Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
2º Semestre de 2012

Trabalho Prático 0

Esse trabalho tem como objetivo familiarizar o aluno com algumas primitivas básicas da linguagem C, assim como os padrões de documentação e codificação esperados ao longo dessa disciplina.

Produto de Kronecker

O produto de Kronecker, também denominado produto tensorial, consiste em uma operação entre duas matrizes de dimensões arbitrárias que resulta em uma matriz em bloco.

Formalmente, seja $\mathbf{A}$ uma matriz $m \times n$ e $\mathbf{B}$ uma matriz $p \times q$, o produto de Kronecker $\mathbf{A} \otimes \mathbf{B}$ é uma matrix $mp \times nq$:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix},$$

Expandindo cada bloco, temos:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix}.$$

Um exemplo é dado abaixo.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1\cdot 0 & 1\cdot 5 & 2\cdot 0 & 2\cdot 5 \\ 1\cdot 6 & 1\cdot 7 & 2\cdot 6 & 2\cdot 7 \\ 3\cdot 0 & 3\cdot 5 & 4\cdot 0 & 4\cdot 5 \\ 3\cdot 6 & 3\cdot 7 & 4\cdot 6 & 4\cdot 7 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}.$$

Nesse trabalho você deve implementar um programa que calcule o produto de Kronecker entre duas matrizes de inteiros dadas de entrada. É obrigatório que a alocação e desalocação das matrizes seja feita dinamicamente através dos comandos *malloc* e *free*.

Entrada e Saída

O programa deverá solucionar múltiplas instâncias do problema em uma única execução. As matrizes a serem multiplicadas devem ser lidas de um arquivo de entrada e as matrizes resultantes devem ser impressas em um arquivo de saída, ambos especificados na linha de comando. Um exemplo de invocação é dado abaixo:

```
./tp0 input.txt output.txt
```

A primeira linha do arquivo de entrada consiste no número k de instâncias (pares de matrizes) que o arquivo contém. A linha seguinte contém as dimensões m e n da matriz A_1. As m próximas linhas contêm os elementos de cada linha de A_1 separados por um espaço. Em seguida as dimensões e os elementos da matrix B_1 são especificados da mesma forma. As instâncias restantes do problema são representadas da mesma forma nas linhas seguintes.

A matriz resultante deve ser impressa seguindo o mesmo padrão no arquivo de saída especificado: a primeira linha com o valor k, seguido das k matrizes resultantes.

Como exemplo, considere os dois pares de matrizes abaixo.

$$A_1 = \begin{bmatrix} 4 & 0 & 1 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 3 & 2 \\ 0 & 5 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Os arquivos de entrada e saída para essas duas instâncias do problema são dados abaixo.

Entrada:

```
2
1 3
4 0 1
2 2
3 2
0 5
3 3
1 0 0
0 1 0
0 0 1
2 1
1
1
```

Saída:

```
2
2 6
12 8 0 0 3 2
0 20 0 0 0 5
6 3
1 0 0
1 0 0
0 1 0
0 1 0
0 0 1
0 0 1
```

Entrega

- O código e a documentação devem ser **submetidos em um arquivo compactado** (zip ou tar.gz) pelo *minha.ufmg* dentro do prazo estipulado.
- Uma cópia da **documentação deve ser entregue na secretaria** (atenção para o horário de funcionamento). Não utilize os escaninhos dos professores. Entregue diretamente para a

secretária.

- **Trabalhos que não tiverem a documentação entregue na secretaria dentro do prazo receberão nota 0.**
- Uma planilha será divulgada no *minha.ufmg* instruindo o **agendamento das entrevistas.**
- Será adotada **média harmônica** entre as notas da **documentação e da execução**, o que implica que a nota final será 0 se uma das partes não for apresentada.

Documentação

A documentação não deve exceder 10 páginas e deve conter pelo menos os seguintes itens:

- Uma **introdução** do problema em questão.
- **Modelagem e solução proposta** para o problema. O algoritmo deve ser explicado de forma clara, possivelmente através de pseudo-código e esquemas ilustrativos.
- **Análise de complexidade** de tempo e espaço da solução implementada.
- **Experimentos** variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem significavelmente a execução.
- Especificação da(s) **máquina(s) utilizada(s)** nos experimentos realizados.
- Uma breve **conclusão** do trabalho implementado.

Código

- O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux dos laboratórios de graduação.
- O utilitário ***make*** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido.
- As estruturas de dados devem ser **alocadas dinamicamente** e o código deve ser **modularizado** (divisão em múltiplos arquivos fonte e uso de arquivos cabeçalho .h)
- Variavéis globais devem ser evitadas.
- Parte da correção poderá ser feita de forma automatizada, portanta **siga rigorosamente os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- O arquivo executável deve ser chamado tp0.
- **Legibilidade e boas práticas** de programação serão avaliadas.