Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
2º Semestre de 2014

Trabalho Prático 0 - Matrizes Complexas

Esse trabalho prático tem como objetivo familiarizar o aluno com conceitos da linguagem C, do ambiente de programação Unix, alocação dinâmica e o utilitário ***make***.

Problema

Este trabalho consiste em implementar um programa para multiplicação de matrizes. As matrizes possuem números complexos. Um número complexo é um número z que pode ser escrito na forma z = x + iy, em que x e y são números reais (tipo double) e i denota a unidade imaginária. A multiplicação de um número complexo (a+bi) por (c+di) = (ac-bd)+(bc+ad)i. As matrizes devem ser alocadas e desalocadas dinâmicamente com as funções malloc() e *free()*.

Entrada e Saída

O programa deverá solucionar múltiplas instâncias do problema em uma única execução. Será passado na entrada de dados os tamanhos das matrizes 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 computadas. Em seguida, as N instâncias são definidas da seguinte forma. A primeira linha possui dois inteiros XY que indicam as 2 dimensões das matrizes. As linhas seguintes terão $X * Y$ números complexos indicando os valores das células da matriz. Os números complexos serão sempre da forma a+bi mesmo que a ou b sejam 0.

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.

Entrada:

```
2
2 3
3+0i 4+0i 5+0i
6+0i 7+0i 8+0i
3 2
```

```
3+0i 4+0i
6+0i 7+0i
8+0i 9+0i
2 2
0+1i -1.5+0i
1+0i 0+0i
2 2
1+1i 2+0i
3+0i 4+1i
```

Saída:

```
2+0i 2+0i
73+0i 85+0i
124+0i 145+0i

-5.5+1i -6+0.5i
1+1i 2+0i
```

Entrega

- A data de entrega desse trabalho é **29 de Agosto**.
- 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.