

RELATÓRIO - ENCADEAMENTO

Universidade Tecnológica Federal do Paraná

Alunos: Rafael Campos Nunes, RA: 1866885

Mikael Messias, RA: 1697650

INTRODUÇÃO

O trabalho tem como objetivo montar um programa capaz de criar uma matriz em que seus elementos apontam um para os outros em quatro sentidos: direita, esquerda, baixo e acima.

Além disso foram implementadas funções para operar sobre essa matriz como: operação de impressão da matriz, remoção, inserção de elementos e pesquisa de um elemento a partir de seu valor.

PROBLEMAS

O primeiro problema com qual nos deparamos foi como criar a matriz encadeada. A solução encontrada foi criar uma listas duplamente encadeadas e criar sub listas para cada elemento da lista anterior e por fim concatená-las uma a uma.

O processo para criação da matriz $n \times m$ pode ser descrito em 3 passos:

1. Cria-se as cabeças de cada lista duplamente encadeada de modo que cada cabeça também seja duplamente encadeada;
2. Criar sub listas respectivas a cada cabeça;
3. Concatenar cada sublista com a próxima até que não haja mais sublistas.

A figura 1 mostra como o primeiro passo se traduz para um conceito visual. Onde temos nós (cabeças), indicadas por: H , H_1 e H_2 , mas podem ser facilmente generalizadas para H , H_1 , ..., H_{n-1} , H_n cabeças.

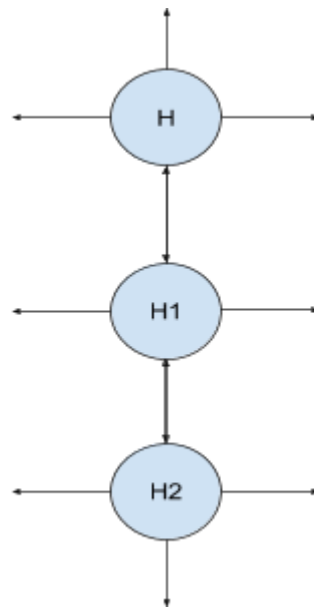


Figura 1: Cabeças da lista duplamente encadeada

Então, para cada cabeça foi criada uma sublista, contendo cada nó da matriz, duplamente encadeada para, na sequência concatenar cada sublista.

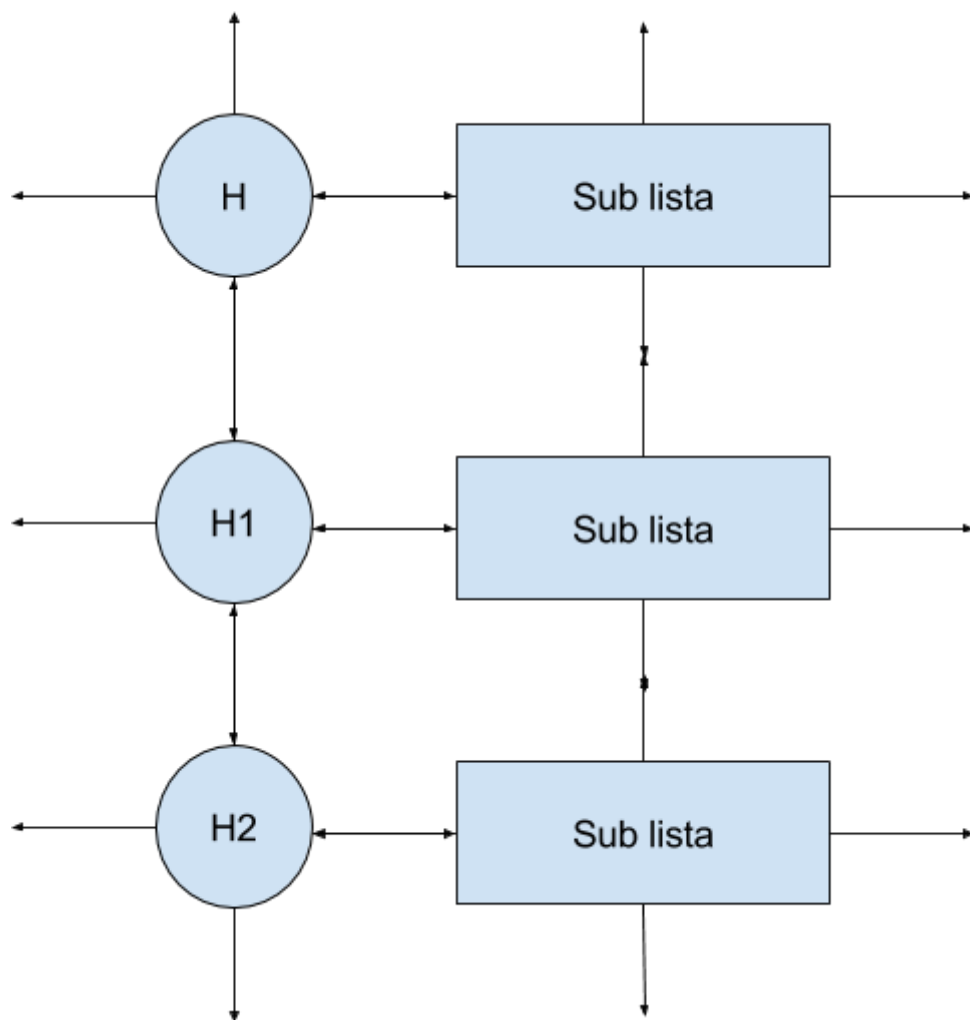


Figura 2: Cabeças com suas sublistas correspondentes

Após esse processo cada sub lista foi concatenada, isto é, os ponteiros de cada nó da sublista de cima e de baixo foram dispostos para apontar para cada elemento da sub lista respectivamente, a próxima imagem explica o resultado final da matriz.

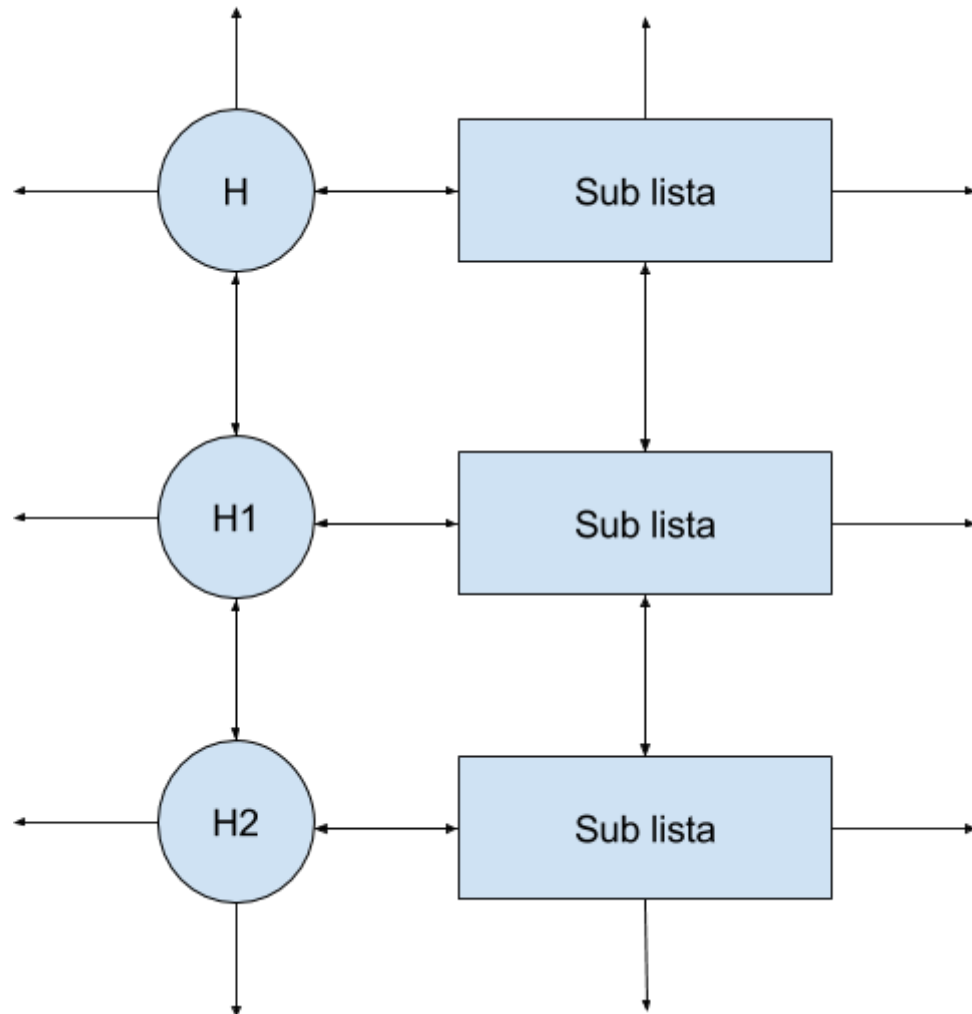


Figura 3: Sub listas concatenadas

Após a criação da matriz foi necessário verificar se tudo funcionava corretamente. Um programa principal em conjunto com macros desenvolvidos ajudaram a realizar os testes. Os testes estão presentes no arquivo `main.c` e utilizaram alguns macros criados para testar se o retorno da função era o retorno esperado especificado, essas definições estão presentes em `macros.h` e `assertions.h`.

A documentação de todas as funções do projeto estão localizadas dentro da pasta docs sob o diretório html no arquivo `index.html`. De lá, pode ser visto na aba *Files* a definição de cada função da matriz como também o significado de seus parâmetros e o que ela retorna. Abaixo pode ser vista a página de documentação do arquivo `matrix.c`

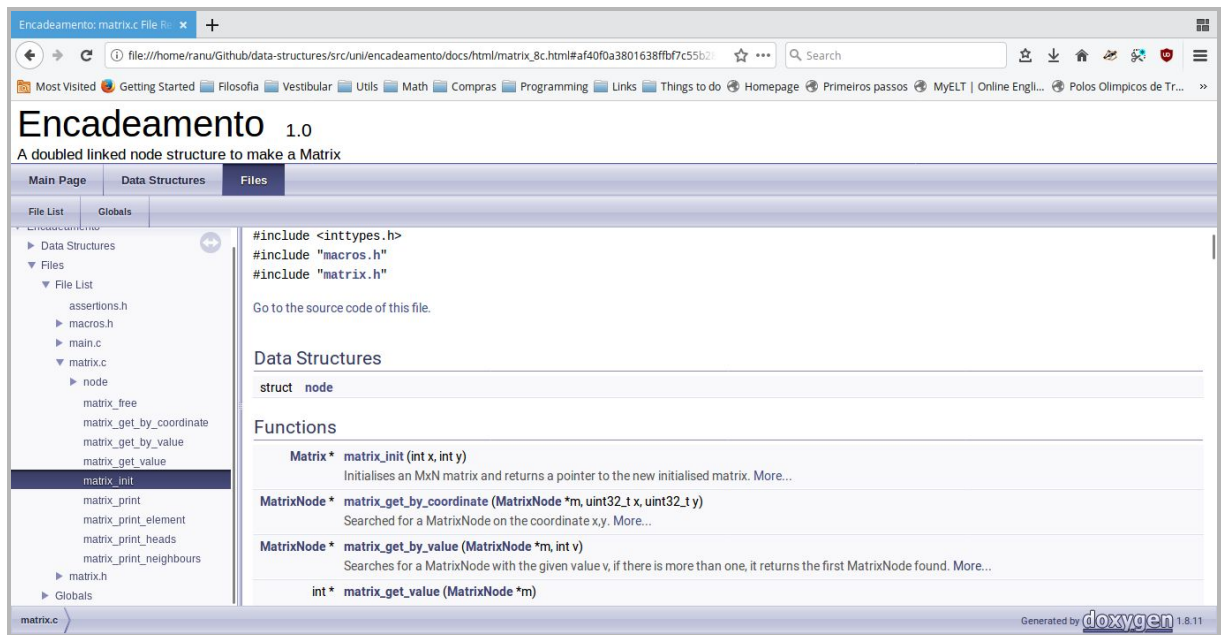


Figura 4: Documentação do programa

O programa principal pode ser executado de duas maneiras, no modo interativo e no modo de testes. O primeiro permite ao usuário interagir com a matriz, criando-a e com a possibilidade de navegar sobre os elementos assim como remover e inserir elementos.

Para entrar no modo interativo é necessário chamar o programa e passar como argumento ao mesmo a flag `--interactive`.

```
$ ./matrix.out --interactive
```

Após executar o programa com a flag de iteratividade ele mostrará um menu com as opções disponíveis:

```

+  x  ..atrix.out --interactive  .../CS/3st period/SGBD  ranu@ranu-laptop: ~

ranu@ranu-laptop ~ dev % : ~/Github/data-structures/src/uni/encadeamento
[0] % ./matrix.out --interactive

1 -> Create a Matrix
2 -> Create a MatrixNode
3 -> Get a MatrixNode by coordinate
4 -> Get a MatrixNode by value
5 -> Print the Matrix
6 -> Print the Matrix HEADS
7 -> Print the neighbours of a given MatrixNode
8 -> Remove a MatrixNode
9 -> Exit
: 1
Rows: 3
Columns: 3

1 -> Create a Matrix
2 -> Create a MatrixNode
3 -> Get a MatrixNode by coordinate
4 -> Get a MatrixNode by value
5 -> Print the Matrix
6 -> Print the Matrix HEADS
7 -> Print the neighbours of a given MatrixNode
8 -> Remove a MatrixNode
9 -> Exit
: 5
-- [0 (0,0)] -- [1 (0,1)] -- [2 (0,2)] --
-- [1 (1,0)] -- [2 (1,1)] -- [3 (1,2)] --
-- [2 (2,0)] -- [3 (2,1)] -- [4 (2,2)] --

1 -> Create a Matrix
2 -> Create a MatrixNode
3 -> Get a MatrixNode by coordinate
4 -> Get a MatrixNode by value
5 -> Print the Matrix
6 -> Print the Matrix HEADS
7 -> Print the neighbours of a given MatrixNode
8 -> Remove a MatrixNode
9 -> Exit
:

```

Figura 5: Programa executando no modo iterativo

Para executar o programa no modo testes basta chamar o programa com o parâmetro `--test` e a quantidade de linhas e colunas da matriz, sendo este segundo e terceiro parâmetro opcionais.

```
+ x .../uni/encadeamento .../CS/3st period/5GBD ranu@ranu-laptop: ~

Testing matrix_print_element | matrix_get_by_coordinate

[3 (1,2)]
[4 (2,2)]
[5 (3,2)]

Testing matrix_print_neighbours

      NULL
[0 (0,0)] [1 (0,1)] [2 (0,2)]
      [2 (1,1)]

      NULL
      [0 (0,0)] [1 (0,1)]
      [1 (1,0)]

[1 (1,0)] [2 (1,1)] [3 (1,2)]
      [3 (2,1)]

[2 (2,0)] [2 (1,1)] [3 (2,1)] [4 (2,2)]
      [4 (3,1)]

[2 (1,1)] [2 (0,2)] [3 (1,2)] [4 (1,3)]
      [4 (2,2)]

Testing: matrix_get_by_value
[1 (0,1)]
[2 (0,2)]
[3 (0,3)]

Testing: matrix_get_value

Testing if the value of the MatrixNode(1,2) == 3
Got: 3. Expect: 3.
Testing if the value of the MatrixNode(1,2) == 4
Got: 4. Expect: 4.
Testing if the value of the MatrixNode(1,2) == 4
Got: 4. Expect: 4.

ranu@ranu-laptop ~$ dev : ~/Github/data-structures/src/uni/encadeamento
[0] %
```

Figura 6: Programa executando no modo iterativo

Caso não seja provido argumento o programa irá inicializar no modo de testes e caso seja inicializado com uma flag desconhecida ele mostra uma lista de flags para uso do programa.

```
+ x .../uni/encadeamento .../CS/3st period/5GBD ranu@ranu-laptop: ~

ranu@ranu-laptop ~$ dev : ~/Github/data-structures/src/uni/encadeamento
[0] % ./matrix.out --testing rafael

Usage: ./matrix --<flag>

Flags available:
--help
--interactive
--test <x> <y>

Where:
x - Rows of the matrix
y - Columns of the matrix

ranu@ranu-laptop ~$ dev : ~/Github/data-structures/src/uni/encadeamento
[0] %
```

Figura 7: Programa mostrando como ser usado