**1 Introdução**

Foi proposto que eu fizesse uma aplicacao que representasse, e fizesse operacoes utilizando matrizes esparsas, carregando-as de arquivos previamente preparados. Depois de carregar as matrizes, é necessario dar a possibilidade de fazer operaces como soma, e multiplicacao, impressao, etc.

**2- Solução Proposta**

main – a main é responsavel por organizar o menu, e atraves de um switch redireciona para as respectivas funcoes externa, tem o papel tambem de receber corretamente o arquivo e carrega-lo na matriz a ou b.

lerMatriz – tem o papel de ler e carregar a matriz da memoria secundaria, para a principal, para que possamos manipula-la. Lê o arquivo e confere se ele realmente existe na pasta origem e se sim chama a funcao setElem, senao gera mensagem de erro e a escreve num arquivo .txt gerado na pasta.

Matriz\_esparsa.h- armazena a estrutura principal do programa e troca de nome atravez do typedef e guarda as chamadas das funcoes.

startHead – Aloca um espaco de memoria referente a uma celula, e a posiciona na cabeca da matriz.

setCollumHead - Aloca um espaco de memoria referente a uma celula, e a posiciona na cabeca da coluna

imprimeMatriz – Imprime o conteudo da matriz, separa as colunas com “|” e entre as linhas existe a separacao por meio de underline “\_\_\_\_\_\_\_\_\_”;

cria – Aloca espaco para uma celula e seta os elementos de inicio, fimlinha e fimcoluna como NULL, depois chama a funcao “startHead”.

setElem- possibilita a insercao dos elementos da matriz, obviamente fazendo testes de validacao .

apagaMatriz – libera o espaco alocado pela matriz tomando cuidado para não perder a referencia nesse processo, por isso é utilizado um auxiliar “aux.”

multiplicaMatriz – multiplica a matriz A pela matriz B e armazena o resultado na matriz C, obviamente fazendo testes de validacao.

somaMatriz- Soma a matriz A com matriz B e armazena o resultado na matriz C, obviamente fazendo testes de validacao assegurando que essa operacao é possivel.

getIndice – obtem o valor do indice da matriz.

rowHead - Aloca um espaco de memoria referente a uma celula, e a posiciona na cabeca da linha.

**3 – Analise de Complexidade**

Utilizei a abordagem do pior caso (big O) para analizar esse programa .

main – complexidade depénde do comportamento do do-while O(1).

lerMatriz –O(1).

Matriz\_esparsa.h- O(1).

startHead – O(n).

setCollumHead –O(1).

imprimeMatriz – O(n^2).

cria –O(1).

setElem- O(n).

apagaMatriz – O(n^2).

multiplicaMatriz – O(n^2).

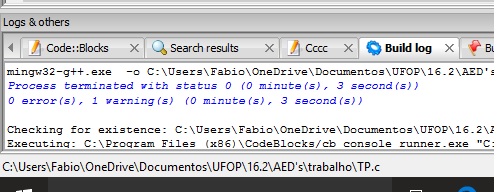
somaMatriz- O(n^2)

getIndice – O(n).

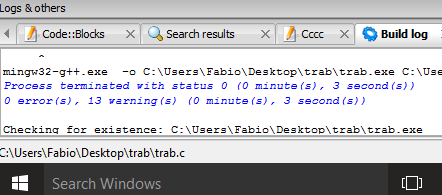
rowHead – O(n)

**4 – Experimento**

Diversos experimentos foram realizados buscando perceber o que o programa poderia executar, e concluí que ele faz tudo que foi pedido, escolhi destacarum ponto interessante sobre o programa, mesmo o trabalho1 (upgrade do trabalho0) tendo muito mais funçoes do que o trabalho0, o tempo de execuçao dos dois é praticamente o mesmo, fato devido à utilização de estruturas de dados, que acessam as informações de forma consideravelmente mais rapida.



Dados de compilação do T0



Dados de compilação do T1

**5 – Maquina Utilizada**

A aplicação foi inteiramente escrita e compilada em um notebook HP 1000, com processador i3 (2,2GHz), 2GB de memoria primária , 500GB de memória secundaria. Sistema operacional windows 10 – 64bits.

**6 - Conclusão**

Minha principal dificuldade foi manipular a lista , e relacionar essas TAD’s entre si, por muitas vezes me perdi no raciocinio para realizar uma simples tarefa do programa, mas com um mapa do programa feito à mão pude entender melhor, e consegui realizar tudo que foi pedido no trabalho.

**7 - Referências**

**- Slides disponibilizados pelo professor**

**-** [**www.cprogressivo.net**](http://www.cprogressivo.net)