DCC013 - Estrutura de Dados / DCC107 - Laboratório de Programação II

Matrizes especiais

Você pode utilizar qualquer ambiente de programação para desenvolver sua atividade. Ao final, **copie e cole o seu código-fonte com a resposta aqui mesmo neste documento**, dentro dos espaços indicados para isso e **preservando a identação do código**. **Depois que terminar sua avaliação, não se esqueça de entregar sua atividade!** Fique atento ao relógio, pois as atividades entregues com atraso não serão aceitas.

A matriz onda ampliada com alternância de sinal é uma matriz com `m` linhas e `n` colunas em que os elementos da primeira coluna aparecem com sinal trocado na terceira, que aparecem com o sinal trocado na sexta, que aparecem com sinal trocado na décima, e assim por diante. Os demais elementos da matriz são iguais a zero. Note que a quantidade de colunas com elementos nulos vai aumentando em progressão aritmética. Assume-se aqui que esse tipo de matriz possui ao menos uma linha e 3 colunas. Veja a seguir um exemplo deste tipo de matriz, $A_{5\times10}$:

$$A_{5\times10} = \begin{bmatrix} 5 & 0 & -5 & 0 & 0 & 5 & 0 & 0 & 0 & -5 \\ 8 & 0 & -8 & 0 & 0 & 8 & 0 & 0 & 0 & -8 \\ -9 & 0 & 9 & 0 & 0 & -9 & 0 & 0 & 0 & 9 \\ -1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 2 & 0 & -2 & 0 & 0 & 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

O TAD `MatrizEspecial` representa esse tipo de matriz. Os valores devem ser armazenados no TAD `MatrizEspecial` por meio de uma representação linear com um único vetor (`vet`) e de modo que a quantidade de elementos armazenados seja mínima. Para esse TAD, desenvolver:

A) Construtor (que recebe as dimensões `m` e `n` da matriz como parâmetro) e destrutor da classe. Não esquecer de indicar no construtor a **forma** de representação dos elementos não nulos da matriz. Imprimir a mensagem de erro “Dimensões inválidas” e sair do programa, caso os valores de `m` e `n` não sejam válidos para esse tipo de matriz.

```
MatrizEspecial::MatrizEspecial(int mm, int nn)
{
    if (mm < 1 && nn < 3) {
        cout << "Dimensões inválidas" << endl;
        exit(2);
    }

    nl = mm;
    nc = nn;
    vet = new float[nl];
```

```
    ///guarda-se os elementos da primeira coluna
}

MatrizEspecial::~MatrizEspecial()
{
    delete [] vet;
}
```

B) A operação `int detInd(int i, int j)` para verificar se os índices `i` e `j` da matriz são válidos (retornar -1 caso não sejam), e determinar o índice do vetor em que se encontra o elemento na posição indicada por `i` e `j`. Retornar -2 se os índices representam uma posição de valor zero. Use outras *flags* se necessário.

```
int MatrizEspecial::detInd(int i, int j)
{
    if(i >= 0 && i < nl && j >= 0 && j < nc)
        //se primeira coluna
        if (j==0) {
            return i;
        } else {
            ///usando uma estrutura de repetição
            ///-3 se sinal trocado
            /*bool mesmoSinal = false;
            int pa = 2;
            for(int k=2; k<=j; k+=pa) {
                if (k==j) {
                    if (mesmoSinal) {
                        return i;
                    } else {
                        return -3;
                    }
                }
                mesmoSinal = !mesmoSinal;
                pa++;
            }
                return -2; ///se não está na primeira coluna e em
nenhuma outra que possui elementos não nulos, então é um elemento
nulo
            */
              ///usando equações de PA
              ///temos uma coluna com valores quando n é inteiro, e
n indica o número de termos (ou seja, há troca de sinal quando n é
ímpar)
            float n = (-3 + sqrt(9+8*j))/2;
            //se n é inteiro
```

```
            if ( n == (int)n ) {
                if ( (int)n%2 == 0 ) {
                    return i;
                } else {
                    return -3;
                }
            } else {
                return -2;
            }
        }
    else
        return -1;
}
```

C) A operação pública **float** `get(`**int** `i,` **int** `j)` para retornar o valor da posição `i` e `j` da matriz. Imprimir a mensagem de erro “Índices inválidos” e sair do programa, caso os índices não representem uma posição válida para a matriz.

```
float MatrizEspecial::get(int i, int j)
{
    int k = detInd(i, j);
    if(k != -1)
        if (k == -2 ) {
            return 0;
        } else if (k==-3) {
            k = detInd(i, 0);
            return -vet[k];
        } else {
            return vet[k];
        }
    else
    {
        cout << "ERRO: Indice invalido!" << endl;
        exit(1);
    }
}
```

D) A operação pública **void** `set(`**int** `i,` **int** `j,` **float** `val)` para atribuir o valor na posição `i` e `j` da matriz. Emitir a mensagem de erro: “Tentando atribuir valor nao zero em posição impropria”, caso o usuário tente atribuir um valor diferente de zero na posição que deve ser zero. Imprimir a mensagem de erro “Índices inválidos”, caso os índices não representem uma posição válida para a matriz. Note que aqui não há a instrução de saída do programa no caso de erro.

```
void MatrizEspecial::set(int i, int j, float val)
{
    int k = detInd(i, j);
    if(k != -1)
        if (k==-2) {
            if (val != 0){
                    cout << "Tentando atribuir valor nao zero em
posição impropria" <<endl;
            }
        } else if ( k==-3 ) {
            k = detInd(i, 0);
            vet[k] = -val;
        } else {
            vet[k] = val;
        }
    else
        cout << "ERRO: Indice invalido!" << endl;
}
```