

**MATRIZ**

Matriz é a junção de vários vetores. Na teoria, vetor é uma matriz de 1 linha e [X] colunas. Ao invés de você criar vários vetores, você pode criar uma matriz que leia várias linhas com colunas. Os itens de uma matriz têm que ser todos do mesmo tipo de dado.

1. Sintaxe da matriz:

Tipo (int, float, char) nome\_da\_matriz[num\_de\_linhas][num\_de\_colunas];

Exemplo:

int numeros[5][2]

1. Para ler uma matriz, é preciso colocar um for dentro de outro for. O primeiro for percorre a linha, o segundo percorre a coluna.

Exemplo:

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 2; j++) {

scanf("%d", &numeros[ i ][ j ]);

    }

}

OBS: i é o número de linhas, j é o número de colunas.

EXEMPLO DE CÓDIGO EM C PARA LER UMA MATRIZ COM 3 LINHAS E 2 COLUNAS:

#include<stdio.h>

int main (){

int matriz[3][3];

printf ("\nDigite valores para os elementos da matriz\n\n");

for (int i=0; i<3; i++){

for (int j=0; j<3; j++){

printf ("\nElemento[%d][%d] = ",i,j);

scanf ("%d", &matriz[i][j]);

}

}

printf("\n\n\*\*\*\*\*\* Saida de Dados \*\*\*\*\*\* \n\n");

for (int i=0; i<3; i++){

for (int j=0; j<3; j++){

printf ("\nElemento[%d][%d] = %d\n", i, j,matriz[ i ][ j ]);

}

}

return(0);

}

**FILA**

**Fila** são estruturas de dados bastante utilizadas na computação, onde o primeiro elemento a ser inserido, será também o primeiro a ser retirado.

Exemplo de código:

#include <stdio.h>

int primeiro\_cadastro = -1;

int first\_cadastrado = 0;

typedef struct {

int pos;

char nome [50];

} Cadastro;

Cadastro vetor [15];

void cadastrar (){

primeiro\_cadastro++;

first\_cadastrado++;

vetor[primeiro\_cadastro].pos=first\_cadastrado;

char nome;

printf ("Nome: ");

scanf ("%c", &nome);

scanf ("%[^\n]", vetor[primeiro\_cadastro].nome);

printf("Cadastrado com sucesso!\n");

}

void listar (){

for (int i=0; i<=primeiro\_cadastro; i++){

printf ("Posição: %d Nome: %s\n", vetor[i-1].pos, vetor[i].nome);

}

}

void excluir (){

for (int i=0; i<primeiro\_cadastro; i++) {

vetor[i] = vetor[i+1];

}

primeiro\_cadastro--;

printf("Excluído\n");

}

int main()

{

int op;

do {

printf ("1- Cadastrar \n2- Listar \n3- Excluir\n4- Sair\n");

scanf ("%d", &op);

switch (op){

case 1:

cadastrar();

break;

case 2:

listar();

break;

case 3:

excluir();

break;

}

} while (op!=4);

    return 0;

}

Na função **cadastrar**, você insere o primeiro elemento:

void cadastrar (){

primeiro\_cadastro++;

first\_cadastrado++;

vetor[primeiro\_cadastro].pos=first\_cadastrado;

char nome;

printf ("Nome: ");

scanf ("%c", &nome);

scanf ("%[^\n]", vetor[primeiro\_cadastro].nome);

printf("Cadastrado com sucesso!\n");

}

Na função **excluir**, você exclui a primeiro elemento inserido na função cadastrar:

void excluir (){

for (int i=0; i<primeiro\_cadastro; i++) {

vetor[i] = vetor[i+1];

}

primeiro\_cadastro--;

printf("Excluído\n");

}

**LISTA**

A lista se inicia de modo vazio, porém vai crescendo com o acrescentamento de  
novos dados inseridos.  
 Uma lista encadeada é um exemplo de uma sequência de uma célula tendo assim cada célula contém uma coisa e todas as coisa são de um tipo só.  
 A célula é vista como um novo tipo de dado e atribui nome a ele.  
 Uma célula c e um ponteiro p para uma célula podem ser declarados assim:  
 celula c;  
 celula \*p;  
 Lista armazena os dados dando nome, sendo um conjunto de estruturas que  
vem a ser chamada de nó que seria a estrutura que armazena as informações que a lista irá gerar.  
 Há dois tipos de lista: ligada e duplamente ligada.  
 Uma lista ligada é uma sequência de nós conectados através de ponteiros para o próximo nó, enquanto uma lista duplamente ligada possui conexões tanto para o próximo quanto para o nó anterior na sequência. Essas estruturas são usadas para implementar estruturas dinâmicas em C, permitindo a inserção, remoção e acesso eficiente aos elementos da lista.

Uma lista encadeada é uma estrutura de dados em que cada elemento, chamado de nó, contém um valor e um ponteiro para o próximo nó na sequência. O último nó da lista sempre aponta para um valor nulo, indicando o final da lista.

Para remover um nó específico da lista, basta ajustar os ponteiros. Não é necessário excluir fisicamente o nó, apenas precisamos fazer com que o nó anterior aponte diretamente para o nó que vem depois do nó que queremos remover.

Dessa forma, o elemento anterior ao nó que desejamos retirar deve apontar diretamente para o elemento que vem depois dele. Isso faz com que o nó que queremos remover seja ignorado na sequência da lista.

Essa é uma maneira eficiente de excluir elementos em uma lista encadeada, pois não requer a realocação de memória. Basta atualizar os ponteiros adequadamente para garantir a continuidade da sequência encadeada.

Exemplo de lista em C:

#include <stdio.h>

#include <stdlib.h>

// Definição da estrutura do nó

struct Node {

int data;

struct Node\* next;

};

// Função para inserir um novo nó no início da lista

void insertAtBeginning(struct Node\*\* head, int value) {

// Criando um novo nó

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

// O próximo do novo nó aponta para a cabeça atual da lista

newNode->next = \*head;

// Atualizando a cabeça para o novo nó

\*head = newNode;

}

// Função para imprimir os elementos da lista

void printList(struct Node\* head) {

struct Node\* current = head;

while (current != NULL) {

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

// Função principal

int main() {

// Cabeça da lista

struct Node\* head = NULL;

// Inserindo elementos na lista

insertAtBeginning(&head, 3);

insertAtBeginning(&head, 2);

insertAtBeginning(&head, 1);

// Imprimindo os elementos da lista

printf("Lista: ");

printList(head);

return 0;

}

Neste exemplo, há uma estrutura de nó (struct Node) que contém um valor inteiro (data) e um ponteiro para o próximo nó (next). A função insertAtBeginning insere um novo nó no início da lista, e a função printList imprime os elementos da lista.

No main, é criada a cabeça da lista (head) e são inseridos três elementos (valores 1, 2 e 3) usando a função insertAtBeginning. Em seguida, a função printList é chamada para imprimir os elementos da lista.

A saída do programa será: "Lista: 1 2 3". Isso indica que os elementos foram inseridos corretamente na lista encadeada e foram impressos em ordem.