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 indentação 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.

Matrizes esparsas (com muitos elementos zeros) surgem em diversos problemas. Considere a matriz a seguir em que os elementos não nulos são apenas os contidos na segunda metade da diagonal principal, na segunda metade da diagonal secundária e na última linha da matriz.

$$M_{[7x7]} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 5 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 10 & 0 \\ 5 & 9 & 9 & 3 & 5 & 6 & 6 \end{bmatrix}$$

Para essa atividade, implemente o TAD `MatrizEspecial` para representar a matriz por meio da representação linear com um único vetor (`vet`) tal que a quantidade de elementos armazenados seja mínima. E desenvolva:

a) O construtor e o destrutor. O construtor deve permitir que a matriz tenha ordem no mínimo 3, deve dimensionar o `vet` corretamente para armazenar apenas o número de elementos necessários e deve preencher a matriz.

```
MatrizEspecial::MatrizEspecial(int ordem)
{
    if(ordem < 3)
    {
        cout << "Dimensões Inválidas" << endl;
        exit(1);
    }

    m = ordem;
    vet = new int[2 * m - 2];
    //Representado como vet = [Lm-1, Dp, Ds] onde o termo em
comum das diagonais está em Dp
}
```

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

b) A operação `int getInd(int i, int j)` para verificar se os índices i e j da matriz são válidos e retornar o índice de `vet` de acordo com o formato armazenado (-2 para índices que não precisam ser armazenados ou -1 caso não sejam válidos).

```
int MatrizEspecial::getInd(int i, int j)
{
    if(i >= 0 && i < m && j >= 0 && j < m)
    {
        if(i == (m -1))
        {
            return j;
        } else if(i == j)
        {
            return m + j;
        } else if(i + j == m - 1)
        {
            return m + (m/2 + 1) + j;
        } else {
            return -2;
        }
    } else {
        return -1;
    }
}
```

c) A operação `int get(int i, int j)` que retorna o valor armazenado na posição i, j da matriz, retorna zero caso o índice seja -2 e imprime uma mensagem de erro e sai do programa caso o índice seja inválido.

```
// Cole aqui sua resposta

int MatrizEspecial::get(int i, int j)
{
    int k = getInd(i, j);
    if(k != -1)
    {
        if(k == -2)
        {
            return 0.0;
        } else {
            return vet[k];
```

```
        }
    } else {
        cout << "Indice Inválido: get" << endl;
        exit(1);
    }
}
```

d) A operação `void set (int i, int j, int val)` que atribui o valor `val` à posição i, j da matriz e imprime mensagem de erro para índice invalido.

```
// Cole aqui sua resposta


void MatrizEspecial::set(int i, int j, int val)
{
    int k = getInd(i, j);
    if(k != -1)
    {
        if(k == -2 && val != 0)
        {
            cout << "Tentando colocar um valor diferente de 0
onde deve ser 0" << endl;
            exit(1);
        } else {
            vet[k] = val;
        }
    }
}
```