Aula Prática 6 - Roteiro

04/05/2021 - Roteiro referente à aula prática 6 - Manipulação de Matrizes

Versões:

- **31/08/2021 - Versão inicial**

Prazo: 08/09/2021 - 18:00

Valor: 10,0 - Peso: 2

Observações:

- Leia este enunciado com **MUITA** atenção até o final antes de iniciar o trabalho.
- Os arquivos solicitados deverão estar disponíveis nos diretórios correspondentes (**Aulas-Praticas** e ***RCS***) até o prazo estipulado acima. Cuidado com os nomes dos diretórios e dos arquivos. Deverão ser exatamente os definidos neste roteiro (maiúsculas, minúsculas, caracteres especiais e extensões, se existentes).
- **As tarefas deverão ser executadas na ordem solicitada neste roteiro.**
- **Os ítens marcados com (*) deverão ser entregues até às 19:00 do dia da aula em questão.**
- Os arquivos de dependências deverão possibilitar que a compilação e a *linkedição* sejam executadas utilizando-se tanto o *gcc*, quanto o *clang*. A seleção da ferramenta utilizada deverá ser realizada no momento da execução do comando *make*. O gcc deverá ser considerado o valor padrão para a compilação e para a *linkedição*.

  Para a definição da ferramenta desejada deverá ser utilizada uma macro (no *FreeBSD*) ou um argumento com o valor desejado (no CentOS). As duas macros utilizadas deverão ser *GCC* e *CLANG* (definidas usando a opção -D). O argumento, identificado por *cc,* deverá ser igual a *GCC* ou *CLANG*.
- Independente da ferramenta utilizada para a compilação, o *flag* de compilação deverá ser definido no instante da execução do comando *make*. O valor padrão para este *flag* deverá ser "*-Wall -ansi*" (sem as aspas).

  Durante a execução do comando *make* poderão ser definidos outros valores para este *flag* (mantendo a opção de exibir todas as mensagens de advertência) através de macros ou através de argumentos (de forma semelhante àquela utilizada para definir o compilador/*linkeditor*). No FreeBSD deverão ser definidas as macros ANSI, C89, C90, C99 e C11, enquanto que no CentOS deverá ser definido o argumento *dialeto* com os valores ANSI, C89, C90, C99 ou C11.
- Crie uma macro, DIALETO, contendo o dialeto a ser utilizado na compilação do código. Esta macro será inicialmente igual a "ansi" e poderá ser alterada para "c89", "c99" ou "c11" de acordo com o esquema definido acima.
- O *flag* de *linkedição* deverá ser igual a "-Wall" (sem as aspas).
- Cuidado com os nomes das macros e dos rótulos. Deverão ser exatamente os definidos neste roteiro (maiúsculas, minúsculas, caracteres especiais e extensões, se existentes).
- Todos os rótulos solicitados no roteiro são obrigatórios. Durante a correção, caso não seja possível alcançar os objetivos (binários e/ou bibliotecas e limpezas de código) solicitados, a nota correspondente ao item/aula questão será zero.
- Seguem alguns exemplos:

  make - compila/linkedita (tanto no FreeBSD, quanto no CentOS) com a ferramenta e dialeto padrões, ou seja, gcc e ANSI respectivamente.
  make -DGCC - compila/linkedita usando o *gcc* e o dialeto ANSI (somente FreeBSD).
  make -DCLANG - compila/linkedita usando o *clang* e o dialeto ANSI (somente FreeBSD).

make cc=GCC - compila/linkedita usando o *gcc* e o dialeto ANSI (somente CentOS).
make cc=CLANG - compila/linkedita usando o *clang* e o dialeto ANSI (somente CentOS).
make -DCLANG -DC89 - compila/linkedita usando o *clang* e o dialeto C89 (somente FreeBSD).
make -DCLANG -DC11 - compila/linkedita usando o *clang* e o dialeto C11 (somente FreeBSD).
make cc=CLANG dialeto=C99 - compila/linkedita usando o *clang* e o dialeto C99 (somente CentOS).

- Inclua, no início de todos os arquivos solicitados, os seguintes comentários:

```
Universidade Federal do Rio de Janeiro
Escola Politecnica
Departamento de Eletronica e de Computacao
EEL270 - Computacao II - Turma 2021/1
Prof. Marcelo Luiz Drumond Lanza
Autor: <nome completo>
Descricao: <descrição sucinta dos objetivos do programa>

$Author$
$Date$
$Log$
```

- Inclua, no final de todos os arquivos solicitados, os seguintes comentários:

```
$RCSfile$
```

1. **(*)** Crie o arquivo "*aula0601.h*" contendo o protótipo definido abaixo. Este arquivo deverá conter também as macros e os tipos necessários para a implementação desta função. A macro referente à combinação *ifndef* e *define*, como por exemplo _AULA0601_, deverá ser definida como uma *string* valendo:

   "@(#)aula0601.h $Revision$"

   tipoErros
   MultiplicarMatrizes (unsigned short, / numero de linhas da matriz 1 (E) */*
   unsigned short, / numero de colunas da matriz 1 (E) */*
   unsigned short, / numero de linhas da matriz 2 (E) */*
   unsigned short, / numero de colunas da matriz 2 (E) */*
   double [ ][ ], / matriz 1 (E) */*
   double [ ][ ], / matriz 2 (E) */*
   double [ ][ ]); / matriz produto (S) */*

   Considere que as matrizes podem ter no máximo 100 linhas por 100 colunas. Defina duas macros correspondendo a estes limites.
2. **(*)** Crie o arquivo "*aula0601.c*" contendo a implementação da função *MultiplicarMatrizes*. As matrizes deverão ser manipuladas através de indexação de vetores. A implementação desta função NÃO poderá utilizar alocação dinâmica de memória.
3. **(*)** Crie o arquivo "*aula0602.c*" contendo o programa de testes para a função *MultiplicarMatrizes*. O programa deverá receber, através dos argumentos de linha de comando, as dimensões de cada matriz e os valores de cada elemento das matrizes de entrada. Após executar a função de multiplicação de matrizes, o programa deverá exibir a matriz produto (em formato de matriz). Os valores desta matriz deverão ser exibidos com no mínimo 5 casas decimais. Este programa NÃO pode utilizar alocação dinâmica de memória.

   Exemplo:
   ./aula0602 3 2 2 4 a_{11} a_{12} a_{13} a_{21} ... b_{31} b_{32} b_{33} b_{34}

   No exemplo acima, a matriz A terá 3 linhas x 2 colunas, enquanto que a matriz B terá 2 linhas x 4 colunas.
4. **(*)** Inclua, nos arquivos de dependências, as macros *AULA0602OBJS* e *AULA06*. Altere o valor da macro *EXECS,* de forma que inclua o valor da macro *AULA06*. Inclua também os objetivos

aula06 e *aula0602* com os comandos correspondentes.

5. **(*)** Gere e teste as 16 versões do executável *aula0602*.
6. **(*)** Submeta os arquivos *"aula0601.h"*, "*aula0601.c"*, "*aula0602.c"* e "**makefile"* ao sistema de controle de versão.
7. **(*)** Recupere uma cópia de leitura do arquivo "*aula0602.c"* e uma cópia de escrita dos arquivos *"aula0601.h*", "*aula0601.c*" e "**makefile"*.
8. Inclua, nos arquivos de dependências, as macros *LIBMATEMATICAOBJS* (correspondendo ao arquivo "*aula0601.o*") e *LIBMATEMATICA (*correspondendo ao arquivo "*libmatematica.a*"). O valor da macro LIBS deverá ser atualizado de forma que inclua o valor desta última macro. Inclua o objetivo correspondente, ou seja, libmatematica.a, com a(s) dependência(s) e comando(s) necessários para atingir este objetivo.
9. Gere as 16 versões da biblioteca.
10. Inclua, no arquivo "aula0601.h", o protótipo da função *ObterMatrizTransposta*.

```
tipoErros
ObterMatrizTransposta (unsigned short, /* numero de linhas da matriz original (E) */
                       unsigned short, /* numero de colunas da matriz original (E) */
                       double [ ][ ], /* matriz original (E) */
                       double [ ][ ]); /* matriz transposta (S) */
```

Considere que as matrizes podem ter no máximo 100 linhas por 100 colunas.

11. Inclua, no arquivo "*aula0601.c*", a implementação da função *ObterMatrizTransposta*. As matrizes deverão ser manipuladas através de indexação de vetores. A implementação desta função NÃO poderá utilizar alocação dinâmica de memória.
12. Crie o arquivo "*aula0603.c*" contendo o programa de testes para a função *ObterMatrizTransposta*. O programa deverá receber, através dos argumentos de linha de comando, as dimensões da matriz e os valores de cada elemento desta matriz. O programa deverá exibir a matriz transposta obtida (em formato de matriz). Os valores desta matriz deverão ser exibidos com no mínimo 5 casas decimais. Este programa NÃO pode utilizar alocação dinâmica de memória.

Exemplo:
./aula0603 3 2 a_{11} a_{12} a_{13} a_{21} ... a_{32}

No exemplo acima, a matriz 3 linhas x 2 colunas.

13. Inclua as declarações necessárias nos arquivos de dependências.
14. Gere e teste as 16 versões do executável *aula0603*.
15. Submeta os arquivos "*aula0601.h*", "*aula0601.c*", "*aula0603.c*" e "**makefile*" ao sistema de controle de versão.
16. Recupere uma cópia de leitura do arquivo "aula0603.c" e uma cópia de escrita dos arquivos "aula0601.h", "aula0601.c" e "*makefile".
17. Inclua, no arquivo "*aula0601.h*", o protótipos abaixo:

```
tipoErros
CalcularMenorComplementar (unsigned short, /* ordem da matriz (E) */
                           unsigned short, /* linha do elemento (E) */
                           unsigned short, /* coluna do elemento (E) */
                           double [ ][ ], /* matriz (E) */
                           double *); /* menor complementar (S) */

tipoErros
CalcularComplementoAlgebrico (unsigned short, /* ordem da matriz (E) */
                              unsigned short, /* linha do elemento (E) */
                              unsigned short, /* coluna do elemento (E) */
                              double [ ][ ], /* matriz (E) */
                              double *); /* complemento algebrico ou cofator (S) */

tipoErros
CalcularDeterminanteMatriz (unsigned short, /* ordem da matriz (E) */
                            double [ ][ ], /* matriz (E) */
                            double *); /* determinante (S) */
```

Considere que as matrizes podem ter no máximo 100 linhas por 100 colunas.

18. Inclua, no arquivo "*aula0601.c*", a implementação da função *CalcularDeterminanteMatriz*. As matrizes deverão ser manipuladas através de indexação de vetores. A implementação desta função NÃO poderá utilizar alocação dinâmica de memória.

    Para a implementação desta função considere que:
    - o determinante de uma matriz de ordem 1 é igual ao único elemento da matriz.
    - o determinante de uma matriz de ordem 2 é igual à diferença entre o produto dos elementos da diagonal principal e o produto dos elementos da diagonal secundária.
    - o determinante de uma matriz de ordem 3 deverá ser calculado utilizando-se a regra de *Sarrus*.
    - o determinante de uma matriz de ordem igual ou superior a 4 deverá ser calculado utilizando-se ao teorema de Laplace.

    Para o cálculo de um determinante usando o teorema de Laplace, deverão ser utilizadas as funções *CalcularMenorComplementar* e *CalcularComplementoAlgebrico* que serão implementadas mais tarde neste roteiro.

19. Inclua, no arquivo "*aula0601.c*", os cabeçalhos e corpos das funções *CalcularMenorComplementar* e *CalcularComplementoAlgebrico*. Por enquanto, o corpo de cada destas funções deverá conter apenas a palavra chave *return* seguida do elemento enumerado *ok*.
20. Crie o arquivo "*aula0604.c*" contendo o programa de testes para a função *CalcularDeterminanteMatriz*. O programa deverá receber, através dos argumentos de linha de comando, a ordem da matriz e os valores de cada elemento desta matriz. O programa deverá exibir a matriz (em formato de matriz) recebida via CLI e seu determinante. Os valores da matriz deverão ser exibidos com no mínimo 5 casas decimais. Este programa NÃO pode utilizar alocação dinâmica de memória. Por enquanto, teste o programa apenas com matrizes de ordem 1, 2 e 3.

    Exemplo:
    ./aula0604 3 a_{11} a_{12} a_{13} a_{21} a_{22} a_{23} a_{31} a_{32} a_{33}

    No exemplo acima, a matriz tem ordem 3.
21. Inclua as declarações necessárias nos arquivos de dependências.
22. Gere e teste as 16 versões do executável *aula0604*.
23. Submeta os arquivos "*aula0601.h*", "*aula0601.c*", "*aula0604.c*" e "**makefile*" ao sistema de controle de versão.
24. Recupere uma cópia de leitura do arquivo "*aula0604.c*" e uma cópia de escrita dos arquivos "*aula0601.h*", "*aula0601.c*" e "**makefile*".
25. Inclua, no arquivo "aula0601.c", a implementação completa da função *CalcularMenorComplementar*. As matrizes deverão ser manipuladas através de indexação de vetores. A implementação desta função NÃO poderá utilizar alocação dinâmica de memória.

    Para a implementação desta função será necessário utilizar a função *CalcularDeterminanteMatriz*.
26. Crie o arquivo "*aula0605.c*" contendo o programa de testes para a função *CalcularMenorComplementar*. O programa deverá receber, através dos argumentos de linha de comando, a ordem da matriz, os valores de cada elemento desta matriz e os valores correspondentes à linha e à coluna do elemento para o qual se deseja calcular o menor complementar. O programa deverá exibir a matriz (em formato de matriz) recebida via CLI, o elemento em questão e o menor complementar correspondente. Os valores da matriz deverão ser exibidos com no mínimo 5 casas decimais. Este programa NÃO pode utilizar alocação dinâmica de memória. Por enquanto, teste o programa apenas com matrizes de ordem 1, 2 e 3.

    Exemplo:
    ./aula0605 3 2 1 a_{11} a_{12} a_{13} a_{21} a_{22} a_{23} a_{31} a_{32} a_{33}

    No exemplo acima, a matriz tem ordem 3, e se deseja o menor complementar para o elemento a_{21}, ou seja, o elemento índices 1 e 0 na matriz correspondente.
27. Inclua as declarações necessárias nos arquivos de dependências.
28. Gere e teste as 16 versões do executável *aula0604*.
29. Submeta os arquivos "*aula0601.h*", "*aula0601.c*", "*aula0605.c*" e "**makefile*" ao sistema de controle de versão.

30. Recupere uma cópia de leitura do arquivo "*aula0605.c*" e uma cópia de escrita dos arquivos "*aula0601.h*", "*aula0601.c*" e "**makefile*".
31. Inclua, no arquivo "aula0601.c", a implementação completa da função *CalcularComplementoAlgebrico*. As matrizes deverão ser manipuladas através de indexação de vetores. A implementação desta função NÃO poderá utilizar alocação dinâmica de memória.

    Para a implementação desta função será necessário utilizar a função *CalcularMenorComplementar*.
32. Crie o arquivo "aula0606.c" contendo o programa de testes para a função *CalcularComplementoAlgebrico*. O programa deverá receber, através dos argumentos de linha de comando, a ordem da matriz, os valores de cada elemento desta matriz e os valores correspondentes à linha e à coluna do elemento para o qual se deseja calcular o cofator (ou complemento algébrico). O programa deverá exibir a matriz (em formato de matriz) recebida via CLI, o elemento em questão e o cofator correspondente. Os valores da matriz deverão ser exibidos com no mínimo 5 casas decimais. Este programa NÃO pode utilizar alocação dinâmica de memória. Por enquanto, teste o programa apenas com matrizes de ordem 1, 2 e 3.

    Exemplo:
    ./aula0606 3 2 1 a_{11} a_{12} a_{13} a_{21} a_{22} a_{23} a_{31} a_{32} a_{33}

    No exemplo acima, a matriz tem ordem 3, e se deseja o cofator para o elemento a_{21}, ou seja, o elemento índices 1 e 0 na matriz correspondente.
33. Inclua as declarações necessárias nos arquivos de dependências.
34. Gere e teste as 16 versões do executável aula0606.
35. Submeta os arquivos "*aula0601.h*", "*aula0601.c*", "*aula0606.c*" e "**makefile*" ao sistema de controle de versão.
36. Recupere uma cópia de leitura dos arquivos "aula0601.h" e "aula0606.c" e uma cópia de escrita dos arquivos "aula0601.c" e "**makefile*".
37. Atualize a implementação da função *CalcularDeterminanteMatriz* com o código necessário para o cálculo do determinante de matrizes com ordem igual ou superior a 4.
38. Atualize o arquivo "aula0604.c" com as declarações necessárias para o teste da nova versão da função *CalcularDeterminanteMatriz*.
39. *Gere e teste as 16 versões do executável aula0604.*
40. Gere novamente as 16 versões da biblioteca *libmatematica.a*.
41. Limpe o diretório (*make clean-all*).