

# Algoritmos - AULA 10

---

## Sumário

1	Vetores.....	1
2	Declaração de vetores.....	1
3	Acesso ao conteúdo de vetores.....	2
4	Inicializando um vetor.....	2
5	Exercícios.....	3
6	Exercícios com strings.....	4

---

## 1 Vetores

Um vetor é uma sequência de vários valores do mesmo tipo, armazenados sequencialmente na memória, e fazendo uso de um mesmo nome de variável para acessar esses valores. Um vetor também pode ser entendido logicamente como uma lista de elementos de um mesmo tipo.

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.

### Exemplo:

Um vetor com 5 posições de memória.

Valores	Valor1	Valor2	Valor3	Valor5	Valor5
Posições (ou índices)	0	1	2	3	4

## 2 Declaração de vetores

A declaração de vetores obedece à mesma sintaxe da declaração de variáveis. A diferença está no valor entre colchetes, que determina quantos elementos ele armazenará (em outras palavras, determina o seu tamanho ou dimensão).

### Sintaxe Básica:

Tipo nomeVetor[tamanho];

### Exemplos:

int valores[20]; //vetor com 20 posições, e todas do tipo int.

float alturas[305]; //vetor com 305 posições, e todas do tipo float

### 3 Acesso ao conteúdo de vetores

Para acessar um determinado elemento de um vetor precisamos o índice do referido vetor, que deve ser indicado entre colchetes. Ou seja, utiliza uma referência de memória (normalmente uma variável do tipo vetor) e um número inteiro (o índice). Ele retorna uma referência para o elemento correspondente ao índice. O tipo do valor retornado é o mesmo tipo da declaração do vetor.

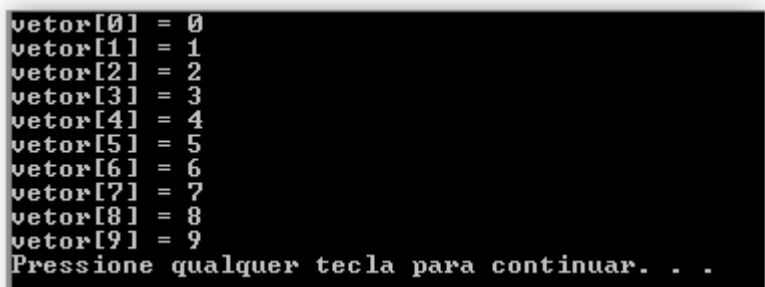
Por exemplo, para atribuir o valor 3 na primeira posição do vetor, escrevemos:  
`vetor[0] = 3;`

É muito comum utilizar a estrutura de repetição `for` para percorrer todos os elementos de um vetor.

**Exemplo:**

Mostrar todos os elementos de um vetor de 10 elementos:

```
#include <stdlib.h>
#include <stdio.h>
int main ( ) {
    int indice;
    int vetor[10];
    for (indice = 0; indice < 10; indice++) {
        vetor[indice]=indice;
        printf("vetor[%d] = %d\n", indice,vetor[indice]);
    }
    system("pause");
    return 0;
}
```



```
vetor[0] = 0
vetor[1] = 1
vetor[2] = 2
vetor[3] = 3
vetor[4] = 4
vetor[5] = 5
vetor[6] = 6
vetor[7] = 7
vetor[8] = 8
vetor[9] = 9
Pressione qualquer tecla para continuar. . .
```

### 4 Inicializando um vetor

Na declaração, pode-se definir o valor inicial de cada elemento de um vetor. A sintaxe é semelhante à declaração de uma variável comum com valor inicial, mas os elementos são listados entre chaves `{ e }` e separados por vírgula.

**Sintaxe Básica:**

`tipo variável[n] = { elem0, elem1, elem2, elem3, ... elemn-1 };`

**Exemplo:**

```
#include <stdlib.h>
#include <stdio.h>
int main ( ) {
    int indice;
    int vetor[]={5,7,9,11};
    for (indice = 0; indice < 4; indice++) {
        printf("vetor[%d] = %d\n", indice,vetor[indice]);
    }
    system("pause");
    return 0;
}
```

```
vetor[0] = 5
vetor[1] = 7
vetor[2] = 9
vetor[3] = 11
Pressione qualquer tecla para continuar. . .
```

**5 Exercícios**

- Gere 50 números aleatórios entre 0 e 99 e armazene-os em um vetor. Mostre cada um dos valores armazenados.
- Gere 30 números aleatórios entre 0 e 999 e armazene-os em um vetor. Mostrar os valores armazenados nas posições 1, 15, 29.
- Crie um vetor com 100 posições e armazenar valores aleatórios. Mostre todos os valores armazenados usando a tabulação ( \t ). Identifique e mostre o maior valor gerado e sua posição no vetor. Dica: inicialize uma variável com um valor muito pequeno e teste se cada elemento do vetor é maior que esse valor, se for a variável passará a assumir o maior valor.
- Gere um vetor com 10 posições de memória. O valor armazenado em cada posição do vetor deve ser igual ao índice da respectiva posição vezes 2.

Exemplo:

Posição do vetor	0	1	2	3	4	5	6	7	8	9
Valor armazenado	0	2	4	6	8	10	12	14	16	18

- Gere 10 números aleatórios entre 0 e 99 e armazene em um vetor, conte quantos desses números são pares, mostre a respectiva posição do vetor e o número par armazenado.
- Desenvolva um programa que: lê 5 valores e armazena no vetor A, lê outros 5 valores e armazena no vetor B e cria um vetor C com 10 posições, onde os cinco primeiros valores são os valores do vetor A e os últimos 5 valores os valores do vetor B. Mostre o vetor C.

Exemplo:

vetor A

5      19      23      45      12

vetor B

18      27      1      18      50

vetor C

5      19      23      45      12      18      27      1      18      50

- g) Dado dois vetores, A (5 elementos) e B (8 elementos), faça um programa em C que imprima todos os elementos comuns aos dois vetores, ou seja, que pertença tanto ao vetor A quanto ao vetor B.
- h) Crie um teclado musical a partir das teclas do seu computador. Sabemos que a escala musical comumente utilizada, chamada escala temperada, divide cada oitava em 12 semitons e a cada oitava a frequência varia de um fator 2, sendo a razão entre as frequências de semitons vizinhos igual a  $2^{1/12}$ . Assim, com a definição da frequência de uma nota, toda a escala musical é construída e o padrão é o Lá fundamental, 440Hz. Ou seja, para obter a escala musical devemos multiplicar cada um dos tons e semitons por  $2^{1/12}$ . Assim, podemos obter a seguinte escala (com arredondamento):

	Dó	Ré	Ré#	Mi	Fá	Fá#	Sol	Sol#	Lá	La#	Si	Dó#
1ª oitava	277	294	311	330	349	370	392	415	440	466	494	523
Uma oitava acima	554	587	622	659	698	740	784	831	880	932	988	1047

Dicas:

- Abra um novo arquivo fonte e salve ele como sendo do tipo C e não C++
- Crie um vetor com 12 posições para armazenar as frequências das notas musicais
- Inicialize a primeira nota com o valor 554, para que o Lá seja algo em torno de 880 Hz
- Use um laço de repetição e atribua os valores das frequências para as demais notas musicais, multiplicando a nota musical anterior por 2 elevado a  $1/12$
- Mostre a escala das notas musicais e veja se os valores estão próximos da 3ª linha da tabela acima
- Crie um laço infinito que será encerrado apenas se o usuário digitar ENTER (tecla igual a 13 na Tabela ASCII). Utilize a função `getch` da biblioteca `conio.h` para ler a tecla digitada. Exemplo: `tecla = getch( );`
- Utilize a função `beep` da biblioteca `windows.h` para gerar o som: `beep(frequência, duração);`
- Atribua às teclas do PC a execução do beep com uma determinada frequência. Adicionalmente, mostre o nome da nota que está sendo tocada. Utilize um `if` para cada possibilidade, sem o `else`, ou utilize o `switch` (matéria extra, para quem quiser correr atrás).

## 6 Exercícios com strings

- i) Leia o primeiro nome de uma pessoa e mostre esse nome na ordem inversa. Dica: você pode usar a função `strlen( )` da biblioteca `string.h` que verifica conta quantos caracteres foram armazenados em um string (procure saber como essa função funciona).  
Exemplo:  
LIMA  
AMIL
- j) Desenvolva um programa que lê uma frase e imprime somente as vogais da respectiva frase.
- k) Desenvolva um programa que define um login e uma senha para um determinado usuário. Depois, o programa deve pedir o login e a senha e verificar se os mesmos são

iguais aos valores armazenados anteriormente. Dica: você deve comparar caractere a caractere dos vetores que armazenaram o login e a senha.

- l) Desenvolva um algoritmo que valida o número de um CPF qualquer. O CPF é composto por onze algarismos, onde os dois últimos são chamados de dígitos verificadores, ou seja, os dois últimos dígitos são criados a partir dos nove primeiros. O cálculo é feito em duas etapas utilizando o módulo de divisão 11.

Para exemplificar melhor iremos calcular os dígitos verificadores de um CPF hipotético, por exemplo, 222.333.666-XX.

### Calculando o Primeiro Dígito Verificador

O primeiro dígito é calculado com a distribuição dos dígitos colocando-se os valores 10,9,8,7,6,5,4,3,2 conforme a representação abaixo:

2	2	2	3	3	3	6	6	6
10	9	8	7	6	5	4	3	2

Na seqüência multiplicaremos os valores de cada coluna, confira:

	2	2	2	3	3	3	6	6	6
x	10	9	8	7	6	5	4	3	2
=	20	18	16	21	18	15	24	18	12

Em seguida efetuaremos o somatório dos resultados (20+18+...+18+12), o resultado obtido (162) será dividido por 11. Considere como quociente apenas o valor inteiro, o resto da divisão será responsável pelo cálculo do primeiro dígito verificador.

Vamos acompanhar: 162 dividido por 11 obtemos 14 de quociente e 8 de resto da divisão. Caso o resto da divisão seja menor que 2, o nosso primeiro dígito verificador se torna 0 (zero), caso contrário subtrai-se o valor obtido de 11, que é nosso caso, sendo assim nosso dígito verificador é 11-8, ou seja, 3 (três), já temos parte do CPF, confira: 222.333.666-3X.

### Calculando o Segundo Dígito Verificador

Para o cálculo do segundo dígito será usado o primeiro dígito verificador já calculado. Montaremos uma tabela semelhante a anterior só que desta vez usaremos na segunda linha os valores 11,10,9,8,7,6,5,4,3,2 já que estamos incorporando mais um algarismo para esse cálculo. Veja:

2	2	2	3	3	3	6	6	6	3
11	10	9	8	7	6	5	4	3	2

Na próxima etapa faremos como na situação do cálculo do primeiro dígito verificador, multiplicaremos os valores de cada coluna e efetuaremos o somatório dos resultados obtidos: 22+20+18+24+21+18+30+24+18+6=201.

	2	2	2	3	3	3	6	6	6	3
x	11	10	9	8	7	6	5	4	3	2
=	22	20	18	24	21	18	30	24	18	6

Agora pegamos esse valor e dividimos por 11. Considere novamente apenas o valor inteiro do quociente, e com o resto da divisão, no nosso caso 3, usaremos para o cálculo do segundo dígito verificador, assim como na primeira parte.

Caso o valor do resto da divisão seja menor que 2, esse valor passa automaticamente a ser zero, que não é o nosso caso, no caso em questão é necessário subtrair o valor obtido de 11 para se obter o dígito verificador, ou seja,  $11 - 3$  (que foi o resto da divisão).

Neste caso chegamos ao final dos cálculos e descobrimos que os dígitos verificadores do nosso CPF hipotético são os números 3 e 8, portanto o CPF ficaria assim: 222.333.666-38.