Relatório - Projeto 01 – Matrizes Esparsas - Estrutura de Dados

Equipe :

Matriculas:

Professor: Atílio Gomes

Departamento de Ciência da Computação
Universidade Federal do Ceará (UFC) – Quixadá, CE – Brasil

Agosto de 2021.

1 Introdução

O problema deste trabalho consiste em implementar um algoritmos para a criação de uma matriz utilizando TAD. Mas o que é uma matrizes esparsas? Matrizes esparsas são matrizes nas quais a maioria das posições é preenchida por zeros afim de economizar espaço de memória, dado tal contexto foi implementada uma solução desenvolvida em C++ para

2 Divisão de Tarefas

A divisão das tarefas na resolução e implementação para o trabalho foi balanceada, onde os alunos dividiram as implementação do algoritmo de forma que os dois fizessem a mesma quantidade. Em razão das circustâncias atuais em relação a pandemia foi necessário um desdobramento de nossa parte para poder realizar de forma eficiente a atividade proposta, para isso procuramos utilizar a tecnologia ao nosso favor, utilizamos o WhatsApp e Google Meet para as definições de tarefas e metas, além de uitilizar-mos o Github para manter o versionamento e compartilhamento de código entre os membros da equipe.

3 Metodologia

Para a criação do projeto foram necessários 3 arquivos:

Projeto.h - contém os cabeçalhos da funções criadas na classe na qual a mesma define a estrutura do projeto.

matriz.cpp – aqui foi onde o código realmente foi implementado as funcionalidades e requisitos exigidos pelo o trabalho.

main.cpp – aqui é onde o código é executado e as funções são chamadas, para fim de patricidade e usabilidade criamos um menu na qual o mesmo chama as funcionalidades e as executa, tudo mais simples e direto possível.

Na parte da **main.cpp** ficou acordado que as matrizes seriam guardadas em um vector para guardar mais matrizes e permtir mais operações.

Também foi criado um arquivo **Makefile** para ajudar na compilação e execução do projeto, criamos isso como forma de praticidade tanto para nós, como para quem avaliar ou testar o projeto. Abaixo segue a explicação com uma tabela de funcionalidades e explicando como cada uma delas funciona.

O código foi implementado utilizando o paradigma da orientação a objetos para permitir uma maior facilidade e reaproveitamento de código, algumas dificuldades para entender o contexto do trabalho foram aparecendo, mas acabaram no final sendo solucionadas.

4 Comandos e execução

Abaixo segue a lista de comandos que podem ser utilizados no programa e seu significado:

Comando.	Significado.
insere <valor> [i] [j] [k]	Insere um valor na linha i e coluna j na matriz k. Ex: insere 5 2 2 1
imprime [k]	Imprime todos os valores contidos na matriz k. Ex: imprime 2
valor [i] [j] [k]	Mostra o valor armazenado na linha i e coluna j na matriz k. Ex: valor 1 0 3
soma [fst] [snd]	Realiza a soma entre duas matrizes e acaba retornando uma terceira, fst e snd são os números das matrizes. Ex: soma 1 2
multiplica [fst] [snd]	Realiza a multiplicação entre duas matrizes e acaba retornando uma terceira, fst e snd são os números das matrizes. Ex: soma 1 2
sair	Libera as matrizes e fecha o programa.
libera	Libera as matrizes armazenadas.
cria <nome_do_arquivo>	Cria uma matriz a partir de um arquivo de texto. Ex: cria cria_matriz_arquivo.txt

OBSERVAÇÃO: TODOS OS COMANDOS DEVEM DIGITADOS EM MINÚSCULO.

4.1 Compilação e execução

Para uma melhor praticidade tanto para execuções e testes, criamos um arquivo makefile que contém 3 funcionalidades que serão descritas abaixo:

4.1.1 Compilação:

Para compilar o código basta apenas utilizar o seguinte comando no terminal:

```
g++  main.cpp Projeto.h  matriz.cpp –o projeto
```

Mas como estamos utilizando o makefile, basta apenas digitar no terminal (dentro do diretório do projeto):

```
make all
```

4.1.2 Execução:

Para executar o código basta apenas utilizar o seguinte comando no terminal:

```
make run
```

4.1.3 Voltando ao estado original:

Para ao estado inicial do projeto basta apenas utilizar o seguinte comando no terminal:

```
make clean
```

5 Funções implementadas

Abaixo segue a lista de quais funções requisitadas foram implementadas com sucesso:

- ✓ Matriz(int m, int n)
- ✓ ~Matriz();
- ✓ void insert(int i, int j, double value);
- ✓ double getValue(int i, int j);
- ✓ void print();
- ✓ • Matriz *soma(Matriz *A, Matriz *B);
- ✓ Matriz *multiplica(Matriz *A, Matriz *B);
- ✓ Matriz *lerMatrizDeArquivo(std::string nome do arquivo)