Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
2º Semestre de 2013

Trabalho Prático 0 - Matrizes 3D

Esse trabalho prático tem como objetivo familiarizar o aluno com alocação dinâmica e o utilitário ***make***.

Problema

A empresa *Matrix3D* precisa realizar operações aritméticas sobre matrizes tridimensionais. A operação de soma de matrizes $C = A + B$ implica em somar o elemento corresponde nas duas matrizes A e B, ou seja, temos $c_{ijk} = a_{ijk} + b_{ijk}$. As matrizes devem ser alocadas e desalocadas dinâmicamente com as funções malloc() e *free()*. O problema consiste em realizar sequências de operações de soma e subtração em matrizes tridimensionais, sendo que o número de matrizes, as suas 3 dimensões e as operações são definidas na entrada.

Entrada e Saída

O programa deverá solucionar múltiplas instâncias do problema em uma única execução. Serão passados na entrada de dados tanto os tamanhos das matrizes como as operações (soma ou subtração) que deverão ser realizadas em cada instância do problema. A saída será a matriz resultante, uma para cada instância da entrada. A entrada será lida de um arquivo e o resultado do programa deve ser impresso em outro arquivo de saída. Ambos arquivos devem ser passados por parâmetro na chamada do executável:

./tp0 input.txt output.txt

O arquivo de entrada possui um inteiro N na primeira linha onde N é o número de instâncias a serem simuladas. Em seguida, as N instâncias são definidas da seguinte forma. A primeira linha possui três inteiros XYZ que indicam as 3 dimensões das matrizes. As linhas seguintes terão $X * Y * Z$ números reais (tipo double) indicando os valores das células da matriz, começando pelo elemento no plano 1, linha 1, coluna 1, depois plano 1, linha 1, coluna 2, ... plano 1, linha 2, coluna 1 ... plano 2, linha 1, coluna 1 ... até plano X, linha Y, coluna Z. Em seguida, poderemos ter o operador + para soma, − para subtração, = para terminar. Caso seja o operador + ou −, teremos outros $X * Y * Z$ números e um novo operador. O operador = significa o fim da instância.

Para cada instância, deve ser impresso no arquivo de saída, a matriz resultante. Entre cada saída das instâncias, incluir uma linha em branco separando as saídas.

Exemplo

A seguir temos um exemplo de funcionamento do programa. Temos 2 instâncias. Na primeira instância, temos $E = A + B + C - D$. Na segunda instância, temos $C = A + B$. A entrada não indica os nomes das matrizes explicitamente.

Entrada:

```
2
2 2 2
0 1
1 0
0 1
1 0
+
0 1
1 0
0 1
1 0
+
0 1
1 0
0 1
1 0
-
0 3
3 0
0 3
3 0
=
2 3 3
1.1 1.1 1.1
1.2 1.2 1.2
1.3 1.3 1.3
1.4 1.4 1.4
1.5 1.5 1.5
1.6 1.6 1.6
+
1.6 1.6 1.6
1.5 1.5 1.5
1.4 1.4 1.4
1.3 1.3 1.3
1.2 1.2 1.2
1.1 1.1 1.1
=
```

Saída:

```
0 0
0 0
0 0
0 0

2.7 2.7 2.7
2.7 2.7 2.7
```

```
2.7 2.7 2.7
2.7 2.7 2.7
2.7 2.7 2.7
2.7 2.7 2.7
```

Entrega

- A data de entrega desse trabalho é **02 de Setembro**.
- A penalização por atraso obedece à seguinte fórmula $2^{d-1}/0.32\%$, onde d são os dias úteis de atraso.
- Submeta apenas um arquivo chamado <numero_matricula>_<nome>.zip. Não utilize espaços no nome do arquivo. Ao invés disso utilize o caractere '_'.
- Não inclua arquivos compilados ou gerados por IDEs. **Apenas** os arquivos abaixo devem estar presentes no arquivo zip.
  - Makefile
  - Arquivos fonte (*.c e *.h)
  - Documentacao.pdf
- Não inclua **nenhuma pasta**. Coloque todos os arquivos na raiz do zip.
- Siga rigorosamente o formato do arquivo de saida descrito na especificação. Tome cuidado com whitespaces e formatação dos dados de saída
- NÃO SERÁ NECESSÁRIO ENTREGAR DOCUMENTAÇÃO IMPRESSA!
- 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.
- **solução proposta** Explique como representou as estruturas de dados.
- **Análise de complexidade** de tempo e espaço da solução implementada.

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)
- **Variáveis globais** devem ser evitadas.
- Parte da correção poderá ser feita de forma automatizada, portanto **siga rigorosamente os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- **Legibilidade e boas práticas** de programação serão avaliadas.