UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE – CCENS/ UFES
Departamento de Computação

Trabalho prático 1 – Estruturas de Dados I

Data de entrega: 10/10/19

OBSERVAÇÕES:

1. Plágio não será tolerado. Trabalhos iguais serão penalizados com **nota zero**.

2. O trabalho poderá ser feito **em duplas (no máximo).**

3. Você pode acrescentar funções que julgar necessárias para completar as tarefas.

4. Não deixe de indentar e documentar seu código para facilitar o entendimento do trabalho.

5. Para entrega do trabalho, o aluno deverá compactar a pasta contendo o projeto e todos os seus arquivos (**incluindo makefile** para compilar e rodar o trabalho) e **dar o nome da dupla** ao arquivo compactado. Enviar para helderamendes@gmail.com.

Matrizes esparsas

Matrizes esparsas são matrizes nas quais a maioria não possuem valor armazenado. Para estas matrizes, podemos economizar um espaço significativo de memória se apenas os termos necessários forem armazenados. As operações usuais sobre estas matrizes (somar, multiplicar, inverter, pivotar) também podem ser feitas em tempo muito menor. Uma maneira eficiente de representar estruturas com tamanho variável e/ou desconhecido é através de alocação encadeada, utilizando listas. Para o caso de matrizes esparsas, cada coluna da matriz será representada por uma lista linear circular com cabeça. Da mesma maneira, cada linha da matriz também será representada por uma lista linear circular com uma célula cabeça. Cada célula da lista representará os termos armazenados na lista:

Como exemplo, considere a seguinte matriz esparsa:

$$A = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 0 & -5 & 3 \\ 18 & 2 & 0 & 0 \\ 0 & 1 & 0 & -2 \end{bmatrix}$$

A representação da matriz *A* pode ser vista na Figura 1.

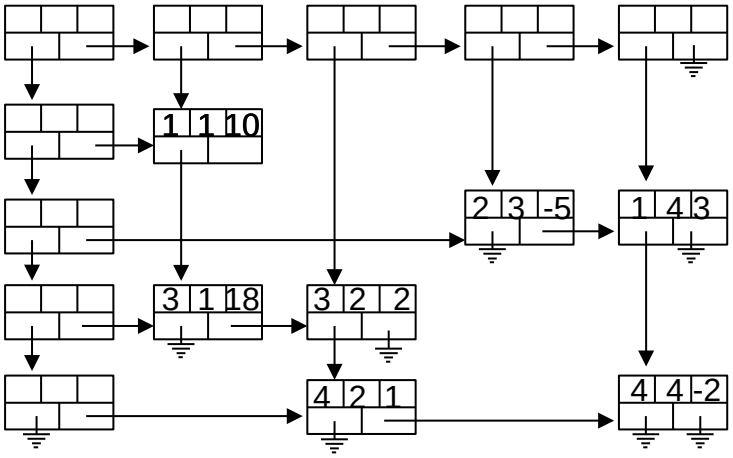

Figura 1: Exemplo de Matriz Esparsa

Com esta representação, uma matrix esparsa m x n com *r* elementos gastará (*m*+*n*+*r*) células.

Dada a representação acima, o trabalho consiste em desenvolver cinco funções em C, conforme especificação abaixo:

- leMatriz (matriz *A); Este procedimento lê do teclado elementos diferentes de zero de uma matriz e monta a estrutura especificada acima. O usuário deverá informar a linha, a coluna e o valor do elemento. Após a entrada dos dados a função deverá armazenar o valor na matriz esparsa A. Para finalizar o usuário deverá digitar o valor -1 para a linha ou para a coluna:
- void apagaMatriz (matriz *A); Este procedimento remove todas as áreas de memória alicadas para as células da matriz *A*
- void somaMatriz (matriz *A, matriz *B, matriz *C); Este procedimento recebe como parâmetros as matrizes *A* e *B*, devolvendo em *C* a soma de *A* com *B*.
- void imprimeMatriz (matriz *A); Este procedimento imprime (uma linha da matriz por linha da saída) a matriz A. Para as posições que não possuírem elementos armazenados a função deverá imprimir o valor zero.

Para inserir e retirar células das listas que formam a matriz, crie procedimentos especiais para este fim.

Implemente sua função principal (main) no arquivo main.c. Ela deve imprimir na tela um menu com as seguintes opções:

----------MENU----------

1 – Criar Matriz

2 – Le Matriz

3 – Apaga Matriz

4 – Soma Matriz

5 - Imprimir Matriz

6 - Sair

Escolha uma opção:

Além do código implementado, deverá ser entregue, também, uma documentação contendo as seguintes informações:

- uma descrição das estruturas de dados
- uma descrição da sintaxe e semântica das operações implementadas