Ficha de Trabalho n.º 1

Tabelas

Relembre a declaração e atribuição de arrays em C:

```
#include <stdio.h>

#define DIM 10


void criaVector(int vec[])
{
    for(int i=0; i<DIM; i++)
        vec[i] = i + 1;
}

void escreveVector(int v[])
{
    printf("vector\n");
    for(int i=0; i<DIM; i++)
        printf("elemento %i:\t%i\n",i,v[i]);
}


int main ()
{
    int vector[DIM];

    criaVector(vector);
    escreveVector(vector);

    return 0;
}
```

Para cada uma das questões seguintes crie os subprogramas que considerar necessários, testando-os no main. Pode atribuir os nomes que considerar adequados, bastando identificar a questão a que se referem.

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

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

NOTA: Aplique a função *random* ou *rand*.

Verifique a biblioteca que deve incluir no programa e tenha em atenção o tipo de valores que esta função devolve.

3. Considere um vector de inteiros de qualquer dimensão.

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

O vector e a sua dimensão devem ser passados como parâmetro.

4. Desenvolva subprogramas que determinem a média dos números ímpares e a média dos números pares indicados por um utilizador, usando um vector:

- Ler número de elementos do vector;
- Ler elementos do vector;
- Calcular médias.

5. Considere o seguinte programa (incompleto).

```
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;
}


// subprograma que escreve uma matriz

int main()
{
      int mat1[5][7],mat2[5][7];

      matriz_1(mat1);
      // escrever matriz mat1

      matriz_2(mat2);
      // escrever matriz mat2

      return 0;
}
```

a. Copie para as secções correctas do programa, o código anterior.

b. Complete o programa, escrevendo o subprograma em falta e a sua chamada no programa principal.

6. Construa uma calculadora que permita realizar várias operações entre matrizes de elementos reais. Elabore subprogramas para efectuar as várias operações.

- **a.** Ler os elementos para uma matriz A_{mxn};
- **b.** Escrever uma matriz A_{mxn};
- **c.** Calcular a média de todos elementos de uma matriz A_{mxn};
- **d.** Calcular a média de todos elementos de uma dada coluna *k* de uma matriz A_{mxn};
- **e.** Calcular a média de todos elementos de uma dada linha *l* de uma matriz A_{mxn};
- **f.** Contar o número de zeros que se encontram acima da diagonal principal de uma matriz A_{mxn};
- **g.** Determinar a linha de uma matriz A_{mxn} que tem a soma dos seus elementos máxima;
- **h.** Trocar as colunas *j* e *k* de uma matriz A_{mxn};
- **i.** Somar duas matrizes A_{mxn} e B_{mxn};

Teste, no main, todos os subprogramas anteriores.

Pesquisa e Ordenação

7. Pretende-se procurar um valor num vector de inteiros inserido por um utilizador.
Desenvolva subprogramas que desempenhem as seguintes tarefas:

- **a.** Conhecendo o número de elementos a colocar no vector, ler os diferentes valores inserindo-os no vector pela ordem indicada pelo utilizador;
- **b.** Dado um valor, indicar o número de ocorrências desse valor;
- **c.** Dado um valor, procurá-lo no vector devolvendo a última ocorrência desse valor (caso o valor não esteja presente no vector, deve ser devolvido –1);
- **d.** Dado um valor, procurá-lo no vector devolvendo a primeira ocorrência desse valor (caso o valor não esteja presente no vector, deve ser devolvido –1);

Conclua o programa de modo que possa testar as diferentes funcionalidades.

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.
Conclua o programa para testar a função.
NOTA: Algoritmo Pesquisa Binária

Dados: tabela *v*
Valor *x*

```
Procura x em v[i1,..., in]
    Enquanto (i1 ≠ in) Faz
        meio ← (i1+in)/2
        Se x = vmeio
        Então x está na posição meio de v
        Senão
            Se x < vmeio
            Então procura x em v[i1,..., imeio-1]
            Senão procura x em v[imeio+1,..., in]
            Fim Se
        Fim Se
    Fim Enquanto
    //o que fazer quando x não está no vector?
Fim Procura
```

9. Pretende-se ordenar um vector por ordem crescente dos seus elementos. Desenvolva funções que realizem as tarefas seguintes.

- **a.** Dado um vector, indicar o índice do maior dos seus n primeiros elementos.
- **b.** Dado um vector e dois índices, trocar os elementos dessas localizações.
- **c.** Usar o algoritmo Selecção Linear para ordenar um dado vector. Apresente o pseudocódigo.
  NOTA: Procure o algoritmo Selecção Linear.

10. Pretende-se ordenar um vector por ordem crescente dos seus elementos.

- **a.** Desenvolva um subprograma que, dado um vector *ordenado* com *n* componentes inteiras e um elemento inteiro, insira o elemento no vector de modo a mantê-lo ordenado.
- **b.** Altere o subprograma anterior de forma que ordene um vector pelo método Inserção Linear:
  - Condições iniciais:
    O primeiro elemento do vector é um subvector ordenado;
    Restante vector é um subvector não ordenado;
  - O primeiro elemento do subvector não ordenado é inserido na posição correcta do subvector ordenado, movendo os elementos maiores uma posição para a direita;

  Neste momento o subvector ordenado tem dois elementos.
  - Repetir sucessivamente o ponto anterior até que o vector esteja ordenado.

  Apresente o pseudocódigo do algoritmo implementado.

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

- **c.** Implemente uma primeira versão do algoritmo em que podem ser efectuadas várias comparações desnecessárias (depois do vector já estar ordenado).
- **d.** Implemente uma versão mais eficiente do algoritmo que termina o processo assim que tiver a garantia de que o vector já está ordenado.

12. Desenvolva um subprograma que remova valores repetidos de um vector.