Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Santa Helena
Bacharelado em Ciência da Computação

Disciplina:	Algoritmos e Estrutura De Dados III		
Aluno:	Igor Carvalho Marchi	**R.A.:**	a2677229
Aluno:	Jefferson korte junior	**R.A.:**	a2651114
Aluno:	Marciano Kuhn	**R.A.:**	a2677245

ATIVIDADE

Foi criado três arquivos(matriz.c, matriz.h, main.c)

Main.c

No main foi utilizado o switch case para poder fazer uma estrutura organizada onde melhora a interação do usuário com o sistema, que por meio do switch o usuário vai digitar uma opção que vai levar a um determinado case, eles dependem da escolha do usuário, que por meio do void menu, vai aparecer as opções que ele pode escolher, dentro dos cases também possui condições para prevenir erros, além da função que o mesmo deseja.

Matriz.h

No matriz.h é onde é chamada às funções e declarada a struct matriz, ele serve para que as funções não entrem em conflito independente da ordem delas.

Matriz.c

No matriz.c é o local onde é estruturado os structs e as funções a serem criadas, nele devemos chamar a matriz.h com um #include “matriz.h” para que as funções não entrem em conflito.

FUNÇÕES

1-Cria matriz (int linhas, int colunas)

Para criar uma matriz, começamos alocando memória para nossa estrutura Mat e verificamos se a alocação foi bem-sucedida. Caso contrário, retornamos NULL para indicar a falha. Em seguida, definimos as dimensões da matriz (linha e coluna) e iniciamos o ponteiro início como NULL. Depois, alocamos um vetor de ponteiros que armazenará os elementos da matriz. Se essa alocação falhar, liberamos a memória já alocada e

retornamos NULL. Após a alocação do vetor, seguimos para a etapa de criação dos elementos da matriz. Percorremos todas as posições e alocamos individualmente cada elemento, inicializando seu valor como 0 e seus ponteiros (prox, ant, cima, baixo) como NULL. Caso ocorra erro na alocação de algum elemento, liberamos todas as alocações anteriores e interrompemos a criação da matriz. Por fim, fazemos a ligação entre os elementos, garantindo que cada um tenha referências corretas para seus vizinhos. Se um elemento estiver na borda da matriz, ele permanecerá com ponteiros NULL indicando ausência de vizinhos naquela direção. Com todas as alocações e conexões finalizadas, definimos o início da matriz (inicio = no[0]) e armazenamos o vetor de elementos na estrutura. Por fim, retornamos o ponteiro para a matriz criada.

2- Inserir elemento(Mat *mat, int linha, int coluna, int valor)

A função inserir_elemento tem como objetivo armazenar um valor em uma posição específica dentro da matriz. Primeiro, ela verifica se a matriz foi criada, pois tentar acessar um espaço não alocado resultaria em erro. Se a matriz for inválida, a função retorna 0, indicando falha na operação. Em seguida, verifica se os índices da linha e da coluna fornecidos estão dentro dos limites da matriz. Caso os índices sejam menores que 0 ou excedam o tamanho máximo permitido, a função retorna 0. Se todas as verificações forem bem-sucedidas, a função calcula o índice correspondente no vetor de elementos da matriz e armazena o novo valor na posição desejada. Ao final, retorna 1 para indicar que o valor foi inserido com sucesso.

3- Consultar Posição (Mat *mat, int linha, int coluna)

A função consultar_posicao permite verificar o valor armazenado em uma posição específica da matriz. Primeiramente, verifica se a matriz foi criada e se seus elementos foram devidamente alocados. Caso contrário, a função retorna imediatamente sem realizar a consulta. Em seguida, checa se os índices fornecidos (linha e coluna) estão dentro dos limites da matriz. Se a posição estiver fora dos limites, uma mensagem informando isso é exibida, e a função retorna. Após passar pelas verificações iniciais, calcula-se o índice correspondente no vetor de elementos da matriz. Antes de acessar o valor, confirma-se se o elemento existe. Se a posição for válida, mas o valor do elemento for igual a 0, significa que nenhum valor foi inserido naquela posição, e a função exibe uma mensagem informando isso. Caso contrário, o valor armazenado na posição é impresso.

4- Busca um valor qual quer (Mat *mat, int valor)

A função buscar_valor tem como objetivo encontrar todas as ocorrências de um valor específico dentro da matriz. Primeiro, verifica se a matriz foi criada e se seus elementos estão devidamente alocados. Caso contrário, a função retorna imediatamente.

Em seguida, inicia uma varredura completa pela matriz, percorrendo todos os elementos. Para cada posição, compara o valor armazenado com o valor buscado. Se houver uma correspondência, exibe a posição (linha, coluna) onde o valor foi encontrado e incrementa um contador para acompanhar quantas vezes ele aparece na matriz. Após percorrer toda a matriz, verifica-se o contador. Se nenhuma ocorrência do valor foi encontrada, exibe uma mensagem informando que o valor não está presente na matriz.

5- Imprimir Valor dos 4 vizinhos de (Mat *mat, int linha, int coluna)

Na função de imprimir os vizinhos tem condições de erro, sendo caso a matriz não foi criada, caso não exista elemento, caso a posição dada seja maior que a matriz, caso ele passe pelas condições ele vai fazer uma busca na matriz para encontrar a posição do elemento que o usuário insere(linha e coluna), verificando se os quatros vizinhos(prox, ant, cima, baixo) são diferente de NULL e se caso o valor igual a zero então o valor não foi inserido, caso contrário ele vai imprimir o valor que tem no vizinho.

6- Liberar matriz(Mat *mat)

Na função de liberar matriz tem a condição de que é necessário a matriz estar criada, caso contrário ela retorna, quando ela continua ela passa por um for para percorrer todos elementos e liberando um por um, depois do for ele vai liberar o vetor de ponteiros e por último vai liberar a matriz(struct).

7- Imprimir Matriz(Mat *mat)

Na função de imprimir matriz tem a condição de que é necessário a matriz estar criada e se tiver elementos, caso contrário ela retorna, quando ela continua ela percorre por um for para passar por todos os elementos, onde vai passar por duas condições, caso o valor do elemento seja igual a 0 então significa que o valor não foi inserido printando a letra “X”, contudo caso contrário ele vai imprimir o valor do elemento.

8- Remover dado de uma posição específica(Mat *mat, int linha, int coluna)

Na função de remover dados de uma posição específica tem condições de erro, sendo caso a matriz não foi criada, caso não exista elemento, caso a posição dada seja maior que a matriz, caso nenhum desses erros aconteça então ele pegar a posição dada pelo usuário(linha e coluna), verificando se o elemento existe assim indo pra duas funções, onde caso o valor já seja zero então ele vai avisar que já não existe um valor, caso contrário ele vai remover o dados antigo substituindo o antigo valor por zero.

9- Limpar dados da matriz(Mat *mat)

Na função de limpar dados da matriz ele vai ter condições de erros, onde deve existir a matriz e ter elementos dentro dela e caso exista ambos, vai percorrer o for por toda

matriz onde caso o valor do elemento seja diferente de zero(Vai ter valor dentro dele) então vai pegar esse valor e igualar a zero deixando com “nenhum valor” repetindo essa condição até o último elemento.

10- void menu()

Dentro do void menu foi organizado a tela do menu que vai aparecer para o usuário determinar qual opção ele deseja escolher, nele mostra em formato de texto(printf) qual função o usuário deseja utilizar.

Dificuldades

As maiores dificuldades foi no momento de criar as structs, pois foi um momento de pensar quais elementos dentro dela seriam mais úteis para facilitar na criação das funções, sendo a mais extensiva a de criar a matriz foi um pouco complicada, pois além de criar os elementos era necessário fazer as 4 ligações entre eles, além disso outra que estava tendo um problema era em imprimir os vizinhos, mas para consertar esse problema foi resolvido utilizando condições de ifs para saber se o usuário não havia inserido um valor(quando estava igual a 0) e caso fosse diferente de “0” e “NULL” ele mostrava o valor, se não se encaixa-se em nenhum deles então o valor é NULL.