Ficha de Trabalho n.º 1

ESTRUTURAS DE DADOS
LEI
2024/2025

Tabelas, Vetores, Matrizes, *Arrays*

Relembre a declaração e atribuição de *arrays* em C:

```c
#include <stdio.h>

#define DIM 10


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

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


int main ()
{
    int vetor[DIM];

    criaArray(vetor);
    escreveArray(vetor, DIM);

    return 0;
}
```

Para cada uma das questões seguintes crie os subprogramas que considerar necessários, testando-os no *main*.

1. Escreva um subprograma que receba como parâmetros de entrada uma tabela de inteiros e um valor inteiro (tabela e dimensão da tabela) e leia os elementos inseridos pelo utilizador para a tabela.

2. Desenvolva uma função que gere aleatoriamente números entre 1 e 50, colocando estes valores numa tabela de 30 elementos.

   NOTA: Aplique a função *random*. Verifique a biblioteca que deve incluir no programa e tenha em atenção o tipo de valores que estas funções devolvem.

3. Considere uma tabela unidimensional de inteiros de qualquer dimensão.

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

   A tabela e a sua dimensão devem ser passadas como parâmetro.

4. Desenvolva subprogramas que determinem a média dos números ímpares e a média dos números pares, registados pelo utilizador, com recurso a uma tabela. Os subprogramas devem permitir:
   - Ler número de elementos da tabela;
   - Ler elementos da tabela;
   - 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 corretas 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 efetuar 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.

Algoritmos de ordenação e pesquisa

7. Pretende-se procurar um valor numa tabela de valores inteiros inseridos por um utilizador.

   Desenvolva subprogramas que desempenhem as seguintes tarefas:

   a. Conhecendo o número de elementos a colocar na tabela, ler os diferentes valores inserindo-os na tabela pela ordem indicada pelo utilizador;
   b. Dado um valor, indicar o número de ocorrências desse valor;
   c. Dado um valor, procurá-lo na tabela devolvendo a última ocorrência desse valor (caso o valor não esteja presente na tabela, deve ser devolvido –1);
   d. Dado um valor, procurá-lo na tabela devolvendo a primeira ocorrência desse valor (caso o valor não esteja presente na tabela, 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[0,..., n-1]
    Início
        esq ← 0
        dir ← n-1
        Enquanto (esq ≤ dir) Faz
            meio ← (esq + dir)/2
            Se x = v_meio
            Então x está na posição meio de v
            Senão
                Se x < v_meio
                Então dir ← meio - 1 (procura x em v[esq,..., meio-1])
                Senão esq ← meio + 1 (procura x em v[meio+1,..., dir])
                Fim Se
            Fim Se
        Fim Enquanto
        //o que fazer quando x não está na tabela?
    Fim
```

9. Pretende-se ordenar uma tabela unidimensional por ordem crescente dos seus elementos. Desenvolva funções que realizem as tarefas seguintes.
   a. Dado uma tabela, indicar o índice do maior dos seus n primeiros elementos.
   b. Dada uma tabela e dois índices, trocar os elementos dessas localizações.
   c. Usar o algoritmo Seleção Linear para ordenar uma dada tabela. Apresente o pseudocódigo.
      NOTA: Pesquise o algoritmo Seleção Linear.

10. Pretende-se ordenar uma tabela unidimensional por ordem crescente dos seus elementos.
   a. Desenvolva um subprograma que, dada uma tabela *ordenada* com *n* componentes inteiras e um elemento inteiro, insira o elemento na tabela de modo a mantê-la ordenada.
   b. Altere o subprograma anterior de forma que ordene uma tabela pelo método Inserção Linear:
      - Condições iniciais:
        O primeiro elemento da tabela é uma tabela ordenada;
        Restante tabela é uma tabela não ordenada;
      - O primeiro elemento da tabela não ordenada é inserido na posição correta da tabela ordenada, movendo os elementos maiores uma posição para a direita;
        Neste momento a tabela ordenada tem dois elementos.
      - Repetir sucessivamente o ponto anterior até que a tabela esteja ordenada.

      Apresente o pseudocódigo do algoritmo implementado.

11. Pretende-se ordenar, por ordem decrescente, uma tabela de inteiros usando o algoritmo *BubbleSort* (ou algoritmo por borbulhamento).
   a. Implemente uma primeira versão do algoritmo em que podem ser efetuadas várias comparações desnecessárias (depois da tabela já estar ordenada).
   b. Implemente uma versão mais eficiente do algoritmo que termina o processo assim que tiver a garantia de que a tabela já está ordenada.

12. Desenvolva um subprograma que remova valores repetidos de uma tabela unidimensional.