

Linguagem e Técnicas de Programação



Matrizes

Uma matriz é uma coleção bidimensional de elementos do mesmo tipo. Em C, uma matriz é essencialmente um array de arrays.

```
1 tipo nome_da_matriz[número_de_linhas][número_de_colunas];
```

Matrizes

Exemplo:

```
4  int matriz[3][4];  
5  //Uma matriz de 3 linhas e 4 colunas
```

Matrizes

Inicialização de Matrizes

Você pode inicializar uma matriz no momento da declaração.



```
1 int matriz[2][3] = {  
2     {1, 2, 3},  
3     {4, 5, 6}  
4 };
```

Matrizes

Acessando Elementos de uma Matriz

Para acessar um elemento específico da matriz, você especifica o índice da **linha** e o **índice da coluna**.



```
1 int valor = matriz[1][2];  
2 //Acessa o elemento na segunda linha e terceira coluna  
3
```

Matrizes

Exemplo Prático

Leitura e Impressão de uma Matriz:

```
1  #include <stdio.h>
2
3  int main() {
4      int matriz[2][3];
5      int i, j;
6      // Leitura
7      for(i = 0; i < 2; i++) {
8          for(j = 0; j < 3; j++) {
9              printf("Digite o elemento [%d][%d]: ", i, j);
10             scanf("%d", &matriz[i][j]);
11         }
12     }
13     // Impressão
14     for(i = 0; i < 2; i++) {
15         for(j = 0; j < 3; j++) {
16             printf("%d ", matriz[i][j]);
17         }
18         printf("\n");
19     }
20     return 0;
21 }
```

Matrizes

Exemplo Prático

Adição de Duas Matrizes:


```
1  #include <stdio.h>
2
3  int main() {
4      int A[2][3] = {{1, 5, 3}, {4, 5, 6}};
5      int B[2][3] = {{6, 5, 4}, {3, 2, 1}};
6      int C[2][3]; // Matriz resultante
7
8      int i, j;
9
10     for( i = 0; i < 2; i++) {
11         for(j = 0; j < 3; j++) {
12             C[i][j] = A[i][j] + B[i][j];
13         }
14     }
15
16     // Impressão da matriz resultante
17     for(i = 0; i < 2; i++) {
18         for(j = 0; j < 3; j++) {
19             printf("%d ", C[i][j]);
20         }
21         printf("\n");
22     }
23     return 0;
24 }
```

Matrizes

Vetor de Caracteres

Aqui, **numero_de_nomes** é o número total de nomes que você deseja armazenar e

tamanho_maximo_do_nome é o tamanho máximo de caracteres que cada nome pode ter, incluindo o caractere nulo ('\0').



```
2 char nomes[numero_de_nomes][tamanho_maximo_do_nome];
```

Vetores

Inicialização no momento da declaração:

Suponha que você queira armazenar três nomes: "Ana", "João" e "Carlos". Considerando que "Carlos" tem 7 caracteres (incluindo o `\0`), você pode usar 7 como tamanho máximo:


```
1 char nomes[3][7] = {  
2     "Ana",  
3     "Joao",  
4     "Carlos"  
5 };
```

Vetores

Vetores

Acessando e modificando nomes:

Para acessar um nome específico, você usaria o índice do vetor. Por exemplo, para acessar o segundo nome ("Joao"):




```
7 printf("%s\n", nomes[1]);  
8 // Imprimirá "Joao"
```

Vetores

Acessando e modificando nomes:


Se quiser acessar um caractere específico de um nome:



```
1 char letra = nomes[2][1];  
2 // Letra receberá 'a' de "Carlos"  
3
```

Vetores

Para modificar um nome:



```
1 strncpy(nomes[1], "Bianca", 7);  
2 // O segundo nome agora será "Bianca"  
3
```

Note que usamos a função **strncpy** para copiar uma string para outra. O último parâmetro (7) é o número máximo de caracteres a serem copiados, incluindo o '\0'.