UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Projeto Matriz Dinâmica

Docente: Dr. Thiago França Naves

Discentes: Diego Lucas Hattori Dallaqua

Mileno Oliveira Matos

Disciplina: Estrutura de Dados

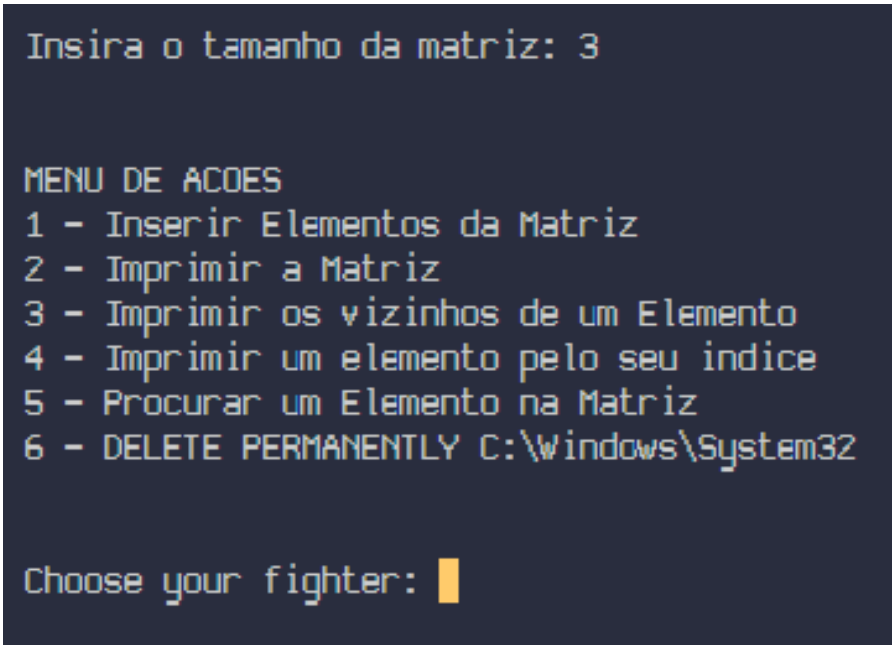
Turma: CC23C

Santa Helena - Paraná

2023

O projeto

O código do projeto funciona tendo como base um Menu de Ações, onde o após o usuário inserir o tamanho da matriz desejada, ele deve escolher entre as opções disponíveis, que são feitas através de um comando de switch.



Caso o usuário escolha a opção 1 do MENU, o comando de switch executará o seguinte bloco de código:

```
case 1:

    for ( contLinha = 0; contLinha < tamanho; contLinha++ ) {

        for ( contColuna = 0; contColuna < tamanho; contColuna++ ) {

            valor.coluna = contColuna;
            valor.linha = contLinha;

            printf ( "Insira o elemento que vai estar na posicao (%i)(%i): ", contLinha, contColuna );
            scanf ( "%d", &num );

            valor.conteudo = num;
            insereMatriz ( mat, valor );

        }

    }

break;
```

Onde a função ***insereMatriz*** irá ser chamada até que a matriz esteja complemente cheia, segue abaixo o código da função ***insereMatriz***:

```
Matriz *insereMatriz ( Matriz *mat, struct dados valor ) {

    // Condicional de segurança pro caso da matriz não existir
   if ( mat == NULL ) {

       return 0;

    }

    // Criação e alocação do auxiliar dinâmicamente, em seguida, outra condicional de segurança é executada para garantir que o auxiliar foi alocado
    Elem *aux;
    aux = ( Elem* ) malloc ( sizeof ( Elem ) );

    if ( aux == NULL ) {

       return 0;

    }

    // O número digitado pelo usuário é armazenado aonde o auxiliar aponta e o ponteiro PROX de auxiliar aponta para NULL, pois ainda não existe um próximo elemento
    aux -> numero = valor;
    aux -> prox = NULL;

    // Caso o elemento inserido for o primeiro elemento da matriz, o ponteiro ANT do auxiliar aponta para NULL e o conteúdo da cabeça da matriz, passa a ser o endereço de aux
    if ( ( *mat ) == NULL ) {

       aux -> ant = NULL;
       *mat = aux;
```

```
    // Caso não for o primeiro elemento, cria um segundo auxiliar e faz com que o auxiliar aponte para a cabeça da matriz
    } else {

        Elem *auxZ;

        auxZ = *mat;

        // Enquanto o próximo elemento de Aux2 for diferente de NULL, aux2 aponta para o próximo
        while ( auxZ -> prox != NULL ) {

            auxZ = auxZ -> prox;

        }

        // O ponteiro PROX de aux2 aponta para aux, e o ponteiro ANT de aux aponta para aux2, logo, o nó aux vem depois do nó aux2
        auxZ -> prox = aux;
        aux -> ant = auxZ;

    }

    return 1;

}
```

Sendo assim, teremos a seguinte resposta no terminal:

```
MENU DE ACOES
1 - Inserir Elementos da Matriz
2 - Imprimir a Matriz
3 - Imprimir os vizinhos de um Elemento
4 - Imprimir um elemento pelo seu indice
5 - Procurar um Elemento na Matriz
6 - DELETE PERMANENTLY C:\Windows\System32


Choose your fighter: 1
Insira o elemento que vai estar na posicao (0)(0): 1
Insira o elemento que vai estar na posicao (0)(1): 2
Insira o elemento que vai estar na posicao (0)(2): 3
Insira o elemento que vai estar na posicao (1)(0): 4
Insira o elemento que vai estar na posicao (1)(1): 5
Insira o elemento que vai estar na posicao (1)(2): 6
Insira o elemento que vai estar na posicao (2)(0): 7
Insira o elemento que vai estar na posicao (2)(1): 8
Insira o elemento que vai estar na posicao (2)(2): 9
```

Caso o usuário escolha a opção 2 do MENU, o comando de switch executará o seguinte bloco de código:

```
case 2:

    imprimeMatriz ( mat, tamanho );

break;
```

Onde a função ***imprimeMatriz*** irá ser chamada até que a matriz seja totalmente exibida, segue abaixo o código da função ***imprimeMatriz***:

```
void imprimeMatriz ( Matriz *mat, int tamanho ) {

    int cont = 0, contAux = 1, linha, coluna;

    // Condicional de segurança pro caso da matriz não existir
    if ( mat == NULL ) {

        return 0;

    }

    // Cria o auxiliar e faz com que ele aponte para onde a cabeça aponta
    Elem *aux = *mat;

    printf ( "\n" );
```

```c
    // Enquanto o auxiliar não apontar para NULL, imprime o número armazenado aonde o auxiliar aponta
    while ( aux != NULL ) {

        printf( "\t" );
        printf( "|%d|", aux -> numero.conteudo );

        // O auxiliar aponta para o próximo elemento
        aux = aux -> prox;
        cont++;

        // Faz com que ocorra uma quebra de linha quando necessário
        if ( cont == tamanho || cont == tamanho * contAux ){

            printf ( "\n\n" );
            contAux++;

        }

    }
}
```

Sendo assim, teremos a seguinte resposta no terminal:

```
MENU DE ACOES
1 - Inserir Elementos da Matriz
2 - Imprimir a Matriz
3 - Imprimir os vizinhos de um Elemento
4 - Imprimir um elemento pelo seu indice
5 - Procurar um Elemento na Matriz
6 - DELETE PERMANENTLY C:\Windows\System32

Choose your fighter: 2

        |1|     |2|     |3|

        |4|     |5|     |6|

        |7|     |8|     |9|
```

Caso o usuário escolha a opção 3 do MENU, o comando de switch executará o seguinte bloco de código:

```c
case 3:

    imprimeMatriz ( mat, tamanho );

    printf ( "Digite a linha: " );
    scanf ( "%d", &linha );
    printf ( "Digite a coluna: " );
    scanf ( "%d", &coluna );

    imprimeVizinho ( mat, linha, coluna, tamanho );

break;
```

Onde a função ***imprimeVizinho*** irá ser chamada até que a matriz seja totalmente exibida, segue abaixo o código da função ***imprimeVizinho***:

```c
int imprimeVizinho ( Matriz *mat, int linha, int coluna, int tamanho ) {

    if ( linha < 0 || linha > tamanho || coluna < 0 || coluna > tamanho || linha * coluna > ( tamanho * tamanho ) ) {

        printf("\nValores Invalidos\n\n\n");
        return 0;

    }

    int posTop = tamanho, posBot = tamanho;

    Elem *aux = *mat;
    Elem *aux2, *top, *bot;

    while ( aux != NULL ) {

        if ( aux -> numero.linha == linha && aux -> numero.coluna == coluna ) {

            printf ( "\n\nPosicao (%d)x(%d) = %d\n", linha, coluna, aux -> numero.conteudo );

            aux2 = aux -> ant;

            if ( aux2 != NULL && aux2 -> numero.linha == aux -> numero.linha ) {

                printf ( "Left = %d\n", aux2 -> numero );

            } else {

                printf ( "Left = NULL\n" );

            }
```

```
        top = aux;

        while ( top -> ant != NULL && posTop > 0 ) {

            top = top -> ant;
            posTop--;

        }

        if ( top == aux ) {

            top = NULL;

        }

        if ( top != NULL && top -> numero.coluna == aux -> numero.coluna ) {

            printf( "Top = %d\n", top -> numero );

        } else {

            printf( "Top = NULL\n" );

        }
```

```
        aux2 = aux -> prox;

        if ( aux2 != NULL && aux2 -> numero.linha == aux -> numero.linha ) {

            printf ( "Right = %d\n", aux2 -> numero );

        } else {

            printf ( "Right = NULL\n" );

        }

        bot = aux;

        while ( bot -> prox != NULL && posBot > 0 ) {

            bot = bot -> prox;
            posBot--;

        }

        if ( bot == aux ) {

            bot = NULL;

        }
```

```
            if ( bot != NULL && bot -> numero.coluna == aux -> numero.coluna ) {

                printf ( "Bot = %d\n", bot -> numero );

            } else {

                printf ( "Bot = NULL\n" );

            }

            return 1;
        }

        aux = aux -> prox;

    }

}
```

Sendo assim, teremos a seguinte resposta no terminal:

```
Choose your fighter: 3

        |12|    |2|     |3|

        |4|     |5|     |6|

        |7|     |8|     |9|

Digite a linha: 1
Digite a coluna: 1


Posicao (1)x(1) = 5
Left = 4
Top = 2
Right = 6
Bot = 8
```

Caso o usuário escolha a opção 4 do MENU, o comando de switch executará o seguinte bloco de código:

```c
case 4:

    printf ( "\Digite a linha: " );
    scanf ( " %d", &linha);

    printf ( "Digite a coluna: " );
    scanf ( " %d", &coluna );

    imprimeMatriz ( mat, tamanho );
    imprimeElemento ( mat, linha, coluna, tamanho );

break;
```

Onde a função ***imprimeElemento*** irá ser chamada até que a matriz seja totalmente exibida, segue abaixo o código da função ***imprimeElemento***:

```c
int imprimeElemento ( Matriz *mat, int linha, int coluna, int tamanho ) {

    // Condicional de Segurança
    if ( linha < 0 || linha > tamanho || coluna < 0 || coluna > tamanho ) {

        printf( "\nValores Invalidos\n\n\n" );

        return 0;

    }

    // Cria o ponteiro auxiliar para percorrer a matriz
    Elem *aux = *mat;

    while ( aux != NULL ) {

        // Quando a linha e a coluna do nó para onde o uxiliar aponta forem iguais as digitadas pelo usuário, imprime o conteúdo do nó
        if ( aux -> numero.linha == linha && aux -> numero.coluna == coluna ) {

            printf ( "\nElemento (%d)x(%d) = %d\n", linha, coluna, aux -> numero.conteudo );

            return 1;

        }

        // Caso contrário, aponta para o próximo nó
        aux = aux -> prox;

    }

    return 0;

}
```

Sendo assim, teremos a seguinte resposta no terminal:

```
MENU DE ACOES
1 - Inserir Elementos da Matriz
2 - Imprimir a Matriz
3 - Imprimir os vizinhos de um Elemento
4 - Imprimir um elemento pelo seu indice
5 - Procurar um Elemento na Matriz
6 - DELETE PERMANENTLY C:\Windows\System32


Choose your fighter: 4
Digite a linha: 2
Digite a coluna: 0

       |12|    |2|     |3|

       |4|     |5|     |6|

       |7|     |8|     |9|


Elemento (2)x(0) = 7
```

Caso o usuário escolha a opção 5 do MENU, o comando de switch executará o seguinte bloco de código:

```c
case 5:

    printf ( "Digite um numero: " );
    scanf ( "%d", &num );

    imprimeMatriz ( mat, tamanho );

    procuraElemento ( mat, num, tamanho );

break;
```

Onde a função ***procuraElemento*** irá ser chamada até que a matriz seja totalmente exibida, segue abaixo o código da função ***procuraElemento***:

```c
int procuraElemento ( Matriz *mat, int num, int tamanho ) {

    // Cria o ponteiro auxiliar e os contadores
    Elem *aux = *mat;


    // Percorre toda a matriz
    while ( aux != NULL ) {

        // Compara o conteúdo do nó e o número digitado
        if ( aux -> numero.conteudo == num ) {

            printf ( "\nLinha: %d\n",aux -> numero.linha );
            printf ( "Coluna: %d\n",aux -> numero.coluna );

            return 1;

        }

        // Caso não for igual, aponta para o próximo nó
        aux = aux -> prox;

    }

    printf ( "Número Inválido, Tente Novamente" );

    return 0;

}
```

Sendo assim, teremos a seguinte resposta no terminal:

```
MENU DE ACOES
1 - Inserir Elementos da Matriz
2 - Imprimir a Matriz
3 - Imprimir os vizinhos de um Elemento
4 - Imprimir um elemento pelo seu indice
5 - Procurar um Elemento na Matriz
6 - DELETE PERMANENTLY C:\Windows\System32


Choose your fighter: 5
```

```
Digite um numero: 3

        |12|     |2|      |3|

        |4|      |5|      |6|

        |7|      |8|      |9|


Linha: 0
Coluna: 2
```

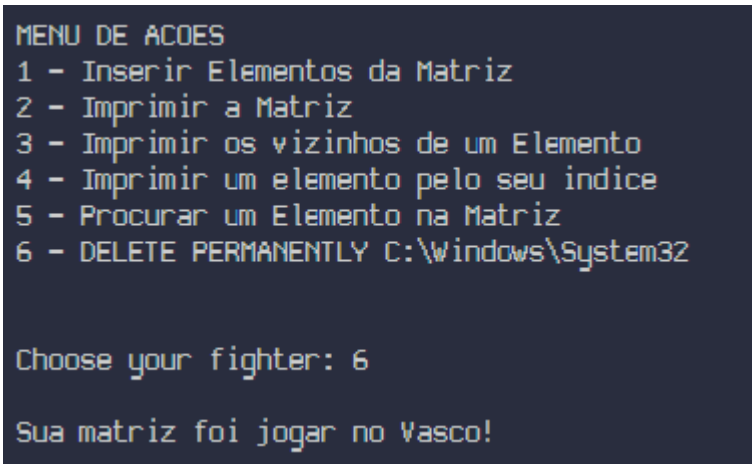
Caso o usuário escolha a opção 6 do MENU, o comando de switch executará o seguinte bloco de código:

```
case 6:

    apagaMatriz ( mat );

break;
```

Onde a função ***apagaMatriz*** irá ser chamada até que a matriz seja totalmente exibida, segue abaixo o código da função ***apagaMatriz***:

```
void apagaMatriz ( Matriz* mat ) {

    // Condicional de segurança
    if ( mat == NULL ) {

        return;

    }

    // Cria os auxiliares que vão percorrer a matriz
    Elem *aux = *mat, *aux2;

    while ( aux != NULL ) {

        // O segundo auxiliar aponta para o próximo elemento após o primeiro auxiliar
        aux2 = aux -> prox;

        // Libera o primeiro auxiliar
        free ( aux );

        // O primeiro auxiliar aponta para o mesmo lugar que o segundo aponta
        aux = aux2;

    }

    // Após isso tudo, libera a matriz
    free ( mat );

    printf ( "\nSua matriz foi jogar no Vasco!\n\n" );

}
```

Sendo assim, teremos a seguinte resposta no terminal e após isso, o programa será encerrado:



Inicialmente, o maior desafio do projeto, de forma geral, residia na concretização da estrutura da matriz, o que, embora esboçado, carecia de clareza na implementação prática. Devido a contratempos pessoais, experimentamos significativos atrasos na conclusão do projeto, o que nos levou a investir uma quantidade considerável de tempo para alcançar seu término. Enfrentamos diversos obstáculos relacionados à visualização de elementos específicos da matriz, demandando extensa pesquisa. No entanto, eventualmente, conseguimos superar esses desafios.

Desde o início, percebemos que a disciplina se revelava altamente complexa, e o grau de dificuldade associado à execução da atividade proposta comprovou essa percepção. Acreditamos que a qualidade da lógica do código poderia ter sido aprimorada substancialmente com uma alocação de tempo mais generosa. Além disso, devido às outras disciplinas que também necessitam da nossa atenção, a conclusão do projeto acabou se dando na data do prazo de entrega estabelecido.