UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

INSTITUTO METRÓPOLE DIGITAL

DISCIPLINA: INTRODUÇÃO ÀS TÉCNICAS DE PROGRAMAÇÃO

ALUNO: DÊNIS ROCHA DA SILVA

RELATÓRIO - CALCULADORA DE MATRIZES

1. Introdução e contexto

O presente relatório se destina a englobar os aprimoramentos feitos neste projeto que vem sido desenvolvido desde o início da disciplina, que consiste em uma calculadora de matrizes em linguagem C, ofertando ao usuário a realização de operações como soma e subtração de matrizes, implementados na etapa anterior, e também: multiplicação de matrizes, cálculo do determinante e da matriz inversa, elaborados nesta nova fase. Além disso, também foram incluídas algumas funcionalidades para melhorar a experiência do usuários e serão melhor detalhadas no decorrer do texto.

2. Análise Técnica

O projeto foi desenvolvido no Visual Studio Code, utilizando a extensão para C, que permite compilar e executar arquivos .c diretamente no terminal embutido. Nesta nova versão do código, foram incorporadas operações adicionais que ampliam o escopo do trabalho: multiplicação de matrizes, cálculo de determinante (por meio do Teorema de Laplace) e cálculo da matriz inversa (através da matriz adjunta).

O programa agora utiliza de maneira mais intensa estruturas fundamentais da linguagem C, incluindo:

- Loops aninhados;
- Manipulação de matrizes;
- Funções com retorno e funções auxiliares;
- Recursão (para o cálculo do determinante);
- Uso de ponteiros para receber tipos diferentes em uma função;
- Controle de fluxo através de estruturas condicionais;
- Uso de funções modulares para cada operação;
- Função de reinicialização da execução.

Esses elementos, junto à reorganização das funções, tornam o programa mais robusto, modular e preparado para futuras expansões.

2.1 Funcionamento Geral do Programa

Ao iniciar a execução, o usuário é requisitado a fornecer o número equivalente a uma das operações oferecidas, são elas:

1 - Soma
2 - Subtração
3 - Multiplicação
4 - Determinante
5 - Matriz Inversa

A entrada é validada por meio de um laço do…while, assegurando que apenas valores numéricos correspondentes às operações existentes serão aceitos. Caso o usuário insira 1 ou 2, ele precisará fornecer posteriormente o número de linhas e colunas, porém se o valor for acima de 2, ele precisa fornecer apenas o tamanho da matriz, pois as operações como determinante e a inversa só fazem sentidos para matrizes quadradas (que tem o mesmo número de linhas e colunas) e o código usado na multiplicação só está funcionando em matrizes de mesma ordem. Portanto, resumindo melhor, se o usuário escolher:

- Soma ou subtração: Exigem números de linhas e de colunas, que serão usados nas duas matrizes.
- Multiplicação, Determinante e inversa: Exige apenas o tamanho, que será atribuído tanto ao número de linhas, quanto ao de colunas.

A próxima etapa é fornecer os elementos da matriz, em que o usuário deve inserir cada um dos termos em sua devida posição, essa etapa faz uso de laços aninhados do tipo for para acessar os elementos nas suas respectivas coordenadas i e j. Uma vez que as matrizes estiverem preenchidas, o código irá executar a operação selecionada, retornando uma matriz para os casos da multiplicação e de inversão, e um número inteiro para o determinante. A seguir será feito um detalhamento quanto aos fundamentos matemáticos por trás dessas três operações.

Multiplicação

Implementada seguindo a definição clássica, e usando como referência o pseudocódigo disponível na página 249 da 6ª edição do Kenneth & Rosen (2010):

ALGORITMO 1 Multiplicação de Matrizes.

procedure *multiplicacao de matriz* (**A**, **B**: matrizes)
for $i := 1$ **to** m
    for $j := 1$ **to** n
    begin
        $c_{ij} := 0$
        for $q := 1$ **to** k
            $c_{ij} := c_{ij} + a_{iq}b_{qj}$
    end
{$\mathbf{C} = [c_{ij}]$ é o produto de **A** e **B**}

Figura 1 - Algoritmo disponível em Kenneth & Rosen (2010) para a multiplicação de matrizes

A operação é efetuada em três laços aninhados, garantindo a multiplicação adequada entre linhas e colunas.

Determinante

O código adota uma abordagem recursiva baseada no Teorema de Laplace, reduzindo o tamanho das matrizes, para formar submatrizes menores até atingir o tamanho 2x2, onde será feito o cálculo do determinante através do produto de cada elemento da matriz pelos seus respectivos cofatores, que foram obtidos conforme a equação abaixo:

$$C_{i,j} = (-1)^{i+j} \det(A_{ij})$$

Equação 1 - Em que C representa o cofator, i e j são os índices da matriz e A é uma submatriz

Para facilitar a programação, optou-se por fixar a primeira linha da matriz, ou seja, deixando o índice 0 fixo, e permutar apenas as colunas para gerar as submatrizes. Além disso, foi usada uma estrutura condicional para verificar se o índice j é par ou ímpar, o que determinaria o sinal usado no cofator. Por fim, a soma de cada um dos valores resulta no determinante da matriz, e essa foi a etapa mais difícil de todo o projeto.

Matriz Inversa

A matriz inversa foi calculada em cima da seguinte equação:

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \cdot \operatorname{adj}(\mathbf{A}).$$

Equação 2 - Em que A elevado a -1 é a inversa da matriz A, e adj(A) é a matriz adjunta de A.

Percebe-se que na equação 2, o determinante da matriz está no denominador, logo para que a matriz tenha inversa, o determinante deve ser um valor diferente de zero, e essa foi a primeira verificação feita dentro da função, caso o valor seja zero, o usuário recebe uma mensagem informando e o código é reiniciado. Se o valor for diferente de zero, a operação procede.

Boa parte dos elementos complexos da inversão já haviam sido programados na operação anterior, e essas funções apenas foram reutilizadas, tanto para obter o determinante, quanto para a matriz de cofatores. Depois, bastou inverter os índices da matriz de cofatores, transpondo suas linhas e colunas, obtendo assim a matriz adjunta. Por fim, bastou dividir os elementos da matriz adjunta pelo determinante da matriz, resultando na matriz inversa.

Adicionalmente, em virtude da operação de divisão, optou-se pelo uso de uma matriz tipo float para armazenar os valores da matriz inversa, o que trouxe a

necessidade de atualizar função de impressão das matrizes para receber ambos os tipos int e float. Para isso, foi criado um typedef chamado TipoMatriz, para identificar se a matriz fornecida é do tipo int ou float. A nova função imprimirMatriz passou então a utilizar um ponteiro genérico (void *) como entrada, permitindo que os dois tipos fossem recebidos e abrindo espaço para adicionar ainda outros tipos, se necessário fosse. Dentro da função, esse ponteiro é convertido para o tipo correto (int* ou float*) com base no valor de TipoMatriz. Além disso, como a matriz é tratada como um vetor linear na memória, cada elemento é acessado através da fórmula i * colunas + j, tornando a função flexível e reaproveitável.

Além das operações descritas, uma nova implementação inserida foi ao final da execução do código, adicionando uma função que pergunta se o usuário deseja reiniciar o código, ou seja, escolher uma nova operação, ou simplesmente encerrar a execução. Esse mecanismo tornou o programa mais prático e facilitou a testagem de múltiplas operações em sequência, o que ajudou a avaliar se os cálculos estavam sendo feitos corretamente.

3. Estrutura Atualizada do Código

Com a expansão do projeto, novas funções foram adicionadas. A seguir um resumo das principais funções, incluindo as recém-criadas:

Funções anteriores (já existentes):

- escolherOperacao() – valida e retorna a operação
- inserirLinhasOuColunas() – define dimensões
- criarMatriz() – preenche matriz
- operacaoMatrizes() – soma/subtração
- imprimirMatriz() – imprime matriz formatada

Novas funções adicionadas:

multiplicarMatrizes() - Recebe duas matrizes quadradas e calcula a multiplicação conforme a regra tradicional.

criarSubmatriz() - Gera as submatrizes necessárias para o cálculo recursivo do determinante.

determinante() - Função recursiva que se aplica a múltiplos tamanhos, calcula os cofatores e acumula resultados a cada chamada.

Inversa() - Combina cálculo do determinante, dos cofatores e da matriz adjunta, retornando, no fim, a matriz inversa.

reiniciarOuEncerrar() - Controla o fluxo final e permite reiniciar a aplicação.

4. Implementação e Reflexão

Nesta etapa de evolução do projeto, mesmo com uma maior familiaridade com a linguagem C, o nível de dificuldade no cálculo das operações excedeu as expectativas, e foi necessário um aprofundamento matemático para construir um fundamento mais robusto e assim conseguir implementar essas soluções. Entre os principais obstáculos encontrados estavam:

- Criação de submatrizes dinamicamente, garantindo que a recursão do determinante não herdasse valores errados;
- Evitar variáveis globais para garantir cálculo consistente;
- Buscar meios permitir o polimorfismo em C, usando funções que recebam tanto float quanto int em um mesmo parâmetro;
- Estudar melhor os conceitos por trás do cálculo do determinante e da inversa.

5. Conclusão

O projeto passou por uma evolução significativa desde sua primeira versão. Além das operações básicas, o programa agora contempla:

- Multiplicar matrizes;
- Calcular determinantes;
- Gerar matrizes inversas;
- Uso de recursão;
- Criar submatrizes dinamicamente;
- Permite o reinício automático no final da execução;

O projeto está mais próximo de uma calculadora matricial completa, interseccionando diversos aspectos da programação em C com conteúdos de álgebra linear. Quanto a possibilidades de ampliação, seria interessante incorporar uma interface gráfica através de alguma biblioteca, e ainda permitir que o usuários escolha o número de matrizes que quer somar ou multiplicar, por exemplo.