VETORES PROGRAMAÇÃO

PROJETO

AMPERE

VETORES / ARRAYS

O QUE SÃO? O QUE COMEM? ONDE VIVEM?

"Um vetor é um conjunto de elementos consecutivos, todos do mesmo tipo, que podem ser acessados individualmente a partir de um único nome."

Também chamados de *arrays*, vetores podem ser pensados como "listas" que agrupam vários elementos de um mesmo tipo.

VETOR UNIDIMENSIONAL

EXEMPLO

Conjunto de comissões mensais associadas a um determinado empregado ao longo de um ano.

12 000	5 000	2 300	1 230	7 400	...

Um vetor de *int*. Cada posição corresponde ao valor recebido em um mês, como se fosse uma tabela.

VETOR MULTIDIMENSIONAL

EXEMPLO

Um jogo da velha. É um vetor de *char* com duas dimensões, como se fosse uma matriz.

x		o
	x	
		o

COMO DECLARAR?

SINTAXE PARA UMA DIMENSÃO

Vetores são declarados de forma muito semelhante que variáveis simples, porém com um elemento a mais.

A sintaxe de declaração de um vetor unidimensional é como a seguir:

tipo nome_da_variável[no_de_elementos];

COMO DECLARAR?

EXPLICAÇÃO DA SINTAXE

tipo nome_da_variável[no_de_elementos];

- Tipo: tipo de dados dos elementos do vetor
- Nome_da_variável: nome pelo qual esse vetor vai ser conhecido (*int*, *char*, *float*, etc)
- No_de_elementos: valor inteiro constante que indica quantos elementos terá o vetor

COMO DECLARAR?

EXEMPLO DA SINTAXE

int g[20];

Nesse caso, g é um vetor de *int* com 20 elementos. Ou seja, é um vetor com 20 posições para que sejam armazenados números inteiros.

ELEMENTOS DO VETOR

COMO ACESSÁ-LOS?

Os elementos de um vetor podem ser identificados por um mesmo nome (aquele com que o vetor foi declarado) e um índice numérico único para cada um.

O índice varia entre 0 e n-1 (em que n é o número de elementos do vetor) e indica a posição de cada elemento.

ELEMENTOS DO VETOR

COMO ACESSÁ-LOS?

Por exemplo, se temos um vetor "int arr[6];" – ou seja, um vetor de *int* com 6 elementos chamado *arr* –, podemos acessar o elemento de índice i fazendo arr[i], desde que 0 <= i < n.

ELEMENTOS DO VETOR

COMO ATRIBUIR VALORES A ELES?

Para atribuir valores aos elementos de um vetor, podemos utilizar a sintaxe de acesso que acabamos de aprender.

Por exemplo, para colocar o valor "123" na primeira posição de *arr*, faríamos:

```
arr[0] = 123;
```

ELEMENTOS DO VETOR

COMO ATRIBUIR VALORES A ELES?

Para colocar na última posição de *arr* o dobro do valor do primeiro elemento, faríamos:

```
arr[5] = arr[0] * 2;
```

Para colocar na terceira posição de *arr* a soma do primeiro com o último elemento, faríamos:

```
arr[2] = arr[0] + arr[5];
```

ELEMENTOS DO VETOR

O QUE É CARGA INICIAL AUTOMÁTICA?

Tal qual as variáveis com que já estamos acostumados, os vetores quando criados contêm lixo em cada uma de suas posições.

E, da mesma forma, é possível iniciar automaticamente todos os elementos de um vetor.

ELEMENTOS DO VETOR

SINTAXE DE CARGA INICIAL AUTOMÁTICA

Para iniciar os elementos de um vetor de forma mais compacta, utilizamos a seguinte sintaxe:

$$tipo\ var[n] = \{ valor_0 , valor_1, valor_2 , \dots , valor_{n-1} \}$$

ELEMENTOS DO VETOR

EXEMPLO DE CARGA INICIAL AUTOMÁTICA

Por exemplo, os dois blocos de código abaixo são equivalentes.

```
char vogal[5] = { 'a', 'e', 'i', 'o', 'u' };
```

```
char vogal[5];      vogal[0] = 'a';      vogal[1] = 'e';
vogal[2] = 'i';     vogal[3] = 'o';      vogal[4] = 'u';
```

ELEMENTOS DO VETOR

CARGA INICIAL AUTOMÁTICA

"Se um vetor for declarado com n elementos e forem colocados apenas k valores (k < n) em sua carga inicial, os primeiros k elementos serão iniciados com os respectivos valores e os demais serão iniciados com o valor 0."

ELEMENTOS DO VETOR

CARGA INICIAL AUTOMÁTICA

Dessa forma, as duas linhas a seguir são equivalentes.

```
int v[10] = {10, 20, 30};
```

```
int v[10] = {10, 20, 30, 0, 0, 0, 0, 0, 0, 0};
```

ELEMENTOS DO VETOR

CARGA INICIAL AUTOMÁTICA

Caso defina-se também a carga inicial de um vetor junto à declaração, é possível omitir o número de elementos da declaração (deixando os colchetes vazios).

Dessa forma, o compilador vai criar um vetor com mesma quantidade de elementos e cargas iniciais.

ELEMENTOS DO VETOR

CARGA INICIAL AUTOMÁTICA

Por exemplo, a linha abaixo vai criar um vetor *arr* com 3 elementos (1.6, 2.3 e 3.9, respectivamente):

float arr[ ] = { 1.6, 2.3, 3.9 };

E a linha abaixo é inválida e, portanto, vai provocar um erro de compilação:

float arr[ ];

EXERCÍCIO 01

VETORES UNIDIMENSIONAIS

Escreva um programa capaz de realizar a soma entre dois vetores $v_1 = (x_1, y_1, z_1)$ e $v_1 = (x_2, y_2, z_2)$, em que x_i, y_i e z_i serão fornecidos pelo usuário.

Ex: (2, 5, 2) + (7, 5, 4) = (9, 10, 6)

EXERCÍCIO 02

VETORES UNIDIMENSIONAIS

Um atleta está se preparando para uma competição. Para treinar, este pretende dar N voltas em um circuito, cronometrando seu tempo em cada volta.

Desenvolva um programa que receba um número natural N responsável por representar a quantidade de voltas que o atleta correu naquele dia, seguido de N

EXERCÍCIO 02

VETORES UNIDIMENSIONAIS

pontos flutuantes que representam seu tempo em cada volta.

Como saída, seu programa deve retornar o tempo total que o atleta correu naquele dia, além do tempo médio que levou por volta.

EXERCÍCIO 03

VETORES UNIDIMENSIONAIS

Escreva um programa que lê um número inteiro n, n números inteiros e um número inteiro x. E imprima:

a. Quantas vezes x apareceu entre os n inteiros
b. Os n inteiros na ordem inversa à que foram inseridos
c. Os n inteiros em ordem crescente

VETORES / ARRAYS

MÚLTIPLAS DIMENSÕES

Não existe limite para o número de dimensões que um vetor pode conter, tendo em vista que um vetor com n dimensões é declarado através da sintaxe abaixo:

$$tipo\ vetor[dim_0][dim_1][dim_2][...][dim_{n-1}];$$

OBS: os índices de cada dimensão começam no 0.

VETORES / ARRAYS

MÚLTIPLAS DIMENSÕES

Pode-se dizer que cada dimensão está "contida" na dimensão anterior.

Por exemplo, na prática um vetor bidimensional ("matriz") é, na verdade, um vetor de vetores. Isto é, um vetor cujos elementos também são vetores. O mesmo vale para vetores com mais dimensões.

VETORES / ARRAYS

MÚLTIPLAS DIMENSÕES

Assim, a declaração abaixo vai gerar um vetor com n dimensões, sendo que a dimensão 0 é um vetor de dim_1 elementos; a dimensão 1 é um vetor de dim_2 elementos; e assim por diante.

tipo vetor$[dim_0][dim_1][dim_2][\ldots][dim_{n-1}]$;

VETORES / ARRAYS

MÚLTIPLAS DIMENSÕES

Os elementos de um vetor multidimensional podem ser acessados com o uso de múltiplos pares de colchetes.

Por exemplo, a linha abaixo vai acessar o primeiro elemento da terceira dimensão do vetor:

vetor[0][0][0];

VETORES / ARRAYS

EXEMPLO DE MATRIZ: JOGO DA VELHA

char velha[3][3];

O vetor *velha* possui três posições e cada uma delas é formada por um vetor de três caracteres.

VETORES / ARRAYS

EXEMPLO DE MATRIZ: JOGO DA VELHA

Pode-se pensar que cada elemento do vetor (*velha[0]*, *velha[1]* e *velha[2]*) é uma "linha" e cada uma dessas linhas possui três colunas (*velha[i][0]*, *velha[i][1]* e *velha[i][2]*).

Dessa forma, o vetor *velha* possui, efetivamente, espaço para armazenar 9 caracteres.

VETORES / ARRAYS

EXEMPLO DE MATRIZ: JOGO DA VELHA

	[0]	[1]	[2]
velha[0]	(velha[0][0])	(velha[0][1])	(velha[0][2])
velha[1]	(velha[1][0])	(velha[1][1])	(velha[1][2])
velha[2]	(velha[2][0])	(velha[2][1])	(velha[2][2])

VETORES / ARRAYS

EXEMPLO DE MATRIZ: JOGO DA VELHA

Assim, temos:

- *velha* → vetor bidimensional de caracteres 3x3
- *velha[i]* → vetor de 3 caracteres localizado na posição i de *velha* (é a linha i deste)
- *velha[i][j]* → caractere localizado na linha i e coluna j do vetor *velha*

VETORES / ARRAYS

EXEMPLO DE MATRIZ: JOGO DA VELHA

Assim, temos:

- *velha* → vetor bidimensional de caracteres 3x3
- *velha[i]* → vetor de 3 caracteres localizado na posição i de *velha* (é a linha i deste)
- *velha[i][j]* → caractere localizado na linha i e coluna j do vetor *velha*

VETORES / ARRAYS

EXEMPLO DE MATRIZ: JOGO DA VELHA

A carga inicial de um vetor multidimensional pode ser feita de duas formas.

Para exemplificar, vamos novamente declarar o vetor *velha*, mas, dessa vez, iniciá-lo com espaços em branco (' ') em todas as posições.

VETORES / ARRAYS

EXEMPLO DE MATRIZ: JOGO DA VELHA

1. Incluindo todos os valores do vetor dispostos dentro de chaves (neste caso, 3x3 = 9 valores):
   char velha[3][3] = { ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ' };

2. Incluindo cada dimensão dentro de suas próprias chaves:
   char velha[3][3] = { { ' ', ' ', ' ' }, { ' ', ' ', ' ' }, { ' ', ' ', ' ' } };

EXERCÍCIO 04

VETORES BIDIMENSIONAIS (MATRIZES)

Implemente um algoritmo que receba do usuário duas matrizes 3x3 A e B e imprima:

a. A + B

b. AB

EXERCÍCIO 05

VETORES BIDIMENSIONAIS (MATRIZES)

Uma civilização precisa eleger seu novo representante em uma eleição democrática.

Nela, cada eleitor indica se concorda ou não com um certo candidato – votando 1 se concordarem e 0 caso contrário.

EXERCÍCIO 05

VETORES BIDIMENSIONAIS (MATRIZES)

Seu papel é colher os votos da população e apresentar um relatório com os votos recebidos por cada candidato.

Serão fornecidos dois números naturais P e E referentes à quantidade de candidatos e eleitores, respectivamente.

EXERCÍCIO 05

VETORES BIDIMENSIONAIS (MATRIZES)

Em seguida, serão fornecidas também E linhas com P números (1 ou 0) separados por espaço. Cada linha representa o voto de um eleitor.

Como saída, seu programa deve imprimir os candidatos e a quantidade de votos que cada um recebeu.

EXERCÍCIO 05

VETORES BIDIMENSIONAIS (MATRIZES)

Exemplo de entrada:

- 2
- 4
- 1 0
- 0 1
- 1 1
- 0 1

Exemplo de saída:

- Candidato 1: 2 votos
- Candidato 2: 3 votos