Vetores e Matrizes

- Nesta aula veremos estruturas de dados homogêneas: vetores (ou *arrays*) e matrizes
- Essas estruturas de dados são homogêneas pois todos os elementos que as compõem têm o mesmo tipo de dado associado

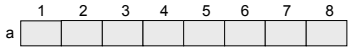
José Augusto Baranauskas
Departamento de Física e Matemática – FFCLRP-USP
Sala 226 – Bloco P2 – Fone (16) 3602-4361
E-mail: augusto@ffclrp.usp.br
URL: http://www.fmrp.usp.br/augusto

Constantes Simbólicas

- Freqüentemente, uma mesma constante é utilizada em várias partes de um programa
- Para facilitar o entendimento, pode-se declarar uma constante simbólica, ou seja, uma constante com um nome associado
- Em C++, a declaração de constantes simbólicas é efetuada através do modificador *const*. Por exemplo:
  - `const int Max = 100;`
  - `const int N = 8;`
- Não é permitido alterar o valor de uma constante!

Vetor (*Array*)

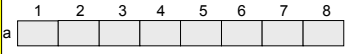
- Um vetor é uma variável composta homogênea **unidimensional**, formada por uma seqüência de variáveis, todas do **mesmo tipo de dados**, com o mesmo identificador (mesmo nome) e alocadas **seqüencialmente** na memória
- Uma vez que as variáveis que compõem o vetor têm o mesmo nome, o que distingue cada um delas é um **índice**, que referencia sua localização dentro da estrutura
- Por exemplo, o seguinte vetor **a** é capaz de armazenar 8 inteiros



Vetor (*Array*)

- Um vetor é uma variável composta homogênea **unidimensional**, formada por uma seqüência de variáveis, todas do **mesmo tipo de dados**, com o mesmo identificador (mesmo nome) e alocadas **seqüencialmente** na memória
- Uma vez que as variáveis que compõem o vetor têm o mesmo nome, o que distingue cada um delas é um **índice**, que referencia sua localização dentro da estrutura
- Por exemplo, o seguinte vetor **a** é capaz de armazenar 8 inteiros

Unidimensional pois possui apenas uma única dimensão



Vetor (*Array*)

- Um vetor é uma variável composta homogênea **unidimensional**, formada por uma seqüência de variáveis, todas do **mesmo tipo de dados**, com o mesmo identificador (mesmo nome) e alocadas **seqüencialmente** na memória
- Uma vez que as variáveis que compõem o vetor têm o mesmo nome, o que distingue cada um delas é um **índice**, que referencia sua localização dentro da estrutura
- Por exemplo, o seguinte vetor **a** é capaz de armazenar 8 inteiros

Mesmo tipo de dados de cada elemento (posição ou entrada) do vetor. Neste exemplo, **a** pode armazenar apenas inteiros

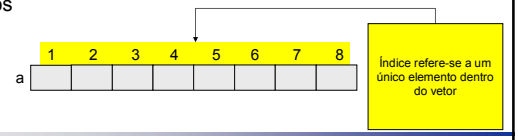
Vetor (*Array*)

- Um vetor é uma variável composta homogênea **unidimensional**, formada por uma seqüência de variáveis, todas do **mesmo tipo de dados**, com o mesmo identificador (mesmo nome) e alocadas **seqüencialmente** na memória
- Uma vez que as variáveis que compõem o vetor têm o mesmo nome, o que distingue cada um delas é um **índice**, que referencia sua localização dentro da estrutura
- Por exemplo, o seguinte vetor **a** é capaz de armazenar 8 inteiros

Seqüencial pois todos os elementos encontram-se consecutivos um ao outro (não há "buracos" entre os elementos)

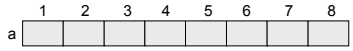
Vetor (*Array*)

- Um vetor é uma variável composta homogênea **unidimensional**, formada por uma seqüência de variáveis, todas do **mesmo tipo de dados**, com o mesmo identificador (mesmo nome) e alocadas **seqüencialmente** na memória
- Uma vez que as variáveis que compõem o vetor têm o mesmo nome, o que distingue cada um delas é um **índice**, que referencia sua localização dentro da estrutura
- Por exemplo, o seguinte vetor **a** é capaz de armazenar 8 inteiros

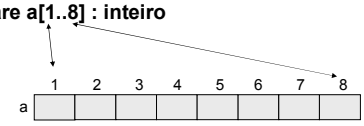


Vetor (*Array*)

- Por exemplo, o seguinte vetor **a** é capaz de armazenar 8 inteiros

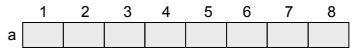


- O vetor pode ser declarado em pseudo-código da seguinte forma:
- **declare a[1..8] : inteiro**

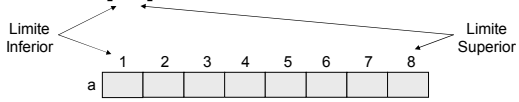


Vetor (*Array*)

- Por exemplo, o seguinte vetor **a** é capaz de armazenar 8 inteiros

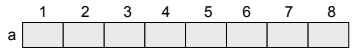


- O vetor pode ser declarado em pseudo-código da seguinte forma:
- **declare a[1..8] : inteiro**

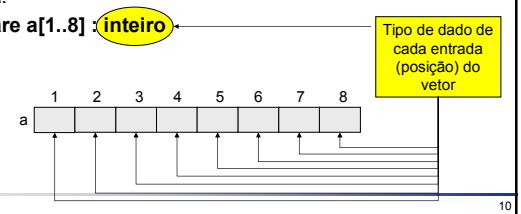


Vetor (*Array*)

- Por exemplo, o seguinte vetor **a** é capaz de armazenar 8 inteiros



- O vetor pode ser declarado em pseudo-código da seguinte forma:
- **declare a[1..8] : inteiro**



Vetor (*Array*)

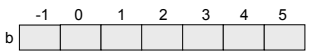
- A regra geral para declaração de um vetor é
  - **declare a[Li..Ls] : Tipo**

- Onde:
  - **Li**: Limite inferior que o índice pode assumir
  - **Ls**: Limite superior que o índice pode assumir
  - **Tipo**: é um tipo de dado primitivo, por exemplo: inteiro, real, caracter, booleano, *string* bem como um tipo de dado composto (estruturas compostas serão vistas nas próximas aulas)

Vetor (*Array*): Exemplos

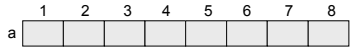
- **declare x[2..9] : inteiro**

- **declare b[-1..5] : real**



Número de Elementos de um Vetor

- Exemplo 1
  - **declare a[1..8] : inteiro**

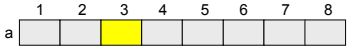


  - Observa-se que o vetor possui 8 – 1 + 1 = 8 elementos
- Exemplo 2
  - **declare b[-1..5] : real**

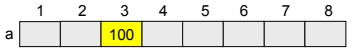
  - Observa-se que o vetor possui 5 - (-1) +1 = 7 elementos
- No caso genérico:
  - **declare a[Li..Ls] : Tipo**
  - O número de elementos de **a** é (Ls – Li + 1)
  - **Li** e **Ls** devem ser constantes

Manipulação de Vetores

- Um elemento do vetor pode ser referenciado com o nome do vetor, seguido do seu índice correspondente colocado entre colchetes
- Por exemplo, para referenciar o 3º elemento do vetor **a**, denotamos por a[3]

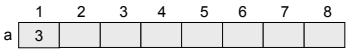


- Por exemplo, se desejamos atribuir o valor 100 ao 3º elemento de **a**, basta utilizar o comando de atribuição:
  - a[3] ← 100



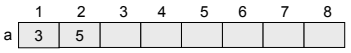
Manipulação de Vetores

- Da mesma forma que podemos atribuir um valor a um elemento de um vetor, ele também pode ser lido ou escrito
- Por exemplo
  - Leia(a[3])
  - Escreva(a[3])
- Assim sendo, cada elemento do vetor se comporta como uma variável qualquer e todas as operações que vimos sobre variáveis são válidas para um elemento do vetor
- Exemplo:
  - a[1] ← 3



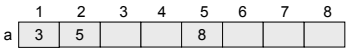
Manipulação de Vetores

- Da mesma forma que podemos atribuir um valor a um elemento de um vetor, ele também pode ser lido ou escrito
- Por exemplo
  - Leia(a[3])
  - Escreva(a[3])
- Assim sendo, cada elemento do vetor se comporta como uma variável qualquer e todas as operações que vimos sobre variáveis são válidas para um elemento do vetor
- Exemplo:
  - a[1] ← 3
  - a[2] ← 5



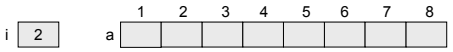
Manipulação de Vetores

- Da mesma forma que podemos atribuir um valor a um elemento de um vetor, ele também pode ser lido ou escrito
- Por exemplo
  - Leia(a[3])
  - Escreva(a[3])
- Assim sendo, cada elemento do vetor se comporta como uma variável qualquer e todas as operações que vimos sobre variáveis são válidas para um elemento do vetor
- Exemplo:
  - a[1] ← 3
  - a[2] ← 5
  - a[5] ← a[2] + a[1]



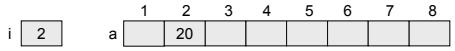
Manipulação de Vetores

- O índice de um vetor não precisa ser necessariamente uma constante
- O índice pode ser uma variável ou uma expressão escalar
- Exemplo
  - i ← 2



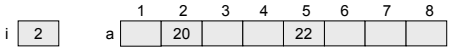
Manipulação de Vetores

- O índice de um vetor não precisa ser necessariamente uma constante
- O índice pode ser uma variável ou uma expressão escalar
- Exemplo
  - i ← 2
  - a[i] ← 20



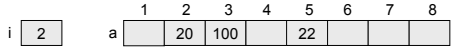
Manipulação de Vetores

- O índice de um vetor não precisa ser necessariamente uma constante
- O índice pode ser uma variável ou uma expressão escalar
- Exemplo
  - i ← 2
  - a[i] ← 20
  - a[5] ← a[i] + 2



Manipulação de Vetores

- O índice de um vetor não precisa ser necessariamente uma constante
- O índice pode ser uma variável ou uma expressão escalar
- Exemplo
  - i ← 2
  - a[i] ← 20
  - a[5] ← a[i] + 2
  - a[i+1] ← 100



Exercício

Sendo as variáveis **a** e **b** e vetor **x** iguais a:

			1	2	3	4	5	6	7	8	9	10
a	2	b	4	x	2	6	8	3	10	9	1	21	33	14

Escreva o valor correspondente de:

a) x[a + 1]
b) x[a + 2]
c) x[a + 3]
d) x[a * 4]
e) x[a * 1]
f) x[a * 2]
g) x[a * 3]
h) x[x[a + b]]
i) x[a + b]
j) x[8 – x[2]]
k) x[x[4]]
l) x[x[x[7]]]
m) x[x[1] * x[4]]
n) x[x[a + 4]]

Solução

Sendo as variáveis **a** e **b** e vetor **x** iguais a:

			1	2	3	4	5	6	7	8	9	10
a	2	b	4	x	2	6	8	3	10	9	1	21	33	14

Escreva o valor correspondente de:

a) x[a + 1] = x[2 + 1] = x[3] = 8
b) x[a + 2] = x[2 + 2] = x[4] = 3
c) x[a + 3] = x[2 + 3] = x[5] = 10
d) x[a * 4] = x[2 * 4] = x[8] = 21
e) x[a * 1] = x[2 * 1] = x[2] = 6
f) x[a * 2] = x[2 * 2] = x[4] = 3
g) x[a * 3] = x[2 * 3] = x[6] = 9
h) x[x[a + b]] = x[x[2 + 4]] = x[x[6]] = x[9] = 33
i) x[a + b] = x[2 + 4] = x[6] = 9
j) x[8 – x[2]] = x[8 – 6] = x[2] = 6
k) x[x[4]] = x[3] = 8
l) x[x[x[7]]] = x[x[1]] = x[2] = 6
m) x[x[1] * x[4]] = x[2 * 3] = x[6] = 9
n) x[x[a + 4]] = x[x[2 + 4]] = x[x[6]] = x[9] = 33

Entrada de Vetores

- Em pseudo-código, é possível expressar que um vetor deve ser totalmente lido através do comando Leia
- Por exemplo

```
declare a[1..8] : inteiro
Leia(a)
```

- Entretanto, as linguagens de programação em geral não possuem um comando deste tipo
- Assim, torna-se necessário ler um vetor através da leitura individual de cada um dos seus elementos
- Por exemplo

```
declare a[1..8] : inteiro
Para i ← 1 até 8 Faça
  Leia(a[i])
Fim Para
```

Saída de Vetores

- Em pseudo-código, é possível expressar que um vetor deve ser totalmente impresso através do comando Escreva
- Por exemplo

```
declare a[1..8] : inteiro
Escreva(a)
```

- Entretanto, as linguagens de programação em geral não possuem um comando deste tipo
- Assim, torna-se necessário imprimir um vetor através da impressão individual de cada um dos seus elementos
- Por exemplo

```
declare a[1..8] : inteiro
Para i ← 1 até 8 Faça
  Escreva(a[i])
Fim Para
```


Exemplo

```
Algoritmo NotasClasse. Dada uma classe de N >= 1 alunos, este algoritmo lê suas
   notas, calcula a média da classe e imprime quais os alunos possuem nota acima
   da média
Início
  declare soma, media, nota[1..100] : real
  declare i, N : inteiro

  Escreva("Numero de alunos? ")
  Leia(N)
  Para i ← 1 até N Faça            // Ler notas dos alunos
    Escreva("Nota do aluno ",i,"? ")
    Leia(nota[i])
  Fim Para
  soma ← 0.0                       // Calcular a média da classe
  Para i ← 1 até N Faça
    soma ← soma + nota[i]
  Fim Para
  media = soma / N
  Escreva("Media = ",media)
  Para i ← 1 até N Faça            // Alunos com nota > media
    Se (nota[i] > media) Então
      Escreva("Aluno ",i," Nota = ",nota[i])
    Fim Se
  Fim Para
Fim
```


Exemplo em C++

```
/* Algoritmo NotasClasse. Dada uma classe de N >= 1 alunos, este algoritmo lê
   suas notas, calcula a média da classe e imprime quais os alunos possuem nota
   acima da média
*/
#include <iostream>
using namespace std;
int main()
{ float soma, media, nota[101];
  int i, N;

  cout << "Numero de alunos? ";
  cin >> N;
  for(i=1; i<=N; i++)              // Ler notas dos alunos
  { cout << "Nota do aluno " << i << "? ";
    cin >> nota[i];
  }
  soma = 0.0;                      // Calcular a média da classe
  for(i=1; i<=N; i++)
    soma = soma + nota[i];
  media = soma / N;
  cout << "Media = " << media << endl;
  for(i=1; i<=N; i++)              // Alunos com nota > media
    if(nota[i] > media)
      cout << "Aluno " << i << " Nota = " << nota[i] << endl;
  return 0;
}
```


Vetores em C/C++

- Como já visto, em pseudo-código, um vetor, é declarado como:
  - **declare a[Li..Ls] : Tipo**
  - O número de elementos de **a** é (Ls – Li + 1)
- Em C/C++, há uma restrição quanto ao limite inferior, que sempre deve ser zero (Li = 0) e não pode ser explicitado na declaração do vetor. Portanto
  - **declare a[Li..Ls] : Tipo**
- Fica restrito a
  - **declare a[0..Ls-Li+1] : Tipo**
- Assim, somente é possível declarar um vetor em C/C++ com o mesmo número de elementos que em pseudo-código
- Exemplo em pseudo-código / Exemplo em C/C++
  - declare a[1..5] : inteiro — int a[5];
  - declare b[0..10] : real — float b[11];
  - declare c[-1..10] : inteiro — int c[12];

Vetores em C/C++

- Exemplo em pseudo-código / Exemplo em C/C++
  - declare a[1..5] : inteiro — int a[5];
  - declare b[0..10] : real — float b[11];
  - declare c[-1..10] : inteiro — int c[12];
- Entretanto, note que o intervalo em que o índice varia pode ser diferente!
  - declare a[1..5] : inteiro — int a[5];
  - a[1],a[2],a[3],a[4],a[5] — a[0],a[1],a[2],a[3],a[4]
  - declare b[0..10] : real — float b[11];
  - b[0],b[1],b[2],...b[9],b[10] — b[0],b[1],b[2],...,b[8],b[9],b[10]
  - declare c[-1..10] : inteiro — int c[12];
  - c[-1],c[0],c[1],c[2],...,c[9],c[10] — c[0],c[1],c[2],...,c[9],c[10],c[11]

Vetores em C/C++

- Assim, para declarar um vetor **x** de inteiros em C/C++ que possa armazenar 10 elementos, com seu índice inicial começando de zero, declara-se:
  - int x[10];
- Com isso, os índices admitidos para o vetor variam
  - de 0 até 9 (x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7], x[8], x[9])
- Para declarar um vetor **x** de inteiros em C/C++ que possa armazenar 10 elementos, com seu índice inicial começando de um, declara-se:
  - int x[11];
- Com isso, os índices admitidos para o vetor variam
  - de 0 até 10 (x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7], x[8], x[9],x[10])
  - O elemento x[0] sempre existe em um vetor em C/C++
- De maneira geral, um vetor declarado como
  - int x[K];
- Onde K é uma **constante**, tem índices admitidos no intervalo
  - de 0 até K-1 (x[0], x[1], ..., x[K-2], x[K-1])

Vetores em C/C++

- Assumindo a declaração de um vetor **x** de inteiros em C/C++ que possa armazenar 10 elementos, com seu índice inicial começando de zero:
  - int x[10];
- Com isso, os índices admitidos para o vetor variam
  - de 0 até 9 (x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7], x[8], x[9])
  - Entretanto nada impede o programador de escrever:
    - x[10] = 100;
    - x[70] = 20;
- Ou seja, o compilador C/C++ não verifica se o índice que você usou está dentro dos limites válidos
- Se o programador não tiver atenção com os limites de validade para os índices ele corre o risco de ter variáveis sobre-escritas, podendo obter resultados inesperados

Exemplo 1

```
Algoritmo Maior. Para um vetor A de n números reais, determina o maior
   elemento de A.
Início
  declare i,n : inteiro
  declare maior, A[1..100] : real

  Escreva("Numero de elementos? ")
  Leia(n)
  Para i ← 1 até n Faça       // Ler elementos do vetor
    Escreva("A[",i,"]=")
    Leia(A[i])
  Fim Para
  maior ← A[1]        // Assumir 1o elemento o maior
  // Procurar maior do 2 elemento em diante
  Para i ← 2 até n Faça
    Se (A[i] > maior) Então
      maior ← A[i]
    Fim Se
  Fim Para
  Escreva("Maior elemento = ",maior)
Fim
```


Exemplo 1 em C++

```
/* Algoritmo Maior. Para um vetor A de n números reais, determina o maior
   elemento de A.
*/
#include <iostream>
using namespace std;
int main()
{ int i,n;
  float maior, A[101];

  cout << "Numero de elementos? ";
  cin >> n;
  for(i=1; i<=n; i++)              // Ler elementos do vetor
  { cout << "A[" << i << "]=";
    cin >> A[i];
  }
  maior = A[1];           // Assumir 1o elemento o maior
  // Procurar maior do 2 elemento em diante
  for(i=2; i<=n; i++)
    if(A[i] > maior)
       maior = A[i];
  cout << "Maior elemento = " << maior;
  return 0;
}
```


Exemplo 2

```
Algoritmo Maior2. Para um vetor A de n números reais, determina o maior elemento
   de A bem como seu indice.
Início
  declare indice_maior,i,n : inteiro
  declare maior, A[1..100] : real

  Escreva("Numero de elementos? ")
  Leia(n)
  Para i ← 1 até n Faça  // Ler elementos do vetor
    Escreva("A[",i,"]=")
    Leia(A[i])
  Fim Para
  indice_maior ← 1              // Assumir 1o elemento o maior
  // Procurar maior do 2 elemento em diante
  Para i ← 2 até N Faça
    Se (A[i] > A[indice_maior]) Então
      indice_maior ← i
    Fim Se
  Fim Para
  maior ← A[indice_maior]
  Escreva("Maior elemento = ",maior," com indice = ",indice_maior)
Fim
```


Exemplo 2 em C++

```
/* Algoritmo Maior2. Para um vetor A de n números reais, determina o maior
   elemento de A bem como seu indice.
*/
#include <iostream>
using namespace std;
int main()
{ int indice_maior,i,n;
  float maior, A[101];

  cout << "Numero de elementos? ";
  cin >> n;
  for(i=1; i<=n; i++)     // Ler elementos do vetor
  { cout << "A[" << i << "]=";
    cin >> A[i];
  }
  indice_maior = 1;              // Assumir 1o elemento o maior
  // Procurar maior do 2 elemento em diante
  for(i=2; i<=n; i++)
    if(A[i] > A[indice_maior])
       indice_maior = i;
  maior = A[indice_maior];
  cout << "Maior elemento = " << maior << "com indice = "
       << indice_maior << endl;
  return 0;
}
```


Matrizes

- Uma matriz é uma variável composta homogênea **multidimensional**, formada por variáveis, todas do **mesmo tipo de dados**, com o mesmo identificador (mesmo nome)
- Por exemplo, o seguinte matriz $a_{3,4}$ é capaz de armazenar (3 - 1 +1) * (4 - 1 + 1)= 12 inteiros

a	1	2	3	4
1				
2				
3				

- Essa matriz pode ser declarada em pseudo-código da seguinte forma:
  - **declare a[1..3, 1..4] : inteiro**

Número de Elementos de uma Matriz

- Exemplos
  - **declare a[2..3, 3..4] : inteiro**
  - **Possui (3 – 2 + 1) * (4 – 3 + 1) = 4 elementos**
  - **declare b[0..5, 1..4, 100..105] : real**
  - **Possui (5-0+1) * (4-1+1) * (105-100+1) = 6*4*6 = 144 elementos**
- Em geral:
  - declare m[Li1..Ls1,Li2..Ls2, ..., Lin..Lsn]
  - Possui (Ls1-Li1+1)*(Ls2-Li2+1)*...*(Lsn-Lin+1) elementos

Manipulando Matrizes

- Um elemento de uma matriz pode ser referenciado com o nome da matriz, seguido do seu índice correspondente colocado entre colchetes
- Por exemplo, para referenciar o elemento $\mathbf{a_{1,3}}$, denotamos por a[1,3] ou a[1][3]
- Por exemplo, se desejamos atribuir o valor 100 ao elemento $\mathbf{a_{1,3}}$, basta utilizar o comando de atribuição:
  - a[1,3] ← 100

a	1	2	3	4
1			100	
2				
3				

Manipulando Matrizes

- Um elemento de uma matriz pode ser referenciado com o nome da matriz, seguido do seu índice correspondente colocado entre colchetes
- Por exemplo, para referenciar o elemento $\mathbf{a_{1,3}}$, denotamos por a[1,3]
- Por exemplo, se desejamos atribuir o valor 100 ao elemento $\mathbf{a_{1,3}}$, basta utilizar o comando de atribuição:
  - a[1,3] ← 100
  - a[3,1] ← 2

a	1	2	3	4
1			100	
2				
3	2			

Manipulando Matrizes

- Um elemento de uma matriz pode ser referenciado com o nome da matriz, seguido do seu índice correspondente colocado entre colchetes
- Por exemplo, para referenciar o elemento $\mathbf{a_{1,3}}$, denotamos por a[1,3]
- Por exemplo, se desejamos atribuir o valor 100 ao elemento $\mathbf{a_{1,3}}$, basta utilizar o comando de atribuição:
  - a[1,3] ← 100
  - a[3,1] ← 2
  - a[2,2] ← 7

a	1	2	3	4
1			100	
2		7		
3	2			

Manipulando Matrizes

- Um elemento de uma matriz pode ser referenciado com o nome da matriz, seguido do seu índice correspondente colocado entre colchetes
- Por exemplo, para referenciar o elemento $\mathbf{a_{1,3}}$, denotamos por a[1,3]
- Por exemplo, se desejamos atribuir o valor 100 ao elemento $\mathbf{a_{1,3}}$, basta utilizar o comando de atribuição:
  - a[1,3] ← 100
  - a[3,1] ← 2
  - a[2,2] ← 7
  - a[2,4] ← 11

a	1	2	3	4
1			100	
2		7		11
3	2			

Entrada de Matrizes

- Em pseudo-código, é possível expressar que uma matriz deve ser totalmente lida através do comando Leia
- Por exemplo

```
declare a[1..3, 1..4] : inteiro
Leia(a)
```

- Entretanto, as linguagens de programação em geral não possuem um comando deste tipo
- Assim, torna-se necessário ler uma matriz através da leitura individual de cada um dos seus elementos
- Por exemplo

```
declare a[1..3, 1..4] : inteiro
Para i ← 1 até 3 Faça
  Para j ← 1 até 4 Faça
    Leia(a[i,j])
  Fim Para
Fim Para
```

Saída de Matrizes

- Em pseudo-código, é possível expressar que uma matriz deve ser totalmente impressa através do comando Escreva
- Por exemplo

```
declare a[1..3, 1..4] : inteiro
Escreva(a)
```

- Entretanto, as linguagens de programação em geral não possuem um comando deste tipo
- Assim, torna-se necessário escrever uma matriz através da impressão individual de cada um dos seus elementos
- Por exemplo

```
declare a[1..3, 1..4] : inteiro
Para i ← 1 até 3 Faça
  Para j ← 1 até 4 Faça
    Escreva(a[i,j])
  Fim Para
Fim Para
```


Matrizes em C/C++

- Nas linguagens C/C++ ANSI, podem ser utilizadas no mínimo 12 dimensões, mas o limite fica por conta de cada compilador
- Assim como em vetores, o limite inferior de cada dimensão de uma matriz é sempre zero e apenas o tamanho de cada dimensão é especificado
- A matriz
  - `declare a[1..3, 1..4] : inteiro`
- Pode ser declarada em C/C++ como
  - `int a[4][5]`

Matrizes em C/C++

- A matriz
  - `declare a[0..3, 0..4] : inteiro`
- Pode ser declarada em C/C++ como
  - `int a[4][5]`
- Para referenciar um elemento da matriz, basta utilizar o nome da matriz, seguido dos índices respectivos, cada índice entre colchetes
  - a[1][2] = 4;

a	0	1	2	3	4
0					
1			4		
2					
3					

Matrizes em C/C++

- A matriz
  - `declare a[0..3, 0..4] : inteiro`
- Pode ser declarada em C/C++ como
  - `int a[4][5]`
- Para referenciar um elemento da matriz, basta utilizar o nome da matriz, seguido dos índices respectivos, cada índice entre colchetes
  - a[1][2] = 4;
  - a[2][3] = 7;

a	0	1	2	3	4
0					
1			4		
2				7	
3					

Matrizes em C/C++

- A matriz
  - `declare a[0..3, 0..4] : inteiro`
- Pode ser declarada em C/C++ como
  - `int a[4][5]`
- Para referenciar um elemento da matriz, basta utilizar o nome da matriz, seguido dos índices respectivos, cada índice entre colchetes
  - a[1][2] = 4;
  - a[2][3] = 7;
  - a[0][0] = 11;

a	0	1	2	3	4
0	11				
1			4		
2				7	
3					

Exemplo 1

```
Algoritmo Preencher. Preenche uma matriz 10x20 (200 elementos)
   com valores de 1 até 200.
Início
  declare i,j,contador,m[1..10, 1..20] : inteiro

  contador ← 1
  Para i ← 1 até 10 Faça
    Para j ← 1 até 20 Faça
      m[i,j] ← contador
      contador ← contador + 1
    Fim Para
  Fim Para
  Para i ← 1 até 10 Faça
    Para j ← 1 até 20 Faça
      Escreva(m[i,j])
    Fim Para
  Fim Para
Fim
```

Exemplo 1 em C++

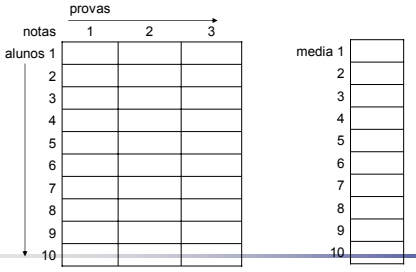
```
/* Algoritmo Preencher. Preenche uma matriz 10x20 (200 elementos) com valores de
   1 até 200.
*/
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{ const int L=10, C=20;
  int i,j,contador,m[L+1][C+1];

  contador = 1;
  for(i=1; i<=L; i++)
    for(j=1; j<=C; j++)
    { m[i][j] = contador;
      contador++;
    }
  for(i=1; i<=L; i++)
  { for(j=1; j<=C; j++)
      cout << setw(4) << m[i][j];
    cout << endl;
  }
  return 0;
}
```

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
```


Exemplo 2

```
Algoritmo Médias. Dada uma classe de 10 alunos e 3 provas por
   aluno, calcular e imprimir a média de cada aluno
Início
  declare i,j : inteiro
  declare notas[1..10, 1..3], media[1..10] : real
```



Exemplo 2

```
Algoritmo Médias. Dada uma classe de 10 alunos e 3 provas por aluno, calcular e imprimir a
    média de cada aluno
Início
  declare i,j : inteiro
  declare notas[1..10, 1..3], media[1..10] : real

  // Leitura da matriz de 10 alunos x 3 provas
  Para i ← 1 até 10 Faça
    Para j ← 1 até 3 Faça
      Escreva("Aluno ",i," Prova ",j)
      Leia(notas[i,j])
    Fim Para
  Fim Para
  // Cálculo da média
  Para i ← 1 até 10 Faça   // para cada aluno
    media[i] ← 0.0
    Para j ← 1 até 3 Faça  // para cada nota
      media[i] ← media[i] + notas[i,j]
    Fim Para
    media[i] ← media[i] / 3.0
  Fim Para
  Para i ← 1 até 10 Faça       // Impressão de resultados
    Escreva("Aluno ",i," Média = ",media[i])
  Fim Para
Fim
```


Exemplo 2 em C++

```
/* Algoritmo Médias. Dada uma classe de 10 alunos e 3 provas por aluno, calcular
   e imprimir a média de cada aluno */
#include <iostream>
using namespace std;
int main()
{ const int ALUNOS=10, PROVAS=3;
  int i,j;
  float notas[ALUNOS+1][PROVAS+1], media[ALUNOS+1];

  // Leitura da matriz de 10 alunos x 3 provas
  for(i=1; i<=ALUNOS; i++)
    for(j=1; j<=PROVAS; j++)
    { cout << "Aluno " << i << " Prova " << j;
      cin >> notas[i][j];
    }
  // Cálculo da média
  for(i=1; i<=ALUNOS; i++)                  // para cada aluno
  { media[i] = 0.0;
    for(j=1; j<=PROVAS; j++)                // para cada nota
      media[i] = media[i] + notas[i][j];
    media[i] = media[i] / PROVAS;
  }
  for(i=1; i<=ALUNOS; i++)          // Impressão de resultados
    cout << "Aluno " << i << " Media = " << media[i] << endl;
  return 0;
}
```


Resumo

- Nesta aula vimos estruturas de dados homogêneas que permitem armazenar e manipular um conjunto de dados por meio de uma mesma variável
- Vetores são estruturas de dados unidimensionais, ou seja, possuem um único índice
- Matrizes são estruturas de dados com duas ou mais dimensões (dois ou mais índices)