# Trabalho Prático 0

## Operações com Matrizes Alocadas Dinamicamente

**Antony Guilherme Costa do Amaral**
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil
antonyguilherme@ufmg.br

# 1 Introdução

Esta documentação possui como objetivo explicitar o funcionamento do programa que realiza operações (**soma**, **multiplicação** e **transposição**) com matrizes, sendo estas alocadas dinamicamente, ou seja, podendo ter qualquer tamanho de linhas e/ou colunas. Contanto que a matriz fornecida respeite as limitações da operação escolhida.

## 2 Implementação

O programa foi escrito utilizando a linguagem C, compilada pelo compilador G++ da GNU Compiller Collection. Além disso, o programa foi testado em um ambiente Linux (Ubuntu-WSL) tendo disponível 8GB de RAM DDR4 e um processador AMD Ryzen 5.

### 2.1 Estrutura de Dados

A implementação do programa teve como base o tipo abstrato de dados de uma matriz. Esta é alocada dinamicamente, pois dessa maneira é possível manipular matrizes de tamanhos variados sem a necessidade estipular um tamanho máximo.

Esse tipo abstrato de dados foi criado utilizando um struct, um tipo que consegue agrupar variáveis sob um mesmo tipo de dado. Como utilizado no exemplo proposto para o trabalho, foi implementado, no tipo de dado especificado, “tamx” e “tamy” as quais definem as dimensões da matriz. Isso tem como intuito evitar o custo linear de algumas operações, ou seja, com esses valores já em memória, não mais é necessário estimar o tamanho da matriz através de iterações.

### 2.2 Organização do código

O código para a implementação do programa requisitado foi realizado em apenas três classes, assim como o exemplo disponibilizado. A classe principal **mat.c,** contém todas as funções desde leitura dos arquivos a operações matriciais. Ademais, as outras duas classes **matop.c** e **memlog.c,** possuem como respectivas responsabilidades controlar o fluxo da aplicação e criar o histórico de leitura e escrita de memória.

## 2.3 Operações Matriciais

O programa possui três operações, sendo estas, **soma**, **multiplicação** e **transposição**. Todas as operações são realizadas, a partir de matrizes obtidas dos arquivos que são fornecidas ao programa e ao final é gerado um arquivo contendo o resultado. Dito isso, a operação de soma espera dois arquivos contendo matrizes que devem possuir as mesmas dimensões. Além disso, a operação de multiplicação espera também dois arquivos contendo matrizes, entretanto a primeira matriz fornecida deve possuir o número de linhas iguais ao número de colunas da segunda matriz. Ademais a operação de transposição somente espera um arquivo contendo uma matriz de qualquer tamanho desejado.

# 3 Instruções Para Compilação e Execução

Para realizar a compilação basta acessar a raiz do projeto e executar o arquivo **makefile** regra **all**. Ademais, para se executar o programa basta digitar o seguinte comando:

**bin/matop -(operação) -p multlog.out -l –1 (arquivo_matriz1) –2 (arquivo_matriz2) -o (arquivo_saída)**

Além disso, as operações são soma**(-s)**, multiplicação**(-m)**, transposição**(-t).** Sendo que estas possuem números de matrizes necessárias diferentes, como explicitado nos tópicos acima. **Outro fator que vale ser destacado, é que todos arquivos considerados devem ficar ou serão gerados na raiz do projeto**.

# 4 Análise de Complexidade

**m- Número de linhas da matriz**
**n- Número de colunas da matriz**

## 4.1 Operação de Soma

A função **opcaoDeSoma** realiza a operação de soma entre duas matrizes e para isso essa função necessita realizar a alocação dinâmica de três matrizes, sendo estas matrizes de soma e resultado. Essa tarefa é executa pela função **criaMatriz** e para isso ela realiza o equivalente assintótico de tempo à **Θ(m)**.

Ademais, a função principal também deve realizar a soma propriamente dita e para isso ela excuta a função **somaMatrizes** a qual tem como equivalente assintótico de tempo **Θ(mxn).** Por fim, também se faz necessário a execução de três funções **imprimeMatriz, criarArquivoDaMatriz e destroiMatrizes** que possuem como equivalentes assintóticos de tempo, **Θ (mxn)**, **Θ(mxn)** e **Θ(m)**. Assim temos, que o equivalente assintótico de tempo da função **OpcaoSoma** é **Θ (mxn)**.

## 4.2 Operação de Multiplicação

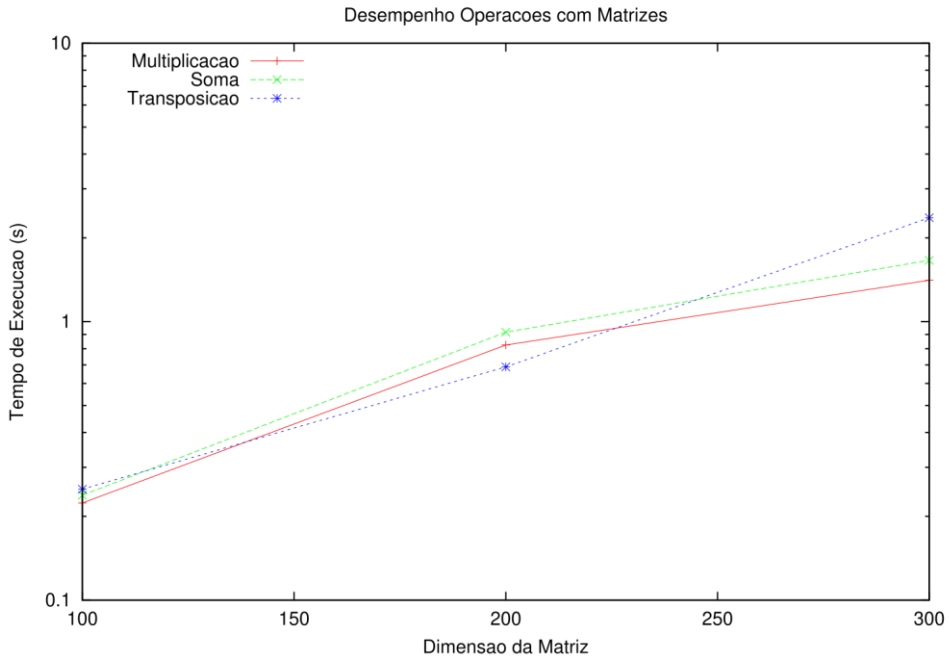
A função **opcaoDeMultiplicacao** realiza a operação de multiplicação entre duas matrizes. Para isso, é necessário realizar a alocação de três matrizes utilizando as funções **criarMatrizDeUmArquivo** e **criaMatriz** as quais possuem como equivalente assintótico de tempo **Θ(mxn)**. Posteriormente, a função que de fato realiza a multiplicação é a **multiplicaMatrizes** que possui como equivalente assintótico **Θ ((m^2 )x n).** Por fim, são executadas as três funções **imprimeMatriz, criarArquivoDaMatriz e destroiMatrizes** que possuem como equivalentes assintóticos, **Θ (mxn)**, **O(mxn)** e **Θ (m).** Assim temos que o equivalente assintótico de tempo da função **opcaoDeMultiplicacao** é **Θ ((m^2)xn).**

## 4.3 Operação de Transposição

A função **opcaoDeTransposicao** realiza a operação de transposição para uma matriz qualquer. Para isso é necessário realizar a locação dinâmica por meio da função **criarMatrizDeUmArquivo** e **criaMatriz**, as quais possuem como equivalente assintótico de tempo **Θ(mxn).** Posteriormente a operação de transposição é realizada, de fato, pela função **transpoeMatriz** a qual possui como equivalente assintótico de tempo **Θ(mxn)**. Por final, são executadas três funções **imprimeMatriz**, **criarArquivoDaMatriz** e **destroiMatriz**. Estas possuem como equivalentes assintóticos de tempo **Θ (mxn)**, **Θ (mxn)** e **Θ (m)**, respectivamente. Assim temos, que a função principal deste caso possui equivalência assintótica de tempo igual à **Θ (mxn)**.

## 5 Gráfico De Desempenho

Pode-se perceber que o tempo de execução cresce conforme o tamanho da matriz. Pois, conforme analisado anteriormente na análise de tempo e espaço, o tempo de execução e o espaço utilizado depende diretamente das dimensões das matrizes envolvidas nas operações.




## 6 Análise de espaço

## 6.1 Criação da Matriz

A criação da matriz é realizada pela função **criarMatrizDeUmArquivo** a qual aloca dinamicamente a matriz propriamente dita. Para isso essa função faz uso da **criaMatriz** que possui como equivalente assintótico de espaço **Θ (mxn)**. Ademais, dois laços são realizados na função principal para a leitura e verificação do conteúdo do arquivo de origem da matriz. Essas iterações possuem como equivalente assintótico de espaço **Θ (mxn)**. Assim temos que essa função possui como equivalente assintótico **Θ (mxn)**;

## 7 Conclusão

O Trabalho teve como principal objetivo não só relembrar os conceitos abordados em disciplinas anteriores sobre c e c++, mas principalmente abordar a alocação dinâmica e seus impactos na execução de um programa. Ademais, o impacto causado pelo acesso desordenado a memória também possui elevada importância para o desempenho do programa. Por fim, pode-se concluir que os recursos de alocação e uso de memória devem ser usados com parcimônia, pois seus impactos na execução de um programa podem causar sérios entraves em seu custo.

# Bibliografia

Algoritmos - Teoria e Prática - 3º Edição Americana traduzida, por Thomas Cormen

C Language Reference | Microsoft Docs