Aula prática #9 – Apontadores e Strings

Problema 1

Escreva uma função que recebe dois valores inteiros por referência e devolve por retorno o endereço do valor maior. Escreva um programa para testar a sua função. Considere o seguinte protótipo para a função:

```
1 int *vmaior(int *valor1, int *valor2);
```

Exemplo

```
1 Insira dois valores: 56 32
2 Endereco das variaveis: 0x0016F838 0x0016F83B
3 Endereco do maior: 0x0016F838
4 Valor: 56
```

Problema 2

Escreva uma função chamada **ordena** que recebe três valores e os ordena por ordem crescente. Teste a sua função com um programa que pede ao utilizador três números, invoca a função ordena e depois imprime o resultado no ecrã. Considere o seguinte protótipo para a função:

```
1 void ordena(int *valorA, int *valorB, int *valorC);
```

Exemplo

```
1 Insira os valores a ordenar: 43 65 17
2 Valores a, b, c ordenados por ordem crescente: 17 43 65
```

Problema 3

Escreva uma função chamada **horasMin** que converte um valor em minutos para o formato *horas:minutos*. A sua função deve ainda retornar 1 se o número de *horas:minutos* for superior a um dia e 0 caso contrário. Para testar a sua função escreva um programa que usa a função **horasMin** para efetuar a conversão e que depois imprime o resultado no ecrã.

```
int hoursMin(int totalMins, int *hours, int *minutes);
```

Exemplo

```
Insira o total de minutos: 568
568 minutos correspondem a 09h:28m e nao e superior a 1 dia.

Insira o total de minutos: 4689
4689 minutos correspondem a 78h:09m e e superior a 1 dia.
```

Problema 4

Implemente duas funções que convertam coordenadas cartesianas em coordenadas polares e vice-versa. Deverá definir as seguintes funções:

```
void cartesianas_polares(float x, float y, float *r, float *theta);
void polares_cartesianas(float r, float theta, float *x, float *y);
```

As relações entre coordenadas cartesianas em coordenadas polares são definidas por:

$$x = r.cos\theta$$

$$y = r.sen\theta$$

$$r = \sqrt{x^2 + y^2}$$

$$\theta = atan2(y, x)$$

Escreva um programa que lhe permita testar as funções que desenvolveu.

Problema 5

Implemente uma função que calcule o quociente e o resto da divisão inteira de dois números do tipo **int**, sem utilizar os operadores % e /. Utilizando o procedimento que definir, implemente uma função que determina a soma dos dígitos de um número inteiro e outra função que teste se um número é par. Portanto, deverá definir as seguintes funções:

```
1 void quociresto(int dividendo, int divisor, int *quociente, int *resto);
2 int soma(int n);
3 int par(int n);
```

Por fim, escreva um programa que lhe permita testar as funções que desenvolveu.

Problema 6

Uma resistência é um componente passivo de 2 terminais utilizado na maioria dos circuitos eletrónicos. Implemente um conversor que transforme um código de cores no valor da resistência em Ω. Para tal, complete o código disponível em **prob6.c**, em particular o procedimento seguinte:

```
1 int converte_codigo_cores(int cor, int pos, float *ret);
```

Este procedimento tem como argumentos um número inteiro representando a cor de cada banda da resistência, um inteiro que corresponde à posição dessa cor (1ª, 2ªou 3ªbanda) e um apontador para **float** contendo o valor atual da resistência. Não é preciso considerar a tolerância. Cada cor é representada por um número inteiro de 0 a 9, de acordo com https://www.digikey.sg/en/resources/conversion-calculators/conversion-calculator-resistor-color-code-4-band, reproduzido em seguida:

0 - Black; 1 - Brown; 2 - Red; 3 - Orange; 4 - Yellow; 5 - Green; 6 - Blue; 7 - Violet; 8 - Grey; 9 - White.

A função deverá ainda retornar 1 em caso de sucesso e -1 em caso de erro (valor da posição ou da resistência inválidos).

Exemplo

```
1 Introduza a cor da banda 1: 0
2 Introduza a cor da banda 2: 1
3 Introduza a cor da banda 3: 2
4 Valor da resistencia: 100.00 Ohms
```

Problema 7

Escreva um programa capaz de ler uma frase (**string**) introduzida pelo utilizador e imprima essa string invertida. Para isso, implemente a seguinte função:

```
1 void inverte(char *strOriginal, char *strInvertida);
```

O primeiro argumento da função representa a string original, enquanto no segundo argumento é devolvida a string invertida.

Exemplo

```
1 Escreva uma frase: Programacao 1 e divertido!
2 A frase invertida e: "!oditrevid e 1 oacamargorP".
```

Problema 8

Escreva e teste a função **conta** que recebe uma frase e uma palavra e retorna quantas vezes essa palavra aparece na frase.

```
1 int conta(char *frase, char *palavra);
```

Utilize esta função em um programa que permita ao utilizador testar várias frases e palavras até que o "." (ponto final) seja introduzido.

Nota: Palavras escritas com letras maiúsculas ou minúsculas devem ser consideradas diferentes.

Exemplo

```
1 Introduza uma frase: A Ana foi ao cinema e Ao parque
2 Qual palavra desejas procurar? ao
3 A palavra "ao" aparece 1 vez.
```

Problema 9

Escreva um programa que determina se uma palavra introduzida pelo utilizador é capicua. O programa deve permitir ao utilizador testar o número de strings arbitrário e apenas termina com "Ctrl+D". Para tal, implemente uma função com o seguinte cabeçalho:

```
1 int capicua(char * str);
```

A função deve retornar 1 se a string **str** é capicua e 0 se não for capicua.

Exemplo

```
1 Introduza uma palavra: Programa
2 A palavra "Programa" nao e capicua.
3 Introduza uma palavra: ana
4 A palavra "ana" e capicua.
5 Introduza uma palavra: .
```

Problema 10

Escreva um programa que pede ao utilizador para escrever uma frase e apresenta no terminal: quantas palavras constituem a frase; a palavra de maior comprimento; e o comprimento médio das palavras.

Exemplo

```
1 Introduza uma frase: O jornal de hoje tem na capa uma fotografia interessante
2 Quantidade de palavras: 10
3 Palavra maior: interessante
4 Comprimento medio: 4.7
```