Programação

Guião laboratorial n.º 1

Ponteiros e Tabelas

Tópicos da matéria:

- Noções básicas sobre ponteiros e endereços.
- Aritmética de ponteiros
- Tabelas de ponteiros para caracter.
- Argumentos da linha de comando.

Bibliografia:

K. N. King, *C Programming: A Modern Approach*: Capítulos 11, 12 e 13.

Nota: antes da implementação deve definir uma estratégia genérica para cada um dos exercícios propostos.

---

1. Considere o seguinte esqueleto da função `main()`:

```
#include <stdio.h>
int main()
{
      int a, b, total, *p = &a, *q = &b, *r = &total;

      /*  completar  */

      printf("a= %d \t b= %d \t total= %d\n", a, b, total);
      return 0;
}
```

O objetivo do programa é pedir dois números inteiros ao utilizador e guardá-los nas variáveis `a` e `b`. A seguir deve somar esses valores e guardar o resultado na variável `total`. Termine a implementação da função `main()` sem nunca se referir explicitamente (i.e., pelo seu nome) às variáveis `a`, `b` ou `total`.

2. Escreva uma função em C que receba informação sobre 3 variáveis reais do tipo *float* e que efetue uma rotação entre elas, i.e., a segunda variável deve ficar com o valor da primeira, a terceira deve ficar com o valor da segunda e a primeira deve ficar com o valor da terceira. Ao implementar o código defina corretamente os parâmetros da função, de forma a garantir que esta realiza a tarefa pretendida.

3. Escreva uma função em C que receba, como argumentos, o nome e a dimensão de uma tabela unidimensional de números inteiros e que coloque a zero todos os elementos cujo valor seja inferior à média dos valores armazenados nessa tabela. Pode assumir que quando a função for chamada a tabela já foi inicializada.

4. Escreva uma função em C que receba o nome e a dimensão de uma tabela de inteiros positivos e verifique quantos dos seus elementos são pares e quantos são ímpares. Deve ainda calcular qual o maior valor presente na tabela e a posição onde ele se encontra.

Por exemplo, se o conteúdo da tabela for:

1	3	7	5	2	10	9	7	7	1

Existem 2 números pares, 8 números ímpares. O maior número é o 10 e está na posição 5.

A função recebe como argumentos um ponteiro para o início da tabela, o número de elementos que esta contém, um ponteiro para o inteiro onde deve colocar o n.º de elementos ímpares, um ponteiro para o inteiro onde deve colocar o n.º de elementos pares, um ponteiro para o inteiro onde deve colocar o maior valor armazenado na tabela e um ponteiro para o inteiro onde deve colocar a posição onde o maior valor se encontra. A sua declaração é a seguinte:

```
void f(int *t, int tam, int *np, int *ni, int *maior, int *pos);
```

5. Escreva uma função em C que receba 2 tabelas de inteiros e que contabilize quantos elementos elas têm em comum. A função tem o seguinte protótipo:

```
int comuns(int *tabA, int tamA, int *tabB, int tamB);
```

Recebe informação sobre as duas tabelas e devolve o número de elementos em comum. As tabelas recebidas estão ordenadas de forma crescente e dentro de cada uma das tabelas não existem elementos repetidos

6. Escreva uma função em C que determine quantos elementos de uma tabela de inteiros são iguais à média dos seus dois vizinhos. A função recebe como argumentos um ponteiro para o início da tabela, o número de elementos que esta contém e um ponteiro para uma variável inteira onde deve ser colocado o resultado (i.e., quantos elementos são iguais à média dos seus vizinhos). A sua declaração é a seguinte:

```
void vizinhos(int *tab, int dim, int *igual);
```

7. Escreva uma função em C que encontre os dois maiores elementos de um vetor de inteiros. A função recebe, como argumentos, o nome e a dimensão do vetor, e os ponteiros para as variáveis onde os dois maiores elementos devem ser armazenados.

```
void procura_dupla(int *tab, int tam, int *prim, int *seg);
```

8. Escreva uma função em C que determine em que posição de uma tabela de inteiros se encontra o elemento que regista a maior subida em relação ao elemento anterior. Esta posição deve ser devolvida como resultado. A função recebe como argumentos um ponteiro para o início da tabela e o número de elementos que esta contém. A sua declaração é a seguinte:

```
int maior_subida(int *tab, int dim);
```

9. Escreva uma função em C que verifique se uma sequência de caracteres representa um número de telefone da rede fixa PT. A sequência deve obedecer à seguinte propriedade:

- É composta por 9 caracteres, em que o primeiro representa o dígito 2 e os restantes qualquer um dos 10 dígitos existentes.

A declaração da função é a seguinte: `void verifica(char *tel, char *c);`

O argumento `tel` aponta para o primeiro elemento da sequência de caracteres que representa o número de telefone (existe um '\0' no final) e o argumento `c` aponta para uma variável do tipo caracter onde deve ser colocado o resultado da avaliação. Se o número de telefone analisado for válido deve aí ser colocado o caracter 'V'. Caso contrário, deve ser colocado o caracter 'I'.

Nota: pode utilizar a função `int isdigit(char c);` da biblioteca `<ctype.h>`. Devolve um valor diferente de 0 se o argumento `c` for um caracter que represente um dígito.

10. Escreva uma função em C que verifique se existem 3 caracteres consecutivos iguais numa frase. A função recebe como argumento um ponteiro para o início da frase (deve assumir que no final existe um '\0'). Deve devolver 1 se existirem 3 caracteres consecutivos iguais, ou 0 no caso contrário. A sua declaração é a seguinte:

```
int tres_consecutivos(char *frase);
```

11. Escreva uma função em C que receba, como argumentos, os nomes e as dimensões de dois vetores de inteiros e verifique se estes são iguais. Considere que dois vetores de inteiros são iguais se tiverem o mesmo número de elementos e se, em posições equivalentes, tiverem elementos com o mesmo valor. A função devolve 1 se os vetores forem iguais, ou 0, no caso contrário.

12. Modifique a função do exercício anterior de modo a permitir comparar matrizes de inteiros (i.e., tabelas com duas dimensões).

13. Escreva uma função em C que calcule o produto escalar de dois vetores de números reais **a, b**, com dimensão **n,** sabendo que:

$$\underline{a} \bullet \underline{b} = a_1 b_1 + a_2 b_2 + .. + a_n b_n \text{ com } \underline{a} = \begin{bmatrix} a_1 \\ a_2 \\ ... \\ a_n \end{bmatrix} \text{ e } \underline{b} = \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{bmatrix}$$

A declaração da função poderá ter o seguinte formato:

```
double produto_escalar (double *a, double *b, int n)
```

14. Complete o seguinte programa:

```
#include <stdio.h>
#define L1  3
#define C1  2
#define L2  4
#define C2  3

void escreve(int n_lin, int n_col, _________________){
      printf("\nFalta implementar a funcao\n");
}

int main(){
      int mat1[L1][C1]={{1,2},{3,4},{5,6}};
      int mat2[L2][C2]={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

      printf("\nMatriz mat1:\n");
      escreve(L1, C1, ______);
      printf("\nMatriz mat2:\n");
      escreve(L2, C2, ______);
      return 0;
}
```

A função `escreve()` deve mostrar o conteúdo de uma matriz de inteiros na consola. A função recebe como argumentos o número de linhas, o número de colunas e o endereço inicial da matriz.
Neste exemplo, o resultado será o seguinte:

Matriz mat1:
1 2
3 4
5 6

Matriz mat2:
1 2 3
4 5 6
7 8 9
10 11 12

15. Escreva uma função em C que verifique se todos os elementos de uma matriz M×N de inteiros são únicos. A função tem o seguinte protótipo:

```
int unicaMat(int nLin, int nCol, int mat[][nCol]);
```

A função devolve 1 se todos os elementos da matriz forem únicos, ou 0, caso contrário.

16. Escreva uma função em C que some duas matrizes de inteiros, A e B. A função deve receber a informação necessária para efetuar a soma. O resultado deve ficar armazenado na matriz A.

17. Escreva uma função em C que efetue a transposição numa matriz N×N de inteiros. A transposição consiste em trocar as linhas pelas colunas. A função recebe como argumentos o endereço inicial da matriz e o valor N (pode assumir que a matriz é quadrada). Por exemplo:

Se tivermos a seguinte matriz 3*3

1	3	5
6	3	2
10	45	4

Após a transposição obtém-se

1	6	10
3	3	45
5	2	4

18. Escreva uma função em C que calcule a média dos valores armazenados em cada uma das colunas de uma matriz de números reais (*float*). A função recebe como argumentos o endereço inicial da matriz e as suas dimensões. Os valores calculados devem ser escritos na consola. A função deve devolver os índices das colunas com a média mais elevada e a média mais baixa.

19. Pretende-se reduzir uma imagem a metade do seu tamanho original. A imagem inicial consiste numa tabela bidimensional com M×M valores inteiros onde cada *pixel* (ponto na imagem) pode tomar um valor inteiro entre 0 e 9 (tonalidades de cinza). A imagem reduzida é armazenada numa tabela bidimensional com M/2×M/2 valores reais onde cada elemento (*pixel*) corresponde á média dos 4 elementos que substitui. Para melhor compreensão do enunciado veja o exemplo apresentado de seguida (neste caso, para M=8):

Imagem original

0	4	9	3	6	7	2	9
9	4	7	8	3	4	7	2
5	1	2	6	3	8	6	2
7	1	4	8	7	2	0	7
4	9	3	5	2	1	5	7
5	3	4	7	8	9	1	0
2	3	5	6	7	9	2	4
2	3	4	5	6	7	8	9

Imagem reduzida

4.3	6.8	5.0	5.0
3.5	5.0	5.0	3.8
5.3	4.8	5.0	3.3
2.5	5.0	7.3	5.8

Escreva uma função em C que efetue esta operação. A função recebe 3 argumentos: endereço inicial da matriz original, endereço inicial da matriz reduzida e o valor M.

20. Escreva uma função em C que, para um dado mês à escolha do utilizador, indique o seu correspondente em Língua Inglesa.

21. Considere que pretende resolver um quebra-cabeças que surgiu no seu jornal. Existe um retângulo, com um determinado número de linhas e um determinado número de colunas, preenchido com caracteres alfabéticos em cada uma das suas posições. Na figura pode ver um exemplo para um quebra-cabeças com 5 linhas e 6 colunas.

e	B	a	u	l	q
l	e	r	r	s	s
u	w	u	q	g	r
a	a	l	l	u	a
p	m	h	u	d	j

Escreva uma função em C que procure todas as ocorrências de uma determinada palavra no quebra-cabeças. A palavra pode ocorrer numa linha ou numa coluna. De cada vez que a função encontrar uma ocorrência da palavra deve escrever no monitor o número da linha e da coluna em que a palavra tem início. Considerando o exemplo da figura, se a palavra a pesquisar for *lua* a função deveria escrever:

```
A palavra lua surge:
   -    Ao longo da coluna 0 com início na linha 1
   -    Ao longo da linha 3 com início na coluna 3
```

A função recebe como argumentos o endereço inicial da matriz de caracteres, as suas dimensões (n.º de linhas e de colunas) e um ponteiro para a palavra a pesquisar.

22. Um programa em C lida com uma estrutura de dados `char *s[][2]` para armazenar uma lista de sinónimos. A tabela é uma variável local da função `main()` e é inicializada na declaração (ver código disponível no *InforEstudante*). Os sinónimos armazenados são os seguintes:

estranho	bizarro
desconfiar	suspeitar
vermelho	encarnado
duvidar	desconfiar
carro	automóvel

a) Escreva uma função que escreva na consola os pares de sinónimos armazenados na tabela. A função recebe como argumentos um ponteiro para o início da tabela e o número de pares de palavras aí armazenados (i.e., o número de linhas).

b) Escreva uma função que verifique se uma determinada palavra tem um sinónimo conhecido. A função recebe, como argumentos, um ponteiro para o início da tabela, o número de pares de palavras armazenados e um ponteiro para a palavra a pesquisar. Devolve como resultado um ponteiro para o sinónimo da palavra recebida como argumento (se existirem vários sinónimos devolve um ponteiro para um deles). Se não existir nenhum sinónimo, a função devolve NULL.

c) Escreva uma função que obtenha uma frase do utilizador e verifique se esta contém palavras que tenham sinónimo conhecido. Sempre que estes existirem, deve escrever no monitor a palavra original e o seu sinónimo. A função recebe como argumentos um ponteiro para o início da tabela e o número de pares de palavras aí armazenados.

d) Escreva uma função que verifique se existem palavras que apareçam em mais do que uma entrada na tabela. No exemplo em cima, a palavra desconfiar aparece duas vezes. A função recebe como argumentos um ponteiro para o início da tabela e o número de pares de palavras aí armazenados. Devolve como resultado o número de palavras que aparecem mais do que uma vez.

23. Desenvolva um programa que receba duas palavras a partir da linha de comando e escreva no monitor uma sequência constituída por caracteres retirados alternadamente de cada uma das palavras originais.
As palavras originais devem ter o mesmo número de caracteres. Se isso não suceder (ou se o número de palavras for diferente de 2), o programa deve terminar imediatamente.
Exemplo: Se o programa receber AAA e BBB, a sequência final será: ABABAB.

24. Desenvolva um programa, chamado `media`, que receba na linha de comando um conjunto de inteiros e apresente a sua média.
Por exemplo, se a chamada do programa tiver o seguinte formato:
`C:\> 1 2 3 4 5 6 7`
obtém-se o resultado: `4.00`

25. Desenvolva um programa que leia um conjunto de palavras a partir da linha de comando. Se o número de palavras for ímpar ou superior a 20, o programa termina imediatamente. Caso contrário, agrupa as palavras, duas a duas (palavras consecutivas), numa tabela de *strings*, `char s[10][50];`
Após ter agrupado todas as palavras, deve escrever na consola as linhas que foram armazenadas na tabela.

26. Modifique o programa anterior, de modo que as palavras sejam agrupadas começando nos limites opostos (a primeira palavra é agrupada com a última, a segunda com a penúltima e assim sucessivamente).