Segunda Atividade Prática

Estrutura de Dados 1

Prof. Paulo Henrique Ribeiro Gabriel

O objetivo deste trabalho é implementar um tipo abstrato de dados, bem como reforçar os conceitos de alocação dinâmica e programação em linguagem C. Implemente sua atividade individualmente sem compartilhar, olhar código de seus colegas, ou buscar na Internet. Procure usar apenas os conceitos já vistos em aula.

TAD Matriz

Nesta atividade prática você deve:

1. Definir um tipo abstrato de dados (TAD) matriz de números inteiros (alocada dinamicamente);
2. Escrever funções para:
   (a) Alocar a memória para a matriz;
   (b) Liberar a memória associada à matriz;
   (c) Ler o número de linhas, de colunas e os elementos da matriz;
   (d) Transpor uma matriz dada;
   (e) Retornar a soma de duas matrizes;
   (f) Retornar o produto de duas matrizes;
   (g) Retornar o produto escalar de uma matriz por um inteiro.

Dadas essas funções, o seu programa deve ler **três matrizes** (A, B e C) e um inteiro k, e imprimir, nessa ordem, o resultado de:

1. $A + B$;
2. $A \times C$;
3. $k(B \times C)$;
4. $C^\top + B$;
5. $B^\top + A^\top$

Note que não há restrições em relação às dimensões das matrizes. Em outras palavras, as suas funções devem verificar se as operações matriciais podem ou não serem executadas.

Entrada

Os dados de entrada do problema consistem nas informações das três matrizes, bem como do escalar k.

- A primeira linha contém o inteiro k (um valor);
- A segunda linha contém o número de linhas e o número de colunas da matriz A (dois valores).
- A terceira linha contém o número de linhas e o número de colunas da matriz B (dois valores).
- A quarta linha contém o número de linhas e o número de colunas da matriz C (dois valores).
- As próximas linhas (a partir da quinta linha) contêm os elementos de cada matriz (A, B e C, nessa sequência), **um único elemento por linha**.

Por exemplo, vamos considerar um inteiro $k = 4$ e as seguintes matrizes:

$$A = \begin{pmatrix} 0 & 2 & 4 \\ 6 & 8 & 10 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \end{pmatrix} \qquad C = \begin{pmatrix} 2 & 3 \\ 5 & 7 \\ 11 & 13 \end{pmatrix}$$

Assim, os dados fornecidos pelo usuário, via entrada padrão, possuem o seguinte formato:

```
4
2 3
0
2
4
6
8
10
2 3
1
3
5
7
9
11
3 2
2
3
5
7
11
13
```

Saída

A saída do programa consiste nas matrizes resultantes das cinco operações descritas anteriormente, na ordem solicitada. Cada operação resultará em uma matriz e cada linha da saída deve conter um único elemento de cada matriz resultante.

Por exemplo, para as matrizes A, B e C, mostrada na seção anterior, teríamos os seguintes resultados:

$$A + B = \begin{pmatrix} 1 & 5 & 9 \\ 13 & 17 & 21 \end{pmatrix} \qquad A \times C = \begin{pmatrix} 54 & 66 \\ 162 & 204 \end{pmatrix} \qquad k(B \times C) = \begin{pmatrix} 288 & 356 \\ 720 & 908 \end{pmatrix}$$

$$C^\top + B = \begin{pmatrix} 3 & 8 & 16 \\ 10 & 16 & 24 \end{pmatrix} \qquad B^\top + A^\top = \begin{pmatrix} 1 & 13 \\ 5 & 17 \\ 9 & 21 \end{pmatrix}$$

o que resultaria na seguinte saída:

```
1
5
9
13
17
21
54
66
162
204
288
356
720
908
3
8
16
10
16
24
1
13
5
17
9
21
```

Note que, no exemplo apresentado aqui, todas as operações puderam ser executadas. Caso alguma dessas operações não seja permitida (ou seja, caso as dimensões das matrizes

não sejam compatíveis), o programa deve ignorar tal operação. Por exemplo, se $A + B$ não fosse uma operação permitida, o programa não iria imprimir seu resultado, seguindo seu fluxo de execução para a próxima operação (no caso, $A \times C$).

Observações

1. O trabalho deve ser feito individualmente e em Linguagem C.
2. Todas as submissões são checadas para evitar plágio; portanto, evite problemas e implemente o seu próprio código.
3. Comente o seu código com uma explicação rápida do que cada função ou trecho importante de código faz (ou deveria fazer). Os comentários e a modularização do código serão checados e valem nota.
4. Todas as funções devem ser implementadas em um único arquivo `.c`; porém, deve-se ficar atendo ao bom uso de TAD.
5. Entradas e saídas devem ser lidas e escritas a partir dos dispositivos padrão, ou seja, use as funções `scanf` e `printf`.
6. Lembre-se de respeitar estritamente o formado de entrada e saída. Uma quebra de linha a mais ou a menos resultará em erro no caso de teste.
7. Todas as operações de leitura e escrita de valores devem ser feitas na função `main()`. Lembre-se: TAD deve ser genérico, ou seja, suas funções devem evitar a escrita de mensagens.
8. Use alocação dinâmica de memória! O programa que não usar alocação valerá **metade** da nota.