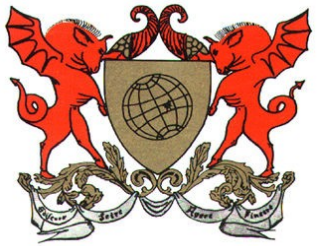
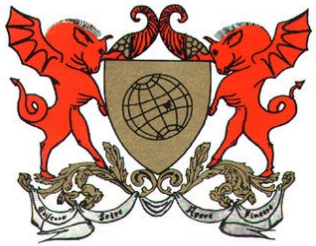


Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

SIN 110
Programação

Sistemas de Informação
Profa. Rachel Reis
rachel.reis@ufv.br



Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

Aula de Hoje

Introdução à Arrays

Material adaptado do Prof. Murilo Naldi

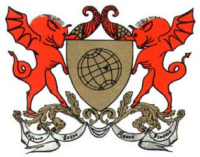
Créditos:

Prof. Guilherme Pena

Introdução à Arrays

- **Arrays** são coleções de dados com algumas características ou propriedades especiais.

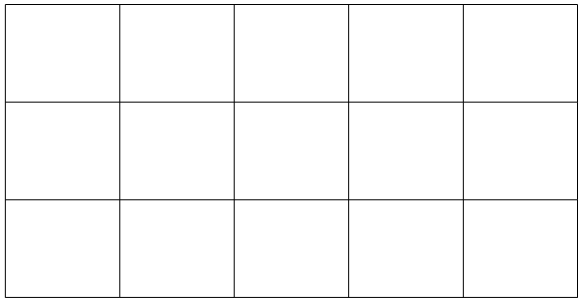
- Eles são também denominados **Arranjos, Vetores ou Matrizes**.



Introdução à Arrays

- **Arrays** podem variar em número de dimensões.
  - Arrays de 1 dimensão = vetores

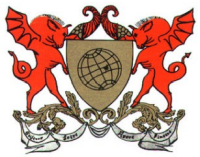
  - Arrays de 2 ou mais dimensões = matrizes



Introdução à Arrays

- Uma definição mais formal para arrays:
  - **Variáveis Compostas Homogêneas**

  Principais propriedades:
  - ✔ Conjunto de elementos do mesmo tipo (int, float, ...)
  - ✔ Posições Indexadas, isto é, cada posição possui um identificador



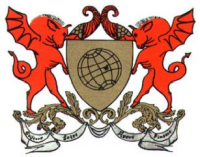
Vetor

- Array de uma dimensão
  - Um vetor é um tipo de dado usado para armazenar elementos do mesmo tipo.
  - Suponha que você precise ler as notas de 5 alunos.
  - Declaração:

```
int nota0, nota1, nota2, nota3, nota4;
```

  - O programa deve ler cada nota separadamente:

```
printf("Digite a nota 0:");
scanf("%d", &nota0);
```



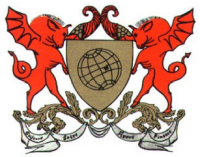
Vetor

- Imagine agora: E se você tivesse que ler as notas de uma classe de 50 alunos ou da escola toda com 200 alunos?
  - O **vetor** é o tipo de dado oferecido por C para este propósito.
  - Solução para o problema anterior:
    - ✔ Ao invés de declarar:

```
int nota0, nota1, nota2, ..., nota49;
```

    - ✔ Use:

```
int notas[50];
```

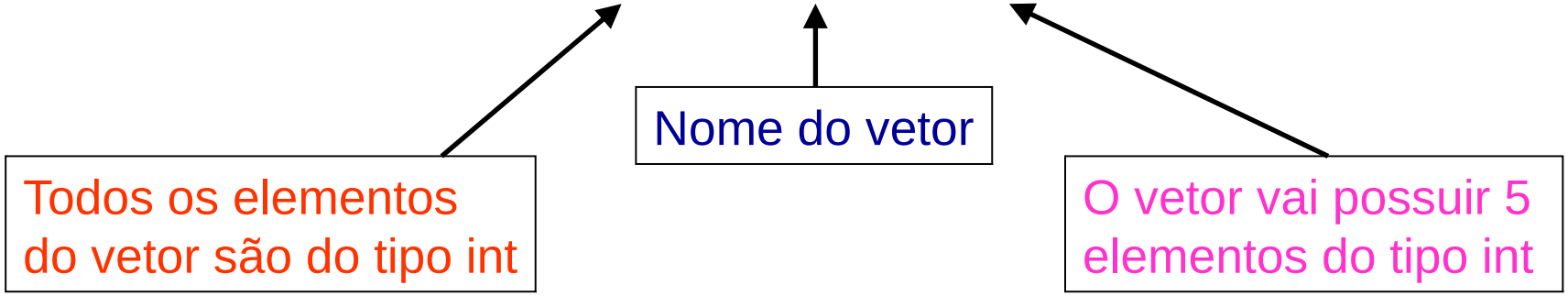


Declaração de um Vetor

- Vetores são declarados por meio do uso de um par de colchetes ([ ]) que seguem o nome da variável.

```
tipo nomeDoArray[tamanhoDoArray];
```

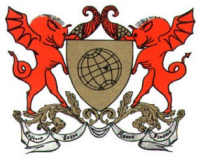
```
int notas[5];
```



Declaração de um Vetor

- Uma das coisas mais importantes que devemos saber sobre vetores e matrizes é que sua utilidade se deve muito ao **reuso de dados**.
- Ou seja, quando guardamos dados que serão usados em mais de um momento no mesmo programa.

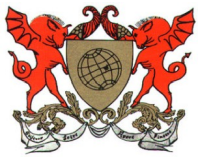
```
int notas[5];
```



Exemplo 1

- Crie um programa em C que declare um vetor de número reais com 10 posições.

```
int main()
{
    float vetor1[10];

    return 0;
}
```

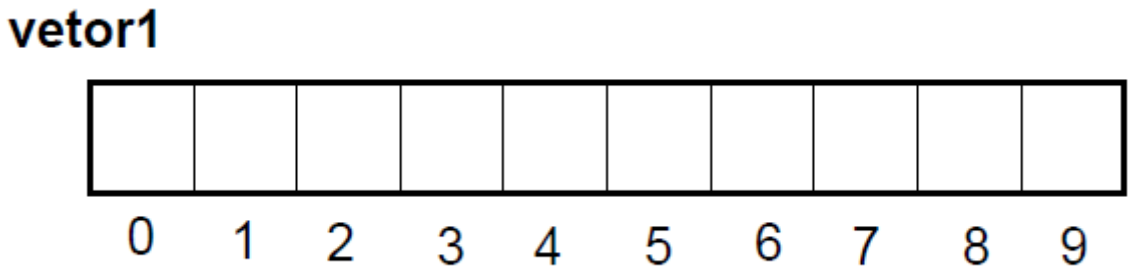


Exemplo 1

- Um vetor de 10 posições:

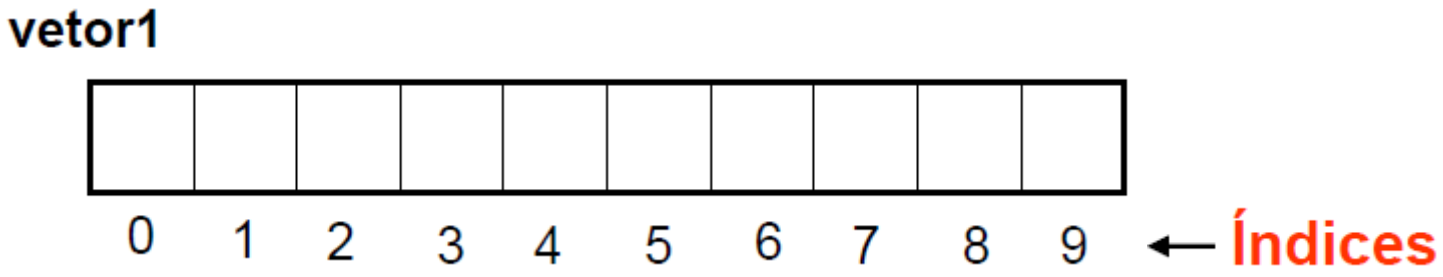
```
float vetor1[10];
```

- Na memória serão alocadas 10 posições do tipo float para a variável **vetor1**.



Referenciando os Elementos

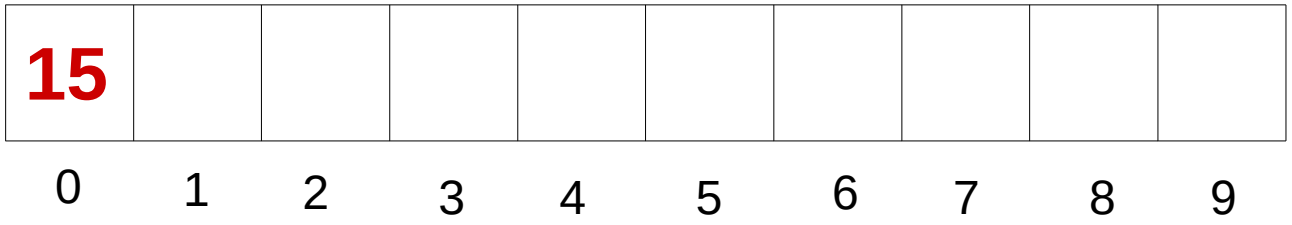
- Os elementos do vetor são sempre numerados por índices iniciados por 0 (zero).



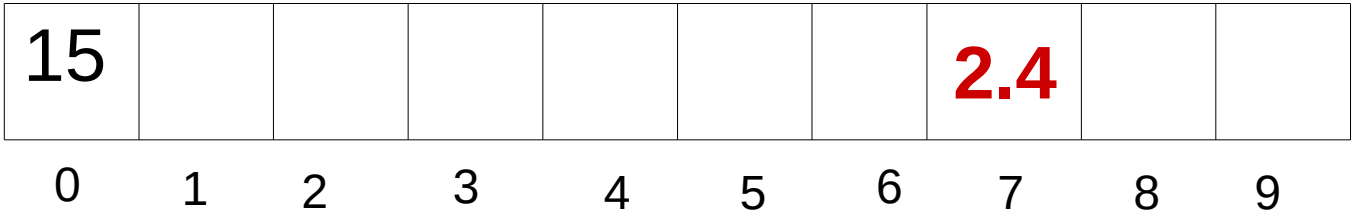
Acessando o Vetor

- Inicializando algumas posições do vetor

Vetor1[0] = 15;



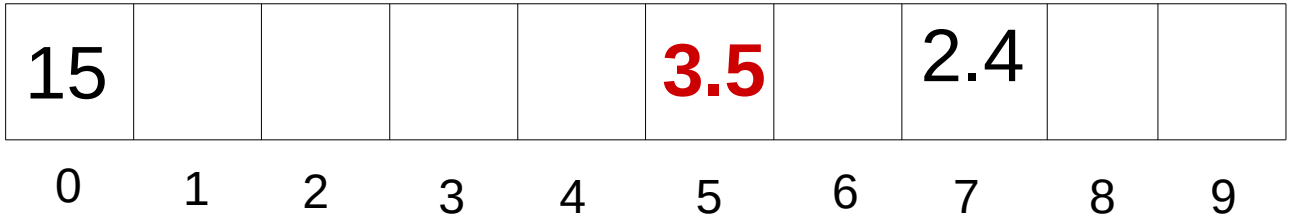
Vetor1[7] = 2.4;



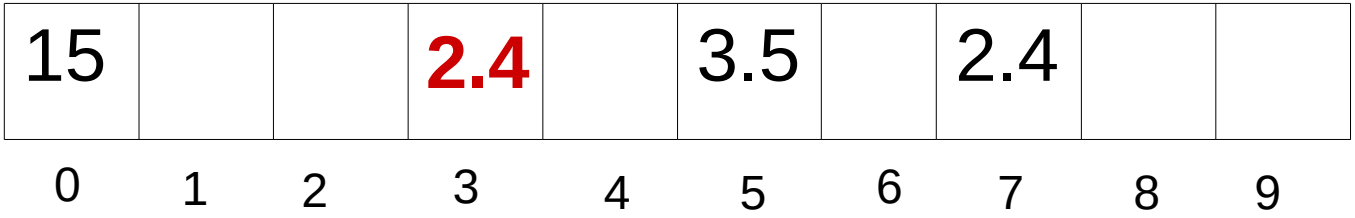
Acessando o Vetor

- Inicializando algumas posições do vetor

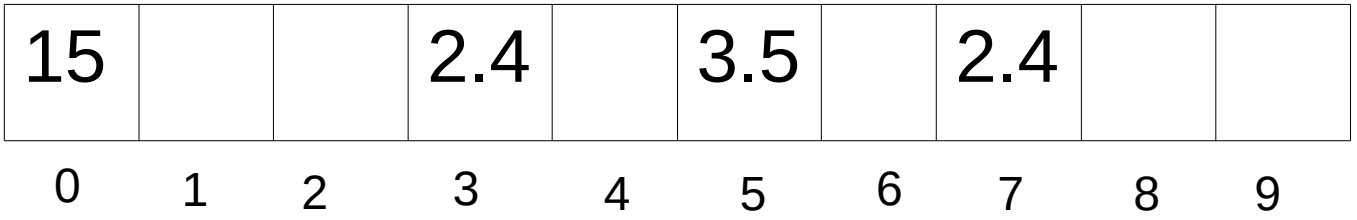
```
pos = 5;
vetor1[pos] = 3.5;
```




```
vetor1[3] = vetor1[7];
```



Acessando o Vetor

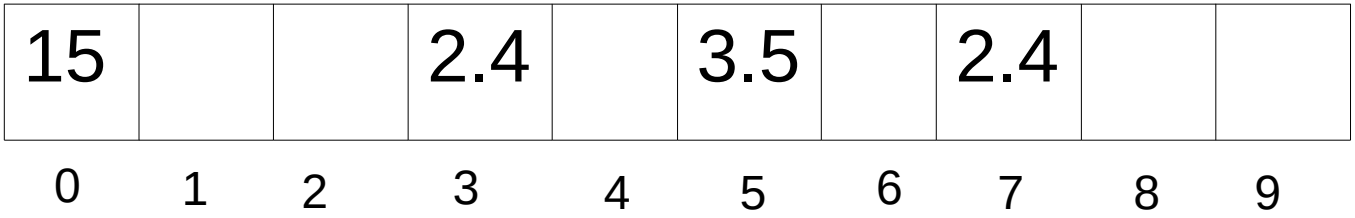



```
pos = 5;
printf("Posicao %d = %.2f", pos, vetor1[pos]);
```

Tela de saída:

```
Posicao 5 = 3.50
```

Acessando o Vetor




```
for (pos=3; pos<=7 ;pos++)
{
    printf("Valor %.1f\n", vetor1[pos]);
}
```

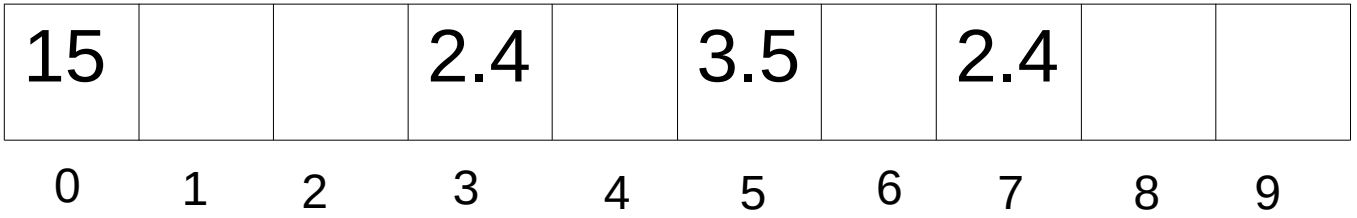
Tela de saída:

```
Valor 2.4
Valor ?
Valor 3.5
Valor ?
Valor 2.4
```

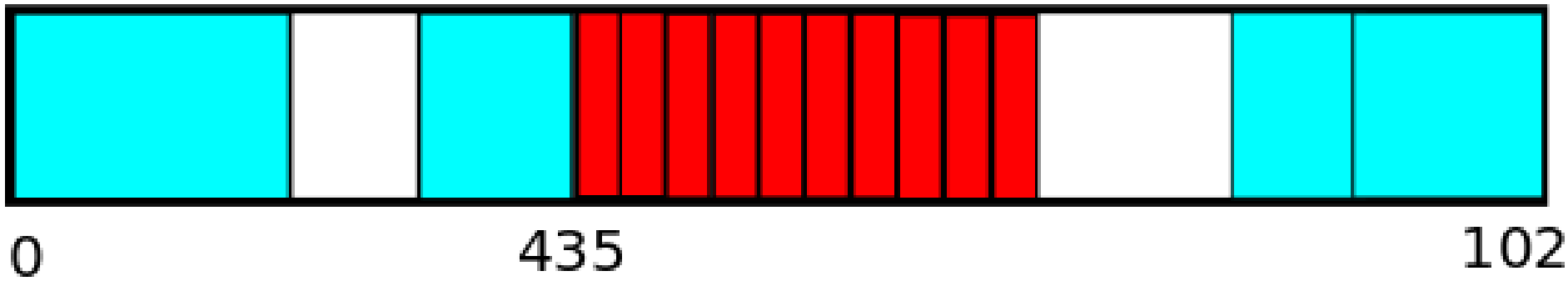
Valor de pos

```
3
4
5
6
7
```

Representação do Vetor na memória



Memória Principal

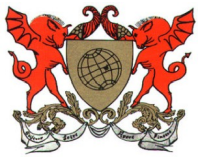



```
float  vetor1[10]; //declara vetor
```

Preenchendo um Vetor

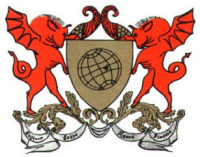
- Complete o código abaixo de forma que o usuário insira o valor das 5 notas. Use o laço **for**.

```
#include <stdio.h>
int main()
{
   float notas[5]; int i;
   for(i=0; i < 5; i++)
   {
    printf("Digite a nota do aluno %d: ", i);
    scanf("%f", &notas[i]);
   }
   return 0;
}
```



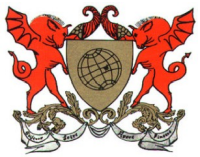
Preenchendo um Vetor

```
#include <stdio.h>
int main()
{
    int vetor2[30];
    int i;
    for (i=0; i<30; i++)
    {
      printf("Entre com o numero %d:", i);
      scanf("%d", &vetor2[i]);
    }
     return 0;
}
```



Exibindo um vetor

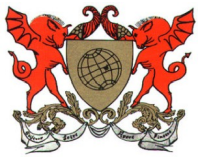
```
#include <stdio.h>
int main()
{
      char vetor3[6] = {'a','b','c','d','e','!'};
      int i;

      for ( i = 0; i < 6; i++)
      {
          printf("Posicao %d: %c\n", i, vetor3[i]);
      }

      return 0;
}
```



Exibindo um Vetor

Altere o programa abaixo para que ele imprima os elementos do vetor.

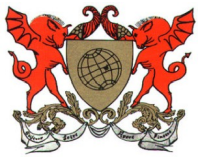
```
#include <stdio.h>
int main() {
    int vetor2[30];
    int i;
    for (i=0; i<30; i++){
      printf("Entre com o numero %d:", i);
      scanf("%d", &vetor2[i]);
    }
     return 0;
}
```



Exibindo um Vetor

Altere o programa abaixo para que ele imprima os elementos do vetor.

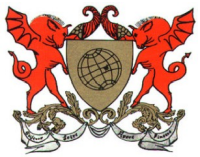
```
#include <stdio.h>
int main(){
    int vetor2[30];
    int i;
    for (i=0; i<30; i++){
      printf("Entre com o numero %d:", i);
      scanf("%d", &vetor2[i]);
    }
    for (i=0; i<30; i++){
      printf("Elemento %d: %d", i, vetor2[i]);
    }
     return 0;
}
```



Usando os tipos

- É possível declarar um array de qualquer tipo de dado:

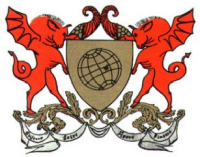
```
int numeros[5];


char letras[26];


float notas[50];
```



Exercícios

Representando o reuso de dados:

Faça um programa que permita entrar com a nota de 50 alunos de uma turma e mostrar no final do programa a maior nota, a menor nota e quantos alunos ficaram acima da média da turma.

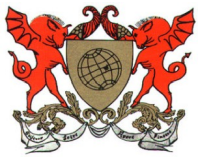


Exercícios

1. Dada uma sequência de 20 números fornecidos pelo usuário, imprimi-la na ordem inversa à da leitura.

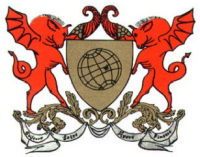
2. Faça um programa que declare um vetor de 5 posições do tipo **char** e execute as operações:

a. Faça um comando for para que o usuário possa preencher o vetor criado.
b. Atribua a letra B no indice 6.
c. Atribua a letra K no indice 3.
d. Escreva na tela o que está no indice 2.
e. Faça um comando for para exibir na tela todos os valores do vetor.



Array de 2 dimensões

- No caso de 2 dimensões, o array é comumente chamado de **Matriz**.
- A matriz é extremamente útil em várias aplicações.
- Principalmente aplicações em que usamos os dados armazenados na matriz para vários processamentos diferentes.

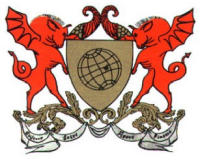


Aplicação de Matrizes

- Matrizes podem ser usadas para simular tabelas.
  - Tabela de distâncias

Distância em km entre as capitais do sudeste

	São Paulo	Rio de Janeiro	Vitória	Belo Horizonte
São Paulo	0	429	882	586
Rio de Janeiro	429	0	521	434
Vitória	882	521	0	524
Belo Horizonte	586	434	524	0



Aplicação de Matrizes

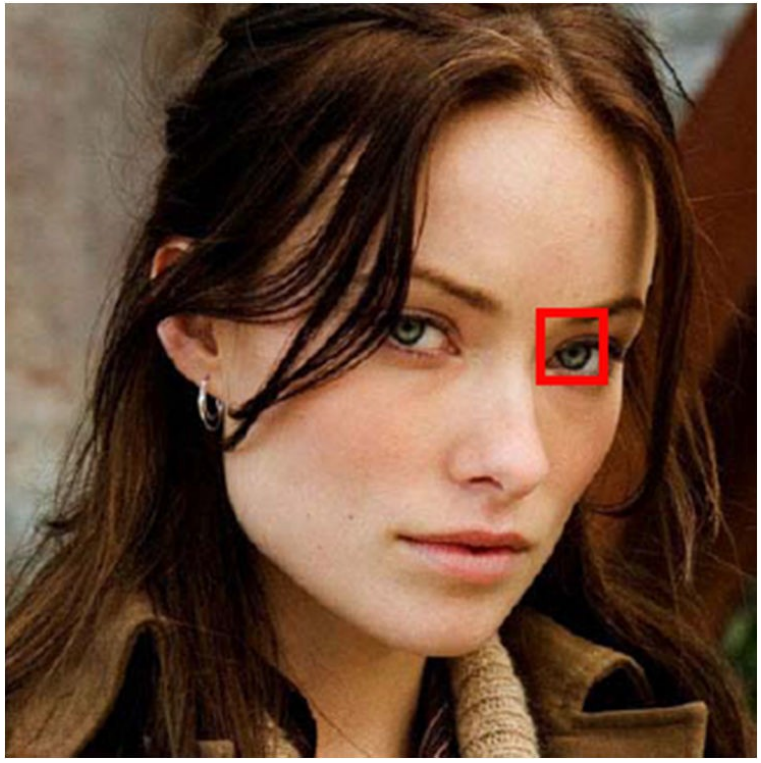
- Matrizes podem ser usadas para simular tabelas.
  - Tabela de valores

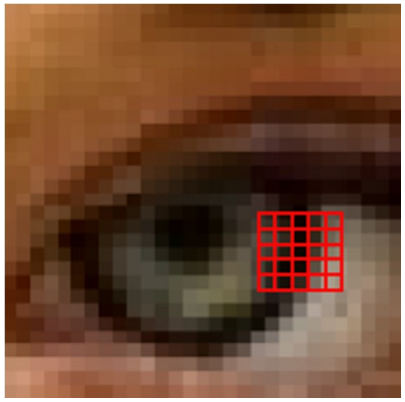
Valores em Reais das Diárias de um Hotel

	Suíte Executiva	Suíte Presidencial
Baixa-temporada	60,00	90,00
Alta-temporada	95,00	130,00
Carnaval	100,00	140,00
Natal/Ano Novo	110,00	150,00

Aplicação de Matrizes

- Imagens são matrizes de pixels
  Cada cor é representada por um número inteiro





100	105	105	100	80
160	160	155	100	80
180	155	155	90	75
180	155	155	90	75
200	180	155	90	75

Sintaxe/Declaração

- A declaração de uma matriz é similar a de um vetor:

```
tipo nome_matriz[NumLinhas][NumColunas];
```

- Na memória serão alocadas Nlin X Ncol posições:

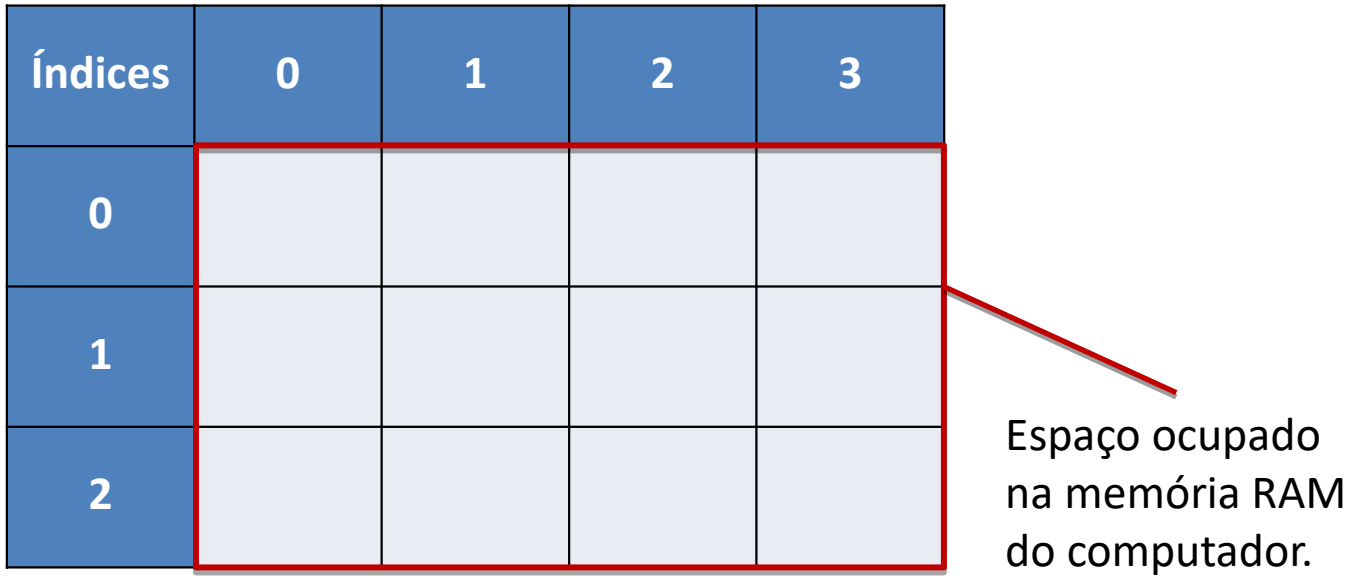
0, 0	0,1	0,2	...	0, NCol -1
1, 0	1,1	1,2	...	1, NCol -1
2, 0	2,1	2,2	...	2, NCol -1
...	...	...		...
NLin -1 , 0	NLin -1 , 1	NLin -1, 2	...	NLin -1 , NCol -1

Exemplo

- A declaração de uma matriz é similar a de um vetor:

```
int matriz[3][4];
```

- Na memória serão alocadas 3x4 posições do tipo **int**:

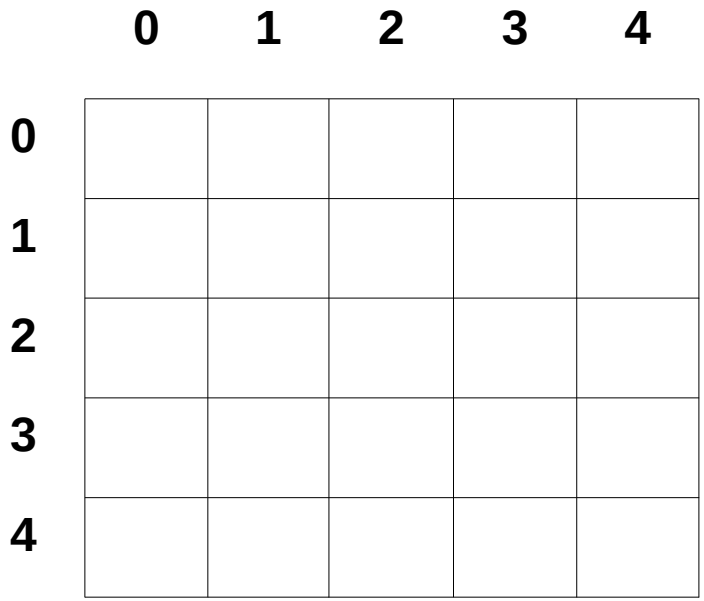


Exemplo

- A declaração de uma matriz é similar a de um vetor:

```
char mat_letras[5][5];
```

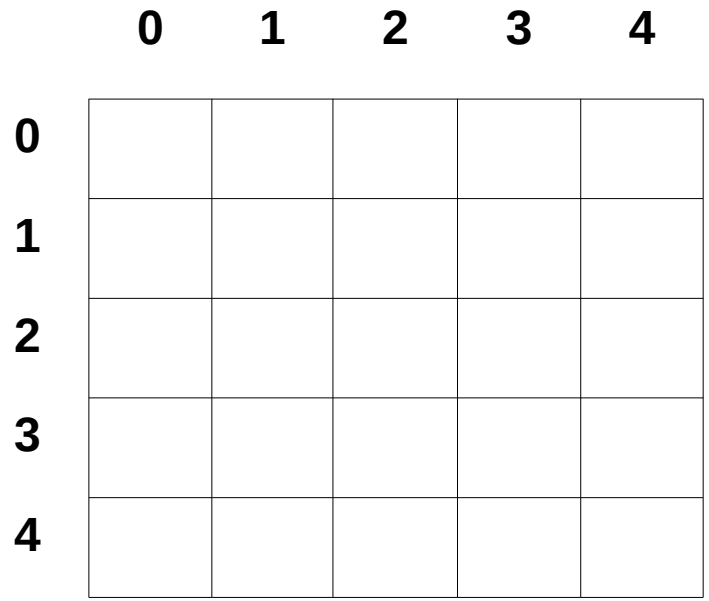
- Na memória serão alocadas 5X5 posições do tipo char:



Exemplo

- Para acessar um valor da matriz é preciso informar 2 posições:

```
mat_letras[0][0] = 'A';
```



Exemplo

- Para acessar um valor da matriz é preciso informar 2 posições:

```
mat_letras[0][0] = 'A';
```

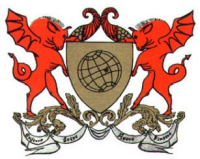
	0	1	2	3	4
0	A				
1					
2					
3					
4					

Exemplo

- Para acessar um valor da matriz é preciso informar 2 posições:

```
mat_letras[3][1] = 'M';
```

	0	1	2	3	4
0	A				
1					
2					
3		M			
4					



Exemplo

- Para acessar um valor da matriz é preciso informar 2 posições:

```
mat_letras[2][4] = 'R';
```

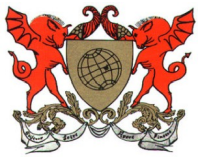
	0	1	2	3	4
0	A				
1					
2					R
3		M			
4					

Exemplo

- Para acessar um valor da matriz é preciso informar 2 posições:

```
mat_letras[1][3] = 'T';
```

	0	1	2	3	4
0	A				
1				T	
2					R
3		M			
4					



Exemplo

- Para acessar um valor da matriz é preciso informar 2 posições:

```
mat_letras[4][4] = 'I';
```

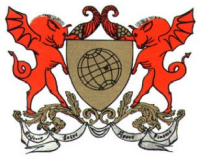
	0	1	2	3	4
0	A				
1				T	
2					R
3		M			
4					I

Exemplo

- É possível informar as posições através de variáveis:

```
p1 = 0;
p2 = 1;
mat_letras[p1][p2] = 'Z';
```

	0	1	2	3	4
0	A	Z			
1				T	
2					R
3		M			
4					I



Exemplo

- É possível informar as posições através de variáveis:

```
p1 = 1;
p2 = 2;
mat_letras[p1][p2] = '1';
```

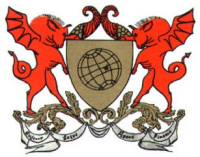
	0	1	2	3	4
0	A	Z			
1			1	T	
2					R
3		M			
4					I

Acessando a matriz

- Que resultado gera esta sequência de comandos?

```
printf("%c",mat_letras[3][1]);

printf("%c",mat_letras[0][0]);

printf("%c",mat_letras[1][3]);

printf("%c",mat_letras[2][4]);

printf("%c",mat_letras[4][4]);

printf("%c",mat_letras[0][1]);
```

	0	1	2	3	4
0	A	Z			
1			1	T	
2					R
3		M			
4					I



“Varrendo” uma linha da matriz

```
int col;
int lin = 0;
for (col=0; col<5; col++)
{
   printf("Valor: %c \n",mat_letras[lin][col]);
}
```

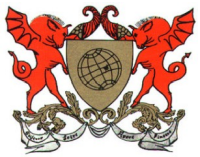
Tela

```
Valor: A
Valor: Z
Valor: ?
Valor: 3
Valor: y
```

Valor de ‘**col**’

```
0
1
2
3
4
```

	0	1	2	3	4
0	A	Z		3	Y
1		6		T	F
2	5	D	V	1	R
3	3			A	5
4	6	J	K	X	I



“Varrendo” uma coluna da matriz

```
int col = 1;
int lin;
for (lin=0; lin<5; lin++)
{
   printf("Valor: %c \n",mat_letras[lin][col]);
}
```

Tela

```
Valor: Z
Valor: 6
Valor: D
Valor: ?
Valor: J
```

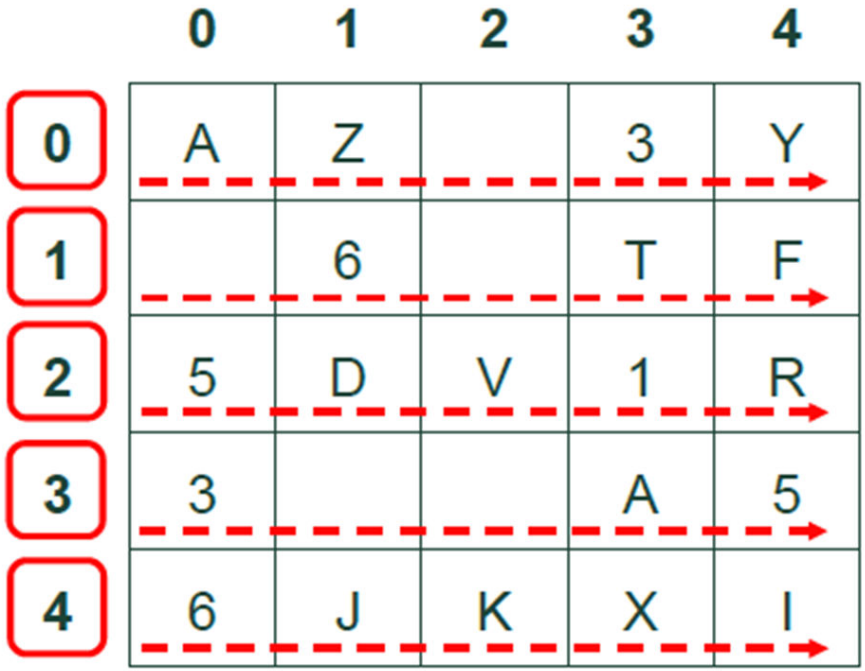
Valor de ‘**lin**’

```
0
1
2
3
4
```

	0	**1**	**2**	**3**	**4**
0	A	Z		3	Y
1		6		T	F
2	5	D	V	1	R
3	3			A	5
4	6	J	K	X	I

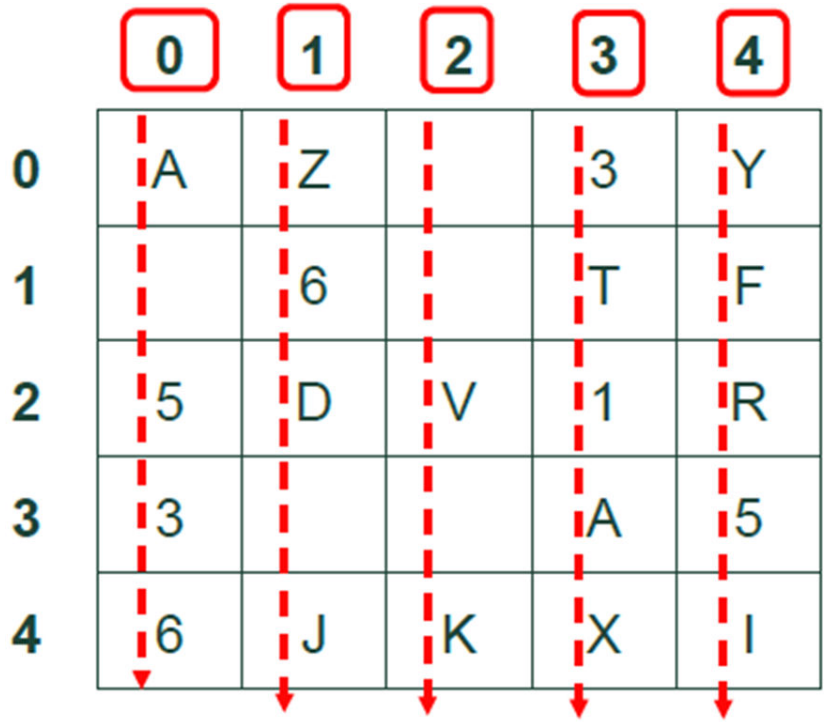
Usando toda a matriz

- Para acessar toda a matriz:
  - Podemos acessar todas as colunas de cada uma das linhas.



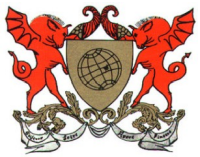
Usando toda a matriz

- OU
  - Podemos acessar todas as linhas de cada uma das colunas.



Preenchendo uma matriz (entrada do usuário)

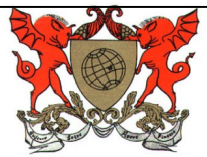
```
int main()
{
    float matF[20][30];
    int l, c;
    for (l=0; l<20; l++)
    {
        for (c=0; c<30; c++)
        {
            printf("\nValor da Posicao (%d,%d):",l,c);
            scanf("%f",&matF[l][c]);
        }
    }
    return 0;
}
```

Exibindo uma matriz (que já tenha valores)

- Como exibir os valores da matriz do código anterior?

```
int main()
{
    ...

    for (l=0; l<20; l++)
    {
        for (c=0; c<30; c++)
        {
            printf("\nPosicao (%d,%d):%f",l,c,matF[l][c]);
        }
    }


    return 0;
}
```

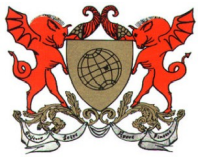



Erros Comuns

Declaração de vetores e matrizes:

```
int vetor[];

int vetor[n];

int matriz[n][m];

int matriz[][];
```

A declaração de vetores e matrizes com o uso do operador [] indica que o espaço de memória é estático e deve ser sempre constante, por esse motivo não se pode fazer as declarações acima, sempre deve-se usar um valor constante.

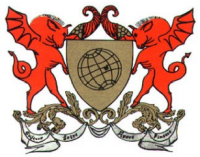


Erros Comuns

Acesso a posições inválidas ou índices inexistentes:

```
int vetor[10];
float a = 1.0;
vetor[a] = 5;
vetor[-1] = 0;

int matriz[5][5];
matriz[6][4] = 1;
```

Toda dimensão de um vetor ou matriz é contada por índices inteiros que iniciam em **0 e vão até o tamanho da declaração menos 1**. É um erro comum usar variáveis reais para acessar elementos e não verificar os limites do array.

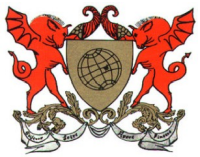


Erros Comuns

Atribuição entre vetores e matrizes:

```
int va[10], vb[10];

va = vb;

int ma[2][2], mb[2][2];

ma = mb;
```

Não é possível fazer atribuição direta entre vetores ou matrizes, mesmo que ambos tenham o mesmo tamanho.
A única forma de atribuir é assegurar que ambos tenham o mesmo tamanho e a atribuição deve ser feita elemento a elemento.



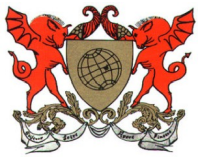
Exemplo

Dada a matriz M abaixo responda:

-22	77	99
31	10	06
02	05	45
34	88	23
72	55	44
18	33	65
07	21	87

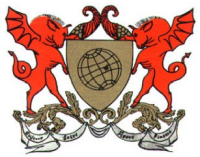
1. Como a matriz deve ser declarada na linguagem C?
2. Determine os seguintes elementos:

a) M[2, 1]; b) M[4, 3]; c) M[6, 2];
d) M[2, 2]; e) M[3, 3]; f) M[4, 2];
g) M[1, 1]; h) M[0, 2]; i) M[0, 0];



Exemplo

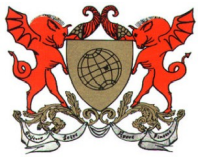
- Faça um programa que:
  - Crie uma matriz de distâncias entre 4 cidades diferentes;
  - Peça para o usuário entrar com as distâncias entre as cidades;
  - Exiba na tela a matriz de distâncias criada;
  - Quando o usuário digitar o número de duas cidades o programa deverá retornar a distância entre elas;



Exercícios

1) Escreva um programa que leia 10 números inteiros e os armazene em um vetor. Imprima o vetor, o maior elemento e a posição que ele se encontra.

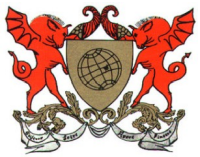
2) Faça um programa que receba do usuário dois vetores, A e B, com 10 números inteiros cada. Crie um novo vetor denominado C calculando C = A - B. Mostre na tela os dados do vetor C.



Exercícios

3) Faça um programa que leia um vetor de 20 posições, mostre a quantidade de valores repetidos e quais são os valores.

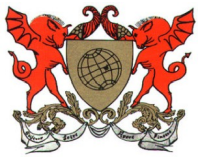
4) Faça um programa que receba do usuário dois vetores, A e B, com n **números reais** cada. Crie quatro novos vetores que receberão a soma, subtração, multiplicação e divisão dos vetores anteriores.



Exercícios

5) Declare uma matriz 5 x 5. Preencha com 1 a diagonal principal e com 0 os demais elementos. Escreva ao final a matriz obtida.

6) Leia uma matriz 4x4, imprima a matriz e retorne a localização (linha e a coluna) do maior e do menor valor.



Exercícios

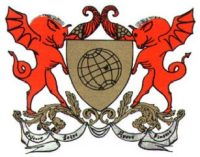
7) Faça programa que leia uma matriz 3x6 com valores reais.

(a) Imprima a soma de todos os elementos das colunas ímpares.

(b) Imprima a média aritmética dos elementos das colunas pares.

(c) Substitua os valores da sexta coluna pela soma entre os valores das colunas 1 e 2.

(d) Imprima a matriz modificada.



Exercícios

8) Faça programa que declare uma matriz 50x50 com valores inteiros.

(a) Permita o usuário entrar com quais dimensões ele quer trabalhar (n x m).

(b) Leia os dados da matriz criada.

(c) Calcule a média de todos os elementos da matriz.

(d) Imprima a quantidade elementos pares e ímpares da matriz que estão acima da média.