Linguagens de Programação

Prof. Miguel Elias Mitre Campista

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

Parte III

Introdução à Programação em C++
(Continuação)

Relembrando da Última Aula...

- Funções
- Classes de armazenamento
- Regras de escopo
- Funções sobrecarregadas
- Templates
- Mais exemplos de programação orientada a objetos...

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

Arrays

- Grupo consecutivo de posições da memória
  - Todas são do mesmo tipo
- Índice
  - Número da posição usado para indicar uma 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].

Array

- Inicialização de um array em uma declaração com uma lista inicializadora
  - Lista inicializadora
    - Os itens encontram-se entre chaves ({})
    - Os itens na lista são separados por vírgula
    - Ex.: `int n[] = { 10, 20, 30, 40, 50 };`
      - Pelo fato de o tamanho do array ser omitido na declaração, o compilador determina o tamanho do array com base no tamanho da lista inicializadora
      - Os valores do índice são `0, 1, 2, 3, 4`
      - São inicializados nos valores `10, 20, 30, 40, 50`, respectivamente.

Array

- Inicialização de um array em uma declaração com uma lista inicializadora
  - Se houver menos inicializadores que elementos no array
    - Os elementos remanescentes são inicializados em zero
      - Ex.: `int n[ 10 ] = { 0 };`
        - » Inicializa explicitamente o primeiro elemento em zero
        - » Inicializa implicitamente os nove elementos restantes em zero
  - Se houver mais inicializadores que elementos no array
    - Ocorrerá erro de compilação

Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 2
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

int main() {
    int n [] = {32, 27, 2, 5, 22, 90, 1, 10, 67, 99}; // Array n de 10 inteiros

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

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

    return 0;
}
```


Exemplo Usando Array em C++

Lista inicializadora utilizada para inicializar o array. Compilador aloca memória conforme o número de elementos declarados

```
/*
 * Aula 7 - Exemplo 2
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

int main() {
    int n [] = {32, 27, 2, 5, 22, 90, 1, 10, 67, 99}; // Array n de 10 inteiros

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

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

    return 0;
}
```


Exemplo Usando Array em C++

```
shell>$ g++ exemplo.cpp –o ex2
shell>$ ./ex2
Elemento          valor
      0             32
      1             27
      2              2
      3              5
      4             22
      5             90
      6              1
      7             10
      8             67
      9             99
shell>$
```


Array

- Fornecer mais inicializadores em uma lista inicializadora de array do que o número de elementos existentes no array é um erro de compilação
- Esquecer de inicializar os elementos de um array cujos elementos deveriam ser inicializados é um erro de lógica

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;

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

    return 0;
}
```

Exemplo Usando Array em C++

Declaração do número de elementos do array utilizando uma variável const

```
/*
 * 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;

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

    return 0;
}
```


Exemplo Usando Array em C++

```
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>$
```


Array

- Variáveis constantes
  - São declaradas usando-se o qualificador `const`
  - São também chamadas de constantes identificadas ou variáveis de leitura
  - Devem ser inicializadas com uma sentença constante quando são declaradas e não podem ser modificadas posteriormente
  - Podem ser colocadas em qualquer lugar em que se espera uma expressão constante
  - Usar variáveis constantes para especificar o tamanho do array torna os programas mais escaláveis e torna o programa mais fácil de modificar

Array

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

```
const int x;
```

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

```
const int x = 1;
x = 2;
```


Array

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

```
const int x;
```

X Erro!

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

```
const int x = 1;
x = 2;
```

X Erro!

Array

- Recomendação
  - Definir o tamanho de um array como uma variável constante torna os programas mais claros
    - Técnica que elimina os chamados "números mágicos"
      - Ex.: Mencionar repetidamente o tamanho 10 em código de processamento de array para um array de 10 elementos dá para o número 10 uma importância artificial que pode confundir o leitor quando o programa incluir outros números 10 que não estão relacionados com o tamanho do array

Array

- Emprego de arrays de caracteres para manipular strings
  - Os arrays podem ser de qualquer tipo, incluindo `char`s.
    - Strings de caracteres podem ser armazenados em arrays char
  - Podem ser inicializados por meio de um literal de string.
    - Ex.: `char string1[] = "Hi";`
    - Equivalente a
      - `char string1[] = { 'H', 'i', '\0' };`
  - O array contém cada caractere mais um caractere especial de terminação de string denominado caractere nulo (`'\0'`)

Array

- Emprego de arrays de caracteres para manipular strings
  - Podem também ser inicializados com constantes de caractere individuais em uma lista inicializadora
    - Ex.: `char string1[] = { 'f', 'i', 'r', 's', 't', '\0' };`
  - Além disso, podem inserir uma string diretamente em um array de caracteres por meio do teclado, usando `cin` e `>>`
    - Ex.: `cin >> string1;`
      - `cin >>` talvez leia mais caracteres que o array pode armazenar

Sétimo Exemplo Usando Array em C++

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

using namespace std;

int main() {
    char string1[20];
    char string2[] = ("string literal"); // reserva 15 caracteres

    // Lê a string fornecida pelo usuário para string1
    cout << "Entre com a string \"Hello there\": ";
    cin >> string1;

    // Strings recebidas
    cout << "string1 eh: " << string1 << "\nstring2 eh: " << string2 << endl;

    cout << "string1 com espacos entre caracteres eh:\n";
    for (int i = 0; string1[i] != '\0'; i++)
        cout << string1[i] << ' ';

    cin >> string1; // Lê there
    cout << "\nstring1 eh: " << string1 << endl;

    return 0;
}
```


Sétimo Exemplo Usando Array em C++

Armazena "string literal" como um array de caracteres

Inicialização de um array por meio de `cin`

Sétimo Exemplo Usando Array em C++

Loop até encontrar o caractere nulo

Sétimo Exemplo Usando Array em C++

```
shell>$ g++ exemplo.cpp –o ex7

shell>$ ./ex7
Entre com a string "Hello there": hello there
string1 eh: hello
string2 eh: string literal
string1 com espacos entre caracteres eh:
h e l l o
string1 eh: there
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

Oitavo 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;
}
```


Oitavo Exemplo Usando Array em C++

```
void staticArrayInit() {
    // Inicializa 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() {
    // Inicializa elementos como 0 na primeira vez que 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 << "array1[" << i << "] = " << (array2[i] += 5) << " ";

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

Cria um array **static**

Cria um array automático

Oitavo Exemplo Usando Array em C++

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

```
C:\Documents and Settings\Campista\Meus documentos\miguel\aula7-ex8.exe
array1[0] = 0 array1[1] = 0 array1[2] = 0
Valores no array estatico
array1[0] = 5 array1[1] = 5 array1[2] = 5
Valores no array estatico ao sair da funcao

array2[0] = 1 array2[1] = 2 array2[2] = 3
Valores atribuidos ao array automatico
array1[0] = 6 array1[1] = 7 array1[2] = 8
Valores no array automatico ao sair da funcao


 ** Segunda chamada para cada funcao:

array1[0] = 5 array1[1] = 5 array1[2] = 5
Valores no array estatico
array1[0] = 10 array1[1] = 10 array1[2] = 10
Valores no array estatico ao sair da funcao

array2[0] = 1 array2[1] = 2 array2[2] = 3
Valores atribuidos ao array automatico
array1[0] = 6 array1[1] = 7 array1[2] = 8
Valores no array automatico ao sair da funcao
```

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

    for (int i = 0; i < 3; i++)
        cout << "array1[" << i << "] = " << (array2[i] += 5) << " ";

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


Passagem de Array para Função

- Para passar um argumento array a uma função
  - Especifique o nome do array sem os colchetes
    - O array `hourlyTemperatures` é declarado como
      - `int hourlyTemperatures[ 24 ];`
    - A chamada de função

      `modifyArray( hourlyTemperatures, 24 );`

      passa o array `hourlyTemperatures` e o seu tamanho à função `modifyArray`
  - O tamanho do array é normalmente passado como outro argumento, de modo que a função possa processar o número específico de elementos no array

Passagem de Array para Função

- Os arrays são passados por referência
  - A chamada de função na verdade passa o endereço inicial do array
    - Portanto, a função sabe em que posição o array se encontra na memória
  - O chamador concede à função chamadora acesso direto aos dados do chamador
    - A função chamada pode manipular esses dados

Passagem de Array para Função

- Passar arrays por referência é mais eficiente
  - Arrays passados por valor requerem uma cópia de cada elemento
    - Arrays grandes passados com frequência demandaria tempo e exigiria espaço de armazenamento considerável para as cópias dos elementos do array

Passagem de Array para Função

- Elementos individuais do array passados por valor
  - Fragmentos de dados simples
    - São chamados escalares ou quantidades escalares
  - Para passar um elemento a uma função
    - Use o elemento do array com o seu índice como um argumento
- Funções que consideram os arrays como argumentos
  - A lista de parâmetros da função deve especificar um parâmetro de array
    - Ex.: `void modArray( int b[], int arraySize );`

Passagem de Array para Função

- Funções que consideram os arrays como argumentos
  - O parâmetro de array pode incluir o tamanho do array
    - O compilador o ignora, pois só se preocupa com o endereço do primeiro elemento

```
void funcao( int a[10], int num );
```

```
void funcao( int a[], int num );
```


Passagem de Array para Função

- Os protótipos de função podem incluir nomes de parâmetro
  - Mas o compilador vai ignorá-los
  - Os nomes de parâmetro devem ser deixados fora dos protótipos de função

```
void funcao( int a[], int num );
```

```
void funcao( int [], int );
```


Nono Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 9
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

void modifyArray(int [], int); // Protótipo de função
void modifyElement(int); // Protótipo de função

int main() {
    const int arraySize = 5;
    int a [arraySize] = {0, 1, 2, 3, 4};

    cout << " ** Passagem de array por referencia\n\n";

    cout << "Array original:\n";
    for (int i = 0; i < arraySize; i++)
        cout << setw(3) << a[i];
    cout << endl;

    modifyArray(a, arraySize);

    cout << "Array modificado:\n";
    for (int i = 0; i < arraySize; i++)
        cout << setw(3) << a[i];
```


Nono Exemplo Usando Array em C++

Função recebe um array como argumento

Passagem do array

```
/*
 * Aula 7 - Exemplo 9
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

void modifyArray(int [], int); // Protótipo de função
void modifyElement(int); // Protótipo de função

int main() {
    const int arraySize = 5;
    int a [arraySize] = {0, 1, 2, 3, 4};

    cout << " ** Passagem de array por referencia\n\n";

    cout << "Array original:\n";
    for (int i = 0; i < arraySize; i++)
        cout << setw(3) << a[i];
    cout << endl;

    modifyArray(a, arraySize);

    cout << "Array modificado:\n";
    for (int i = 0; i < arraySize; i++)
        cout << setw(3) << a[i];
```

Nono Exemplo Usando Array em C++

```
    cout << "\n\n ** Passagem de elemento por valor\n\n";

    cout << "Elemento original:\n";
    cout << setw(3) << a[3] << endl;

    modifyElement(a[3]);

    cout << "Elemento modificado:\n";
    cout << setw(3) << a[3] << endl;

    return 0;
}

void modifyArray(int b[], int numberElements) {
    // Multiplica cada elemento do array por dois
    for (int i = 0; i < numberElements; i++)
        b[i] *= 2;
}

void modifyElement(int e) {
    cout << "Valor do elemento em modifyElement: " << (e *= 2) << endl;
}
```


Nono Exemplo Usando Array em C++

```
    cout << "\n\n ** Passagem de elemento por valor\n\n";

    cout << "Elemento original:\n";
    cout << setw(3) << a[3] << endl;

    modifyElement(a[3]);

    cout << "Elemento modificado:\n";
    cout << setw(3) << a[3] << endl;

    return 0;
}

void modifyArray(int b[], int numberElements) {
    // Multiplica cada elemento do array por dois
    for (int i = 0; i < numberElements; i++)
        b[i] *= 2;
}

void modifyElement(int e) {
    cout << "Valor do elemento em modifyElement: " << (e *= 2) << endl;
}
```

Passagem de elemento

Nono Exemplo Usando Array em C++

```
    cout << "\n\n ** Passagem de elemento por valor\n\n";
```

```
shell>$ g++ exemplo.cpp –o ex9
shell>$ ./ex9
 ** Passagem de array por referencia

Array original:
   0   1   2   3   4
Array modificado:
   0   2   4   6   8

 ** Passagem de elemento por valor

Elemento original:
   6
Valor do elemento em modifyElement: 12
Elemento modificado:
   6
shell>$
```


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 no corpo da função
  - Permite que o programador evite alterações acidentais nos dados

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 no corpo da 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

Décimo 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;
}
```

Décimo 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

Décimo Exemplo Usando Array em C++

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

using namespace std;

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

    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;
}
```


Estudo de Caso: Classe `GradeBook`

- Classe `GradeBook`
  - Representa um livro de notas que armazena e analisa as 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
    - Use o nome da classe seguido do operador binário de resolução de escopo e o nome dos membros de dados `static`

Décimo Primeiro 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 configura 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();
```

Décimo Primeiro 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 configura 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 da classe `static`

Décimo Primeiro Exemplo Usando Array em C++

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

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

Décimo Primeiro Exemplo Usando Array em C++

`array grades` para armazenar as notas

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

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


Décimo Primeiro 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;
}
```


Décimo Primeiro Exemplo Usando Array em C++

Copia elementos de `gradesArray` para o atributo `grades`

Loop em `grades` para o encontrar a nota mínima

```
/*
 * 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;
}
```


Décimo Primeiro 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 maximo de 25 caracteres..." << endl <<
            "Nome limitado aos primeiros 25 caracteres: " << courseName <<
            endl;
      }
}
```


Décimo Primeiro Exemplo Usando Array em C++

Loop em `grades` para o encontrar a nota máxima

Loop para a soma e posterior divisão

```
// 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 maximo de 25 caracteres..." << endl <<
            "Nome limitado aos primeiros 25 caracteres: " << courseName <<
            endl;
      }
}
```


Décimo Primeiro Exemplo Usando Array em C++

```
// Função que obtém 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 << "\nMedia 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;
      }
}
```

Décimo Primeiro Exemplo Usando Array em C++

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

            }
      }
```


Décimo Primeiro Exemplo Usando Array em C++

```
// Função para exibir a distribuição de notas da turma
void GradeBook::displayBarChart() {
      cout << "Distribuicao 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; cou
            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 o encontrar a frequência

Décimo Primeiro 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;
}
```


Décimo Primeiro 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;
}
```

Usa **students** declarado como **static** na classe

Décimo Primeiro 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;
}
```

```
C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagem\projetos\aula7-ex11.exe
Bem-vindo ao seu primeiro programa com classes em Linguagens de Programacao!
Student  1:  87
Student  2:  68
Student  3:  94
Student  4: 100
Student  5:  83
Student  6:  78
Student  7:  85
Student  8:  91
Student  9:  76
Student 10:  87

Media da turma eh: 84.90
Menor nota foi: 68
Maior nota foi: 100

Distribuicao de notas:
  0-9:
 10-19:
 20-29:
 30-39:
 40-49:
 50-59:
 60-69:*
 70-79:**
 80-89:****
 90-99:**
  100:*
```


Busca em Array

- Os arrays podem armazenar grande quantidade de dados
  - Talvez seja necessário determinar se existe um certo valor-chave no array
- Busca linear
  - Compara cada elemento de um array com uma chave de pesquisa

Busca Linear em Array

- Caso os elementos do array não estejam organizados segundo alguma classificação
  - Para encontrar um elemento, é igualmente provável que o valor seja encontrado tanto na primeira quanto na última posição
    - Em média, o programa deve comparar a chave de pesquisa com metade dos elementos no array
  - Para determinar se o valor não se encontra no array
    - O programa deve comparar a chave de pesquisa com todos os elementos no array
- Busca linear é uma solução eficiente em arrays menores ou arrays não classificados

Décimo Segundo Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 12
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int linearSearch(const int [], int, int);

int main() {
    const int arraySize = 100;
    int a [arraySize];
    int searchKey;

    for (int i = 0; i < arraySize; i++)
        a [i] = 2 * i;

    cout << "Entre com o elemento a ser buscado: ";
    cin >> searchKey;

    int element = linearSearch(a, searchKey, arraySize);

    if (element != -1)
        cout << "Elemento encontrado na posicao: " << element << endl;
    else
        cout << "Elemento nao encontrado." << endl;

    return 0;
}
```

Décimo Segundo Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 12
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int linearSearch(const int [], int, int);

int main() {
    const int arraySize = 100;
    int a [arraySize];
    int searchKey;

    for (int i = 0; i < arraySize; i++)
        a [i] = 2 * i;

    cout << "Entre com o elemento a ser buscado: ";
    cin >> searchKey;

    int element = linearSearch(a, searchKey, arraySize);

    if (element != -1)
        cout << "Elemento encontrado na posicao: " << element << endl;
    else
        cout << "Elemento nao encontrado." << endl;

    return 0;
}
```

A função recebe o array, o valor chave e o tamanho do array como argumentos

A função retorna o a posição do elemento encontrado ou -1 se não encontrar

Décimo Segundo Exemplo Usando Array em C++

```
int linearSearch(const int array[], int key, int sizeOfArray) {
    for (int i = 0; i < sizeOfArray; i++) {
        if (array[i] == key)
            return i;
    }
    return -1;
}
```


Décimo Segundo Exemplo Usando Array em C++

Pesquisa em todo o array

```
int linearSearch(const int array[], int key, int sizeOfArray) {
    for (int i = 0; i < sizeOfArray; i++) {
        if (array[i] == key)
            return i;
    }
    return -1;
}
```


Décimo Segundo Exemplo Usando Array em C++

```
int linearSearch(const int array[], int key, int sizeOfArray) {
    for (int i = 0; i < sizeOfArray; i++) {
        if (array[i] == key)
            return i;
    }
    return -1;
}
```

```
shell>$ g++ exemplo.cpp –o ex12
shell>$./ex1
Entre com o elemento a ser buscado: 4
Elemento encontrado na posicao: 2
shell>$
```

Classificação de Arrays por Inserção

- Classificando dados
  - Ordenamento de elementos de um array
  - Uma das aplicações mais importantes da computação
    - Praticamente toda organização tem de classificar algum tipo de dado

Classificação de Arrays por Inserção

- Classificação por inserção
  - Simples, mas ineficaz
  - A primeira iteração pega o segundo elemento
    - Se for menor que o primeiro elemento, troca-o por este
  - A segunda iteração examina o terceiro elemento
    - Insere-o na posição correta com relação aos dois primeiros elementos
  - ...
  - Na enésima iteração desse algoritmo, os primeiros i elementos no array original serão classificados

Décimo Terceiro Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 13
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

int *insertionSort(int [], int);

int main() {
    const int arraySize = 10;
    int unsorted [arraySize] = {34, 56, 4, 10, 77, 51, 93, 30, 5, 52};
    int *sorted;

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

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

    sorted  = insertionSort(unsorted, arraySize);

    cout << "\nSorted array:" << endl;

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

    return 0;
}
```

Décimo Terceiro Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 13
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

int *insertionSort(int [], int);

int main() {
    const int arraySize = 10;
    int unsorted [arraySize] = {34, 56, 4, 10, 77, 51, 93, 30, 5, 52};
    int *sorted;

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

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

    sorted  = insertionSort(unsorted, arraySize);

    cout << "\nSorted array:" << endl;

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

    return 0;
}
```

Função que retorna a referência para o array ordenado

Décimo Terceiro Exemplo Usando Array em C++

```
int *insertionSort(int array [], int sizeOfArray) {
    int insert;
    for (int next = 1; next < sizeOfArray; 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;
    }
    return array;
}
```


Décimo Terceiro Exemplo Usando Array em C++

```
int *insertionSort(int array [], int sizeOfArray) {
    int insert;
    for (int next = 1; next < sizeOfArray; 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;
    }
    return array;
}
```

Para cada elemento do array, a localização segundo a ordenação deve ser encontrada e o elemento deve ser posicionado nessa posição

Décimo Terceiro Exemplo Usando Array em C++

```
shell>$ g++ exemplo.cpp –o ex13
shell>$ ./ex13
Unsorted array:
  34  56   4  10  77  51  93  30   5  52

Sorted array:
   4   5  10  30  34  51  52  56  77  93
shell>$
```

```
            array [moveItem] = insert;
        }
        return array;
    }
```


Décimo Terceiro Exemplo Usando Array em C++

```
shell>$ g++ exemplo.cpp –o ex13
shell>$ ./ex13
Unsorted array:
  34  56   4  10  77  51  93  30   5  52

Sorted array:
   4   5  10  30  34  51  52  56  77  93
shell>$
```

```
            array [moveItem] = insert;
        }
        return array;
    }
```

Foi necessário a criação de um array ordenado?

Décimo Terceiro Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 13
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

int *insertionSort(int [], int);

int main() {
    const int arraySize = 10;
    int unsorted [arraySize] = {34, 56, 4, 10, 77, 51, 93, 30, 5, 52};
    int *sorted;

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

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

    sorted  = insertionSort(unsorted, arraySize);

    cout << "\nSorted array:" << endl;

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

    return 0;
}
```

Aqui poderia ser o próprio array **unsorted** porque a passagem de parâmetro é por referência

Décimo Terceiro Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 13
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

using namespace std;

int *insertionSort(int [], int);

int main() {
    const int arraySize = 10;
    int unsorted [arraySize] = {34, 56, 4, 10, 77, 51, 93, 30, 5, 52};
    int *sorted;

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

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

    sorted  = insertionSort(unsorted, arraySize);

    return 0;
}
```

Aqui poderia ser o próprio array **unsorted** porque a passagem de parâmetro é por referência

Como ficaria usando templates?

Décimo Terceiro Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 13 c/ template-sort.h
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include "template-sort.h"

using namespace std;

int main() {
    const int arraySize = 10;
    int unsorted [arraySize] = {34, 56, 4, 10, 77, 51, 93, 30, 5, 52};
    int *sorted;

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

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

    sorted  = insertionSort(unsorted, arraySize);

    cout << "\nSorted array:" << endl;

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

    return 0;
}
```


Décimo Terceiro Exemplo Usando Array em C++

```
template <class X>
X *insertionSort(X array [], int sizeOfArray) {
    X insert;
    for (int next = 1; next < sizeOfArray; 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;
    }
    return array;
}
```

Arrays Multidimensionais

- Arrays multidimensionais com duas dimensões
  - Denominados arrays bidimensionais ou arrays 2-D.
  - Representam tabelas de valores com linhas e colunas.
  - Os elementos são referenciados com dois subscritos (`[ x ][ y ]`).
  - Em geral, um array com *m* linhas e *n* colunas é chamado de array *m* por *n*.
- Os arrays multidimensionais podem ter mais de duas dimensões

Arrays Multidimensionais

- Declarando e inicializando arrays bidimensionais
  - Declarando um array bidimensional `b`
    - `int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };`
      - `1` e `2` inicializam `b[ 0 ][ 0 ]` e `b[ 0 ][ 1 ]`
      - `3` e `4` inicializam `b[ 1 ][ 0 ]` e `b[ 1 ][ 1 ]`
    - `int b[ 2 ][ 2 ] = { { 1 }, { 3, 4 } };`
      - A linha `0` contém valores `1` e `0` (implicitamente incializados em zero).
      - A linha `1` contém os valores `3` e `4`

Arrays Multidimensionais

	Coluna 1	Coluna 2	Coluna 3	Coluna 4
Linha 1	a [0][0]	a [0][1]	a [0][2]	a [0][3]
Linha 2	a [1][0]	a [1][1]	a [1][2]	a [1][3]
Linha 3	a [2][0]	a [2][1]	a [2][2]	a [2][3]

Décimo Quarto Exemplo Usando Array em C++

```
/*
 * Aula 7 - Exemplo 14
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void printArray (const int [][3]);

int main() {
    int array1 [2][3] = {{1, 2, 3}, {4, 5, 6}};
    int array2 [2][3] = {1, 2, 3, 4, 5};
    int array3 [2][3] = {{1, 2}, {4}};

    cout << "Os valores do array1 por linha sao: " << endl;
    printArray(array1);

    cout << "Os valores do array2 por linha sao: " << endl;
    printArray(array2);

    cout << "Os valores do array3 por linha sao: " << endl;
    printArray(array3);

    return 0;
}
```


Décimo Quarto Exemplo Usando Array em C++

Uso de inicializadores de array

```
/*
 * Aula 7 - Exemplo 14
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void printArray (const int [][3]);

int main() {
    int array1 [2][3] = {{1, 2, 3}, {4, 5, 6}};
    int array2 [2][3] = {1, 2, 3, 4, 5};
    int array3 [2][3] = {{1, 2}, {4}};

    cout << "Os valores do array1 por linha sao: " << endl;
    printArray(array1);

    cout << "Os valores do array2 por linha sao: " << endl;
    printArray(array2);

    cout << "Os valores do array3 por linha sao: " << endl;
    printArray(array3);

    return 0;
}
```


Décimo Quarto Exemplo Usando Array em C++

```
void printArray(const int a [][3]) {
     for (int i = 0; i < 2; i++) {
          for (int j = 0; j < 3; j++)
               cout << a [i][j] << ' ';
     cout << endl;
     }
}
```

Décimo Quarto Exemplo Usando Array em C++

```
void printArray(const int a [][3]) {
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++)
            cout << a [i][j] << ' ';
    cout << endl;
    }
}
```

Uso de loops aninhados para imprimir um array

Décimo Quarto Exemplo Usando Array em C++

```
void printArray(const int a [][3]) {
    for (int i = 0; i < 2; i++) {
```

```
shell>$ g++ exemplo.cpp –o ex14
shell>$ ./ex14
Os valores do array1 por linha sao:
1  2  3
4  5  6
Os valores do array2 por linha sao:
1  2  3
4  5  0
Os valores do array3 por linha sao:
1  2  0
4  0  0
shell>$
```


Arrays Multidimensionais

- Parâmetros de array multidimensional
  - O tamanho da primeira dimensão não é necessário
    - É igual ao array unidimensional
  - O tamanho das dimensões subseqüentes é necessário
    - O compilador tem que saber quantos elementos deve pular para mover-se para o segundo elemento na primeira dimensão
  - Ex.: `void printArray( const int a[][ 3 ] );`
    - A função pulará 3 elementos da linha 0 para acessar os elementos da linha 1 (`a[ 1 ][ x ]`)

Arrays Multidimensionais

- Manipulações de array multidimensional
  - Comumente executadas com instruções `for`
    - Exemplo
      - Modificar todos os elementos em uma linha

```
» for ( int col = 0; col < 4; col++ )
      a[ 2 ][ col ] = 0;
```

    - Exemplo
      - Total de todos os elementos

```
» total = 0;
  for ( row = 0; row < 3; row++ )
     for ( col = 0; col < 4; col++ )
        total += a[ row ][ col ];
```


Estudo de Caso: Classe `GradeBook`

- Classe `GradeBook`
  - Array unidimensional
    - Armazena as notas do alunos em um único exame
  - Array bidimensional
    - Armazena várias notas de um único aluno e de vários alunos da classe como um todo.
      - Cada linha representa as notas de um único aluno
      - Cada coluna representa todas as notas tiradas pelos alunos em um determinado exame

Décimo Quinto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 15
 * Arquivo: GradeBookCap7Ex15.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;
            // Constante - Número de alunos que fizeram o teste
            const static int tests = 3;
            // Construtor inicializa courseName com a string-argumento
            GradeBook(string, const int [][tests]);
            // Função que configura 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(const int [], const int);
            // Função para exibir os conceitos
            void displayGrades();
            // Função para retornar a menor nota dos alunos
            int getMinimum();
```

Décimo Quinto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 15
 * Arquivo: GradeBookCap7Ex15.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;
        // Constante - Número de alunos que fizeram o teste
        const static int tests = 3;
        // Construtor inicializa courseName com a string-argumento
        GradeBook(string, const int [][tests]);
        // Função que configura 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(const int [], const int);
        // Função para exibir os conceitos
        void displayGrades();
        // Função para retornar a menor nota dos alunos
        int getMinimum();
```

O construtor **GradeBook** recebe um array bidimensional

Décimo Quinto Exemplo Usando Array em C++

```
        // Função para retornar a maior nota dos alunos
        int getMaximum();
        // Função para realizar funções nas notas
        void processGrades();
        // Função para exibir a distribuição de notas da turma
        void displayBarChart();
        void displayMessage();

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


Décimo Quinto Exemplo Usando Array em C++

```
        // Função para retornar a maior nota dos alunos
        int getMaximum();
        // Função para realizar funções nas notas
        void processGrades();
        // Função para exibir a distribuição de notas da turma
        void displayBarChart();
        void displayMessage();

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

Declara um array bidimensional para **grades**

Décimo Quinto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 15
 * Arquivo: GradeBookCap7Ex15.cpp
 * Autor: Miguel Campista
 */

#include <iostream>
#include <string>
#include "GradeBookCap7Ex15.h"

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

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


Décimo Quinto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 15
 * Arquivo: GradeBookCap7Ex15.cpp
 * Autor: Miguel Campista
 */

#include <iostream>
#include <string>
#include "GradeBookCap7Ex15.h"

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

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

Usa loops aninhados para copiar elementos de **gradesArray** para **grades**

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

Loop nas linhas e nas colunas de **grades** para encontrar a nota mínima

Décimo Quinto 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++) {
        for (int test = 0; test < tests; test++) {
            if (grades [i][test] > maximum)
                maximum = grades [i][test];
        }
    }
    return maximum;
}
// Função que calcula a média das notas da turma
double GradeBook::getAverage(const int setOfGrades [], const int grades) {
    int total = 0;

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

    return static_cast <double> (total)/grades;
}
// 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 maximo de 25 caracteres..." << endl <<
        "Nome limitado aos primeiros 25 caracteres: " << courseName <<
        endl;
    }
}
```

Décimo Quinto 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++) {
          for (int test = 0; test < tests; test++) {
              if (grades [i][test] > maximum)
                    maximum = grades [i][test];
          }
      }
      return maximum;
}
// Função que calcula a média das notas da turma
double GradeBook::getAverage(const int setOfGrades [], const int grades) {
      int total = 0;

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

      return static_cast <double> (total)/grades;
}
// 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 maximo de 25 caracteres..." << endl <<
            "Nome limitado aos primeiros 25 caracteres: " << courseName <<
            endl;
      }
}
```

Loop nas linhas e nas colunas de **grades** para encontrar a nota máxima

Loop para calcular a média de um dos testes

Décimo Quinto Exemplo Usando Array em C++

```
// Função que obtém 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 << "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;

          for (int test = 0; test < tests; test++) {
              cout << setw(8) << grades [i][test];
          }
          double average = getAverage(grades[i], tests);

          cout << setw(9) << setprecision(2) << fixed << average << endl;
      }
}
```


Décimo Quinto Exemplo Usando Array em C++

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

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

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

      }
}
```


Décimo Quinto Exemplo Usando Array em C++

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

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

      for (int i = 0; i < students; i++) {
          for (int test = 0; test < tests; test++)
              frequency [grades [i][test]/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 nas linhas e nas colunas de **grades** para calcular a distribuição das notas

Décimo Quinto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 15
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>
#include "GradeBookCap7Ex15.h" // Inclui a definição da classe

using namespace std;

int main() {
    // Array de notas dos alunos
    int gradesArray [GradeBook::students][GradeBook::tests] =
        {{87, 96, 70}, {68, 87, 90},
         {94, 100, 90}, {100, 81, 82},
         {83, 65, 85}, {78, 87, 65},
         {85, 75, 83}, {91, 94, 100},
         {76, 72, 84}, {87, 93, 73}};

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

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

    return 0;
}
```

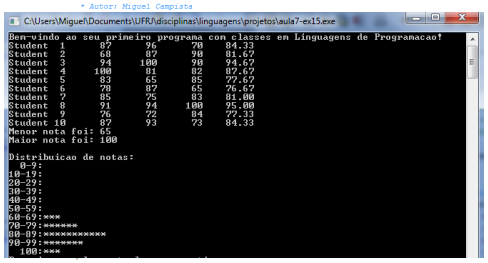

Décimo Quinto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 15
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>
#include "GradeBookCap7Ex15.h" // Inclui a definição da classe

using namespace std;

int main() {
    int gradesArray [GradeBook::students][GradeBook::tests] =
        {{87, 96, 70}, {68, 87, 90},
         {94, 100, 90}, {100, 81, 82},
         {83, 65, 85}, {78, 87, 65},
         {85, 75, 83}, {91, 94, 100},
         {76, 72, 84}, {87, 93, 73}};

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

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

    return 0;
}
```

Declaração do array 3 x 10

Décimo Quinto Exemplo Usando Array em C++

```
/*
 * Aula 7 -- Exemplo 15
 * Arquivo principal
 * Autor: Miguel Campista
```



Introdução ao 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) {
    ...
}
```

✗ Erro!

```
int a[10], b[10];
int b[10] = a;
```

✗ Erro!

Introdução ao 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`
  - Os objetos `vector` podem ser comparados por meio dos operadores de igualdade e relacionais
  - O operador de atribuição pode ser usado em `vector`s

Introdução ao Template `vector` da C++ *Standard Library*

- Elementos `vector` podem ser obtidos como um *lvalue* (valor à esquerda) não modificável ou um *lvalue* modificável
  - *lvalue* não modificável
    - Expressão que identifica um objeto na memória, mas não pode ser usada para modificar esse objeto
  - *lvalue* modificável
    - Expressão que identifica um objeto na memória, mas pode ser usada para modificar o objeto

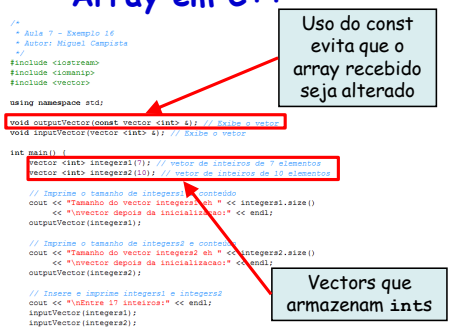
Introdução ao Template `vector` da C++ *Standard Library*

- 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

Décimo Sexto Exemplo Usando Array em C++

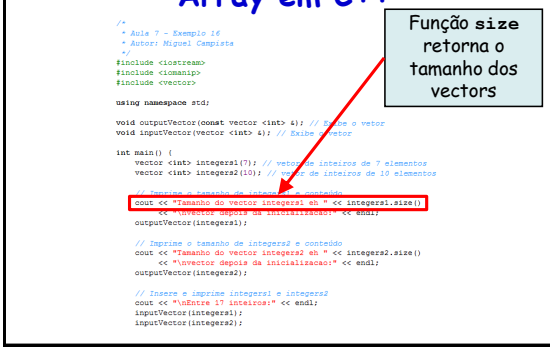
```
/*
 * Aula 7 - Exemplo 16
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include <vector>

using namespace std;

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

int main() {
    vector <int> integers1(7); // vetor de inteiros de 7 elementos
    vector <int> integers2(10); // vetor de inteiros de 10 elementos

    // Imprime o tamanho de integers1 e conteúdo
    cout << "Tamanho do vector integers1 eh " << integers1.size()
         << "\nvector depois da inicializacao:" << endl;
    outputVector(integers1);

    // Imprime o tamanho de integers2 e conteúdo
    cout << "Tamanho do vector integers2 eh " << integers2.size()
         << "\nvector depois da inicializacao:" << endl;
    outputVector(integers2);

    // Insere e imprime integers1 e integers2
    cout << "\nEntre 17 inteiros:" << endl;
    inputVector(integers1);
    inputVector(integers2);
```

Décimo Sexto Exemplo Usando Array em C++




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

using namespace std;

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

int main() {
   vector <int> integers1(7); // vetor de inteiros de 7 elementos
   vector <int> integers2(10); // vetor de inteiros de 10 elementos

   // Imprime o tamanho de integers1 e conteúdo
   cout << "Tamanho do vector integers1 eh " << integers1.size()
        << "\nvector depois da inicializacao:" << endl;
   outputVector(integers1);

   // Imprime o tamanho de integers2 e conteúdo
   cout << "Tamanho do vector integers2 eh " << integers2.size()
        << "\nvector depois da inicializacao:" << endl;
   outputVector(integers2);

   // Insere e imprime integers1 e integers2
   cout << "\nEntre 17 inteiros:" << endl;
   inputVector(integers1);
   inputVector(integers2);
```

Décimo Sexto Exemplo Usando Array em C++

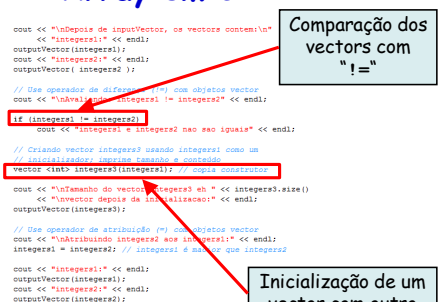



```
/*
 * Aula 7 - Exemplo 16
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include <vector>

using namespace std;

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

int main() {
   vector <int> integers1(7); // vetor de inteiros de 7 elementos
   vector <int> integers2(10); // vetor de inteiros de 10 elementos

   // Imprime o tamanho de integers1 e conteúdo
   cout << "Tamanho do vector integers1 eh " << integers1.size()
        << "\nvector depois da inicializacao:" << endl;
   outputVector(integers1);

   // Imprime o tamanho de integers2 e conteúdo
   cout << "Tamanho do vector integers2 eh " << integers2.size()
        << "\nvector depois da inicializacao:" << endl;
   outputVector(integers2);

   // Insere e imprime integers1 e integers2
   cout << "\nEntre 17 inteiros:" << endl;
   inputVector(integers1);
   inputVector(integers2);
```

Décimo Sexto Exemplo Usando Array em C++

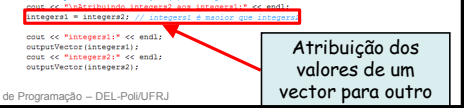
```
cout << "\nDepois de inputVector, os vectors contem:\n"
     << "integers1:" << endl;
outputVector(integers1);
cout << "integers2:" << endl;
outputVector( integers2 );

// Use operador de diferença (!=) com objetos vector
cout << "\nAvaliando: integers1 != integers2" << endl;

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

// Criando vector integers3 usando integers1 como um
// inicializador; imprime tamanho e conteúdo
vector <int> integers3(integers1); // copia construtor

cout << "\nTamanho do vector integers3 eh " << integers3.size()
     << "\nvector depois da inicializacao:" << endl;
outputVector(integers3);

// Use operador de atribuição (=) com objetos vector
cout << "\nAtribuindo integers2 aos integers1:" << endl;
integers1 = integers2; // integers1 é maoior que integers2

cout << "integers1:" << endl;
outputVector(integers1);
cout << "integers2:" << endl;
outputVector(integers2);
```


Décimo Sexto Exemplo Usando Array em C++




```
cout << "\nDepois de inputVector, os vectors contem:\n"
     << "integers1:" << endl;
outputVector(integers1);
cout << "integers2:" << endl;
outputVector( integers2 );

// Use operador de diferença (!=) com objetos vector
cout << "\nAvaliando: integers1 != integers2" << endl;

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

// Criando vector integers3 usando integers1 como um
// inicializador; imprime tamanho e conteúdo
vector <int> integers3(integers1); // copia construtor

cout << "\nTamanho do vector integers3 eh " << integers3.size()
     << "\nvector depois da inicializacao:" << endl;
outputVector(integers3);

// Use operador de atribuição (=) com objetos vector
cout << "\nAtribuindo integers2 aos integers1:" << endl;
integers1 = integers2; // integers1 é maoior que integers2

cout << "integers1:" << endl;
outputVector(integers1);
cout << "integers2:" << endl;
outputVector(integers2);
```


Décimo Sexto Exemplo Usando Array em C++




```
cout << "\nDepois de inputVector, os vectors contem:\n"
     << "integers1:" << endl;
outputVector(integers1);
cout << "integers2:" << endl;
outputVector( integers2 );

// Use operador de diferença (!=) com objetos vector
cout << "\nAvaliando: integers1 != integers2" << endl;

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

// Criando vector integers3 usando integers1 como um
// inicializador; imprime tamanho e conteúdo
vector <int> integers3(integers1); // copia construtor

cout << "\nTamanho do vector integers3 eh " << integers3.size()
     << "\nvector depois da inicializacao:" << endl;
outputVector(integers3);

// Use operador de atribuição (=) com objetos vector
cout << "\nAtribuindo integers2 aos integers1:" << endl;
integers1 = integers2; // integers1 é maoior que integers2

cout << "integers1:" << endl;
outputVector(integers1);
cout << "integers2:" << endl;
outputVector(integers2);
```


Décimo Sexto Exemplo Usando Array em C++

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

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

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

// Use colchetes para criar lvalue
cout << "\n\nAtribuindo 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 integers1.at(15)" << endl;
integers1.at(15) = 1000; // ERROR: fora do intervalo

return 0;
}
```

Décimo Sexto Exemplo Usando Array em C++

Comparação dos vectors com "=="

Exibindo um elemento de um vector

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

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

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

// Use colchetes para criar lvalue
cout << "\n\nAtribuindo 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 integers1.at(15)" << endl;
integers1.at(15) = 1000; // ERROR: fora do intervalo

return 0;
}
```


Décimo Sexto Exemplo Usando Array em C++

Atualizando o valor

Tentativa de atualizar um valor fora do intervalo

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

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

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

// Use colchetes para criar lvalue
cout << "\n\nAtribuindo 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 integers1.at(15)" << endl;
integers1.at(15) = 1000; // ERROR: fora do intervalo

return 0;
}
```


Décimo Sexto Exemplo Usando Array em C++

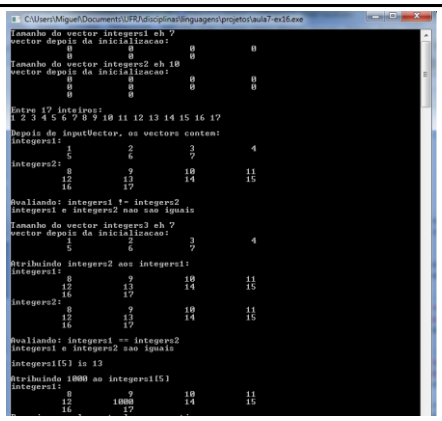
Exibe os elementos do array

Inserção de elementos com o `cin`

```
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 da 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];
}
```


Décimo Sexto 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 da 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];
}
```

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`**

Décimo Sétimo 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;
}
```


Décimo Sétimo Exemplo Usando Array em C++11

```
itaqua:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex17.cpp -o a
itaqua:~/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;
}
```


Décimo Oitavo 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;
}
```


Décimo Oitavo 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;
}
```


Décimo Oitavo Exemplo Usando Array em C++11

```
itaqua:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex18.cpp -o a
itaqua:~/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;
}
```


Geração de Números Aleatórios em C++11

- Função **rand** é considerada insegura
  - É possível conhecer a priori a sequência
- C++11 oferece:
  - Classes para geração de sequências pseudo-aleatórias
  - Classes para distribuição uniforme de números inteiros e outras distribuições

Décimo Nono Exemplo Usando Array em C++11

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

using namespace std;

int main () {
        // Cria um objeto para geração de números pseudo-aleatórios
        default_random_engine engine (static_cast <unsigned int> (time (0)));
        // Cria um ojeto para distribuição uniforme de inteiros
        uniform_int_distribution <unsigned int> randomInt (1, 6);

        array <unsigned int, 7> frequency {}; // Cria um array com 0s

        for (unsigned int i=0; i < 6000000; i++) {
                frequency [randomInt (engine)]++;
        }

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

        for (size_t i=1; i < frequency.size (); i++) {
                cout << setw (7) << i << setw(10) << frequency [i] << endl;
        }

        return 0;
}
```


Décimo Nono Exemplo Usando Array em C++11

```
itagua:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex19.cpp -o a
itagua:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> ./a
Element     Value
      1   1001051
      2    999236
      3   1000288
      4    999379
      5    999248
      6   1000798
```

```
int main () {
        // Cria um objeto para geração de números pseudo-aleatórios
        default_random_engine engine (static_cast <unsigned int> (time (0)));
        // Cria um ojeto para distribuição uniforme de inteiros
        uniform_int_distribution <unsigned int> randomInt (1, 6);

        array <unsigned int, 7> frequency {}; // Cria um array com 0s

        for (unsigned int i=0; i < 6000000; i++) {
                frequency [randomInt (engine)]++;
        }

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

        for (size_t i=1; i < frequency.size (); i++) {
                cout << setw (7) << i << setw(10) << frequency [i] << endl;
        }

        return 0;
}
```


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

Vigésimo 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;
}
```


Vigésimo Exemplo Usando Array em C++11

```
itagua:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex20.cpp -o a
itagua:~/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;
}
```


Vigésimo 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;
}
```

Vigésimo Exemplo Usando Array em C++11

```
itaqua:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex20.cpp -o a
itaqua:~/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)
```


Vigésimo 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;
}
```


Vigésimo Exemplo Usando Array em C++11

```
itaqua:~/disciplinas/linguagens/aulas/2017/programasC++11-C++14> g++ -Wall -std=c++11 aula7-ex20.cpp -o a
itaqua:~/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;
}
```


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 [] = {34, 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 << "\nSorted 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;
    }
}
```


Leitura Recomendada

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