Aula Prática 5 – Roteiro

Objetivos:

- Manipulação de Matrizes

Versão Inicial: 05/06/2023

Prazo: 12/05/2023 – 8:00

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 (incluindo maiúsculas, minúsculas, caracteres especiais e extensões, se existentes).
- **As tarefas deverão ser executadas na ordem solicitada neste roteiro.**
- Os arquivos de dependências deverão possibilitar que a compilação e que 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 como o valor padrão para a ferramenta de compilação e de *linkedição*.

  Para a definição da ferramenta desejada, deverá ser utilizada uma macro no *FreeBSD* e um argumento com o valor desejado no *Linux*. As duas macros utilizadas deverão ser ***GCC*** e ***CLANG*** (definidas usando a opção de linha de comando **-D** do comando ***make***). O argumento, identificado por ***cc***, deverá ser igual a ***GCC*** ou a ***CLANG***.
- Independente da ferramenta utilizada para a compilação, as opções de compilação poderão ser redefinidas no instante da execução do comando ***make*** (mantendo-se a exibição de todas as mensagens de advertência, definida pelo valor ***-Wall***). O valor padrão para estas opções deverá ser ***-Wall -ansi***.

  Estas opções poderão ser redefinidas 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 *Linux* deverá ser definido o argumento ***dialeto*** com um dos seguintes valores ***ANSI***, ***C89***, ***C90***, ***C99*** ou ***C11***.
- Os arquivos de dependências deverão incluir a macro ***DIALECT*** contendo o dialeto a ser utilizado na compilação do código. Esta macro será inicialmente igual a ***ansi*** e poderá ser alterada para ***c89***, ***c90***, ***c99*** ou ***c11*** de acordo com o esquema definido acima.
- Os arquivos de dependências deverão incluir também a macro ***STANDARD*** contendo a opção de linha de comando correspondente ao dialeto selecionado. Se, por exemplo, o dialeto selecionado for o ***ANSI***, esta macro deverá ser igual a ***-ansi***. Por outro lado, se o dialeto for uma das outras quatro opções, esta macro deverá ser igual a ***-std=CXX***, onde ***XX*** deverá ser substituído pelo número correspondente (se o dialeto for igual a ***C89***, *XX* deverá ser igual a ***89***, se o dialeto for igual a ***C90***, ***XX*** deverá igual a ***90*** e assim por diante).
- A *linkedição* deverá utilizar a opção ***-Wall***.
- 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 em questão será igual a zero.
- Seguem alguns exemplos (todos devem funcionar):

- ***make*** - compila/*linkedita* (tanto no *FreeBSD*, quanto no *Linux*) com a ferramenta e dialeto padrões, ou seja, ***gcc*** e **ANSI** respectivamente.
- ***make clean-all all***
- ***make clean-all aula01***
- ***make clean aula0101***
- ***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 *Linux*).
- ***make cc=CLANG*** - compila/*linkedita* usando o ***clang*** e o dialeto **ANSI** (somente *Linux*).
- ***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 dialero=C99*** - compila/*linkedita* usando o ***clang*** e o dialeto **C99** (somente *Linux*).
- ***make cc=GCC dialeto=C90*** - compila/*linkedita* usando o ***gcc*** e o dialeto **ANSI** (somente *Linux*).

- Inclua, no início de todos os arquivos solicitados (código-fonte e arquivos de dependências), os seguintes comentários (**sem caracteres especiais**):

```
Universidade Federal do Rio de Janeiro
Escola Politecnica
Departamento de Eletronica e de Computacao
EEL270 - Computacao II - Turma 2023/1
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, o seguinte comentário:

```
$RCSfile$
```

1. Crie o arquivo ***aula0501.h*** contendo o protótipo definido abaixo. Este arquivo deverá conter também as macros e os tipos necessários para a implementação da função *ExibirMatriz*.

   A macro referente à combinação ***ifndef*** e ***define***, ou seja, **_AULA0501_**, deverá ser definida como uma *string* valendo: "*@(#)aula0501.h $Revision$*".

```
tipoErros
ExibirMatriz (unsigned short,        /* numero de linhas da matriz (E) */
              unsigned short,        /* numero de colunas da matriz 1 (E) */
              long double [ ][ ]);   /* matriz (E) */
```

   Nos protótipos definidos neste roteiro os argumentos de entrada serão identificados pela letra E, os de saída pela letra S e os de entrada/saída por E/S.

   Considere, neste e nos demais exercícios, que as matrizes podem ter no máximo 1000 linhas por 1000 colunas. Estes dois valores devem ser definidos através de duas macros, ou seja, ***NUMERO_MAXIMO_LINHAS*** e ***NUMERO_MAXIMO_COLUNAS***, respectivamente.

   O tipo *tipoErros* deverá ser definido como um enumerado que deverá conter os elementos correspondentes a todas as condições de erros identificadas durante a implementação das

funções solicitadas neste roteiro. Este tipo deverá ser criado com um elemento correspondendo à execução com sucesso das funções (por exemplo, ***ok***) valendo zero. Os demais elementos deverão ser valores inteiros positivos.

2. Crie o arquivo ***aula0501.c*** contendo a implementação da função *ExibirMatriz*. A matriz deverá ser exibida como mostrado no exemplo abaixo:

```
| 0.55000        14.29800        -12.90000 |
| -4.70000       65.00000        7.86000   |
```

   Note que os valores das matrizes deverão ser exibidos com no mínimo 10 casas decimais.

3. Crie o arquivo ***aula0502.c*** contendo a implementação do programa de testes para a função *ExibirMatriz*. Este programa deverá receber, através de argumentos de linha de comando, o número de linhas, o número de colunas e o conteúdo da matriz. O conteúdo deverá ser recebido de forma que os elementos da primeira linha sejam recebidos primeiro, depois os da segunda linha e assim por diante.

   Exemplo:

   ./aula0502 2 3 a_{11} a_{12} a_{13} a_{21} a_{22} a_{23}

   Neste exemplo, serão recebidos os valores dos elementos de uma matriz 2 x 3. Lembre-se que o elemento a_{ij} da matriz (Matemática) corresponde ao elemento $a_{i-1\ j-1}$ na matriz definida na linguagem C.

4. Inclua, nos arquivos de dependências, as macros ***AULA0502OBJS*** e ***AULA05***. Altere o valor da macro ***EXECS***, de forma que inclua o valor da macro ***AULA05***. Inclua também os objetivos ***aula05*** e ***aula0502*** com os comandos correspondentes.

5. Gere e teste as 20 versões do executável ***aula0502***.

6. Submeta os arquivos ***aula0501.h***, ***aula0501.c***, ***aula0502.c*** e ****makefile*** ao sistema de controle de versão.

7. Recupere uma cópia de leitura do arquivo ***aula0502.c*** e uma cópia de escrita dos arquivos ***aula0501.h***, ***aula0501.c*** e dos arquivos de dependências.

8. Inclua, nos arquivos de dependências, as macros ***LIBMATEMATICAOBJS*** (correspondendo ao arquivo ***aula0501.o***) e ***LIBMATEMATICA*** (correspondendo ao arquivo ***libmatematica.a***). O valor da macro ***LIBS*** deverá ser atualizado de forma que inclua esta 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 20 versões da biblioteca.

10. Inclua, no arquivo ***aula0501.h***, o protótipo da função *MultiplicarMatrizes*.

```
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) */
                     long double [ ][ ],  /* matriz 1 (E) */
                     long double [ ][ ],  /* matriz 2 (E) */
                     long double [ ][ ]); /* matriz produto (S) */
```

11. Inclua, no arquivo ***aula0501.c***, 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.

12. Crie o arquivo ***aula0503.c*** contendo o programa de testes para a função solicitada no item anterior. 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, utilizando a função *ExibirMatriz*, a primeira matriz, a segunda matriz e a matriz produto. Este programa **NÃO** poderá utilizar alocação dinâmica de memória.

    Exemplo:

    ./aula0503 3 2 2 4 a_{11} a_{12} a_{21} a_{22} a_{31} a_{32} b_{11} b_{12} b_{13} b_{14} b_{21} b_{22} b_{23} b_{24}

    No exemplo acima, a matriz A terá 3 linhas x 2 colunas, enquanto que a matriz B terá 2 linhas x 4 colunas.

13. Inclua as declarações necessárias nos arquivos de dependências.

14. Gere e teste as 20 versões do executável ***aula0503***.

15. Submeta os arquivos ***aula0501.h***, ***aula0501.c***, ***aula0503.c*** e ****makefile*** ao sistema de controle de versão.

16. Recupere uma cópia de leitura do arquivo ***aula0503.c*** e uma cópia de escrita dos arquivos ***aula0501.h***, ***aula0501.c*** e dos arquivos de dependências.

17. Inclua, no arquivo ***aula0501.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) */
                      long double [ ][ ],   /* matriz original (E) */
                      long double [ ][ ]);  /* matriz transposta (S) */
```

18. Inclua, no arquivo ***aula0501.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.

19. Crie o arquivo **aula0504.c** contendo o programa de testes para a função solicitada no item anterior. 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 (utilizando a função *ExibirMatriz*). Este programa **NÃO** pode utilizar alocação dinâmica de memória.

    Exemplo:

    ./aula0504 3 2 2 4 a_{11} a_{12} a_{21} a_{22} a_{31} a_{32} b_{11} b_{12} b_{13} b_{14} b_{21} b_{22} b_{23} b_{24}

    No exemplo acima, tem-se uma matriz 3 x 2.

20. Inclua as declarações necessárias nos arquivos de dependências.

21. Gere e teste as 20 versões do executável ***aula0504***.

22. Submeta os arquivos ***aula0501.h***, ***aula0501.c***, ***aula0504.c*** e ****makefile*** ao sistema de controle de versão.

23. Recupere uma cópia de leitura do arquivo ***aula0504.c*** e uma cópia de escrita dos arquivos ***aula0501.h***, ***aula0501.c*** e dos arquivos de dependências.

24. Inclua, no arquivo ***aula0501.h***, os protótipos abaixo:

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

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


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

25. Inclua, no arquivo **aula0501.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. O código correspondente ao cálculo do determinante de matrizes com ordem igual ou superior a 4 deverá ser incluído na função quando solicitado neste roteiro.

26. Inclua, no arquivo ***aula0501.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 correspondente à execução com sucesso (por exemplo, ***ok)***.

27. Crie o arquivo **aula0505.c** contendo o programa de testes para a função de cálculo do determinante. 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:

./aula0505 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.

28. Inclua as declarações necessárias nos arquivos de dependências.
29. Gere e teste as 20 versões do executável ***aula0505***.
30. Submeta os arquivos ***aula0501.h***, ***aula0501.c***, ***aula0505.c*** e ****makefile*** ao sistema de controle de versão.
31. Recupere uma cópia de leitura do arquivo ***aula0505.c*** e uma cópia de escrita dos arquivos ***aula0501.h***, ***aula0501.c*** e dos arquivos de dependências..
32. Inclua, no arquivo ***aula0501.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*.
33. Crie o arquivo ***aula0506.c*** contendo o programa de testes para a função solicitada no item anterior. 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:

    ./aula0506 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.

34. Inclua as declarações necessárias nos arquivos de dependências.
35. Gere e teste as 20 versões do executável ***aula0506***.
36. Submeta os arquivos ***aula0501.h***, ***aula0501.c***, ***aula0506.c*** e ****makefile*** ao sistema de controle de versão.
37. Recupere uma cópia de leitura do arquivo ***aula0506.c*** e uma cópia de escrita dos arquivos ***aula0501.h***, ***aula0501.c*** e dos arquivos de dependências.
38. Inclua, no arquivo ***aula0501.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*.

39. Crie o arquivo ***aula0507.c*** contendo o programa de testes para a função solicitada no item anterior. 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:

    ./aula0507 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 a21, ou seja, o elemento índices 1 e 0 na matriz correspondente.

40. Inclua as declarações necessárias nos arquivos de dependências.
41. Gere e teste as 20 versões do executável ***aula0507***.
42. Submeta os arquivos ***aula0501.h***, ***aula0501.c***, ***aula0507.c*** e ****makefile*** ao sistema de controle de versão.
43. Recupere uma cópia de leitura do arquivo ***aula0507.c*** e uma cópia de escrita dos arquivos ***aula0501.h***, ***aula0501.c*** e dos arquivos de dependências.
44. Atualize a implementação da função *CalcularDeterminanteMatriz* com o código necessário para o cálculo do determinante de matrizes com ordem igual ou superior a 4.
45. Gere e teste as 20 versões do executável ***aula0505***.
46. Submeta o arquivo ***aula0501.c*** ao sistema de controle de versão.
47. Recupere uma cópia de leitura do arquivo ***aula0501.c***
48. Gere novamente as 20 versões da biblioteca ***libmatematica.a***.
49. Gere e teste as funções (complemento algébrico e menor complementar) anteriores com matrizes com ordem igual ou superior a 4.
50. Limpe o diretório (make clean-all).
51. Arquivos que devem ser disponíveis ao final da aula:

    Subdiretório "~/private/EEL270/2023-1/Aulas-Praticas"

    - *aula0501.h*
    - *aula0501.c*
    - *aula0502.c*
    - *aula0503.c*
    - *aula0504.c*
    - *aula0505.c*

- *aula0506.c*
- *aula0507.c*
- *BSDmakefile*
- *GNUmakefile*

Além dos correspondentes gerados pela ferramenta de controle de versão (localizados no subdiretório RCS) e dos arquivos gerados nas aulas práticas anteriores.