro::clearFile () {
    if (!remove (fileName.c_str())) {
        cout << "Arquivo removido..." << endl;
    }
    else
        cout << "Nao foi possivel remover o arquivo..." << endl;
}
```

## Exemplo 2

```
/*
 * Aula 8 - Exemplo 25
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include "Cap8Ex25.h"

int main () {
    string file, op;

    cout << "Entre com o nome do arquivo: ";
    getline (cin, file);

    Cadastro cad (file);

    while (1) {
        cout << "Entre com a operacao desejada: ";
        getline (cin, op);
        cout << "Operacao selecionada: " << op << endl;

        if (top.compare ("inserir")) {
            string name, age;

            cout << "Entre com o nome: ";
            getline (cin, name);

            cout << "Entre com a idade: ";
            getline (cin, age);

            cout << "Cadastro a ser inserido:\n"
                << left << setw(25) << name << setw(3) << age << endl;

            cad.writeLinesToFile (name, age);
        }
    }
}
```

## Exemplo 2

```
        } else if (top.compare ("mostrar")) {
            cad.readLinesFromFile ();
        }
        } else if (top.compare ("limpar")) {
            cad.clearFile ();
        }
        } else if (top.compare ("terminar")) {
            cout << "Saindo do programa..." << endl;
            break;
        }
        } else
            cout << "Operacao desconhecida..." << endl;
    }

    return 0;
}
```

## Leitura Recomendada

- Capítulos 7 e 8 do livro
  - Deitel, "*C++ How to Program*", 5th edition, Editora Prentice Hall, 2005