

Engenharia Informática

Estruturas de Dados

2019/2020

Relatório

Pesquisa e Ordenação

António Pereira Nº2019141051

David Ferreira Nº2019141281

Data: 17/03/2020

Índice

1	INTRODUÇÃO.....	3
2	TRABALHO REALIZADO	4
2.1	ESCREVA UMA FUNÇÃO COM UM VETOR DE INTEIROS E UM INTEIRO COMO ARGUMENTOS QUE LEIA OS ELEMENTOS DE UM VETOR INSERIDOS PELO UTILIZADOR.....	4
2.2	DESENVOLVA UMA FUNÇÃO QUE GERE ALEATORIAMENTE NÚMEROS ENTRE 1 E 50, COLOCANDO ESTES VALORES NUM VETOR DE DIMENSÃO 30.....	5
2.3	ESCREVA UMA FUNÇÃO QUE DESLOQUE TODOS OS ELEMENTOS DO VETOR UMA POSIÇÃO PARA A ESQUERDA: O PRIMEIRO ELEMENTO DEVE PASSAR PARA ÚLTIMO, O ÚLTIMO PARA PENÚLTIMO, ..., O SEGUNDO PARA O PRIMEIRO.	7
2.4	DESENVOLVA SUBPROGRAMAS QUE DETERMINEM A MÉDIA DOS NÚMEROS ÍMPARES E A MÉDIA DOS NÚMEROS PARES INDICADOS POR UM UTILIZADOR.....	9
2.5	COPIE PARA AS SECÇÕES CORRETAS DO PROGRAMA, O CÓDIGO ANTERIOR E COMPLETE O PROGRAMA, ESCRIVENDO O SUBPROGRAMA EM FALTA E A SUA CHAMADA NO PROGRAMA PRINCIPAL	10
2.6	CONSTRUA UMA CALCULADORA QUE PERMITA REALIZAR VÁRIAS OPERAÇÕES ENTRE MATRIZES DE ELEMENTOS REAIS.	12
2.6.1	<i>Ler os elementos para uma matriz $A_{m \times n}$;</i>	13
2.6.2	<i>Escrever uma matriz $A_{m \times n}$;</i>	14
2.6.3	<i>Calcular a média de todos elementos de uma matriz $A_{m \times n}$;</i>	15
2.6.4	<i>Calcular a média de todos elementos de uma dada coluna k de uma matriz $A_{m \times n}$;</i>	16
2.6.5	<i>Calcular a média de todos elementos de uma dada linha l de uma matriz $A_{m \times n}$;</i>	17
2.6.6	<i>Contar o número de zeros que se encontram acima da diagonal principal de uma matriz $A_{m \times n}$;</i> 17	
2.6.7	<i>Determinar a linha de uma matriz $A_{m \times n}$ que tem a soma dos seus elementos máxima;</i>	18
2.6.8	<i>Trocar as colunas j e k de uma matriz $A_{m \times n}$;</i>	19
2.6.9	<i>Somar duas matrizes $A_{m \times n}$ e $B_{m \times n}$;</i>	20
2.7	PRETENDE-SE PROCURAR UM VALOR NUM VETOR DE INTEIROS INSERIDO POR UM UTILIZADOR.	21
2.8	ESCREVA UMA FUNÇÃO QUE, RECEBENDO COMO PARÂMETROS UMA TABELA UNIDIMENSIONAL, A SUA DIMENSÃO E UM VALOR, UTILIZE UM ALGORITMO DE PESQUISA BINÁRIA PARA LOCALIZAR O VALOR NA TABELA E DEVOLVER O SEU ÍNDICE. CASO O VALOR NÃO ESTEJA PRESENTE NA TABELA, DEVE SER DEVOLVIDO – I. 25	
2.9	PRETENDE-SE ORDENAR UM VETOR POR ORDEM CRESCENTE DOS SEUS ELEMENTOS.....	26
2.10	PRETENDE-SE ORDENAR UM VETOR POR ORDEM CRESCENTE DOS SEUS ELEMENTOS.....	30
2.11	PRETENDE ORDENAR-SE, POR ORDEM DECRESCENTE, UM VETOR DE INTEIROS USANDO O ALGORITMO <i>BUBBLE SORT</i> (OU ALGORITMO POR BORBULHAMENTO).....	33
2.12	DESENVOLVA UM SUBPROGRAMA QUE REMOVA VALORES REPETIDOS DE UM VETOR.....	35
3	CONCLUSÃO	36
4	REFERÊNCIAS	37

1 Introdução

O objetivo deste primeiro trabalho é fazer programas com vetores, por exemplo ler os elementos dos vetores, mover os elementos dos vetores. Também se pretende com este trabalho usar vetores bidimensionais para conseguir ler, escrever matrizes, somar duas matrizes entre outras coisas. Por último, ordenar os elementos dos vetores através do método de seleção e inserção linear e através do algoritmo bubble sort.

2 Trabalho realizado

2.1 Escreva uma função com um vetor de inteiros e um inteiro como argumentos que leia os elementos de um vetor inseridos pelo utilizador.

Neste primeiro exercício é pedido que criemos uma função para ler um vetor de inteiros inserido pelo utilizador. Para isso criamos duas funções uma função para escrever os elementos do vetor(`escrever_elementos`) e outra função para ler esses elementos(`ler_string`).

```
#include <stdio.h>

int escrever_elementos(int escrever[], int dim_string)
{
    for (int i = 0; i < dim_string; i++)
    {
        printf("Introduza o elemento: ");
        scanf("%d", &escrever[i]);
    }
}

int ler_string(int ler[], int dim_string)
{
    for (int j = 0; j < dim_string; j++)
    {
        printf("elemento %d: %d \n", j, ler[j]);
    }
}
```

Figura 1- Funções para escrever e ler os elementos do vetor

Exemplo de output do primeiro exercício:

```
Introduza quantos elementos quer inserir: 4

Introduza o elemento: 23
Introduza o elemento: 12
Introduza o elemento: 14
Introduza o elemento: 56

elemento 0: 23
elemento 1: 12
elemento 2: 14
elemento 3: 56
```

Figura 2- Interface das funções `escrever_elementos` e `ler_string`

2.2 Desenvolva uma função que gere aleatoriamente números entre 1 e 50, colocando estes valores num vetor de dimensão 30.

No exercício 2 é pedido para utilizar a função rand ou ramdom. Nos decidimos utilizar a função rand. Esta função permite que sejam gerados números aleatórios, sendo que no nosso caso só vão ser gerados 30 números aleatórios entre 1 e 50 pois o tamanho do vetor é de 30 elementos. Para utilizar esta função é preciso incluir a biblioteca "stdlib.h".

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 30
int main()
{
    int vec[MAX];
    for (int i = 0; i < MAX; i++)
    {
        printf("%d\n", rand() % 50 + 1);
        vec[i] = rand();
    }
}
```

Figura 3- Programa para gerar números aleatórios com a função rand

Aqui esta a parte do programa que vai gerar os números aleatórios entre 1 e 50:

```
printf("%d\n", rand() % 50 + 1);
```

Figura 4- Parte do programa que vai gerar os números aleatórios entre 1 e 50

Exemplo do output do exercício 2:

```
0% exe2_ED
42
35
20
29
13
6
32
12
46
28
42
3
43
22
19
48
22
20
18
36
4
23
24
42
4
48
13
38
24
30
```

2.3 Escreva uma função que desloque todos os elementos do vetor uma posição para a esquerda: o primeiro elemento deve passar para último, o último para penúltimo, ..., o segundo para o primeiro.

Neste exercício temos que criar um programa que desloque os elementos do vetor uma posição para esquerda, isto é, o primeiro elemento passa para o último, o último para o penúltimo etc. para fazer isto criamos duas funções:

A primeira função(`escrever_elementos`) serve para o utilizador conseguir escrever os elementos do vetor que tem como dimensão a que o utilizador inserir no início do programa:

```
#include <stdio.h>
int escrever_elementos(int numero[], int num_elementos)
{
    for (int m=0; m < num_elementos; m++)
    {
        printf("Introduza o numero: ");
        scanf("%d", &numero[m]);
    }
}
```

Figura 5- Função para escrever os elementos do vetor

A segunda função é onde os elementos vão ser movidos uma posição para a esquerda, para isso, criamos uma variável(`última_posição`) que vai ter o valor da posição zero do vetor. Depois o elemento da última posição vai ter o valor da variável `última_posição`. Dentro do for os elementos do vetor vão ser iguais aos elementos da posição a seguir, isto é, o primeiro elemento vai ser igual ao segundo elemento, o segundo ao terceiro etc.

```
int mover_elementos(int numero[], int num_elementos)
{
    int ultima_posicao;
    ultima_posicao = numero[0];
    for (int i= 0; i < num_elementos; i++)
    {
        numero[i]=numero[i+1];
    }
    numero[num_elementos-1]=ultima_posicao;
    for (int i= 0; i < num_elementos; i++)
    {
        printf("elemento %d: %d\n", i, numero[i]);
    }
}
```

Figura 6- Função para mover os elementos do vetor

```
int main()
{
    int num_elementos;
    printf("Introduza o tamanho do vetor: ");
    scanf("%d", &num_elementos);
    int numero[num_elementos];
    escrever_elementos(numero, num_elementos);
    mover_elementos(numero, num_elementos);
    return 0;
}
```

Figura 7- Função main onde são chamadas as funções

Exemplo de output do exercício 3:

```
exe3_ED
Introduza o tamanho do vetor: 4
Introduza o numero: 4
Introduza o numero: 12
Introduza o numero: 56
Introduza o numero: 2
elemento 0: 12
elemento 1: 56
elemento 2: 2
elemento 3: 4
```

Figura 8- Interface das funções escrever_elementos e mover_elementos

2.4 Desenvolva subprogramas que determinem a média dos números ímpares e a média dos números pares indicados por um utilizador.

O exercício 4 pede que criemos um programa para fazer a media dos números pares e ímpares. Para isso, primeiro temos que ler o tamanho do vetor e os elementos do vetor que o utilizador vai inserir. Apos para ver se os números são pares ou ímpares realiza-se um if e se for par o resto da divisão dos elementos com 2 tem de ser igual a zero e se for ímpar tem de ser diferente de zero. Depois disto é realizado a media.

```
#include <stdio.h>

int main()
{
    int num;
    printf("Introduza quantos numeros quer inserir: ");
    scanf("%d", &num );
    printf("\n");
    int string[num];
    float media_impares, media_pares;
    int soma =0, soma_pares = 0;
    int somou = 0, somou_pares = 0;
    for(int i = 0; i < num; i++)
    {
        printf("Introduza um numero: ");
        scanf("%d", &string[i]);
    }
}
```

Figura 10- Parte do programa para ler o tamanho e os elementos do vetor

```
for (int j = 0; j < num; j++)
{
    if(string[j] % 2 != 0)
    {
        soma = soma + string[j];
        somou++;
        media_impares= soma / somou;
    }
    if(string[j] % 2 == 0)
    {
        soma_pares = soma_pares + string[j];
        somou_pares++;
        media_pares= soma_pares / somou_pares;
    }
}
printf("media impares: %0.1f\n", media_impares );
```

Figura 9- Parte do programa onde vai ser verificada se os números são pares ou ímpares e onde é feita a media desses números

Este é um exemplo do output do exercício 4:

```
Introduza quantos numeros quer inserir: 5
Introduza um numero: 11
Introduza um numero: 15
Introduza um numero: 66
Introduza um numero: 12
Introduza um numero: 4
media impares: 13.0
media pares: 27.0
```

Figura 11- Output do exercício 4

- 2.5 Copie para as secções corretas do programa, o código anterior e complete o programa, escrevendo o subprograma em falta e a sua chamada no programa principal

Neste exercício pusemos as funções nas secções corretas e depois criamos uma função para ler os elementos das duas matrizes. Esta função é depois chamada no main com os parâmetros corretos para o seu funcionamento.

```
void matriz_1(int m1[5][7])
{
    int i,j;
    for(i=0;i<5;i++)
        for (j=0;j<7;j++)
            m1[i][j]=j;
}
void matriz_2(int m2[][7])
{
    int i,j;
    for(i=0;i<5;i++)
        for (j=0;j<7;j++)
            m2[i][j]=i;
}
```

Figura 13- Funções das matrizes

```
void escrever_matriz(int m[5][7])
{
    for(int i=0;i<5;i++){
        printf("\n");
        for (int j=0;j<7;j++)
            printf("%d ", m[i][j]);
    }
}
```

Figura 12- Função para escrever a matriz

```
int main()
{
    int mat1[5][7],mat2[5][7];
    matriz_1(mat1);
    escrever_matriz(mat1);
    printf("\n");
    matriz_2(mat2);
    escrever_matriz(mat2);
    return 0;
}
```

Figura 14- Função main onde são chamadas as funções

Output deste exercício:

```
0 1 2 3 4 5 6
0 1 2 3 4 5 6
0 1 2 3 4 5 6
0 1 2 3 4 5 6
0 1 2 3 4 5 6

0 0 0 0 0 0 0
1 1 1 1 1 1 1
2 2 2 2 2 2 2
3 3 3 3 3 3 3
4 4 4 4 4 4 4
```

Figura 15- Interface da função escrever_matriz

2.6 Construa uma calculadora que permita realizar várias operações entre matrizes de elementos reais.

No exercício 6 é pedido que criemos um programa que realiza várias operações com matrizes. Para isso fizemos um menu de opções pois trona-se mais fácil e simples para o utilizador usar e para nós.

```
do
{
    printf("Menu do Programa\n");
    printf("1. Ler Matriz\n");
    printf("2. Escrever Matriz\n");
    printf("3. Media dos elementos da matriz\n");
    printf("4. Media dos elementos da coluna da matriz\n");
    printf("5. Media dos elementos da linha da matriz\n");
    printf("6. Zeros acima da diagonal principal da matriz\n");
    printf("7. Determinar a linha que tem a soma dos seus elementos maxima\n");
    printf("8. Trocar Colunas\n");
    printf("9. Somar Matrizes\n");
    printf("0. Sair\n");
    printf("Entre com a opcao: ");
    scanf("%d", &opcao);
    fflush(stdin);
}
```

Figura 16- Menu do programa

Para o utilizador poder escolher a opção e ser redirecionado para a respetiva função, decidimos utilizar cases. Assim sempre que o utilizador inserir uma opção será redirecionado para a interface dessa opção, ou seja, vai ser realizado a função dessa opção.

```
switch(opcao)
{
    case 1: ler_elementos_matriz(linhas_matriz, colunas_matriz, matriz); break;
    case 2: escrever_matriz(linhas_matriz, colunas_matriz, matriz); break;
    case 3: calcular_media_matriz(linhas_matriz, colunas_matriz, matriz); break;
    case 4: calcular_media_colunas(linhas_matriz, colunas_matriz, matriz); break;
    case 5: calcular_media_linhas(linhas_matriz, colunas_matriz, matriz); break;
    case 6: zeros_diagonal_matriz(linhas_matriz, colunas_matriz, matriz); break;
    case 7: soma_maxima_linha_matriz(linhas_matriz, colunas_matriz, matriz); break;
    case 8: trocar_colunas_matriz(linhas_matriz, colunas_matriz, matriz); break;
    case 9: somar_matrizes(linhas_matriz, colunas_matriz, matriz); break;
    case 0: printf("opcao Sair selecionada\n"); break;
    default: printf("opcao invalida\n");
}
getchar();
}
while(opcao != 0);
```

Figura 17- Cases onde são chamadas as funções

2.6.1 Ler os elementos para uma matriz $A_{m \times n}$;

A primeira opção do nosso menu é de ler os elementos da matriz. Esta função(`ler_elementos_matriz`) serve para o utilizador conseguir inserir os elementos da matriz, que tem como dimensão a que o utilizador inserir. A função é a seguinte:

```
void ler_elementos_matriz(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    printf("Introduza os elementos da matriz: \n");
    for(int i = 0; i < linhas_matriz; i++)
        for(int j = 0; j < colunas_matriz; j++)
            scanf("%d", &matriz[i][j]);
}
```

Figura 18- Função para inserir os elementos da matriz

Este é um exemplo de output da função(`ler_elementos_matriz`):

```
Introduza o numero de linhas da matriz: 3
Introduza o numero de colunas da matriz: 3
Menu do Programa
1. Ler Matriz
2. Escrever Matriz
3. Media dos elementos da matriz
4. Media dos elementos da coluna da matriz
5. Media dos elementos da linha da matriz
6. Zeros acima da diagonal principal da matriz
7. Determinar a linha que tem a soma dos seus elementos maxima
8. Trocar Colunas
9. Somar Matrices
0. Sair
Entre com a opcao: 1
Introduza os elementos da matriz:
1
2
3
4
5
6
7
8
9
```

Figura 19- Interface da opção “Ler Matriz”

2.6.2 Escrever uma matriz $A_{m \times n}$;

A segunda opção do menu serve para ler e apresentar os elementos da matriz na forma de uma matriz com a dimensão que o utilizador inseriu, por exemplo, se o utilizador introduziu três linhas e três colunas a matriz será 3 por 3.

```
void escrever_matriz(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    for(int i=0;i<linhas_matriz;i++){
        for (int j=0;j<colunas_matriz;j++)
            printf("%2d ", matriz[i][j]);
        printf("\r\n");
    }
}
```

Figura 20- Função para apresentar a matriz

Output da opção “Escrever Matriz”:

```
Introduza os elementos da matriz:
12
32
4
3
0
8
0
2
14
Menu do Programa
1. Ler Matriz
2. Escrever Matriz
3. Media dos elementos da matriz
4. Media dos elementos da coluna da matriz
5. Media dos elementos da linha da matriz
6. Zeros acima da diagonal principal da matriz
7. Determinar a linha que tem a soma dos seus elementos maxima
8. Trocar Colunas
9. Somar Matrizes
0. Sair
Entre com a opcao: 2
12 32 4
3 0 8
0 2 14
```

Figura 21- Interface da função escrever_matriz

2.6.3 Calcular a média de todos elementos de uma matriz $A_{m \times n}$;

Na terceira opção, o programa vai apresentar a media dos elementos da matriz. A função é a seguinte:

```
void calcular_media_matriz(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    float soma = 0;
    float media;
    for(int i = 0; i < linhas_matriz; i++)
        for(int j = 0; j < colunas_matriz; j++)
        {
            soma = soma + matriz[i][j];
            media = soma / (linhas_matriz * colunas_matriz);
        }
    printf("Media da Matriz: %0.1f", media);
}
```

Figura 22- Função para calcular a media dos elementos da matriz

Com isto, o output é o seguinte:

```
Entre com a opcao: 2
12 32 4
3 0 8
0 2 14

Menu do Programa
1. Ler Matriz
2. Escrever Matriz
3. Media dos elementos da matriz
4. Media dos elementos da coluna da matriz
5. Media dos elementos da linha da matriz
6. Zeros acima da diagonal principal da matriz
7. Determinar a linha que tem a soma dos seus elementos maxima
8. Trocar Colunas
9. Somar Matrices
0. Sair
Entre com a opcao: 3
Media da Matriz: 8.3_
```

Figura 23- Interface da função calcular_media_matriz

2.6.4 Calcular a média de todos elementos de uma dada coluna k de uma matriz $A_{m \times n}$;

A quarta opção é semelhante a terceira opção só que em vez de fazer a media dos elementos da matriz é a media dos elementos de uma dada coluna inserida pelo utilizador. A função é a seguinte:

```
void calcular_media_colunas(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    float media;
    int coluna;
    float somou = 0, soma = 0;
    printf("Introduza a coluna para fazer a media: ");
    scanf("%d", &coluna);
    while (coluna < 0 || coluna >= colunas_matriz)
    {
        printf("Numero da coluna inserido invalido, introduza novamente: ");
        scanf("%d", &coluna);
    }
    for (int i = 0; i < linhas_matriz; i++)
        for(int j = coluna; j <= coluna ; j++)
        {
            soma = soma + matriz[i][coluna];
            somou ++;
            media = soma / somou;
        }
    printf("media coluna: %0.1f\n", media);
}
```

Figura 24- Função para fazer a media dos elementos de uma dada coluna

O output desta função é a seguinte:

```
Entre com a opcao: 4
Introduza a coluna para fazer a media: 0
media coluna: 5.0
```

Figura 25- Output da função calcular_media_colunas

2.6.5 Calcular a média de todos elementos de uma dada linha l de uma matriz $A_{m \times n}$;

Na quinta opção, é mesma que a quarta só que em vez das colunas é as linhas e a função é a seguinte:

```
void calcular_media_linhas(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    float media_linhas;
    int linha;
    float somou = 0, soma_linhas = 0;
    printf("Introduza a linha para fazer a media: ");
    scanf("%d", &linha);
    while (linha < 0 || linha >= linhas_matriz)
    {
        printf("Numero da linha inserida invalida, introduza novamente: ");
        scanf("%d", &linha);
    }
    for (int i = linha; i <= linhas_matriz; i++)
    {
        for (int j = 0; j < colunas_matriz; j++)
        {
            soma_linhas = soma_linhas + matriz[i][j];
            somou++;
            media_linhas = soma_linhas / somou;
        }
        printf("media coluna: %.1f\n", media_linhas);
    }
}
```

Figura 26- Função para calcular media dos elementos de uma dada linha

O output é o seguinte:

```
Entre com a opcao: 5
Introduza a linha para fazer a media: 2
media coluna: 5.3
```

Figura 27- Interface da função calcular_media_linhas

2.6.6 Contar o número de zeros que se encontram acima da diagonal principal de uma matriz $A_{m \times n}$;

A sexta opção serve para determinar quantos zeros é que tem acima da diagonal principal. A função que escrevemos é esta:

```
void zeros_diagonal_matriz(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    int existe = 0;
    if (linhas_matriz != colunas_matriz)
        printf("Nao e possivel contar o numero de zeros acima da diagonal principal\n");
    else
    {
        for (int i = 0; i < linhas_matriz; i++)
        {
            for (int j = 0; j < colunas_matriz; j++)
            {
                if (j > i)
                {
                    if (matriz[i][j] == 0)
                    {
                        existe++;
                    }
                }
            }
        }
        printf("numero de zeros acima da diagonal principal: %d\n", existe);
    }
}
```

Figura 28- Função para ver quantos zeros é que tem acima da diagonal principal

O output desta função é o seguinte:

```
Entre com a opcao: 2
12 32 4
3 0 8
0 2 14

Menu do Programa
1. Ler Matriz
2. Escrever Matriz
3. Media dos elementos da matriz
4. Media dos elementos da coluna da matriz
5. Media dos elementos da linha da matriz
6. Zeros acima da diagonal principal da matriz
7. Determinar a linha que tem a soma dos seus elementos maxima
8. Trocar Colunas
9. Somar Matrices
0. Sair
Entre com a opcao: 6
numero de zeros acima da diagonal principal: 0
```

Figura 29- Interface da função zeros_diagonal_matriz

2.6.7 Determinar a linha de uma matriz $A_{m \times n}$ que tem a soma dos seus elementos máxima;

A opção sete do programa faz a soma dos elementos de cada linha e depois vê qual é a linha que tem a soma dos elementos maior. A função é a seguinte:

```
void soma_maxima_linha_matriz(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    int soma_linha = 0, maior = 0, linha;
    int string[linhas_matriz];
    for(int i = 0; i < linhas_matriz; i++)
    {
        for(int j = 0; j < colunas_matriz; j++)
        {
            soma_linha = soma_linha + matriz[i][j];
            string[i] = soma_linha;
        }
        printf(" a soma da linha %d e : %d\n", i, string[i]);
        soma_linha = 0;
    }
    for(int i = 0; i < linhas_matriz; i++)
    {
        if (string[i] > maior)
        {
            maior = string[i];
            linha = i;
        }
    }
    printf(" a linha que tem a soma dos elementos maxima e: %d\n", linha);
}
```

Figura 30- Função para ver qual é a linha que a soma dos seus elementos maior

O output é o seguinte:

```
Entre com a opcao: 7
a soma da linha 0 e : 48
a soma da linha 1 e : 11
a soma da linha 2 e : 16
a linha que tem a soma dos elementos maxima e: 0
```

Figura 31- Interface da função soma_maxima_linha_matriz

2.6.8 Trocar as colunas j e k de uma matriz $A_{m \times n}$;

Na opção sete o nosso programa vai trocar os elementos de duas colunas inseridas pelo utilizador, ou seja, se o utilizador inserir a coluna 1 e a coluna 2 o programa vai trocar os elementos da coluna 1 pelo os elementos da coluna 2 e vice-versa. A nossa função é esta:

```
void trocar_colunas_matriz(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    int coluna1, coluna2, troca;
    printf("Introduza as colunas que quer trocar: \n");
    scanf("%d%d", &coluna1, &coluna2);
    while ((coluna1 >= colunas_matriz || coluna2 >= colunas_matriz) || (coluna1 < 0 || coluna2 < 0))
    {
        printf("nao foi possivel encontrar a coluna, por favor introduza novamente: \n");
        scanf("%d%d", &coluna1, &coluna2);
    }
    printf(" colunas da matriz trocada: \n");
    for(int i = 0; i < linhas_matriz; i++)
    {
        for(int j = 0; j < colunas_matriz; j++)
        {
            troca = matriz[i][coluna1];
            matriz[i][coluna1] = matriz[i][coluna2];
            matriz[i][coluna2] = troca;
        }
        for(int i=0;i<linhas_matriz;i++)
        {
            for (int j=0;j<colunas_matriz;j++)
            {
                printf("%2d ", matriz[i][j]);
                printf("\r\n");
            }
        }
        printf("\n");
    }
}
```

Figura 32- Função para trocar os elementos de duas colunas

Um exemplo de output desta opção é o seguinte:

```
12 32 4
3 0 8
0 2 14

Menu do Programa
1. Ler Matriz
2. Escrever Matriz
3. Media dos elementos da matriz
4. Media dos elementos da coluna da matriz
5. Media dos elementos da linha da matriz
6. Zeros acima da diagonal principal da matriz
7. Determinar a linha que tem a soma dos seus elementos maxima
8. Trocar Colunas
9. Somar Matrices
0. Sair
Entre com a opcao: 8
Introduza as colunas que quer trocar:
1
2
colunas da matriz trocada:
12 4 32
3 8 0
0 14 2
```

Figura 33- Interface da opção "Trocar Colunas"

2.6.9 Somar duas matrizes $A_{m \times n}$ e $B_{m \times n}$;

A última opção do nosso programa é para somar duas matrizes. Nesta opção o programa vai somar a matriz que o utilizador inseriu na primeira opção com a matriz que o utilizador inserir nesta opção que vai ter a mesma dimensão da outra matriz. A função é a seguinte:

```
void somar_matrizes(int linhas_matriz, int colunas_matriz, int matriz[][colunas_matriz])
{
    int matriz_2[linhas_matriz][colunas_matriz], soma_matrizes[linhas_matriz][colunas_matriz];
    printf("Introduza os elementos da segunda matriz: \n");
    ler_elementos_matriz(linhas_matriz, colunas_matriz, matriz_2);
    printf("Segunda Matriz:\n");
    escrever_matriz(linhas_matriz, colunas_matriz, matriz_2);
    for(int i = 0; i < linhas_matriz; i++)
    {
        for(int j = 0; j < colunas_matriz; j++)
            soma_matrizes[i][j] = matriz[i][j] + matriz_2[i][j];
    }
    printf("\n");
    printf("Soma das matrizes: \n");
    for(int i = 0; i < linhas_matriz; i++)
    {
        for(int j = 0; j < colunas_matriz; j++)
            printf("%2d ", soma_matrizes[i][j]);
        printf("\r\n");
    }
    printf("\n");
}
```

Figura 34- Função para somar duas matrizes

O output desta última função é o seguinte:

```
Entre com a opcao: 2
 1  3  6
10 22 14
15 19  5
```

Figura 35- Primeira Matriz

```
Entre com a opcao: 9
Introduza os elementos da matriz:
23
1
4
7
9
0
23
16
4
Segunda Matriz:
23  1  4
 7  9  0
23 16  4

Soma das matrizes:
24  4 10
17 31 14
```

Figura 36- Soma da primeira matriz com a segunda matriz

2.7 Pretende-se procurar um valor num vetor de inteiros inserido por um utilizador.

Neste exercício, foi-nos pretendido pela professora trabalharmos com um vetor e procurarmos nesse mesmo vetor, as ocorrências do mesmo e os índices da última ocorrência do valor e da primeira ocorrência. Pretende-se procurar um valor num vetor de inteiros inserido por um utilizador. O menu é o seguinte:

```
printf("\n\nMenu\n");
printf("1. Inserir o vetor\n");
printf("2. Numero de ocorrencias de valor\n");
printf("3. Procurar a ultima ocorrencia de x valor\n");
printf("4. Procurar primeira ocorrencia de x valor\n");
printf("0. Sair do menu\n");
printf("\nEscolha a sua opcao: ");
scanf("%d", &opcao);
```

Figura 37- Menu do exercício 7

Para a alínea a), simplesmente criámos um vetor que o utilizador terá que preencher pela ordem pretendida. Chamamento da função no menu, código feito e exemplo de output respetivamente

```
switch(opcao) {
    case 1:
        printf("Insira o numero de elementos do vetor: ");
        scanf("%d", &nmrelementos);
        printf("\nPreencha o vetor: ");
        InserirVetor();
        printf("\nO vetor inserido foi o seguinte:\n");
        MostrarVetor();
}
```

Figura 38- Case da primeira opção do menu

```
void InserirVetor()
{
    for(int i=0;i<nmrelementos;i++)
        scanf("%d", &vetor[i]);
}
```

Figura 39- Função para introduzir os elementos do vetor

```
Menu
1. Inserir o vetor
2. Numero de ocorrencias de valor
3. Procurar a ultima ocorrencia de x valor
4. Procurar primeira ocorrencia de x valor
0. Sair do menu

Escolha a sua opcao: 1
Insira o numero de elementos do vetor: 5

Preencha o vetor: 111
222
333
333
444

O vetor inserido foi o seguinte:
111 222 333 333 444
```

Figura 40- Output da primeira opção do menu

Para a alínea b), como nos pedia para vermos o número de ocorrências de qualquer valor naquele vetor, não foi muito difícil de produzir a função, que é a seguinte:

```
void NmrOcorrencias()
{
    int IncOcorrencia=0;
    printf("\nInsira o numero desejado: ");
    scanf("%d", &ocorrencia);
    for(int i=0;i<nmrelementos;i++)
    {
        if(vetor[i]==ocorrencia)
        {
            IncOcorrencia++;
        }
    }
    printf("\nO numero escolhido aparece %d vezes no vetor", IncOcorrencia);
}
```

Figura 41- Função para ver o número de ocorrências de qualquer elemento do vetor

Simplesmente incrementámos um valor todas as vezes que no vetor tivesse esse valor. Exemplo de output é o seguinte:

```
0 vetor inserido foi o seguinte:
111 222 333 333 444

Menu
1. Inserir o vetor
2. Numero de ocorrencias de valor
3. Procurar a ultima ocorrencia de x valor
4. Procurar primeira ocorrencia de x valor
0. Sair do menu

Escolha a sua opcao: 2

Insira o numero desejado: 333

0 numero escolhido aparece 2 vezes no vetor
```

Figura 42- Exemplo de output da função NmrOcorrencias

Para a alínea c), a professora pediu-nos o índice da última ocorrência de x valor naquele vetor. Para tal, simplesmente usámos outro incremento, que para quando se encontra o valor desejado. Para apresentar o índice (que vai de 0 até ao número de elementos -1) simplesmente chamávamos a variável que continha esse incremento. Caso o valor esteja presente no array, a função retornava o índice do valor, caso contrário, a função retorna -1. Para fazermos isto, tivemos que percorrer o vetor a partir do fim, ou seja, decrementar a partir do último elemento. A função é a seguinte:

```
int UltimaOcorrencia()
{
    int UltimaOcorr;
    for(UltimaOcorr=nmrelementos; UltimaOcorr>0; UltimaOcorr--)
    {
        if(vetor[UltimaOcorr]==ocorrenciaa)
        {
            return UltimaOcorr;
            break;
        }
    }
    return -1;
}
```

Figura 43- Função para ver o índice da última ocorrência de um número introduzido pelo o utilizador

Exemplo de output da função de cima é o seguinte:

```
Menu
1. Inserir o vetor
2. Numero de ocorrencias de valor
3. Procurar a ultima ocorrencia de x valor
4. Procurar primeira ocorrencia de x valor
0. Sair do menu

Escolha a sua opcao: 3

Insira o numero desejado: 333

Valor esta presente no vetor, indice 3
```

Figura 44- Output da função UltimaOcorrencia

Para a alínea d), é apresentar o índice da primeira ocorrência deste vetor. A função é muito semelhante à função anterior, no entanto, temos que começar a procurar no vetor a partir do início:

```
int PrimeiraOcorrencia()
{
    int PrimeiraOcorr;
    for(PrimeiraOcorr=0; PrimeiraOcorr<nmrelementos; PrimeiraOcorr++)
    {
        if(vetor[PrimeiraOcorr]==ocorrenciaaaa)
        {
            return PrimeiraOcorr;
            break;
        }
    }
    return -1;
}
```

Figura 45- Função para ver qual é o índice da primeira ocorrência de um número

Exemplo de output:

```
Menu
1. Inserir o vetor
2. Numero de ocorrencias de valor
3. Procurar a ultima ocorrencia de x valor
4. Procurar primeira ocorrencia de x valor
0. Sair do menu

Escolha a sua opcao: 4

Insira o numero desejado: 333

Valor esta presente no vetor, indice 2
```

Figura 46- Interface da função PrimeiraOcorrencia

- 2.8 Escreva uma função que, recebendo como parâmetros uma tabela unidimensional, a sua dimensão e um valor, utilize um algoritmo de pesquisa binária para localizar o valor na tabela e devolver o seu índice. Caso o valor não esteja presente na tabela, deve ser devolvido -1.

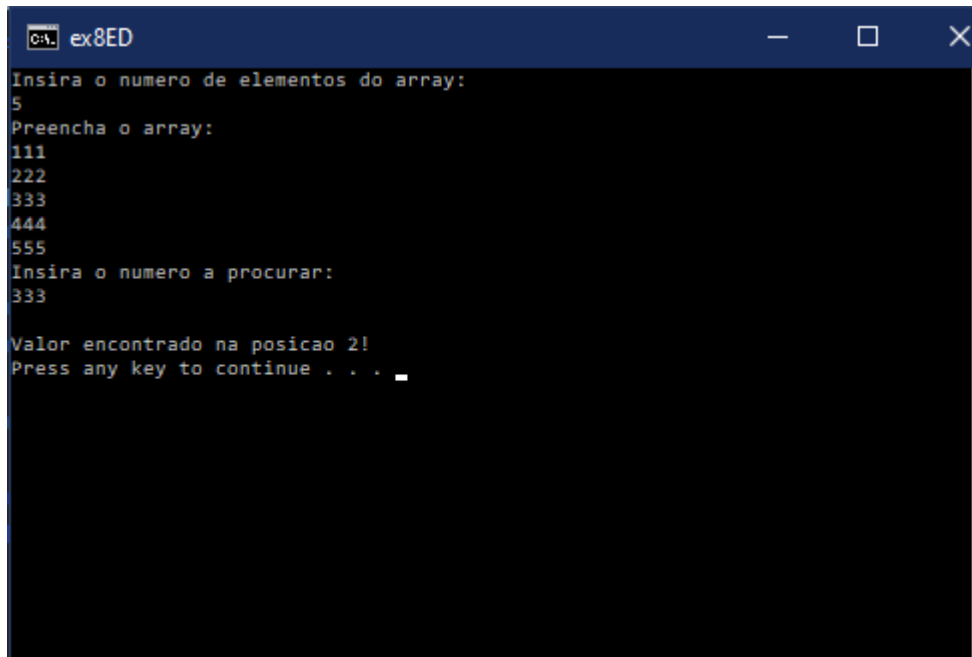
Para este exercício, como nos é pedido um algoritmo desconhecido, tivemos como base o pseudo-código que a professora forneceu no PDF e adaptámos para linguagem C. O resultado foi o seguinte:

```
int PesquisaBin()
{
    primeiro = 0;
    ultimo = n - 1;
    meio = (primeiro+ultimo)/2;
    while (primeiro <= ultimo)
    {
        if (array[meio] < procurar)
            primeiro = meio + 1;
        else
            if (array[meio] == procurar)
            {
                return procurar;
                break;
            }
        else
            ultimo = meio - 1;
        meio = (primeiro + ultimo)/2;
    }
    if (primeiro > ultimo)
        return -1;
}
```

Figura 47- Função para indicar em que posição está o número introduzido pelo o utilizador

O que este algoritmo faz não é difícil de entender. É importante salientar que este parte do sobreposto que o vetor está ordenado. Como funciona? Simples. O algoritmo vai fazer a média do vetor para encontrar o número que está a meio do vetor. A seguir, caso o número encontrado seja menor que o número que se pretender procurar, o algoritmo procura na metade inferior do vetor, caso contrário, procura na metade superior. Caso o vetor não exista, a função terá que retornar -1. Como é que o programa sabe que não existe? Caso o primeiro número seja já maior que o último, significa que o programa já percorreu o vetor todo e não encontrou o valor pretendido pelo utilizador.

Exemplo de output:



```

C:\> ex8ED
Insira o numero de elementos do array:
5
Preencha o array:
111
222
333
444
555
Insira o numero a procurar:
333

Valor encontrado na posicao 2!
Press any key to continue . . .

```

Figura 48- Interface da função PesquisaBin

2.9 Pretende-se ordenar um vetor por ordem crescente dos seus elementos.

Neste exercício também fizemos um menu de opções pois é mais fácil para nós e para o utilizador. Este menu só tem três opções. O menu é o seguinte:

```

int main()
{
    int num_elementos, opcao;
    printf("Introduza quantos elementos quer inserir: ");
    scanf("%d", &num_elementos);
    int num[num_elementos];
    do
    {
        printf("Menu do Programa\n");
        printf("1. Escrever Elementos e Indicar maior indice\n");
        printf("2. Trocar elementos\n");
        printf("3. Ordenar vetor\n");
        printf("0. Sair\n");
        printf("Entre com a opcao: ");
        scanf("%d", &opcao);
        fflush(stdin);
    }
}

```

Figura 49- Menu do programa

Para o utilizador conseguir executar cada opção decidimos também utilizar cases e são os seguintes:

```
switch(opcao)
{
    case 1: indicar_indice_elementos(num, num_elementos); break;
    case 2: trocar_elementos(num, num_elementos); break;
    case 3: selecao_linear(num, num_elementos); break;
    case 0: printf("opcao Sair selecionada\n"); break;
    default: printf("opcao invalida\n");
}
getchar();
}
while(opcao != 0);
}
```

Figura 50- Cases onde são chamadas as funções

A primeira opção do nosso menu é para verificar qual é a posição (índice) onde se encontra o maior elemento do vetor. A função é a seguinte:

```
void indicar_indice_elementos(int num[], int num_elementos)
{
    for (int i = 0; i < num_elementos; i++)
    {
        printf("Introduza um numero: ");
        scanf("%d", &num[i]);
    }
    int indice ,maior =0;
    for (int i = 0; i < num_elementos; i++)
    {
        if(num[i] > maior)
        {
            maior = num[i];
            indice = i;
        }
    }
    printf("o indice do maior elemento e: %d\n", indice);
}
```

Figura 51- Função para ver em que posição esta o maior elemento do vetor

Exemplo de output da primeira opção:

```
Introduza quantos elementos quer inserir: 4
Menu do Programa
1. Escrever Elementos e Indicar maior indice
2. Trocar elementos
3. Ordenar vetor
0. Sair
Entre com a opcao: 1
Introduza um numero: 12
Introduza um numero: 34
Introduza um numero: 2
Introduza um numero: 56
o indice do maior elemento e: 3
```

Figura 52- Interface da função indicar_indice_elementos

A segunda opção serve para trocar dois elementos do vetor, para isso o utilizador tem de inserir a posição onde se encontram os dois elementos. Depois o que programa faz é trocar os elementos dessas posições um pelo outro. A função é a seguinte:

```
void trocar_elementos(int num[], int num_elementos )
{
    int indice1, indice2, troca;
    printf("introduza os dois indices para trocar os elementos desses indices: \n");
    scanf("%d%d", &indice1, &indice2);
    while((indice1 >= num_elementos || indice2 >= num_elementos) || (indice1 < 0 || indice2 < 0))
    {
        printf("Nao foi possivel encontrar o indice introduzido, por favor introduza novamente: \n");
        scanf("%d%d", &indice1, &indice2);
    }
    for (int j = 0; j < num_elementos; j++)
    {
        troca = num[indice1];
        num[indice1] = num[indice2];
        num[indice2] = troca;
    }
    for (int i = 0; i < num_elementos; i++)
    {
        printf("elemento na posicao %d e: %d\n", i, num[i]);
    }
}
```

Figura 53- Função para trocar dois elementos do vetor

Um exemplo de output da segunda opção é o seguinte:

```
Introduza um numero: 13
Introduza um numero: 2
Introduza um numero: 67
Introduza um numero: 89
Introduza um numero: 23
o indice do maior elemento e: 3
Menu do Programa
1. Escrever Elementos e Indicar maior indice
2. Trocar elementos
3. Ordenar vetor
0. Sair
Entre com a opcao: 2
introduza os dois indices para trocar os elementos desses indices:
0
1
elemento na posicao 0 e: 2
elemento na posicao 1 e: 13
elemento na posicao 2 e: 67
elemento na posicao 3 e: 89
elemento na posicao 4 e: 23
```

Figura 54- Output da função trocar_elementos

A última opção do nosso programa é ordenar os elementos do vetor de forma crescente através da seleção linear. Este algoritmo consiste em passar sempre o menor valor do vetor para a primeira posição, a seguir o segundo menor valor vai para a segunda posição e assim sucessivamente até o vetor ficar ordenado. A função desta opção é a seguinte:

```
void selecao_linear(int num[], int num_elementos)
{
    int i, j, min, ordenar;
    for (i = 0; i < (num_elementos-1); i++)
    {
        min = i;
        for (j = (i+1); j < num_elementos; j++) {
            if(num[j] < num[min])
                min = j;
        }
        if (num[i] != num[min])
        {
            ordenar = num[i];
            num[i] = num[min];
            num[min] = ordenar;
        }
    }
    for (int m = 0; m < num_elementos; m++)
    {
        printf("elementos ordenados: %d\n", num[m]);
    }
}
```

Figura 55- Função para ordenar o vetor por ordem crescente

Pseudocódigo do programa da seleção linear:

```
FUNÇÃO seleção_linear (num [], num_elementos)
Variáveis
i, j, min, ordenar: Integer;
Início
Para i <- 0 Até num_elementos -1 Faça
    Min <- i;
    Para j <- i + 1 Até num_elementos Faça
        Se num[j] < num[min] então
            Min <- j;
    Fim_Para
    Se num[i] <> num[min] então
        ordenar = num[i];
        num[i] = num[min];
        num[min] = ordenar;
    Fim_Se
Fim_Para
Para m <- 0 Até num_elementos Faça
    Escreva ("elementos ordenados:", num[j]);
Fim_Para
Fim
```

2.10 Pretende-se ordenar um vetor por ordem crescente dos seus elementos.

Neste exercício primeiro tivemos que fazer um programa que dado um vetor ordenado introduzido pelo utilizador insere-se qualquer número e o programa tem que inserir esse elemento no vetor de modo a este ficar ordenado. A função desta primeira alínea é a seguinte:

```
void ordenar_vetor()
{
    for(i=0;i<tamanho;i++)
    {
        if(inval<vetor[i])
        {
            p = i;
            break;
        }
    }
    for(i=tamanho;i>=p;i--)
    vetor[i]= vetor[i-1];
    vetor[p]=inval;
    if(inval>vetor[tamanho])
    {
        for(int y=0;y<tamanho;y++)
        vetor[y]=vetor[y+1];
        vetor[tamanho]=inval;
    }
}
```

Figura 56- Função para ordenar vetor

Exemplo de output desta função:

```
Menu do Programa
1. Ordenar Elementos
2. Insercao Linear
0. Sair
Entre com a opcao: 1
Introduza o tamanho do vetor: 4
Insira os elementos por ordem crescente:
1
2
3
4
Insira o valor novo do array : 6

Array depois do elemento selecionado :
1 2 3 4 6
```

Figura 57- Interface da função ordenar_vetor

A segunda parte deste exercício é ordenar o vetor por ordem crescente através da inserção linear.

No que é que consiste? Basicamente, a inserção linear serve para comparar quaisquer números que o utilizador insira no array, com os números que estão por trás. O programa irá comparando os números dentro do array com o número inserido pelo utilizador até encontrar o lugar correto do número.

```
void insercao_linear(int num[], int num_elementos)
{
    for(int i = 0; i < num_elementos; i++)
    {
        printf("Introduza o elemento: \n");
        scanf("%d", &num[i]);
    }
    int i, aux, j;
    for (i = 1; i < num_elementos; i++)
    {
        aux = num[i];
        j = i - 1;
        while (j >= 0 && num[j] > aux) {
            num[j + 1] = num[j];
            j = j - 1;
        }
        num[j + 1] = aux;
    }
    for (int m = 0; m < num_elementos; m++)
        printf("vetor por ordem crescente: %d\n", num[m]);
}
```

Figura 58- Função para ordenar os elementos através da inserção linear

Exemplo de output do algoritmo da inserção linear:

```
Entre com a opcao: 2
Introduza o tamanho do vetor: 5
Introduza o elemento:
12
Introduza o elemento:
3
Introduza o elemento:
15
Introduza o elemento:
7
Introduza o elemento:
4
vetor por ordem crescente: 3
vetor por ordem crescente: 4
vetor por ordem crescente: 7
vetor por ordem crescente: 12
vetor por ordem crescente: 15
```

Figura 59- Interface da função insercao_linear

Pseudocódigo do programa da inserção linear:

```
FUNÇÃO inserção_linear (num [], num_elementos)
Variáveis
i, j, aux: Integer;
Início
Para i <- 0 Até num_elementos -1 Faça
  Escrever ("Introduza o elemento");
  Ler (num[i]);
Fim_Para
Para i <- 1 Até num_elementos Faça
  aux <- num[j];
  j <- i - 1;
  Enquanto j >= 0 e num[j] > aux Faça
    num[j+1] <- num[j];
    j <- j - 1;
  Fim_Enquanto
  num[j+1] <- aux;
Fim_Para
Para m <- 0 Até num_elementos Faça
  Escrever("vetor por ordem crescente: num[m]");
Fim_Para
Fim
```


2.11 Pretende ordenar-se, por ordem decrescente, um vetor de inteiros usando o algoritmo *Bubble Sort* (ou algoritmo por borbulhamento).

Neste penúltimo exercício foi pedido que fizéssemos um programa que ordena-se os elementos de um vetor por ordem decrescente através do Bubble Sort. O Bubble Sort consiste em comparar se o primeiro elemento é maior que o segundo, o segundo maior que o terceiro assim sucessivamente até o vetor estar ordenado. Este exercício tem duas alíneas.

Na primeira alínea tivemos que fazer um programa que ordena-se o vetor com o algoritmo Bubble Sort, no entanto, depois de o vetor já estar ordenado o programa continuava a fazer comparações desnecessárias, isto é, ele continuava a ver se o primeiro elemento era maior que o segundo, o segundo maior que o terceiro e assim sucessivamente. A função é a seguinte:

```
void bubble_sort(int vetor[], int tamanho)
{
    int aux;
    for (int i = 0; i < tamanho; i++)
    {
        printf("Introduza o elemento: ");
        scanf("%d", &vetor[i]);
    }
    for (int j = 0; j < tamanho-1; j++)
    {
        for (int i = 0; i < tamanho-1; i++)
            if(vetor[i] < vetor[i+1])
            {
                aux = vetor[i];
                vetor[i] = vetor[i+1];
                vetor[i+1] = aux;
            }
    }
    printf("Vetor ordenado por ordem decrescente: ");
    for (int i = 0; i < tamanho; i++)
        printf("%d ", vetor[i]);
}
```

Figura 60- Função para ordenar os elementos do vetor por ordem decrescente

Exemplo de output:

```
Introduza o tamanho do vetor: 4
Introduza o elemento: 5
Introduza o elemento: 2
Introduza o elemento: 90
Introduza o elemento: 45
Vetor ordenado por ordem decrescente: 90 45 5 2
Press any key to continue . . .
```

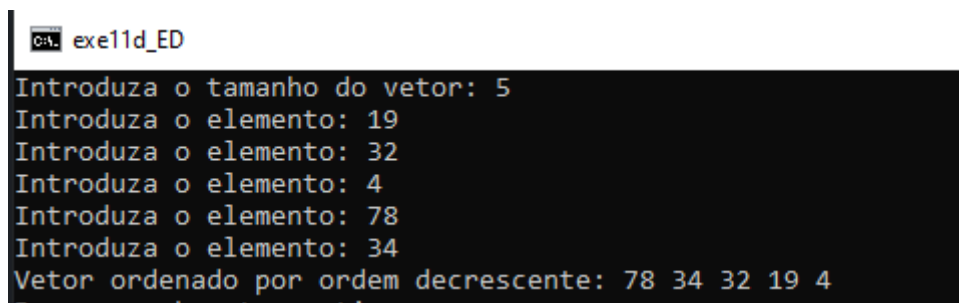
Figura 61- Interface da função bubble_sort

Já na segunda alínea foi nos pedido um programa que fizesse a mesma coisa que o primeiro programa, mas que termina-se o processo quando o vetor já estiver ordenado para evitar assim as tais comparações desnecessárias. Para impedir essas comparações o que fizemos foi a variável tamanho tiramos o valor de j e decrementamos 1. A função desta segunda alínea é a seguinte:

```
void bubble_sort(int vetor[], int tamanho)
{
    int aux;
    for (int i = 0; i < tamanho; i++)
    {
        printf("Introduza o elemento: ");
        scanf("%d", &vetor[i]);
    }
    for (int j = 0; j < tamanho; j++)
    {
        for (int i = 0; i < tamanho-j-1; i++)
            if(vetor[i] < vetor[i+1])
            {
                aux = vetor[i];
                vetor[i] = vetor[i+1];
                vetor[i+1] = aux;
            }
        printf("Vetor ordenado por ordem decrescente: ");
        for (int i = 0; i < tamanho; i++)
            printf("%d ", vetor[i]);
    }
}
```

Figura 62- Função para ordenar os elementos do vetor por ordem decrescente

Exemplo de output desta função:



```
C:\> exe11d_ED
Introduza o tamanho do vetor: 5
Introduza o elemento: 19
Introduza o elemento: 32
Introduza o elemento: 4
Introduza o elemento: 78
Introduza o elemento: 34
Vetor ordenado por ordem decrescente: 78 34 32 19 4
```

Figura 63- Interface da função da segunda alínea

2.12 Desenvolva um subprograma que remova valores repetidos de um vetor.

Para este programa, foi-nos pedido para removermos elementos repetidos de um vetor, para tal, fizemos o seguinte:

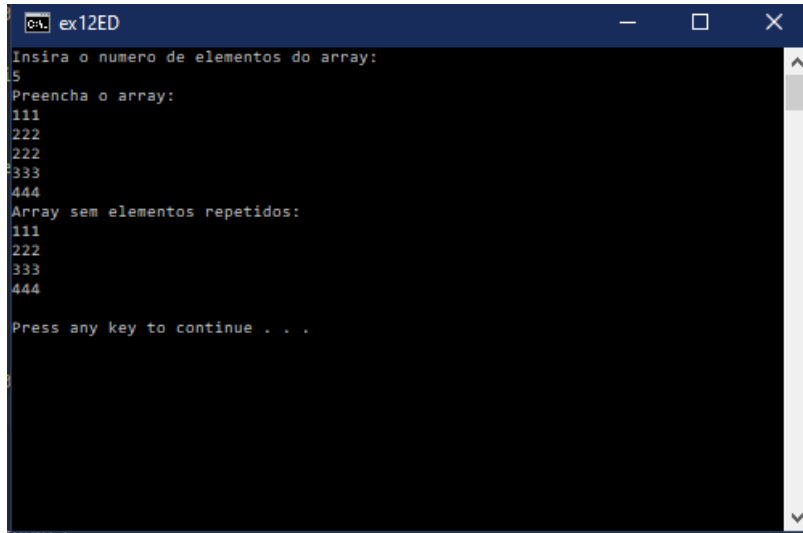
```
for (c = 0; c < n; c++)
{
    for (d = 0; d < count; d++)
    {
        if(a[c] == b[d])
            break;
    }
    if (d == count)
    {
        b[count] = a[c];
        count++;
    }
}

printf("Array sem elementos repetidos:\n");

for (c = 0; c < count; c++)
    printf("%d\n", b[c]);
}
```

Como é que funciona? Primeiro, o programa vai percorrer 2 ciclos, um para o vetor normal, outro para o count (já se explica o que é o count). O que se passa neste programa é que este está a inspecionar o ciclo. Ao inspecionar o ciclo, está a passar os valores do vetor “a” para o vetor “b”. Cada vez que faz esta passagem, o programa irá incrementar a variável “count”. Os vetores “a” e “b” são comparados, e caso os números sejam iguais, o programa deixa de incrementar a variável count e volta ao início. Daí, depois do programa acabar, o ciclo que utilizamos para apresentar o array sem os elementos repetidos, vai até ao número incrementado “count”.

Exemplo de output para o programa é o seguinte:



```
ex12ED
Insira o numero de elementos do array:
5
Preencha o array:
111
222
222
333
444
Array sem elementos repetidos:
111
222
333
444
Press any key to continue . . .
```

3 Conclusão

Chegamos assim ao fim do nosso relatório. Achamos que fizemos um bom trabalho e que estes exercícios foram importantes não só para relembrarmos do trabalho com vetores mas também para aprofundarmos os nossos conhecimentos sobre ordenação e pesquisa, atributos que serão bastante úteis quer para o nosso futuro académico, quer talvez para o nosso futuro no mundo do trabalho.

4 Referências

PDF's disponibilizados pelo professor e Internet.

