



# Linguagem C

## Vetores e Matrizes

MsC. Douglas Santiago Kridi

Programação I - 2018.2

Bacharelado em Ciência da Computação

Universidade Estadual do Piauí

*douglaskridi@gmail.com*

# Introdução

- Um vetor é uma *sequência* de vários valores do *mesmo tipo*.
- São armazenados *sequencialmente* na memória.
- Acessados por meio de um *mesmo nome* de variável.
- Um vetor também pode ser entendido logicamente como uma *lista de elementos* de um mesmo tipo.

# Introdução

- Cada elemento desta sequência pode ser acessado individualmente através de um *índice* dado por um número inteiro.
  - Os elementos são indexados de *0* até *n-1*, onde *n* é a *quantidade* de elementos do vetor.
  - O valor de *n* também é chamado de *dimensão* ou *tamanho* do vetor.

Vet[10]	v <sub>0</sub>	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>	v <sub>5</sub>	v <sub>6</sub>	v <sub>7</sub>	v <sub>8</sub>	v <sub>9</sub>
---------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

- O vetor tem tamanho fixo durante a execução do programa, definido na declaração.
  - Durante a execução não é possível aumentar ou diminuir o tamanho do vetor.

# Declaração

- A declaração de vetores obedece à mesma sintaxe da declaração de variáveis. A diferença está no valor entre colchetes (dimensão):

*tipo variável[tamanho];*

- Por exemplo, para declarar um vetor com 10 números inteiros:

*int vetor[10];*

- Outro exemplo, declarar um vetor com 100 medidas em double:

*double medidas[100];*

# Acesso

- Para acessar os elementos contidos em um vetor usamos o operador de índice [ ].
- Ele retorna uma referência para o elemento correspondente ao índice:

- Elementos de um vetor:

*vetor[0], vetor[1], vetor[2], ...*

- Atribuição:

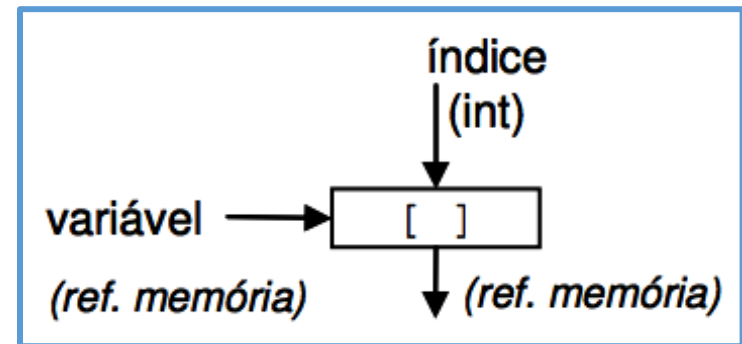
*vetor[indice] = valor;*

- Exemplo:

*int vetor[10];*

*vetor[5] = 3;*


*vetor[0] = vetor[1] + vetor[2];*



# Acesso

- É muito comum utilizar a estrutura de repetição `for` para percorrer todos os elementos de um vetor.
  - Por exemplo, para imprimir todos os elementos de um vetor de 100 elementos:

```
int indice;  
int vetor[100];  
...  
for (indice = 0; indice < 100; indice++) {  
    printf("%d", vetor[indice]);  
}
```



Lembre-se que para um vetor de tamanho 100, o primeiro elemento tem índice 0 e o último elemento, índice 99.

# Exemplo

- Faça um programa que lê dez números e os imprime em ordem inversa.
  - Primeiramente use um for para armazenar os 10 números.
  - Em seguida, usamos outro for para imprimi-los de 9 a 0.

```
int main() {  
    int valores[10];  
    int indice;  
  
    printf("Escreva 10 números inteiros: ");  
    for (indice = 0; indice < 10; indice++) {  
        scanf("%d", &valores[indice] );  
    }  
    printf("Valores em ordem reversa:\n");  
    for (indice = 9; indice >= 0; indice--) {  
        printf("%d ", valores[indice]);  
    }  
    return 0;  
}
```

# Declaração

- Na declaração, pode-se definir o valor inicial de cada elemento, sempre listados entre chaves { e } e separados por vírgula.

*tipo* variável[n] = { *elem<sub>0</sub>*, *elem<sub>1</sub>*, *elem<sub>2</sub>*, *elem<sub>3</sub>*, ... *elem<sub>n-1</sub>* };

- Como o número de elementos do vetor pode ser inferido a partir da lista entre chaves, podemos omitir o tamanho do vetor:

*tipo* variável[] = { *elem<sub>0</sub>*, *elem<sub>1</sub>*, *elem<sub>2</sub>*, *elem<sub>3</sub>*, ... *elem<sub>n-1</sub>* };

- Exemplo:

*int* impares[5] = {1, 3, 5, 7, 9};

*int* impares[] = {1, 3, 5, 7, 9};



# Exemplo:

- Suponha dois vetores, A e B, cada um com 10 elementos. Faça a cópia dos valores de um vetor para outro:

```
Int main() {  
    int vetorA[10], vetorB[10];  
    Int indice;  
  
    for (indice = 0; indice < 10; indice++) {  
        vetorA[indice] = vetorB[indice];  
    }  
}
```

# Exemplo:

- Programa que lê n valores, e armazena-os em um vetor:
  - Como definir um vetor com tamanho variável?

```
int main() {  
    int valores[100];  
    int numero_valores, indice;  
  
    printf("Quantos valores? (no máximo 100): ");  
    scanf("%d", &numero_valores);  
  
    if ( (numero_valores > 100) || (numero_valores < 0) ) {  
        printf("Número de valores inválido\n");  
        return 1;  
    }  
    printf("Escreva %d números inteiros: ", numero_valores);  
    for (indice = 0; indice < numero_valores; indice++) {  
        scanf("%d", &valores[indice] );  
    }  
}
```

# Pratique:

1. Crie um programa que lê 6 valores inteiros, guarde-os em um vetor e, em seguida, mostre na tela os valores lidos.

# Matrizes (vetores multidimensionais)

- Uma matriz consiste de uma tabela de vários valores do mesmo tipo, armazenados sequencialmente.
- Cada elemento da tabela pode ser acessado individualmente através de dois índices determinados por números inteiros.
  - Estes índices poderiam ser interpretados como a linha e a coluna da matriz.
- A linguagem C define uma matriz como um vetor, cujos elementos são novamente vetores de mesmo tamanho e tipo.

Int [4] [10]

Int [10]	v <sub>00</sub>	v <sub>01</sub>	v <sub>02</sub>	v <sub>03</sub>	v <sub>04</sub>	v <sub>05</sub>	v <sub>06</sub>	v <sub>07</sub>	v <sub>08</sub>	v <sub>09</sub>
Int [10]	v <sub>10</sub>	v <sub>11</sub>	v <sub>12</sub>	v <sub>13</sub>	v <sub>14</sub>	v <sub>15</sub>	v <sub>16</sub>	v <sub>17</sub>	v <sub>18</sub>	v <sub>19</sub>
Int [10]	v <sub>20</sub>	v <sub>21</sub>	v <sub>22</sub>	v <sub>23</sub>	v <sub>24</sub>	v <sub>25</sub>	v <sub>26</sub>	v <sub>27</sub>	v <sub>28</sub>	v <sub>29</sub>
Int [10]	v <sub>30</sub>	v <sub>31</sub>	v <sub>32</sub>	v <sub>33</sub>	v <sub>34</sub>	v <sub>35</sub>	v <sub>36</sub>	v <sub>37</sub>	v <sub>38</sub>	v <sub>39</sub>

# Declaração

- A declaração de matrizes inclui uma nova dimensão escrita entre colchetes[ ].

tipo variável[linhas][colunas];

- Exemplo, matriz de inteiros com 4 linhas e 10 colunas:

```
int matriz[4][10];
```

# Acesso

- A atribuição ou leitura de valores em uma matriz utiliza dois índices.
- O primeiro elemento é armazenado em *matriz[0][0]* o segundo em *matriz[0][1]* e assim por diante, até o último elemento, em *matriz[linhas-1][colunas-1]*.
  - Para atribuir o valor 3 na linha 2, coluna 5, escrevemos:  
`matriz[1][4] = 3;`

# Acesso

- Para percorrer os elementos de uma matriz, são necessárias duas estruturas de repetição for, uma dentro da outra.
- O for *externo* percorre as *linhas da matriz*, o for *interno* percorre as *colunas* de uma *determinada linha* que está fixada pelo for externo.
  - Por exemplo, para imprimir todos os elementos de uma matriz 4x10, linha por linha:

```
Int main() {  
    int linha, coluna;  
    int matriz[4][10];  
  
    ...  
    for (linha = 0; linha < 4; linha++) {  
        for (coluna = 0; coluna < 10; coluna++) {  
            printf("%d ", matriz[linha][coluna]);  
        }  
        printf("\n");  
    }  
}
```

# Vetores de char (Strings)

- Na linguagem C, armazenamos o texto como um vetor de caracteres, onde cada caractere do texto é um elemento do vetor.
- O fim do texto deve ser demarcado por um caractere adicional e especial: o caractere nulo (`\0`).

char[20]	O	I	á		m	u	n	d	o	\0
	0	1	2	3	4	5	6	7	8	9

- Declarando:

```
char texto[tamanho];  
char texto[tamanho] = "texto";  
char texto[] = "texto";
```



# Vetores de char (Strings)

- Para **escrita**, podemos utilizar o printf e o formatador %s:

```
char texto[] = "Um texto em C";  
printf("A variavel texto contem: %s", texto);
```

- Para **leitura**, o comando scanf, com o modificador %s, é capaz de ler uma sequência de caracteres, terminada com *fim de linha* ou *espaço em branco*.

```
char texto[20];  
printf("Escreva uma palavra: ");  
scanf("%s", texto);
```

Observe, que no *scanf*,  
não foi usado &.

# Pratique:

1. Faça um programa que leia um valor  $n$  e depois leia uma sequência de  $n$  números inteiros e os armazene em um vetor. Posteriormente, seu programa deve determinar o maior e menor elemento deste vetor.
2. Escreva um programa que inverta um vetor  $v$ , isto é, troca o valor  $v_{[1]}$  com  $v_{[n]}$ ,  $v_{[2]}$  com  $v_{[n-1]}$ , etc.
3. Crie um programa que lê uma matriz  $3 \times 3$  e, em seguida, mostre na tela os valores lidos.
4. Leia uma matriz  $4 \times 4$ , imprima a matriz e retorne a localização (linha e a coluna) do maior valor.

# Referências Bibliográficas



## C: COMO PROGRAMAR

DEITEL, Harvey M.; DEITEL, Paul J. Editora Pearson - 6ª ed. 2011



## Fundamentos da Programação de Computadores

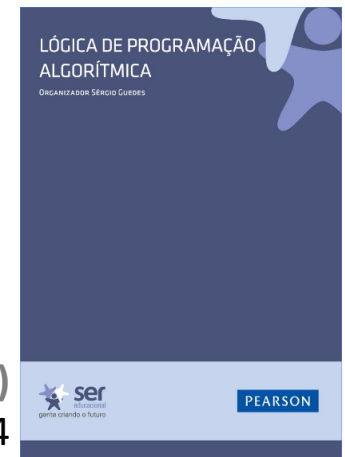
Ascencio, Ana F. G., Campos, Edilene A. V. de, - Editora Pearson

2012



## Lógica de Programação e Estrutura de Dados

Puga, Sandra. Riseti, Gerson. – Ed. Pearson - 2016



## Lógica de Programação Algorítmica (Apostila)

Guedes, Sergio. - Editora Pearson/Ser - 2014