RELATÓRIO TÉCNICO

Matriz Encadeada

João Pedro Martins da Silva

Prof. Dr. Thiago França Naves

Ciência da Computação

Estrutura de Dados

CAMPUS Santa Helena, 2021

RELATÓRIO TÉCNICO

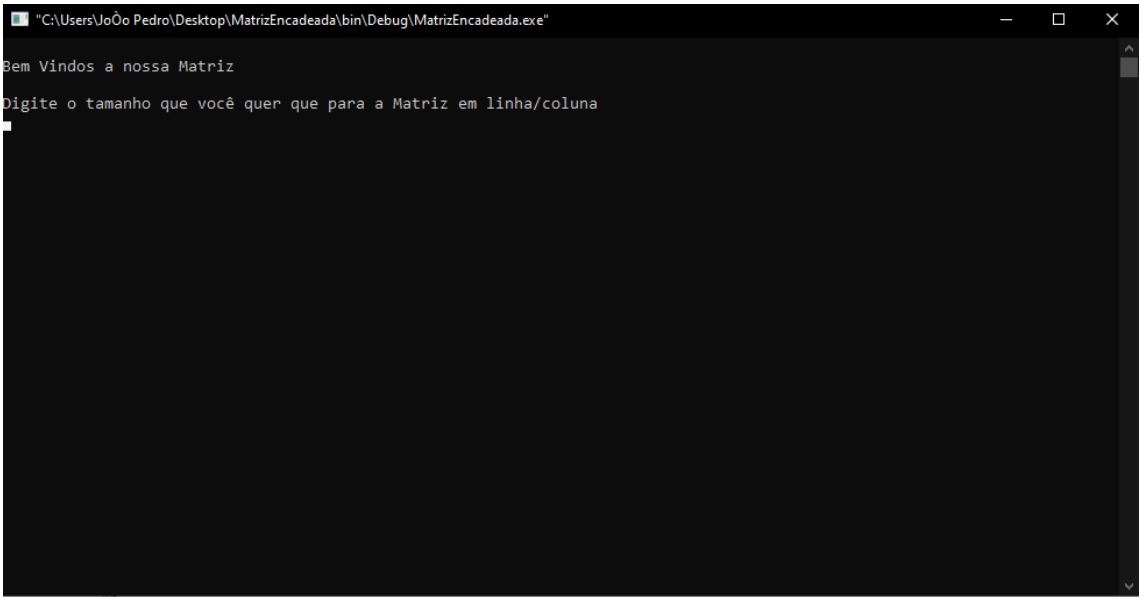
Descrição e contextualização da ferramenta

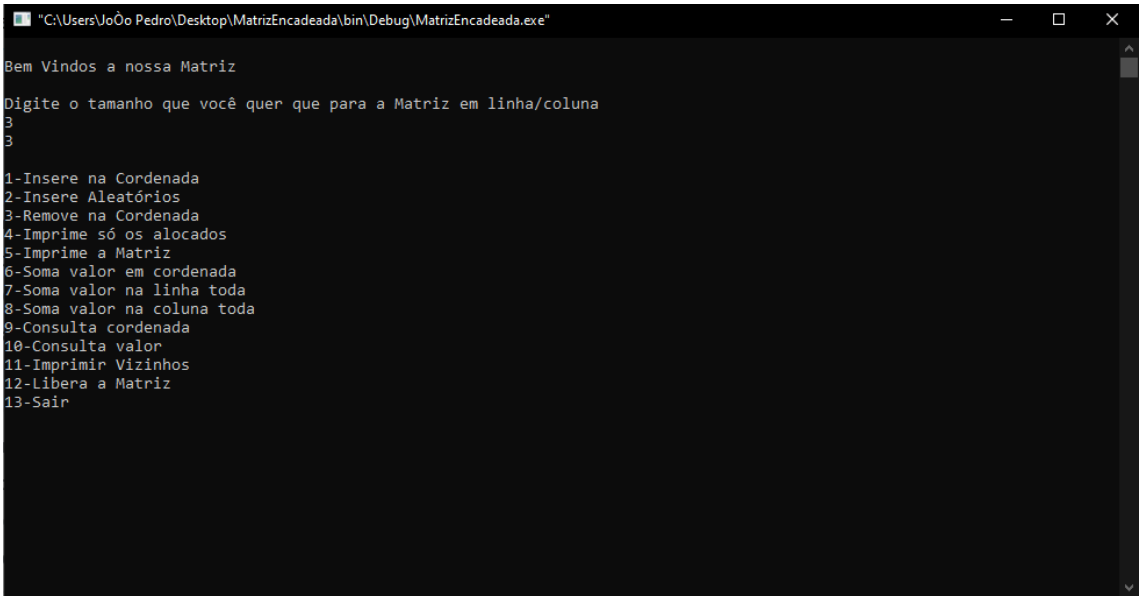
Este projeto se trata de uma matriz esparsa. O usuário ao interagir com o sistema, pode inserir ou remover um número em uma coordenada ou números aleatórios, pode imprimir somente os nós que existem na matriz ou a matriz inteira com os campos que são iguais a zero, somar valores em uma coordenada específica e em uma linha ou coluna inteira, consultar por coordenada ou por valor um número ou uma coordenada respectivamente, consultar quais são os vizinhos de uma determinada coordenada e liberar a matriz.

Arquitetura do Sistema

Seguem abaixo algumas das funcionalidades do sistema:

Tela Inicial do Sistema:

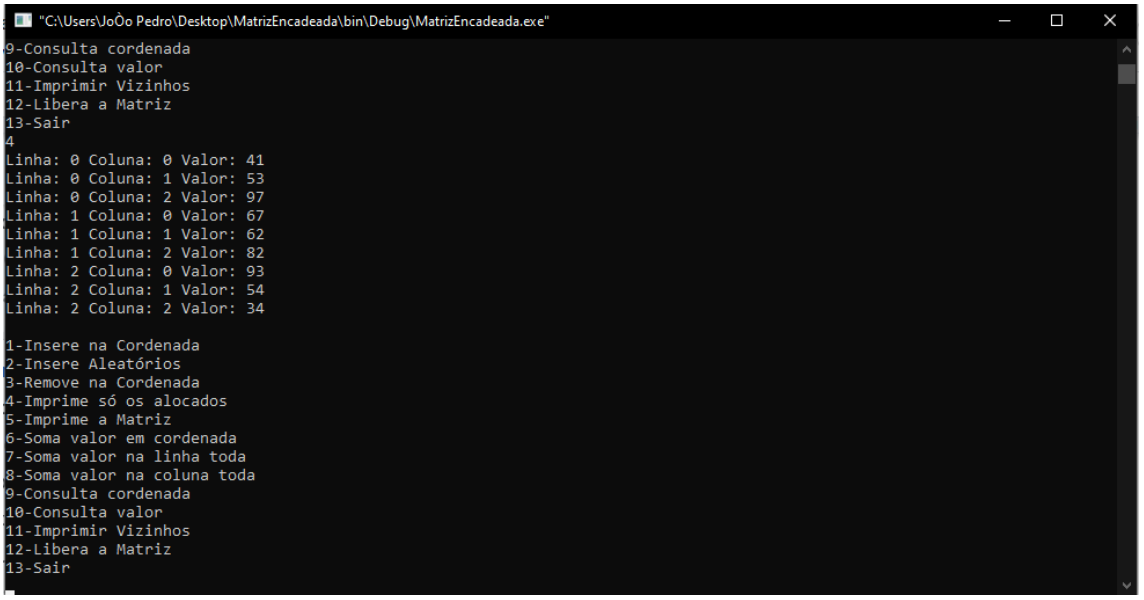




Inserindo números coordenadamente o aleatoriamente:

Removendo uma coordenada:

Imprimindo os valores com nós e todos os valores:



Somando valores:

Em linha:

Em coluna:

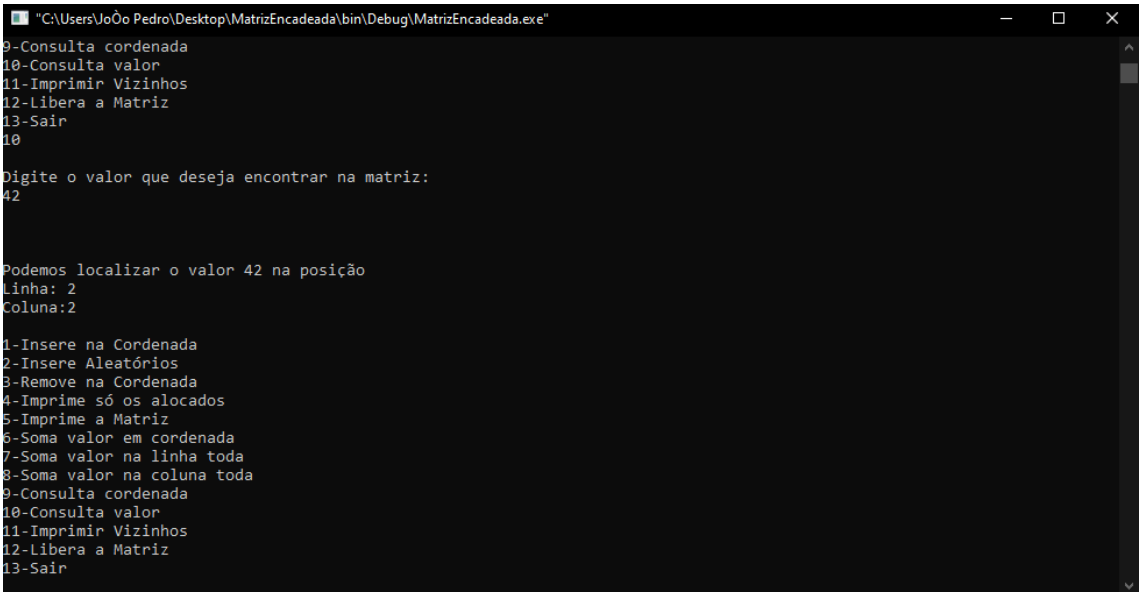
```
"C:\Users\JoÒo Pedro\Desktop\MatrizEncadeada\bin\Debug\MatrizEncadeada.exe"
8

Sua matriz tem dimensão :
Linha: 3
Coluna: 3

Digite a coluna e o valor que deseja somar considerando coluna inicial 0:
2
8

Valor inserido com sucesso!

Valor inserido com sucesso!

Valor inserido com sucesso!

1-Insere na Cordenada
2-Insere Aleatórios
3-Remove na Cordenada
4-Imprime só os alocados
5-Imprime a Matriz
6-Soma valor em cordenada
7-Soma valor na linha toda
8-Soma valor na coluna toda
9-Consulta cordenada
10-Consulta valor
11-Imprimir Vizinhos
12-Libera a Matriz
13-Sair
```

Consulta de uma coordenada:

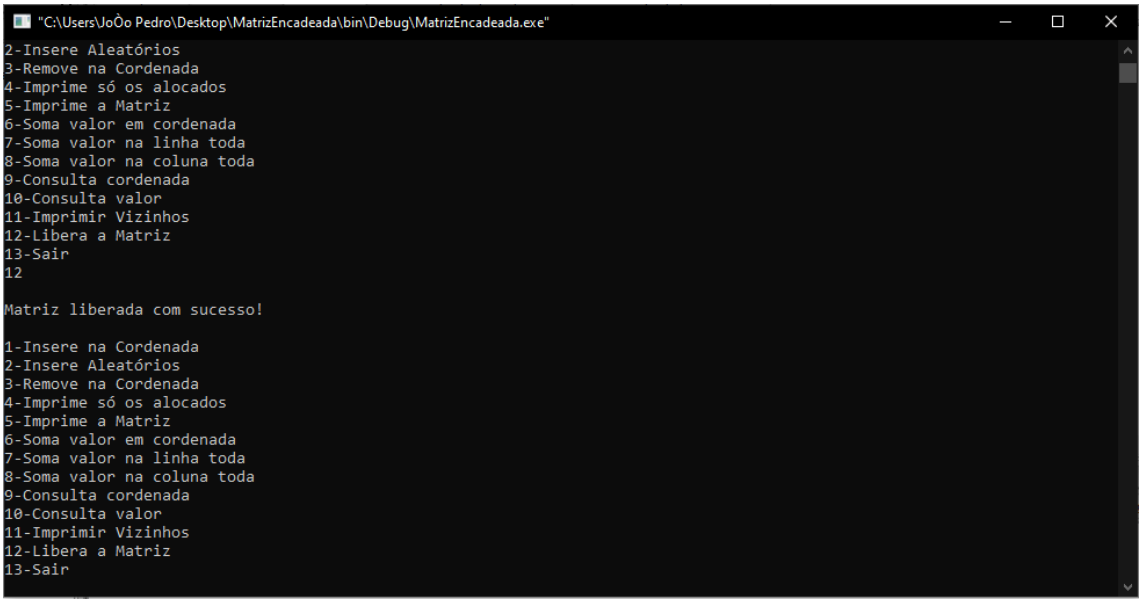
Matriz:

```
41      53      105
67      62      90
93      54      42
```

```
"C:\Users\JoÒo Pedro\Desktop\MatrizEncadeada\bin\Debug\MatrizEncadeada.exe"
10-Consulta valor
11-Imprimir Vizinhos
12-Libera a Matriz
13-Sair
9

Sua matriz tem dimensão :
Linha: 3
Coluna: 3

Digite a linha e coluna e o valor que deseja saber considerando linha inicial 0 e coluna inicial 0:
2
2

 O valor buscado na matriz é 42

1-Insere na Cordenada
2-Insere Aleatórios
3-Remove na Cordenada
4-Imprime só os alocados
5-Imprime a Matriz
6-Soma valor em cordenada
7-Soma valor na linha toda
8-Soma valor na coluna toda
9-Consulta cordenada
10-Consulta valor
11-Imprimir Vizinhos
12-Libera a Matriz
13-Sair
```



Liberar Matriz:



Objetivos geral e específicos

Metodologia utilizada:

Foram criados os códigos e os headers. O header tem a função de servir de declaração de nossa matriz.c. Nele é chamado o comando typedef, para definir o nome das estruturas e listados o cabeçalho das funções.

No código *matriz.c* as biblioteca stdio.h e stdlib.h são incluídas, além do header matriz.h e é criado duas estruturas, elemento e matriz. Na *struct* elemento, que foi nomeada como Elem é colocado cinco campos, dois do tipo inteiro que representará a linha e a coluna que o Nó participará, uma do tipo inteiro que representará o valor que será guardado neste nó e dois ponteiros do tipo estrutura Elem que apontaram para a próxima coluna e próxima linha. Na struct matriz, nomeada como Matriz, é atribuído quatro campos, dois do tipo Elem que serão ponteiros para ponteiros para a matriz onde poderão percorrer duas dimensões e dois do tipo inteiro representando a quantidade de linhas e colunas.

Na primeira função(criar_matriz) é passado a quantidade de linhas e colunas como parâmetro. Logo em seguida é criado um ponteiro de matriz que aloca espaço de memoria dinamicamente. A quantidade de linhas e colunas deste ponteiro que foi criado recebe os valores que foram passados como parâmetro. Os ponteiros para ponteiro deste ponteiro, são alocados alocados dinamicamente e possuirão inicialmente em números de ponteiros que serão criados, o número de linhas e colunas passados como parâmetro, para ponteiro lin e ponteiro col, respectivamente. Dois laços de repetição são criados pra percorrer a posição i de cada ponteiro e setar como NULL e em seguida o ponteiro m é retornado, pois esta função e uma função Matriz.

Na função insere_cordenada é passado como parâmetro um ponteiro para matriz, as linhas e colunas e o valor a ser inserido. É verificado se as linhas estão certas e criado um NO ponteiro p alocado dinamicamente do tipo Elem se estiverem. Este NO recebe os valores passados como parâmetro e seus ponteiros para próxima linha e próxima coluna apontam para NULL.

É declarado um NO ponteiro aux , que apontará para o ponteiro da matriz em linha e um No ponteiro ant que inicialmente aponta para NULL mas servirá para verificar o nó anterior. Um laço de repetição while é criado para percorrer esse ponteiro e extrair as informações e o ponteiro p é alocado de acordo com as informações no início, no meio ou no fim das linhas. Depois de alocar o ponteiro p em linha os ponteiros *aux e *ant recebem agora o ponteiro da matrix em coluna e NULL, respectivamente para fazer o mesmo processo que o da linha, porém agora em coluna.

Para imprimir valores, foi utilizado duas funções, uma para imprimir só os valores dos nós que existem e uma para imprimir a função completa , assumindo que os nós que são NULL valem zero. Na primeira, um for atribui todos os ponteiros da matriz em linha em um NO *aux e dentro deste for, um while percorre este *aux em suas colunas, printando na tela os valores das coordenadas em linha e coluna. Na função para imprimir a matriz toda, um for atribui todos os ponteiros da matriz em linha em um NO *aux e quando esse *aux é nulo é digitado o zero e um espaço entre aproxima coordenada. Se ele não for nulo a coluna dele de valor tem que ser igual a coluna em que o for está

percorrendo para ele pois, caso contrário é preciso colocar o valor de zero para representar as colunas e linhas que são nulas.

Para liberar a matriz, é passada como parâmetro em uma função e se cria dois elementos *aux, um para receber o ponteiro e liberá-lo e o outro para apontar para o ponteiro e ir andando pela matriz. No fim , se libera os dois ponteiros de ponteiro de linha e coluna da matriz e a matriz.

Para remover uma coordenada se usa uma lógica parecida com a lógica de inserção mas, neste caso não é preciso criar um novo nó de alocação dinâmica, somente ponteiros para servirem de referência e serem liberados, são eles o aux e o ant que percorreram a matriz em linha e coluna e liberam a coordenada pedida pelo usuário.

Na função soma_em_cordenada é passado a matriz, a linha e a coluna e o valor a ser somado. Cria-se um NO *aux apontando para o ponteiro da matriz na linha passada como parâmetro e percorre no while até achar o elemento que a linha e a coluna são os mesmos passados na função. Utiliza-se das funções já criadas de inserir e remover para manipular de acordo com o valor do *aux.

Em somalinha é passado como parâmetro a matriz, a linha e o valor a ser somado, um laço de repetição for é criado e percorre essa linha passada utilizando a função soma_em_cordenada repassando os parâmetros e adicionando os parâmetros para a coluna. Em somacoluna utiliza-se da mesma lógica mas, ao invés de linha se utiliza a coluna como parâmetro.

Para consultar o valor de uma posição, cria-se dois ponteiros, um para percorrer as colunas e um para percorrer as linhas, chamados de auxL e auxC. O laço While percorre as colunas e linhas até que os aux's sejam nulos ou menores que a coluna e que a linha passadas como parâmetro. Se algum dos dois forem nulos é sinal de que o nó desta posição não existe, então será zero. Caso a linha e coluna destes nós sejam iguais as passadas como parâmetro, assume-se que este nó na posição existe e é imprimido o valor desta coordenada.

Na função para buscar um valor qualquer e retornar sua coordenada, é passado este valor e a matriz como parâmetro, se o valor for zero é atribuído como NULL. Em um laço for o *aux recebe o ponteiro das linhas da matriz e percorre sua coluna quando é diferente de NULL, retornando as coordenadas em que existe o número passado como parâmetro.

A última função, para atribuir quatro vizinhos(cima,baixo,esquerda,direita), declara-se três ponteiros, *aux, *ant e *prox, o *aux vai receber o ponteiro de linhas da matriz para saber os valores à esquerda e à direita e vai percorrê-lo em um while até que a coluna seja igual ou maior do que a passada como parâmetro ou que ele seja null, ou seja, que a linha seja vazia. Ao percorrer o while ele atribui valor ao ant de acordo com a sua posição e o *ant só será diferente de NULL quando sua coluna for uma posição atrás da coluna passada como parâmetro. Para o valor de *prox, verifica se a coluna do próximo do *aux é uma posição à frente da coluna passada como parâmetro, se sim *prox recebe o valor da próxima posição de *aux. Validando esses valores em estruturas de condição os valores são impressos na tela com seus valores. O mesmo processo é utilizado para saber os valores acima e abaixo porém é atribuído ao *aux o ponteiro de matriz de

coluna e depois de percorrido e validado é impresso os valores de cima e baixo na tela.

No código *main* foram utilizadas as bibliotecas stdio.h, stdlib.h e locale.h, além do header matriz.h. É declarado o cabeçalho da função menu que faz uma interface de todos os métodos descritos acima. Esta função retorna um inteiro que será utilizado em um switch/case que implementa cada método de acordo com a escolha do usuário em um loop do/while que se encerra caso a escolha deste seja de sair, escolhendo o número “13”.

Ferramentas utilizadas para o desenvolvimento:

O Software utilizado para a criação do código foi o ***Code::Blocks 17.12.****Code::Blocks é um ambiente de desenvolvimento integrado de código aberto e multiplataforma. Ele foi desenvolvido em C++, usando wxWidgets. Sua arquitetura é orientada a plugin, de forma que suas funcionalidades são definidas pelos plugins fornecidos a ele.*
O link para o download é: https://www.codeblocks.org/downloads/binaries/

Técnicas utilizadas para o desenvolvimento:

As técnicas de Estrutura de Dados que foram utilizadas são Ponteiros e Listas Encadeadas além de outras implementações como Bibliotecas e Headers (stdio.h, stdlib.h, locale.h,matriz.h); Estruturas (Struct); Funções; Laços de Repetição (do/while, for, switch); Entrada e saída de dados (printf, scanf); Estrutura de Condição (if/else);

Resultados Alcançados

O sistema pode alocar números em uma matriz sem utilizar tanto espaço de memória, pois quando um nó tem seu valor nulo, não será necessário aloca-ló como em uma matriz convencional.

Informações Adicionais

Código do programa:

Header:

```c
typedef struct matriz Matriz;
typedef struct elemento Elem;


Matriz *cria_matriz(int nlinhas, int ncolunas);

void insere_cordenada(Matriz *m, int linha, int coluna, int val);

void imprime_alocados(Matriz *m);

void imprime_completo(Matriz *m);

void libera_matriz(Matriz *m);

void remove_cordenada(Matriz *m,int linha, int coluna);

void soma_em_cordenada(Matriz *m, int linha, int coluna , int val);

void somalinha(Matriz *m, int linha, int val);

void somacoluna(Matriz *m, int coluna, int val);

void consultaposi(Matriz *m, int linha, int coluna);

void buscar_valor(Matriz *m, int valor);

void imprimir_vizinhos(Matriz *m, int linha, int coluna);
```

Sources:

matriz.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include "matriz.h"


struct elemento{
 int lin,col, info;
 Elem *pcol, *plin;
};

struct matriz{
   Elem **lin, **col;
   int nlin,ncol;
};

Matriz *cria_matriz(int nlinhas, int ncolunas){
    Matriz *m = (Matriz*) malloc(sizeof(Matriz));
    m->nlin= nlinhas;
    m->ncol=ncolunas;

    m->lin = (Elem **) malloc(sizeof(Elem*)*nlinhas);
    m->col = (Elem **) malloc(sizeof(Elem*)*ncolunas);

    for(int i=0;i<nlinhas;i++){
        m->lin[i]=NULL;
    }
    for(int i=0;i<ncolunas;i++){
        m->col[i]=NULL;
    }
    return m;
}
//assume-se que o no da linha e coluna nao existe...
void insere_cordenada(Matriz *m, int linha, int coluna, int info){
    if(linha>m->nlin-1 || linha<0 || coluna>m->ncol-1 || coluna<0){
        printf("\nEstá linha ou coluna não existem\n");
```

```
        return;
    }
    Elem *p=(Elem*) malloc(sizeof(Elem));
    p->lin= linha;
    p->col= coluna;
    p->info= info;
    p->pcol=NULL;
    p->plin=NULL;

    Elem *aux = m->lin[linha];
    Elem *ant= NULL;
    while(aux!=NULL && aux->col < coluna){
        ant= aux;
        aux=aux->pcol;
    }
    if(ant==NULL){//insere no inicio
        if(aux!=NULL){ // lista ja tem pelo menos 1 elem
            p->pcol=aux;
            m->lin[linha] = p;
        }else{//lista vazia
            m->lin[linha] =p;
        }
    }else{//inserção no meio ou no fim da lista
        ant->pcol = p;
        p->pcol = aux;
    }
    aux = m->col[coluna];
    ant= NULL;
    while(aux!=NULL && aux->lin < linha){
        ant= aux;
        aux=aux->plin;
    }
    if(ant==NULL){//insere no inicio
        if(aux!=NULL){ // lista ja tem pelo menos 1 elem
            p->plin=aux;
            m->col[coluna] = p;
        }else{//lista vazia
            m->col[coluna] =p;
        }
    }else{//inserção no meio ou no fim da lista
        ant->plin = p;
        p->plin = aux;
    }
    printf("\nValor inserido com sucesso!\n");
}
//imprime os elementos diferentes de zero
void imprime_alocados(Matriz *m){
    for(int i=0;i<m->nlin;i++){
        Elem *aux = m->lin[i];
        while(aux!=NULL){
            printf("Linha: %d Coluna: %d Valor: %d\n", aux->lin, aux->col, aux->info);
            aux=aux->pcol;
        }
    }
}

void imprime_completo(Matriz *m){
    for(int i=0; i < m->nlin;i++){
        Elem *aux = m->lin[i];
        for(int j=0; j<m->ncol;j++){
            if(aux == NULL){
                printf("0");
            }else{
                if(aux->col == j){
                    printf("%d", aux->info);
                    aux=aux->pcol;
                }else{
                    printf("0");
                }
            }
            if(j<m->ncol-1)printf("\t");
        }
        printf("\n");
    }
}

void libera_matriz(Matriz *m){
    Elem *aux2;
    for(int i=0; i<m->nlin; i++){
        Elem *aux= m->lin[i];
        while(aux!=NULL){
            aux2=aux;
            aux= aux->pcol;
            free(aux2);
        }
    }
    free(m->lin);
    free(m->col);
    free(m);
    printf("\nMatriz liberada com sucesso!\n");
}
void remove_cordenada(Matriz *m,int linha, int coluna){
    if(linha>m->nlin-1 || linha<0 || coluna>m->ncol-1 || coluna<0){
        printf("\nEstá linha ou coluna não existem\n");
        return;
    }
    Elem *aux= m->lin[linha];
    Elem *ant= NULL;
    while(aux!=NULL && aux->col != coluna){
        ant=aux;
        aux=aux->pcol;
    }
    if(aux==NULL){
            printf("\nValor já não existe\n");
            return;
    }
    if(ant==NULL){
        m->lin[linha]=aux->pcol;
```

```
145         }else{
146             ant->pcol = aux->pcol;
147         }
148         aux= m->col[coluna];
149         ant= NULL;
150         while(aux!=NULL && aux->lin != linha){
151             ant=aux;
152             aux=aux->plin;
153         }
154         if(aux==NULL){
155                 printf("\nValor já não existe\n");
156                 return;
157         }
158         if(ant==NULL){
159             m->col[coluna]=aux->plin;
160         }else{
161             ant->plin = aux->plin;
162         }
163         free(aux);
164         printf("\nValor removido com sucesso!\n");
165     }
166 void soma_em_cordenada(Matriz *m, int linha, int coluna, int info){
167         if(linha>m->nlin-1 || linha<0 || coluna>m->ncol-1 || coluna<0){
168         printf("\nEstá linha ou coluna não existem\n");
169         return;
170         }
171         Elem *aux= m->lin[linha];
172         if(aux==NULL){
173             insere_cordenada(m,linha, coluna, info);
174             printf("\nValor inserido com sucesso!\n");
175             return ;
176         }
177         while(aux!=NULL){
178             if(coluna == aux->col){
179                 aux->info += info;
180                 if(aux->info==0){
```

```
181                     remove_cordenada(m,linha,coluna);
182                 }
183                 printf("\nValor inserido com sucesso!\n");
184                 return;
185             }
186             if(coluna < aux->col){
187                 insere_cordenada(m,linha,coluna,info);
188                 printf("\nValor inserido com sucesso!\n");
189                 return ;
190             }
191             aux=aux->pcol;
192         }
193         insere_cordenada(m,linha,coluna,info);
194     }
195
196 void somalinha(Matriz *m, int linha, int info){
197         if(linha>m->nlin-1 || linha<0){
198             printf("\nEsta linha não existe\n");
199         }
200         for(int i=0 ; i<m->ncol;i++){
201             soma_em_cordenada(m,linha,i,info);
202         }
203     }
204
205 void somacoluna(Matriz *m, int coluna, int info){
206         if(coluna>m->ncol-1 || coluna<0){
207             printf("\nEsta coluna não existe\n");
208         }
209         for(int i=0; i<m->nlin; i++){
210             soma_em_cordenada(m,i,coluna,info);
211         }
212     }
213 void consultaposi(Matriz *m, int linha, int coluna){
214         if(linha>m->nlin-1 || linha<0 || coluna>m->ncol-1 || coluna<0){
215         printf("\nEstá linha ou coluna não existem\n");
216         return;
```

```
217     }
218         Elem *auxl= m->lin[linha];
219         Elem *auxc= m->col[coluna];
220         while(auxl!=NULL && auxl->col!=coluna){
221             auxl=auxl->pcol;
222         }
223         while(auxc!=NULL && auxc->lin!=linha){
224             auxc=auxc->plin;
225         }
226             if(auxl==NULL || auxc == NULL){
227             printf("\n O valor buscado na matriz é 0\n");
228             return;
229             }
230             if(auxl->col == coluna && auxc->lin==linha)
231                 printf("\n O valor buscado na matriz é %d\n", auxc->info);
232     }
233 void buscar_valor(Matriz *m, int valor){
234         if(valor==0){
235             printf("\nO valor é NULL\n");
236             return;
237         }
238         for(int i=0; i< m->nlin;i++){
239         Elem *aux = m->lin[i];
240         for(int j=0; j<m->ncol;j++){
241             if(aux == NULL){
242             }else{
243                 if(aux->col == j){
244                     if(aux->info==valor)
245                         printf("\nPodemos localizar o valor %d na posição\nLinha: %d\nColuna:%d", aux->info, aux->lin,aux->col);
246                     aux=aux->pcol;
247                 }
248             }
249         }
250         printf("\n");
251     }
252 }
```

```c
void imprimir_vizinhos(Matriz *m, int linha, int coluna){
        if(linha>m->nlin-1 || linha<0 || coluna>m->ncol-1 || coluna<0){
        printf("\nEstá linha ou coluna não existem\n");
        return;
    }
        Elem *aux = m->lin[linha];
        Elem *ant = NULL;
        Elem *prox = NULL;
        while(aux!=NULL && aux->col<coluna){
            if(aux->col==coluna-1){
            ant=aux;
            }
            aux=aux->pcol;
        }

        if(aux==NULL)
            prox=NULL;
        if(aux!=NULL && aux->pcol!=NULL && aux->pcol->col==coluna+1){
                prox=aux->pcol;
        }
        if(aux!=NULL && aux->col==coluna+1){
            prox=aux;
        }
         if(prox==NULL)
            printf("\nO valor a direira é : NULL \n");
         else
            printf("\nO valor a direita é : %d\n",prox->info);
         if(ant==NULL)
            printf("\nO valor a esquerda é : NULL\n");
         else
            printf("\nO valor a esquerda é : %d\n", ant->info);

        aux = m->col[coluna];
        ant = NULL;
        prox = NULL;
        while(aux!=NULL && aux->lin<linha){
            if(aux->lin==linha-1){
            ant=aux;
            }
            aux=aux->plin;
        }

        if(aux==NULL)
            prox=NULL;
        if(aux!=NULL && aux->plin!=NULL && aux->plin->lin==linha+1){
                prox=aux->plin;
        }
        if(aux!=NULL && aux->lin==linha+1){
            prox=aux;
        }
         if(prox==NULL)
            printf("\nO valor abaixo é : NULL \n");
         else
            printf("\nO valor abaixo é : %d\n",prox->info);
         if(ant==NULL)
            printf("\nO valor acima é : NULL\n");
         else
            printf("\nO valor acima é : %d\n", ant->info);
}

```

main.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include "matriz.h"
#include <locale.h>
int menu();
int main(){
        setlocale(LC_ALL, "Portuguese");
        int linha,coluna,valor;
        int l,c;
        printf("\nBem Vindos a nossa Matriz\n");
        printf("\nDigite o tamanho que você quer que para a Matriz em linha/coluna\n");
        scanf("%d%d",&l,&c);
        Matriz *m=cria_matriz(l,c);
        int op;
        do{
            op=menu();
        switch(op){
        case 1:
            printf("\nSua matriz tem dimensão : \nLinha: %d\nColuna: %d\n",l,c);
            printf("\nDigite a linha e coluna e o valor que deseja inserir considerando linha inicial 0 e coluna inicial 0: \n");
            scanf("%d%d%d",&linha,&coluna,&valor);
            insere_cordenada(m,linha,coluna,valor);
            break;
        case 2:
            for(int i=0;i<l;i++){
                for(int j=0;j<c;j++){
                    insere_cordenada(m,i,j, rand() % 99);
                }
            }
            break;
        case 3:
            printf("\nSua matriz tem dimensão : \nLinha: %d\nColuna: %d\n",l,c);
            printf("\nDigite a linha e coluna e o valor que deseja remover considerando linha inicial 0 e coluna inicial 0: \n");
            scanf("%d%d",&linha,&coluna);
            remove_cordenada(m,linha,coluna);
            break;
```

```
        case 4:
            imprime_alocados(m);
            break;
        case 5:
            imprime_completo(m);
            break;
        case 6:
            printf("\nSua matriz tem dimensão : \nLinha: %d\nColuna: %d\n",l,c);
            printf("\nDigite a linha e coluna e o valor que deseja somar considerando linha inicial 0 e coluna inicial 0: \n");
            scanf("%d%d%d",&linha,&coluna,&valor);
            soma_em_cordenada(m,linha,coluna,valor);
            break;
        case 7:
            printf("\nSua matriz tem dimensão : \nLinha: %d\nColuna: %d\n",l,c);
            printf("\nDigite a linha e o valor que deseja somar considerando linha inicial 0: \n");
            scanf("%d%d",&linha,&valor);
            somalinha(m,linha,valor);
            break;
        case 8:
            printf("\nSua matriz tem dimensão : \nLinha: %d\nColuna: %d\n",l,c);
            printf("\nDigite a coluna e o valor que deseja somar considerando coluna inicial 0: \n");
            scanf("%d%d",&coluna,&valor);
            somacoluna(m,coluna,valor);
            break;
        case 9:
            printf("\nSua matriz tem dimensão : \nLinha: %d\nColuna: %d\n",l,c);
            printf("\nDigite a linha e coluna e o valor que deseja saber considerando linha inicial 0 e coluna inicial 0: \n");
            scanf("%d%d",&linha,&coluna);
            consultaposi(m,linha,coluna);
            break;
        case 10:
            printf("\nDigite o valor que deseja encontrar na matriz: \n");
            scanf("%d",&valor);
            buscar_valor(m,valor);
            break;
        case 11:
            printf("\nSua matriz tem dimensão : \nLinha: %d\nColuna: %d\n",l,c);
            printf("\nDigite a linha e coluna que deseja saber os 4 vizinhos considerando linha inicial 0 e coluna inicial 0: \n");
            scanf("%d%d",&linha,&coluna);
            imprimir_vizinhos(m,linha,coluna);
            break;
        case 12:
            libera_matriz(m);
            break;
    }
    }while(op!=13);
    return 0;
}
int menu(){
    printf("\n1-Insere na Cordenada\n");
    printf("2-Insere Aleatorios\n");
    printf("3-Remove na Cordenada\n");
    printf("4-Imprime só os alocados\n");
    printf("5-Imprime a Matriz\n");
    printf("6-Soma valor em cordenada\n");
    printf("7-Soma valor na linha toda\n");
    printf("8-Soma valor na coluna toda\n");
    printf("9-Consulta cordenada\n");
    printf("10-Consulta valor\n");
    printf("11-Imprimir Vizinhos\n");
    printf("12-Libera a Matriz\n");
    printf("13-Sair\n");
    int op;
    scanf("%d",&op);
    return op;
}
```

Conclusão

Ao desenvolver o projeto tive algumas dificuldades na parte de filas mas, as aulas disponibilizadas pelo professor no youtube e conteúdos indicados ajudaram a perseverar no objetivo e a praticar cada vez mais e a prática é uma forma muito eficaz de sanar dificuldades de maneira rápida e fácil.