Relatório do projeto de Matrizes Esparsas

**Danilo de Oliveira Nunes – 18203**

**João Victor Javitti Alves - 18207**

**Introdução:**

O projeto desenvolvido tem como objetivo solucionar problemas variados utilizando o que se aprendeu nas aulas do professor. Portanto, seria uma aplicação prática da programação a fim de exercitar a lógica e adaptá-la para situações diversas, sempre buscando a formação profissional.

Sentimo-nos agradecidos por termos a oportunidade de fazer o projeto e praticar programação. Também termos a oportunidade de programar um aplicativo que nos ajuda a nos organizarmos.

**Desenvolvimento:**

* **Lógica:**

Em ambas as classes nos métodos e atributos foram feitos comentários que explicam com detalhes mais profundos o funcionamento e a lógica deles, portanto apenas será explicado os detalhes mais abrangentes e gerais.

**- Classe Celula:**

Na classe célula foram utilizados os atributos: valor(double, número real) que armazena o valor desejado dentro da célula, linha(int, número inteiro) que armazena a linha na qual a célula se encontra dentro da matriz, coluna(int, número inteiro) que armazena a coluna na qual a célula se encontra dentro da matriz, direita(Celula, um objeto de mesma classe) que armazena um ponteiro para outra célula da matriz a direita desta na linha, abaixo (Celula, um objeto de mesma classe) que armazena um ponteiro para outra célula da matriz abaixo desta na coluna.

Para todos estes atributos foram feitos getters e setter os quais tem a finalidade de serem usados para manipular os valores da célula, mudando o valor nela guardado, ou a sua posição na matriz ou a de outra referênciada na mesma, sendo ambas excluída, incluída na matriz.

O construtor desta classe tem como parâmetros o valor a linha e a coluna, que são os únicos necessários para a criação de uma célula na matriz, sendo posteriormente através da aplicação alterado os atributos direita e abaixo.

A classe possui um método ToString que escreve um par ordenado com o valor, a linha e coluna respectivamente para fins de consulta na aplicação.

**- Classe ListaLigadaCruzada:**

O construtor desta classe tem como parâmetros o valor a linha e a coluna, que são os únicos necessários para a criação de uma matriz, sendo posteriormente através da aplicação alterado os atributos cabeça no construtor que criará a partir dela uma sequência de células que representará as linhas e outra que representará as colunas, todas valendo o valor padrão, que é zero, sendo utilizadas para a inserção, remoção e consulta das outras células, sendo usadas como base para todas as operações, seguindo a regra de deixar a matriz com as linhas e colunas circulares.

O Atributo Desalocada() retorna se a matriz foi instanciada ou se possui um nó de celula cabeça não nulo, boolean.

Já o método ApagarMatriz() void, desaloca a matriz da memória por tornar nulo o atributo cabeça, assim removendo o ponteiro principal de único acesso a matriz da memória, então o garbage collector da linguagem apaga a matriz da memória stack e heap.

ExibirDataGridView() é um método que retorna void, usado para percorrer a matriz consultando seus valores e colocando em um vetor que armazenará os valores da linha da matriz , exibindo linha por linha no gridView que é passado como parâmetro.

O método inserir elemento recebe como parâmetros um double elemento que será o valor a ser armazenado pelo novo elemento, dois inteiros que serão a linha e a coluna desejadas, respectivamente, assim, percorreremos as bases até chegar na linha e coluna desejada, depois percorrermos a partir delas até encontrar a posição desejada, assim inserindo o elemento criado na matriz, atualizando os ponteiros dos elementos posteriores e anteriores a este na coluna e na linha em que se encontra.

Já o método ValorDe() retorna double, é responsável por percorrer a matriz achando a posição desejada da linha e coluna que são passadas por parâmetro, retornando o valor armazenado na célula correspondente

O método RemoverEm() retorna boolean, recebe como parâmetros a linha e a coluna do elemento desejado para a exclusão, então percorre a matriz armazenando os anteriores e os atuais que será o próprio elemento só que na linha e na coluna da matriz, então o eliminaremos por desalocá-lo da memória mas sem perder os posteriores a ele por fazer os anteriores receber o ponteiro dele pros próximos, caso dê certo retornará verdadeiro mas se o elemento tiver como o valor padrão 0, ele não será removido, retornando falso. Pois não estará alocado.

Já o método SomarNaColuna() retorna void, recebe como parâmetro o valor a ser somado e a coluna desejada para que se efetue a soma, então a partir disso ele percorre as colunas da matriz até achar a coluna desejada, então ele percorre a linha da coluna somando o valor em todos os elementos, mas se a soma de um elemento resultar em 0, ele será exluido da matriz, caso não ele continuará alocado nela.

O método SomarMatrizes() e MultiplicarMatrizes() retornam ListaLigadaCruzada(), seguem uma lógica bem parecida, ambos percorrem a matriz passada por parâmetro assim como a matriz que vai ser somada com a do parâmetro e consulta todos os valores de ambas as duas, mesma linha e coluna, se a soma ou a multiplicação de ambos resultar em 0, ele não é inserido na matriz resultante, caso contrário ele será inserido na posição em que os valores originais sem encontravam nas matrizes. Quando acabar ele retornará a matriz resultado.

**- Formulário Windows:**

Foram utilizadas 3 listaLigadaCruzada como atributo para essa classe, dois para matrizes criadas pelo usuário e uma para o resultado. Um método necessário foi o LerArquivo que lia um arquivo fornecido pelo usuário com formatação padrão:

“Linhas

Colunas

(Linha, Coluna, Valor) [...]”

Sendo os valores de Linha e Coluna da célula tendo como formatação obrigatória 2 algarismos (caso menor que dez um zero antes), já o Valor teria 4 algarismos sendo um deles o sinal caso o número fosse positivo, dois números antes da vírgula e um depois.

Este era lido e convertido para uma listaLigadaCruzada que depois era passada para o atributo requisitado pelo usuário e o exibia em seu respectivo DataGridView.

O Formulário também tinha como função realizar as operações requisitadas no enunciado do projeto, chamando os métodos já implementados na ListaLigadaCruzada, como Inserir e Excluir Elementos. Desejamos fazê-lo da maneira mais prática possível, portanto mantivemos um único textBox para valores fornecidos pelo usuário ou pelo programa e apenas dois NumericUpDown sendo um para Linhas e outro para colunas.

* **Erros e dificuldades:**

Enquanto desenvolviamos o projeto encontramos várias dificuldades e erros, das mais diversas, dentre elas estão:

- Entendimento do funcionamento da matriz e da sua utilização através dos métodos desenvolvidos, demoramos a entender e deixar os métodos corretos, cometendo vários erros que até mesmo foram de operadores

- Nos comandos de for colocávamos várias vezes operadores errados como > sendo que se aplicava >= para se percorrer a matriz e achar o valor de linha ou de coluna desejado

- Invertemos nos getters e setters as colunas e linhas, isso foi causado por um erro de atenção.

-Nos métodos Multiplicar e Somar Matrizes mudávamos a linha de uma única matriz o que resultava em operarmos com o cabeçalho da outra. Provavelmente erro de atenção, fora que o código de um foi usado de base para o outro o que fez o erro se repetir.

-Ao inserir elementos na matriz, devido a um erro de atenção, percorríamos as linhas com base nas colunas. Resultava em vários erros na inserção e, consequentemente, em toda a utilização da matriz.

-Ao Somar uma Constante a Coluna, não havíamos codificado para somar a constante as células vazias.

-No método InserirElemento esquecemos de levar em consideração que a célula poderia ser incluída no meio de duas. Foi devidamente corrigido.

-Em vários métodos, ao percorrer a matriz não levávamos em consideração as células vazias no meio, o que levou a erros em vários outros métodos. Devidamente corrigido.

**Auxilio dos monitores:**

Durante o desenvolvimento do projeto encontramos várias dificuldades e dúvidas em relação aos mais diversos elementos do projeto, com isso nos sentimos confortável em pedir a ajuda da monitoria que me recebeu com muito carinho e dedicação, nos deram atenção e ajudaram bastante com todas as minhas questões, devemos muito a eles, sem eles não teria desenvolvido o projeto da maneira esperada e muitos erros seriam encontrados na correção.

* **Conclusão:**

Com este trabalho chegamos a várias conclusões, dentre elas:

* A lógica é algo que não se aprende, mas se aprimora com o treinamento, que consiste em aplicar a programação para solucionar problemas diversos.
* Embora o repertório de funções e comandos dos alunos não seja tão grande, é possível se atingir qualquer meta, desde que haja dedicação e esforço.
* Em programação, é essencial que o código esteja bem feito e organizado para que os monitores e professores possam entender e ajudar. O uso de comentários, por exemplo, torna o programa muito mais esclarecedor para os olhos de quem o vê.
* É possível desenvolver diversas aplicações com funções diversas com a linguagem de programação, sendo muito úteis para solucionar os nossos problemas do dia a dia.
* Quanto mais nos desafiamos com exercícios diversos, mais aprendemos na prática e aprimoramos nossas habilidades com programação e nosso conhecimento da área, assim nos tornamos profissionais melhores e mais preparados para o mercado de trabalho. Assim como diz um velho ditado: “A pratica leva a perfeição”.
* Com o desenvolvimento do trabalho aprendemos muitas coisas novas e relembramos como realizar certas operações que já vimos, aprimoramos assim a nossa lógica e nossa capacidade de resolver problemas.
* Com esse trabalho podemos aprender novos comandos e formas variadas de resolver um mesmo problema, desenvolver uma ferramenta de auxilio, etc... Pois assim podemos nos desafiar com novas propostas, o que nos torna programadores melhores e mais capacitados, e consequentemente, profissionais melhores.
* Ao fazer trabalhos, nós aprendemos a fazê-los da maneira correta, nós entendemos como devemos prosseguir para realizar um relatório técnico, como o código deve estar organizado, como devemos comentar o código, assim nós aprimoramos nossas habilidades de fazer projetos, programas e aplicações que mais tarde será cobrada no mercado de trabalho. Assim nos tornamos trabalhadores mais eficientes e profissionais mais valiosos.