Matrizes, Strings e Vetores Estáticos em C

1. Matrizes Estáticas em C

Em C, matrizes estáticas são blocos de memória fixos onde armazenamos valores, tipicamente números, em uma estrutura de dados bidimensional. Elas são alocadas de forma contínua na memória e acessadas através de índices. Aqui abordaremos a representação de memória, manipulação, limitações, operações básicas e avançadas, além de algumas boas práticas.

1.1 Declaração e Representação de Matrizes Estáticas

Uma matriz estática em C é declarada especificando o tipo de dados, seguido do nome e das dimensões. Por exemplo:

int matriz[3][3];

Aqui declaramos uma matriz 3x3, onde cada elemento é do tipo int. A memória é alocada em ordem de linha (row-major order), e o endereço de cada elemento pode ser calculado pela fórmula:

Endereço de matriz[i][j] = Endereço de matriz[0][0] + ((i * Número de Colunas) + j) * tamanho de int

1.2 Operações com Matrizes

Além de acessar elementos individuais, operações matemáticas podem ser realizadas. Abaixo, exemplos:
- Soma de Matrizes
- Multiplicação de Matrizes (Respeitando a Condição de Tamanho)
- Transposição de uma Matriz
- Exemplo de cálculo do determinante (para matrizes 2x2 ou 3x3)

2. Métodos para Manipulação de Strings

Em C, strings são manipuladas como vetores de caracteres. As funções mais comuns incluem scanf, fgets, e getline para leitura, bem como funções de manipulação da biblioteca <string.h>, como strcpy, strcat, strcmp, e strlen. Cada método possui vantagens e desvantagens.

2.1 Leitura de Strings

Métodos de leitura disponíveis incluem scanf (útil para uma palavra), fgets (captura espaços e linhas inteiras), e getline (com buffer dinâmico).

2.2 Manipulação de Strings

A biblioteca <string.h> fornece funções como:

- strcpy: copia uma string para outra

- strcat: concatena duas strings
- strcmp: compara duas strings
- strlen: retorna o comprimento da string

Exemplos incluem contagem de palavras, remoção de espaços duplicados e substituição de caracteres.

3. Vetores Estáticos

Vetores são estruturas unidimensionais. Operações comuns incluem somar elementos, encontrar mínimo e máximo, calcular a média, e ordenação. Algoritmos de ordenação populares são bubble sort e selection sort.
- Exemplo de ordenação de vetor com Bubble Sort
- Exemplo de busca linear e binária

4. Funções com Matrizes e Vetores

Para passar matrizes e vetores a funções, especificar o número de colunas é importante. Exemplo:

```
void imprimirMatriz(int matriz[3][3], int linhas, int colunas) {
   for (int i = 0; i < linhas; i++) {
      for (int j = 0; j < colunas; j++) {
         printf("%d ", matriz[i][j]);
      }
      printf("\n");
   }
}
```

5. Otimização e Acesso à Memória

Acesso sequencial é geralmente mais eficiente devido ao cache de memória. Isso é especialmente relevante para estruturas grandes, onde acesso aleatório pode degradar a performance.

Exemplo de Declaração e Inicialização de Matrizes

```c
#include <stdio.h>

int main() {
   int matriz[3][3] = {
      {1, 2, 3},
      {4, 5, 6},
      {7, 8, 9}
   };
```

```c
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Este código imprime a matriz 3x3 na tela.

Exemplo: Soma de Matrizes

```c
#include <stdio.h>

int main() {
    int matrizA[2][2] = {{1, 2}, {3, 4}};
    int matrizB[2][2] = {{5, 6}, {7, 8}};
    int resultado[2][2];

    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            resultado[i][j] = matrizA[i][j] + matrizB[i][j];
        }
    }

    printf("Resultado da Soma:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%d ", resultado[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Exemplo de Manipulação de Strings com scanf e fgets

```c
#include <stdio.h>
```

```c
int main() {
   char nome[50];

   printf("Digite o seu nome: ");
   scanf("%s", nome); // Lê até o primeiro espaço
   printf("Nome (scanf): %s\n", nome);

   getchar(); // Limpa o buffer de entrada

   printf("Digite seu nome completo: ");
   fgets(nome, sizeof(nome), stdin); // Lê a linha completa
   printf("Nome completo (fgets): %s", nome);

   return 0;
}
```

Exemplo de Ordenação de Vetor com Bubble Sort

```c
#include <stdio.h>

int main() {
   int vetor[] = {5, 2, 9, 1, 5};
   int n = sizeof(vetor) / sizeof(vetor[0]);

   for (int i = 0; i < n-1; i++) {
      for (int j = 0; j < n-i-1; j++) {
         if (vetor[j] > vetor[j+1]) {
            int temp = vetor[j];
            vetor[j] = vetor[j+1];
            vetor[j+1] = temp;
         }
      }
   }

   printf("Vetor ordenado: ");
   for (int i = 0; i < n; i++) {
      printf("%d ", vetor[i]);
   }

   return 0;
}
```

```

### Exemplo: Função para Imprimir Matriz

```c
#include <stdio.h>

void imprimirMatriz(int *matriz, int linhas, int colunas) {
    for (int i = 0; i < linhas; i++) {
        for (int j = 0; j < colunas; j++) {
            printf("%d ", *(matriz + i*colunas + j));
        }
        printf("\n");
    }
}

int main() {
    int matriz[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    imprimirMatriz(&matriz[0][0], 3, 3);
    return 0;
}
```

Exemplo: Acesso Sequencial em Matrizes

Acessar uma matriz de forma sequencial ajuda a aproveitar o cache de memória, o que pode melhorar o desempenho.
No exemplo abaixo, a matriz é preenchida e percorrida de maneira sequencial.

```c
#include <stdio.h>

int main() {
    int matriz[100][100];
    int linhas = 100, colunas = 100;

    for (int i = 0; i < linhas; i++) {
        for (int j = 0; j < colunas; j++) {
            matriz[i][j] = i + j;
```
```

```
        }
    }

    int soma = 0;
    for (int i = 0; i < linhas; i++) {
        for (int j = 0; j < colunas; j++) {
            soma += matriz[i][j];
        }
    }

    printf("Soma de todos os elementos: %d\n", soma);
    return 0;
}
```