AULA TEÓRICA+PRÁTICA 15

FUNÇÕES E MATRIZES

Na aula 14 vimos funções e vetores e destacamos como estas estruturas são tratadas de forma diferenciada neste caso. Em especial, é muito importante lembrar que vetores são sempre parâmetros passados por referência às funções na linguagem C. Como veremos adiante, isso vale para qualquer variável composta homogênea, isto é, para variáveis compostas homogêneas de qualquer dimensão. Esta aula é baseada nas referências [9, 6].

15.1 Matrizes

Um elemento de uma matriz pode ser passado como parâmetro para uma função assim como fizemos com uma variável de um tipo básico ou como um elemento de um vetor. Ou seja, a sentença

```
p = paridade(matriz[i][j]);
```

chama a função `paridade` passando o valor contido em `matriz[i][j]` como um argumento para a função.

Uma matriz toda pode ser passada a uma função da mesma forma que um vetor todo pode ser passado, bastando listar o identificador da matriz. Por exemplo, se a matriz `A` é declarada como uma matriz bidimensional de números inteiros, a sentença

```
multiplicaEscalar(A, escalar);
```

pode ser usada para chamar uma função que multiplica cada elemento de uma matriz `A` pelo valor armazenado na variável `escalar`. Assim como com vetores, uma modificação realizada no corpo de uma função sobre qualquer elemento de uma matriz que é um parâmetro dessa função provoca uma modificação no argumento da chamada da função, ou seja, na matriz que foi passada à função durante sua chamada.

O programa 15.1 recebe uma matriz $A_{4\times4}$ de números inteiros e um escalar e realiza o produto da matriz por esse escalar.

Programa 15.1: Uma função com um parâmetro que é uma matriz.

```
#include <stdio.h>

/* Recebe uma matriz de dimensão 4x4 de números inteiros e um escalar e
   devolve uma matriz resultante do produto entre a matriz e o escalar */
void multiplicaEscalar(int A[4][4], int escalar)
{
   int i, j;

   for (i = 0; i < 4; i++)
      for (j = 0; j < 4; j++)
         A[i][j] = A[i][j] * escalar;
}

/* Recebe uma traiz de dimensão 4x4 de números
   inteiros e mostra seu conteúdo na saída */
void imprimeMatriz(int A[4][4])
{
   int i, j;

   for (i = 0; i < 4; i++) {
      for (j = 0; j < 4; j++)
         printf("%3d ", A[i][j]);
      printf("\n");
   }
}

/* Recebe um número inteiro e uma matriz de dimensão 4x4
   de números inteiros e mostra a matriz de entrada e a
   matriz resultante do produto desse número pela matriz */
int main(void)
{
   int i, j, esc, matriz[4][4];

   scanf("%d", &esc);
   for(i = 0; i < 4; i++)
      for(j = 0; j < 4; j++)
         scanf("%d", &matriz[i][j]);

   imprimeMatriz(matriz);
   multiplicaEscalar(matriz, esc);
   imprimeMatriz(matriz);

   return 0;
}
```

15.2 Matrizes como parâmetros com uma dimensão omitida

Se um parâmetro de uma função é uma variável composta homogênea k-dimensional, com $k \geqslant 2$, então somente o tamanho da primeira dimensão pode ser omitida quando o parâmetro é declarado. Por exemplo, na função `multiplicaEscalar` da seção anterior, onde A é um de seus parâmetros e é uma matriz, o número de colunas de A deve ser espeficiado, embora seu número de linhas não seja necessário. Assim, na definição da função, poderíamos escrever:

```
void multiplicaEscalar(int A[][4], int escalar)
```

Exercícios

15.1 (a) Escreva uma função com a seguinte interface:

```
void troca(int *a, int *b)
```

que receba dois números inteiros a e b e troque os seus conteúdos.

(b) Usando a função anterior, escreva um programa que receba uma matriz de números inteiros A, de dimensão $m \times n$, com $1 \leqslant m, n \leqslant 100$, e dois números inteiros i e j, troque os conteúdos das linhas i e j da matriz A e imprima a matriz resultante.

Programa 15.2: Solução do exercício 15.1.

```
#include <stdio.h>

#define MAX 100

/* Recebe dois números inteiros a e b e devolve esses valores trocados */
void troca(int *a, int *b)
{
   int aux;

   aux = *a;
   *a = *b;
   *b = aux;
}

/* Recebe uma matriz de dimensão mxn de números inteiros
   e os índices i e j de duas linhas da matriz, troca o
   conteúdo dessas linhas e mostra a matriz resultante */
int main(void)
{
   int m, n, i, j, k, A[MAX][MAX];

   scanf("%d%d", &m, &n);
   for (i = 0; i < m; i++)
      for (j = 0; j < n; j++)
         scanf("%d", &A[i][j]);
   scanf("%d%d", &i, &j);
   for (k = 0; k < n; k++)
      troca(&A[i][k], &A[j][k]);
   for (i = 0; i < m; i++) {
      for (j = 0; j < n; j++)
         printf("%d ", A[i][j]);
      printf("\n");
   }

   return 0;
}
```

15.2 A k-ésima potência de uma matriz quadrada $A_{n\times n}$, denotada por A^k, é a matriz obtida a partir da matriz A da seguinte forma:

$$A^k = I_n \times \overbrace{A \times A \times \cdots \times A}^{k},$$

onde I_n é a matriz identidade de ordem n.

Para facilitar a computação de A^k, podemos usar a seguinte fórmula:

$$A^k = \begin{cases} I_n, & \text{se } k = 0\,, \\ A^{k-1} \times A, & \text{se } k > 0\,. \end{cases}$$

(a) Escreva uma função com a seguinte interface:

```
void identidade(int I[MAX][MAX], int n)
```

que receba uma matriz I e uma dimensão n e preencha essa matriz com os valores da matriz identidade de ordem n.

(b) Escreva uma função com a seguinte interface:

```
void multMat(int C[MAX][MAX], int A[MAX][MAX], int B[MAX][MAX], int n)
```

que receba as matrizes A, B e C, todas de ordem n, e compute $C = A \times B$.

(c) Escreva uma função com a seguinte interface:

```
void copia(int A[MAX][MAX], int B[MAX][MAX], int n)
```

que receba as matrizes A e B de ordem n, e copie os elementos da matriz B na matriz A.

(d) Escreva um programa que, usando as funções em (a) e (b), receba uma matriz de números inteiros A de dimensão n e um inteiro k e compute e imprima A^k.

15.3 Dizemos que uma matriz $A_{n\times n}$ é um **quadrado latino de ordem** n se em cada linha e em cada coluna aparecem todos os inteiros $1, 2, 3, \ldots, n$, ou seja, cada linha e coluna é permutação dos inteiros $1, 2, \ldots, n$.

Exemplo:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \end{pmatrix}$$

A matriz acima é um quadrado latino de ordem 4.

(a) Escreva uma função com a seguinte interface:

```
int linha(int A[MAX][MAX], int n, int i)
```

que receba como parâmetros uma matriz $A_{n\times n}$ de números inteiros e um índice i, e verifique se na linha i de A ocorrem todos os números inteiros de 1 a n, devolvendo 1 em caso positivo e 0 caso contrário.

(b) Escreva uma função com a seguinte interface:

```
int coluna(int A[MAX][MAX], int n, int j)
```

que receba como parâmetros uma matriz $A_{n\times n}$ de números inteiros e um índice j, e verifique se na coluna j de A ocorrem todos os números inteiros de 1 a n, devolvendo 1 em caso positivo e 0 caso contrário.

(c) Usando as funções dos itens (a) e (b), escreva um programa que verifique se uma dada matriz $A_{n\times n}$ de números inteiros, com $1 \leqslant n \leqslant 100$, é um quadrado latino de ordem n.