

Aula prática #8 – Vetores

Problema 1

Implemente uma função que procure e retorne todas as ocorrências de um dado elemento num vetor. Utilize para esse efeito a seguinte função:

```
1 int procuraTodos(int *v, int N, int x, int posicoes[], int *nPosicoes);
```

- v e N indicam o vetor e tamanho, respetivamente;
- x é o valor que se pretende procurar;
- posicoes e nPosicoes são respetivamente o vetor para guardar as posições das ocorrências encontradas e o número de posições;
- o valor de retorno deverá ser 1 se encontrou algum elemento e 0 se não encontrou.

Implemente um programa para testar esta função.

Problema 2

Implemente uma função que remova elementos adjacentes iguais num vetor de caracteres. Utilize para esse efeito a seguinte função:

```
1 int compacta(char orig[], int N, char dst[]);
```

- orig e N indicam o vetor de entrada e tamanho, respetivamente;
- dst é o vetor onde será guardado o resultado;
- o valor de retorno deverá ser o tamanho do vetor dst.

Implemente um programa para testar esta função. Considere que a leitura do vetor termina com um ponto final ('.')

Input	Output
aaaaabbbbbccccc.	abc
abccccc.	abc
abc.	abc
carro.	caro

Problema 3

Escreva um programa que registre, num vetor multidimensional, o número mecanográfico e as notas obtidas por cada aluno. Cada um dos 4 alunos de uma turma realizou duas provas de avaliação. Imprima o conteúdo do vetor, bem como uma coluna extra onde mostre a média das notas obtidas por cada aluno.

Exemplo

```
1 Dados do aluno 1: 09001 5 15
2 Dados do aluno 2: 10701 10 12
3 Dados do aluno 3: 10321 16 17
4 Dados do aluno 4: 10452 8 11
5
6 Numero  Nota 1  Nota 2  Media
7 09001    5      15     10.0
8 10701   10      12     11.0
9 10321   16      17     16.5
10 10452    8      11      9.5
```

Problema 4

Tendo por base o programa do exercício 3, implemente uma função que ordene por ordem decrescente as notas obtidas pelos alunos. Utilize para esse efeito a seguinte função:

```
1 void ordena(float notas[][2], int N);
```

As notas são guardadas numa matriz $N \times 2$, onde primeira coluna guarda o número do aluno e a segunda coluna a respetiva média. Altere o programa para testar esta função.

Exemplo

```
1 Numero Media
2 10321 16.5
3 10701 11.0
4 09001 10.0
5 10452 9.5
```

Problema 5

Implemente um programa que permita ao utilizador introduzir os valores de uma matriz quadrada de tamanho $N \times N$, com $N \leq 10$ (inicialmente especificado pelo utilizador). A matriz deverá ser guardada num vetor unidimensional e para conseguir fazer o mapeamento entre bi/unidimensional deverá usar a seguinte função:

```
1 int pos(int x, int y, int tamanho) { return x + y * tamanho; }
```

O programa deve permitir a realização das seguintes operações:

a) Visualizar a matriz. A visualização de uma matriz deve ser efetuada por uma função para esse efeito:

```
1 void imprimeMatriz(int matriz[], int N);
```

b) Multiplicar a matriz por um escalar (o valor em cada posição da matriz é multiplicado pelo escalar);

```
1 void produtoEscalar(int matriz[], int N, int escalar);
```

c) Multiplicação da matriz por uma matriz de $N \times 1$ (sendo N o tamanho do lado da matriz quadrada), cujos valores devem ser introduzidos pelo utilizador.

```
1 void multMatrizes(int matriz[], int mult[], int resultado[], int N);
```

Problema 6

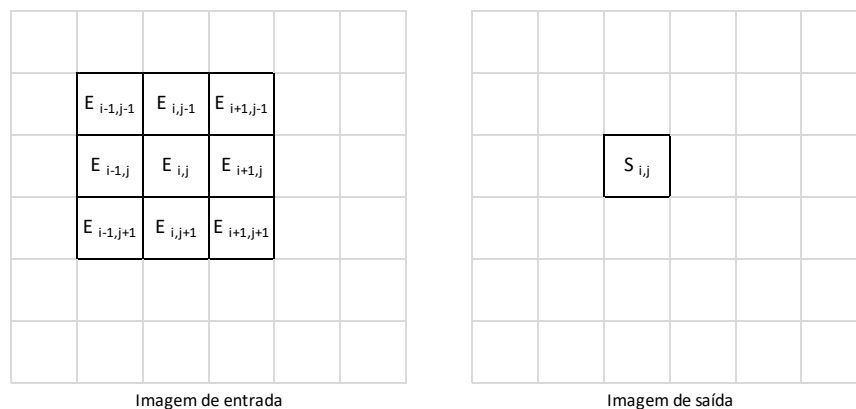
O ficheiro `ex6.c` apresenta uma implementação parcial de um programa que testa algumas operações de processamento de imagem. A implementação fornecida lê um ficheiro com uma imagem e grava os resultados em imagens PNG. Para compilar, deverá usar o comando `clang ex6.c pnglib.c`; depois de executar o programa, a imagem original é gravada no ficheiro `resultado0.png`.

Considere que a imagem é representada por uma matriz com $altura \times largura$ pixéis. Tal como no problema anterior, essa matriz está guardada num vetor unidimensional, e portanto será necessário fazer o mapeamento entre linha/coluna e a respetiva posição no vetor. Nota: pode usar `POS(x, y)` para fazer esse mapeamento.

Implemente as seguintes funções, que transformam uma imagem de entrada numa de saída:

a) `filtro_medio`: para cada pixel da imagem de saída, calcule o respetivo valor de acordo com:

$$S_{i,j} = \frac{E_{i-1,j-1} + E_{i,j-1} + E_{i+1,j-1} + E_{i-1,j} + E_{i,j} + E_{i+1,j} + E_{i-1,j+1} + E_{i,j+1} + E_{i+1,j+1}}{9}$$



Nas linhas e colunas nos bordos da imagem de saída, onde não é possível usar todos os valores de entrada, não calcule os respetivos valores. A imagem de saída é guardada no ficheiro `resultado1.png` → qual é o efeito deste filtro?

b) `filtro_sobel`: para cada pixel da imagem de saída, calcule o respetivo valor de acordo com:

$$S_{i,j} = | -E_{i-1,j-1} + E_{i+1,j-1} - 2 * E_{i-1,j} + 2 * E_{i+1,j} - E_{i-1,j+1} + E_{i+1,j+1} |$$

Tal como na alínea anterior, não calcule os valores de saída nos bordos da imagem. A imagem de saída é guardada no ficheiro `resultado2.png` → qual é o efeito deste filtro?

c) `filtro_threshold`: para cada pixel da imagem de saída, calcule o respetivo valor de acordo com:

$$S_{i,j} = \begin{cases} 0, & E_{i,j} < threshold \\ 255, & E_{i,j} \geq threshold \end{cases}$$

A imagem de saída é guardada no ficheiro `resultado3.png` → qual é o efeito deste filtro?

Problema 7

O ficheiro `ex7.c` apresenta uma implementação parcial do jogo do galo (“Tic Tac Toe” em Inglês). Em primeiro lugar, analise em detalhe a lógica de jogo já implementada. De seguida, implemente as funções em falta no ficheiro de forma a garantir que o programa executa corretamente.

a) Implemente a função que imprime o tabuleiro de jogo:

```
1 void imprimeTabuleiro(char tabuleiro[LINHAS][COLUNAS])
```

b) Implemente a função que verifica se existe um vencedor. Esta deve retornar “1” em caso positivo ou “0” em caso negativo.

```
1 int haVencedor(char tabuleiro[LINHAS][COLUNAS])
```