Vetor e Matriz em C

Vetores

Um vetor é uma variável composta homogênea unidimensional formada por uma sequência de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome) e alocadas sequencialmente na memória.

Como as variáveis têm o mesmo nome, o que as distingue é um índice, que referencia sua localização dentro da sequência.

Os índices utilizados na linguagem C/C++ para identificar as posições de um vetor começam sempre em 0 (zero) e vão até o tamanho do vetor menos uma unidade.

A linguagem C/C++ não tem verificação de limites em vetores (o compilador não acusa este tipo de erro), o programador é quem deve prover a verificação dos limites onde for necessário, pois a ultrapassagem deste limite pode resultar nos mais variados erros durante a execução do programa.

Declaração de Vetor

Em C/C++, os vetores são identificados pela existência de colchetes logo após o nome da variável no momento da declaração. Dentro dos colchetes deve-se colocar o número de posições do vetor.

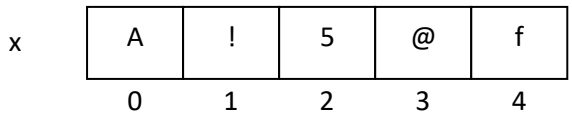
Exemplo 01) float nota[6];

A variável nota contém seis posições, começando pela posição 0 e indo até 5 (6-1). Como foi declarada sendo do tipo float, em cada posição poderão ser armazenados números reais, por exemplo:

nota	5.0	7.5	6.2	4.0	2.0	3.0
	0	1	2	3	4	5

Exemplo 02) char x[5];

A variável x contém cinco posições (de 0 a 4). Em cada posição poderão ser armazenados caracteres, conforme especificado pelo tipo char na declaração:



Manipulação de Vetor

Atribuição

```
vet[0] = 1;
/* atribui o valor 1 ao primeiro elemento do vetor vet */
```

Entrada de dados

```
for (i=0; i<10; i++)
        scanf("%f",&A[i]);
/* O usuário entrará com o valor dos 10 elementos do vetor A do tipo float */
```

OU

```
for (i=0; i<10; i++)
        cin >> A[i];
```

Escrevendo os elementos

```
for (i=0; i<10; i++)
        printf("%5.2f ",A[i]);
```

OU

```
for (i=0; i<10; i++)
        cout << A[i];
```

Exemplo: Dados dois vetores a e b de dimensão 10, calcule o vetor c obtido da soma de a e b.

```
#include <iostream>

using namespace std;

#define TAM 10

main() {

	float a[TAM], b[TAM], c[TAM];

	int i;

	cout << "\nDigite o vetor a:" << endl;

	for (i=0; i<TAM; i++) {

		cout << "a[" << i << "] = ";

		cin >> a[i];

	}

	cout << "\nDigite o vetor b:" << endl;

	for (i=0; i<TAM; i++) {

		cout << "b[" << i << "] = ";

		cin >> b[i];

	}

	for (i=0; i<TAM; i++) {

		c[i] = a[i] + b[i];

	}

	cout << "\nNovo Vetor c = a + b:\n" << endl;

	for (i=0; i<TAM; i++) {

		cout << "c[" << i << "] = " << c[i] << endl;

	}

}
```

Matrizes

São estruturas de dados homogêneas multidimensionais que necessitam de mais de um índice para a individualização (referência) de seus elementos.

A linguagem C/C++ permite declarar matrizes unidimensionais (vetores), bidimensionais e multidimensionais. As matrizes mais utilizadas são as bidimensionais.

A declaração de uma matriz é feita da forma:

tipo nome_da_variável[dim_1][dim_2]...[dim_N];

Onde:

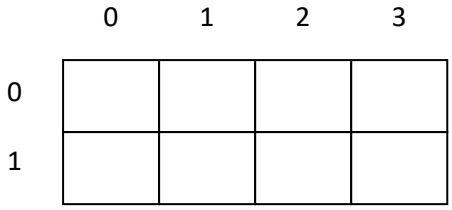
tipo: tipo a que pertencem todos os elementos da matriz.

nome_da_variável: nome dado a variável do tipo matriz.

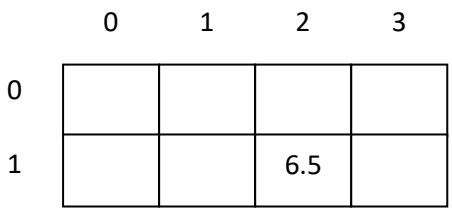
[dim_1][dim_2]...[dim_N]: representam as possíveis dimensões da matriz.

Exemplo) float A[2][4];

A é uma matriz com duas linhas e quatro colunas, cujos elementos são do tipo float. Devemos lembrar que, assim como ocorre com os vetores, os índices começam sempre em zero.



A atribuição de valores a uma matriz é realizada como nos vetores. Assim, A[1][2] = 6.5 representa:



Para entrar com os dados de uma matriz e mostrá-los na tela também procedemos de forma análoga aos vetores, porém, como a matriz possui duas dimensões é necessária a utilização de dois contadores (um para a linha e outro para a coluna).

Para a matriz A declarada anteriormente temos:

```
#include <iostream>

using namespace std;

#define QTDL 2

#define QTDC 4

main() {

	float a[QTDL][QTDC];

	int i, j;

	cout << "\nDigite a matriz:" << endl;

	for (i=0; i<QTDL; i++) {

		for (j=0; j<QTDC; j++) {

			cout << "a[" << i << "][" << j << "] = ";

			cin >> a[i][j];

		}

	}

	cout << "\nMatriz:" << endl;

	for (i=0; i<QTDL; i++) {

		for (j=0; j<QTDC; j++) {

			cout << " " << a[i][j];

		}

		cout << endl;

	}

}
```

Iniciação de Matrizes

A linguagem C/C++ permite a iniciação de matrizes (e vetores) no momento da declaração. A forma geral de uma iniciação de matriz é semelhante a de outras variáveis:

tipo nome_da_variável[dim_1][dim_2]...[dim_N] = { lista_de valores };

A lista de valores é uma lista separada por vírgulas de constantes cujo tipo é compatível com o tipo especificado para a matriz. A primeira constante é colocada na primeira posição da matriz, a segunda na segunda posição, e assim por diante.

Exemplo 01) Vetor

```
int x[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

Esta declaração representa: x[0] = 1, x[1] = 2, ..., x[9] = 10.

Exemplo 02) Matriz

```
int A[5][2] =   {
                    1, 5,
                    2, 6,
                    3, 7,
                    4, 8,
                    5, 9
                };
```

Ou seja, A[0][0]=1, A[0][1]=5, A[1][0]=2, A[1][1]=6, ..., A[4][0]=5, A[4][1]=9.

Exercícios

01) Leia uma variável de 100 elementos reais e verifique se existem elementos iguais a 64. Se existirem, escreva as posições em que estão armazenados.

02) Dadas as matrizes A e B, 5x3, determine a matriz C = A + B. Imprima A, B e C.

03) Dado um vetor de N números inteiros (N <= 20), calcule e escreva o somatório dos valores deste vetor.

04) Dada uma matriz 3x6, calcule a soma de cada coluna dessa matriz e armazene esses valores num vetor. Imprima a matriz e o vetor.

05) Dada A(nxm) de elementos inteiros, calcule e imprima a soma dos elementos situados abaixo da diagonal principal de A, incluindo os elementos da própria diagonal principal.