

ALGORITMOS E **PROGRAMAÇÃO**

Me. Matheus de Melo Machado

INICIAR

introdução

Introdução

Ao realizar o desenvolvimento de um programa na linguagem C, é necessário a utilização de métodos de desenvolvimento que agilizem e haja um bom entendimento. A utilização de vetores e matrizes não é fácil, porém é essencial para um bom desempenho, código limpo e legível por outros desenvolvedores. Com o tempo, ao trabalhar com matrizes será mais fácil o entendimento, do que ao ver código com colchetes (pode ser um pouco complexo), definições e até mesmo tipos de matrizes na passagem por parâmetros, funções entre outras coisas. Será demonstrado exemplos que poderão ser um base para a criação de novas ideias com interações com usuários, leituras e carregamentos de informações.

Vetores



Introdução

Podemos definir um vetor como uma sequência de valores do mesmo tipo, no qual estes valores são armazenados na memória e faz o uso de um mesmo nome da variável, ou seja, com apenas um nome de uma variável é possível acessar inúmero valores.

É possível afirmar também que um vetor pode ser entendido de uma maneira lógica como uma lista de elementos, sendo todos do mesmo tipo. Cada elemento da lista pode ser acessado através de um número inteiro dado como índice.

A lista possui elementos que com suas posições vai de 0 até $n-1$, no qual n é a quantidade de elementos do vetor. Por exemplo, se um vetor possui uma lista de números inteiros que 1 até 9, os índices desse vetor irá começar em 0 e irá até 8.

O valor de n atribuído ao vetor anteriormente pode ser chamado de dimensão ou tamanho do vetor. Outra característica do vetor é que ele possui tamanho fixo durante a execução do programa, sendo definido na declaração. A partir do momento que é executado não poderá diminuir ou até mesmo aumentar o tamanho de um vetor.

Os elementos do vetor como mencionado anteriormente, estes são armazenados de maneira sequencial na memória do computador, ou seja, em um vetor de 10 posições, se cada posição ocupa 4 bytes de memória, o vetor de 10 posições irá ocupar 40 bytes de memória de uma só vez.

É muito importante lembrar que o vetor se inicia em 0, pois a maior parte dos erros e também mais comum é que a posição número 1 não é o índice 1 e sim o 0.



Figura 4.1 - Exemplificação de um vetor com tamanho 10

Fonte: Autor

Declaração

Podemos visualizar a declaração de um vetor em C.

```
#include <stdio.h>

int main()
{
    int vetor[10];

    return 0;
}
```

Observamos que é bem simples a declaração de um vetor, no qual é inserido o tipo do vetor, o nome utilizado na variável e também a quantidade de posições que esse vetor irá ter.

Atribuição

Podemos ver um exemplo de atribuição simples de valor também imprimindo na tela o índice do vetor.

```
#include <stdio.h>

int main()
{
    float vetor[10];

    // Criação de um vetor realizando a declaração de
    valores.
    // float vetor[10] =
    {0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0};

    vetor[2] = 3;

    printf("O valor do vetor na posição 0 é %d\n",
    vetor[0]);

    printf("O valor do vetor na posição 1 é %d\n",
vetor[1]);

    printf("O valor do vetor na posição 2 é %.20f",
    vetor[2]);

    return 0;
}
```

No exemplo descrito anteriormente, podemos ver que foi atribuído o valor inteiro 3 no índice do vetor 2. Ao realizar a impressão conforme mostrado nas 3 linhas do printf, a única linha que irá imprimir o número 3 é a 3°. As linhas onde está realizando a impressão do índice 0 e 1, como não foi declarado valor, a impressão é de um valor qualquer da memória.

atividade

Atividade

Conforme visto em aula, podemos realizar a declaração de um vetor com inúmeras posições, porém é necessário incluir o tipo que o vetor será. Ao realizar a declaração de um vetor do tipo float com 5 posições sem qualquer atribuição de valor e ao executar o comando printf no índice 3, o que será exibido na tela?

- ☐ **a)** Valor qualquer da memória.
 - ☐ **b)** 5.
 - ☐ **c)** 3.
 - ☐ **d)** 2.
 - ☐ **e)** Não será exibido nada.
-

Vetores

Carregando

Podemos carregar um vetor, ou seja, carregar valores dentro de um vetor com a utilização de um laço de repetição. É possível definir o tamanho do vetor em tempo de execução.

```
#include <stdio.h>
int main()
{
    int i,tamanho;

    printf("Digite um número inteiro para definir o tamanho do vetor.");

    scanf("%d\n", &tamanho);

    int vetor[tamanho];
```



```

// ...
// ...

for (i = 0; i < tamanho; i++) {
    vetor[i] = i;

    printf("O valor do vetor no índice %d é de
%d.\n", i, vetor[i]);
}
}

```

No exemplo mostrado anteriormente, podemos observar que foi criado um vetor do tipo inteiro e também criamos uma variável auxiliar para percorrer um laço de repetição.

Neste laço informamos que em cada posição do índice do vetor, este terá o mesmo valor de acordo com o tamanho do vetor criado. Será necessário informar o tamanho do vetor, pois este será criado em tempo de execução.

Lendo

Também podemos realizar a interação com o usuário, ou seja, solicitamos que seja escrito 5 números inteiros, e através dos números digitado, podemos guardar em nosso vetor.

```

#include <stdio.h>
int main()
{
    int vetor[5];
    int indice;

    printf("Escreva 5 números inteiros: ");

    for (indice = 0; indice < 5; indice++) {
        scanf("%d", &vetor[indice] );
    }
}

```

```
        return 0;
    }
```

Como podemos observar, é possível ler os valores com o comando `scanf` com a ajuda do `for`, podemos ler os valores e atribuir a cada posição.

Mostrando

Por fim, podemos deixar o nosso programa um pouco mais complexo, no qual além da interação do usuário, iremos mostrar os números digitados com o seguinte código.

```
#include <stdio.h>
int main()
{
    int vetor[5];
    int indice;

    printf("Escreva 5 números inteiros: ");

    for (indice = 0; indice < 5; indice++) {
        scanf("%d", &vetor[indice] );
    }

    printf("Valores em ordem:\n");
    for (indice = 0; indice < 5; indice++) {
        printf("%d ", vetor[indice]);
    }
    return 0;
}
```

Ao realizar a execução do código, iremos ter como retorno os 5 números inteiros digitados pelo usuários, caso os números digitado seja 1,2,3,4 e 5, o retorno será de 1 2 3 4 5.

atividade

Atividade

Podemos imaginar a criação de um vetor de tamanho 5 com o seu tipo sendo char. Para realizarmos a interação com o usuário será exibido uma mensagem para o usuário escrever uma frase. Conforme o código abaixo, qual será a saída desse vetor se a palavra digitada pelo o usuário for “Programação é legal”.

```
-----  
#include <stdio.h>  
int main()  
{  
    char vetor[5];  
    int indice;  
    printf("Escreva uma frase: ");  
    for (indice = 0; indice < 5; indice++) {  
        scanf("%c", &vetor[indice] );  
    }  
  
    printf("O resultado é:\n");  
  
    for (indice = 0; indice < 5; indice++) {  
        printf("%c ", vetor[indice]);  
    }  
  
    return 0;  
}
```

```
-----
```

Assinale a alternativa correta.

- ☐ **a)** P r o g r a m a ç ã o.
- ☐ **b)** P r o.
- ☐ **c)** P r o g r.

- ☐ **d)** Programa.
 - ☐ **e)** Nenhuma das anteriores.
-

Matrizes

Introdução

Matriz é uma generalização de um vetor, ou seja, uma matriz é uma tabela de vários valores que possuem o mesmo tipo, e são armazenados sequencialmente assim como um vetor. Contudo, entendemos que os vetores são matrizes de apenas uma dimensão. As posições na memória são criadas sequencialmente a partir da definição de uma matriz.

Podemos entender matriz também da seguinte forma.

Uma matriz é uma coleção de variáveis de mesmo tipo, acessíveis com um único nome e armazenados contiguamente na memória.

A individualização de cada variável de um vetor é feita através do uso de índices.

(MÁRCIO SARROGLIA, 2018)

Por fim, a matriz é a junção de vários vetores. Cada elemento da matriz pode ser acessado através de dois índices com valor inteiro. Podemos entender que cada índice um irá se referir a coluna e outro a linha.

A linguagem C define uma matriz como um vetor multidimensional, no qual ambos os elementos são vetores do mesmo tipo e também do mesmo tamanho. Por exemplo, uma matriz que possui um vetor do tipo char com tamanho 5, todos os outros serão iguais.

É importante lembrar que uma matriz pode ter múltiplas dimensões, um exemplo na declaração abaixo.

int matriz[4][4][4];

O número de linhas de uma matriz corresponde ao número de elementos do vetor externamente, e o número de colunas é o tamanho dos vetores internos que irão constituir cada elemento dos vetores externos.

Podemos observar a figura.

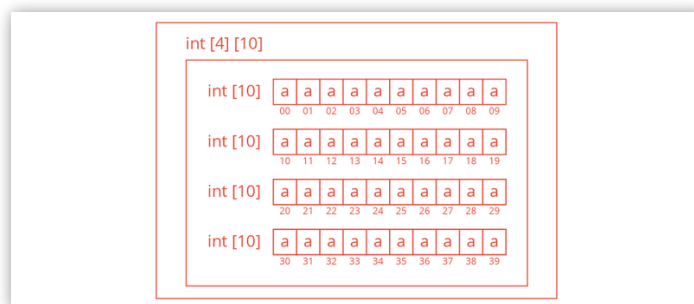


Figura 4.2 - Exemplificação de uma matriz com de 4 linhas e 10 colunas.

Fonte: Autor

Declarando

Podemos realizar a declaração de uma matriz que tenha 3 linhas e 5 colunas do tipo inteiro da seguinte forma.

```
#include <stdio.h>
int main()
{
    int matriz[3][5];
    return 0;
}
```

Também podemos declarar uma matriz de 8 linhas com 15 colunas do tipo char da seguinte maneira.

```
#include <stdio.h>
int main()
{
    char matriz[8][15];
    return 0;
}
```

Atribuindo

Para atribuir valores a uma matriz, iremos utilizar um exemplo simples no qual será criado uma matriz de 3 linhas com 2 colunas do tipo inteiro. Nesta matriz as linhas serão numeradas de 0 a 2. Quando a coluna for o índice 0, o seu valor será índice + 5, quando o seu valor for 1, será índice + 8. O código ficará da seguinte maneira.

	Coluna 0	Coluna 1
Linha 0	$0 + 5 = 5$	$0 + 8 = 8$
Linha 1	$1 + 5 = 6$	$1 + 8 = 9$
Linha 2	$2 + 6 = 7$	$2 + 8 = 10$

Tabela - Demonstração da execução do código em forma de tabela.

Fonte: Autor

```
#include <stdio.h>
int main()
{
    int matriz[3][2];
    int y;
    for (y = 0; y < 3; y++) {
        matriz[y][0] = y + 5;
        matriz[y][1] = y + 8;
    }
    return 0;
}
```

atividade

Atividade

Dado uma matriz do tipo inteiro sendo ela com as dimensões de 10 linhas e 14 colunas, esta pode ser definida como uma tabela não muito simples. Com essa quantidade de informações, mesmo que não utilizada durante a execução do programa será consumida a memória do hardware normalmente. Qual a alternativa que corresponde com a declaração da matriz informada.

- ☐ **a)** `int matriz[14][10]`.
 - ☐ **b)** `float matriz[14][10]`.
 - ☐ **c)** `int matriz[10][14]`.
 - ☐ **d)** `char matriz[10][14]`.
 - ☐ **e)** `int matriz[10,14]`.
-

Matrizes



Carregando

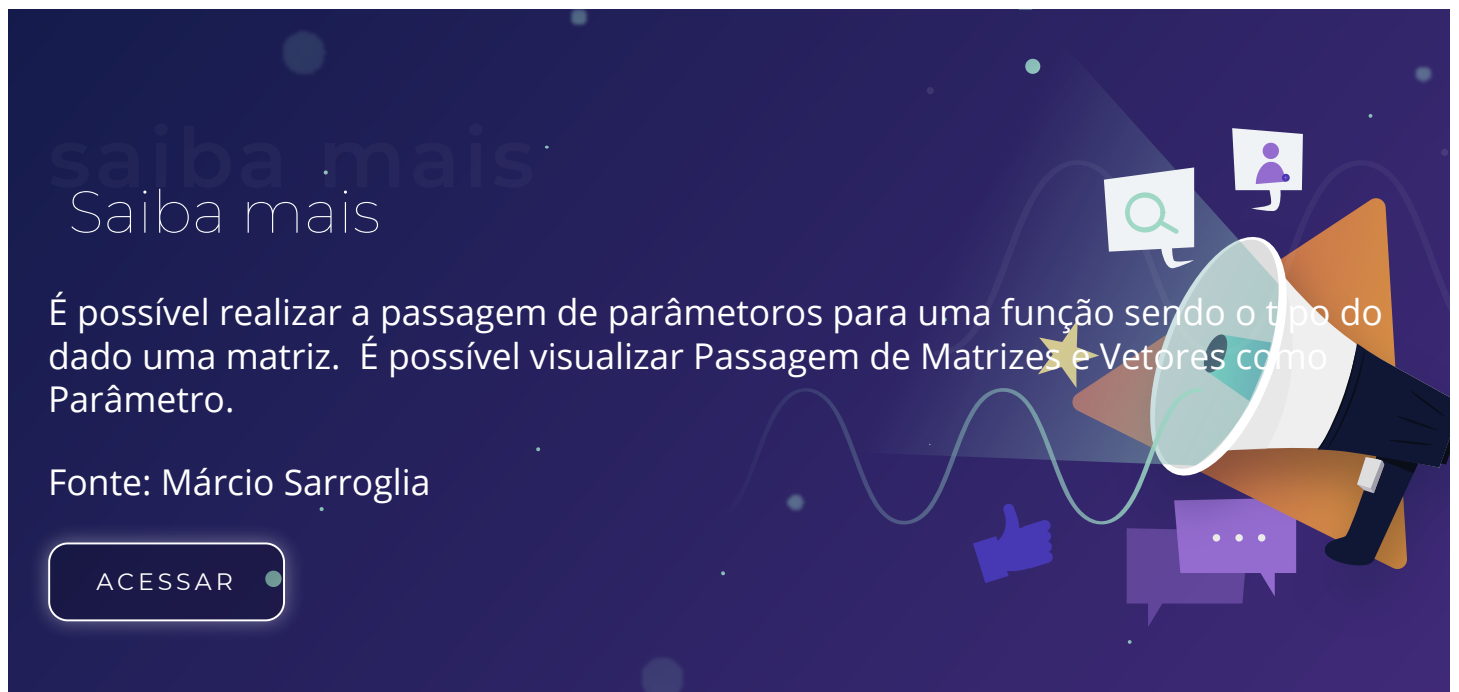
Podemos carregar uma matriz assim como carregamos um vetor com valores. Iremos utilizar o exemplo anterior preenchendo uma matriz de 2 linhas com 3 colunas.

```
#include <stdio.h>
int main()
{
    int matriz[2][3];
    int i;

    for (i = 0; i < 2; i++) {
        matriz[i][0] = i + 2;
        matriz[i][1] = i + 4;
        matriz[i][2] = i + 6;
    }
```

```
        return 0;  
    }
```

Este exemplo é simples, pois apenas realizamos o carregamento de uma matriz de dimensões 2 por 3 e atribuímos os seguintes valores, quando a coluna for índice 0 o valor será o índice + 2. Quando o índice da coluna for 1 o valor será índice + 4. E por fim, quando o índice da coluna for 2 o valor será índice + 6.



saiba mais
Saiba mais

É possível realizar a passagem de parâmetros para uma função sendo o tipo do dado uma matriz. É possível visualizar Passagem de Matrizes e Vetores como Parâmetro.

Fonte: Márcio Sarroglia

[ACESSAR](#)

The banner features a dark blue background with a large megaphone on the right side. Above the megaphone are icons for a magnifying glass and a person. Below the megaphone are icons for a thumbs up and a speech bubble. The text 'saiba mais' is written in a light blue, sans-serif font. The main text is in a white, sans-serif font. The source 'Fonte: Márcio Sarroglia' is in a white, sans-serif font. The button 'ACESSAR' is in a white, sans-serif font with a green dot to its right.

Lendo

Podemos realizar uma interação com o usuário no qual ele irá informar o valor de 2 números inteiros. Na primeira linha da matriz iremos informar os dois números digitados e na 3ª coluna iremos apresentar a soma. Na segunda linha da matriz, iremos realizar o mesmo processo, porém iremos realizar a subtração.

O código ficará da seguinte maneira.

```
#include <stdio.h>
int main()
{
    int matriz[2][3];
    int i;
    int x;
    printf("Digite 2 números inteiros.");
    for (i = 0; i < 2; i++){
        for (j = 0; j < 2; j++){
            scanf("%d",&matriz[i][j]);
        }
    }
    for (i = 0; i < 2; i++) {
        if (i == 0) {
            matriz[i][2] = matriz[i][0] + matriz[i][1];
        } else {
            matriz[i][2] = matriz[i][0] -matriz[i][1];
        }
    }
    return 0;
}
```

O primeiro passo é ler os valores digitados pelo usuário e guardar em um vetor. Após isso percorremos a matriz para guardar os valores e realizar a soma e subtração conforme definido pela regra anteriormente.

Mostrando

Utilizando o exemplo anterior, para realizar a impressão será adicionado dois laços de repetição, um dentro do outro para listar as informações que precisamos.

Observe o código.

```
#include <stdio.h>
int main()
{
    int matriz[2][3];
    int valores[2];
    int i;
    int x;

    printf("Digite 2 números inteiros.");

    for (i = 0; i < 2; i++){
        scanf("%d",&valores[i]);
    }

    for (i = 0; i < 2; i++) {
        matriz[i][0] = valores[0];
        matriz[i][1] = valores[1];

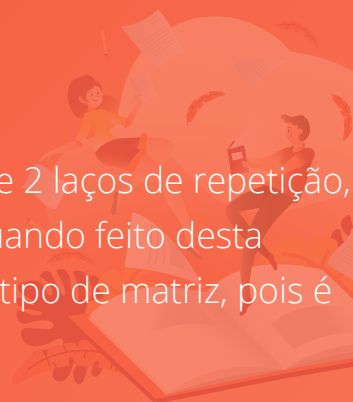
        if (i == 0) {
            matriz[i][2] = valores[0] + valores[1];
        } else {
            matriz[i][2] = valores[0] - valores[1];
        }
    }

    for (i = 0; i < 2; i++) {
        for (x = 0; x < 3; x++) {
            printf("Na linha %d o valor da coluna %d  
é de %d\n",i,x,matriz[i][x]);
        }
    }
    return 0;
}
```

reflita

Reflita

Para realizar a exibição de uma matriz, é necessário a utilização de 2 laços de repetição, uma para exibir as linhas e o outro laço para exibir as colunas. Quando feito desta maneira, não é necessário reescrever código para nenhum outro tipo de matriz, pois é uma forma genérica.



atividade

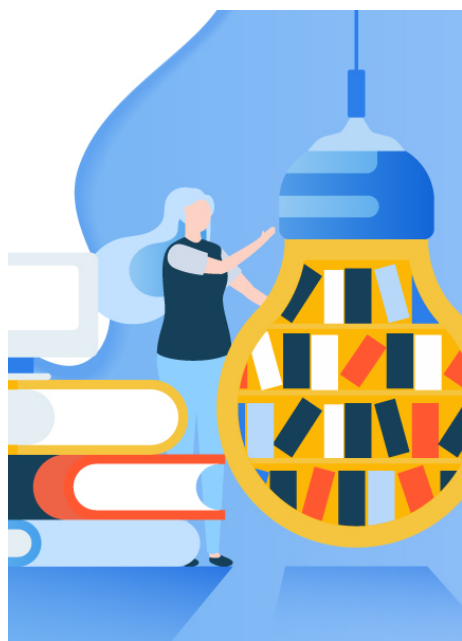
Atividade

Em uma matriz com dimensões de 5 linhas e 3 colunas do tipo inteiro, será necessário realizar a exibição das informações para o usuário. Qual será a melhor maneira de realizar a exibição das informações das linhas de uma maneira, que não seja necessário reescrever o código para quando exibir outra matriz com outras dimensões. Assinale a alternativa correta.

- ☐ **a)** Escrever todas as linhas fixas no código.
 - ☐ **b)** Utilizar 2 laços de repetição, um para exibir as linhas e o outro para exibir as colunas.
 - ☐ **c)** Utilizar 1 laço de repetição e deixar as linhas fixas.
 - ☐ **d)** Utilizar 4 laços de repetição.
 - ☐ **e)** Nenhuma das anteriores.
-

indicações

Material Complementar



LIVRO

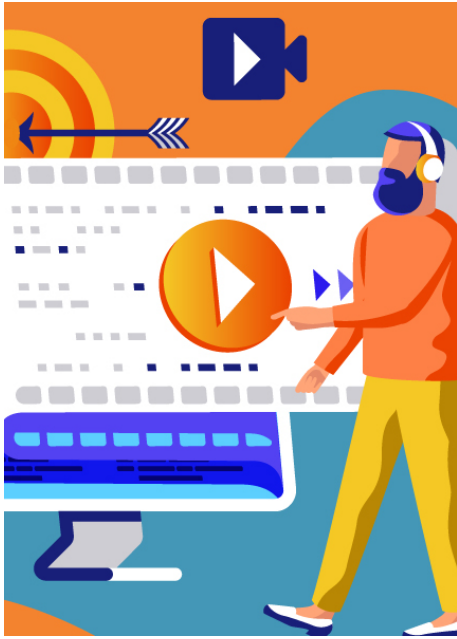
Linguagem C - 2º Edição

Editora: Elsevier

Autor: André Backes

ISBN: 9788535291063

Comentário: O livro mencionado de André Backes é praticamente um curso completo de programação em C, no qual vai do básico ao mais avançado. Pode ser utilizado por atuantes na área de desenvolvimento pois poderá ajudar bastante com funções, métodos e boas práticas utilizadas no desenvolvimento da plataforma.



FILME

Programação C - Aula 10 - Matriz bidimensional - Peterson Lobato

Ano: 2012

Comentário: Para trabalhar com matrizes ou vetores não é fácil, muita das vezes pode se confundir devido aos seus índices como mesmo nome. Para trabalhar com estes assuntos é essencial a utilização de estrutura de repetição para o seu tratamento, exibição de informações entre outras coisas.

TRAILER

conclusão

Conclusão

Com a utilização de matrizes e vetores, podemos observar que é indispensável a utilização de laços de repetição, pois ao termos a necessidade de exibir informações de um array ou uma tabela com inúmeros campos, não podemos escrever o código fixo até mesmo para que não sirva apenas para um caso. Ao utilizarmos a criação de um código de maneira genérica, irá servir para vários casos, desde matrizes simples com dimensões 2 por 2 até para matrizes com dimensões 10 por 30. É necessário o entendimento que os códigos mostrados são apenas uma base simples e que é possível criar códigos muito mais complexos utilizando vários outros recursos que a linguagem C oferece para os desenvolvedores.

referências

Referências Bibliográficas

SARROGLIA, Márcio. Título. **Programação de baixo nível**. Disponível em <<http://www.inf.pucrs.br/~pinho/PRGBAIXONIVEL/>> Acesso em 06 de maio, 2019.

IMPRIMIR

