Programação em C - Exercícios sobre Estruturas

Ligações úteis:

- Slides de *Visão geral da linguagem C* disponibilizados no moodle;
- C Programming Videos (Neso Academy) - Estruturas - Video 149 e seguintes

---

1. Complete o seguinte código definindo as funções:
   - `distanciaEuclideana()`, para cálculo da distância euclideana entre os pontos p1 e p2;
   - `soma()`, para calcular o vector soma dos vectores representados pelos pontos p1 e p2;
   - e completando a função `main()`.

```
#include <stdio.h>
#include <math.h>
struct Ponto{
    float x;
    float y;
};
typedef struct Ponto PONTO;

float distanciaEuclideana(PONTO p1, PONTO p2);
float distancia_total(PONTO percurso[], int size);
PONTO soma(PONTO p1, PONTO p2);

int main(){
  PONTO p1, p2, p3;
  PONTO caminho[] = {{0,0},{0,1}, {1,1},{4,5}};
  int size = 4;
  float dist;

  scanf(/*....*/); // ler coordenada x e y de um ponto para p1
  scanf(/*....*/); // ler coordenada x e y de um ponto para p2

  dist = distanciaEuclideana(p1,p2);
  p3 = soma(p1,p2);

  printf (/*...*/); // imprimir distância euclideana entre os pontos p1 e
p2
  printf (/*...*/); // imprimir vector soma dos vectores representados
pelos pontos p1 e p2

  dist = distancia_total(caminho, size);
```

```
  printf (/*...*/); // imprimir distância euclideana total dos segmentos do
percurso

  return 0;
}
```

Seguidamente defina uma função: `int distancia_total(PONTO percurso[], int size)` que retorna a distância total de um percurso representado por uma sequência de pontos.

2. *Frações*

- Defina uma estrutura `Frac` (e um novo tipo `FRAC`, usando `typedef`) que seja adequada para conter uma fração com numerador e denominador inteiros. Deve ser incluída a seguinte informação: sinal, numerador (positivo), denominador (positivo), indicação de erro (e.g., devido a divisão por zero). Por exemplo, -1/4 é representado por:

  Sinal : -1, Num: 1, Den: 4, Erro: 0 (não tem erro)

- Defina uma função com o protótipo

  ```
  FRAC simp(FRAC f);
  ```

  que retorna o argumento simplificado.
- Defina uma função com o protótipo

  ```
  FRAC soma(FRAC, FRAC);
  ```

  que retorna a soma dos dois argumentos.
- Defina uma função com o protótipo

  ```
  FRAC sub(FRAC, FRAC);
  ```

  que retorna a diferença dos dois argumentos.
- Defina uma função com o protótipo

  ```
  FRAC mult(FRAC, FRAC);
  ```

  que retorna o produto dos dois argumentos.
- Defina uma função com o protótipo

  ```
  FRAC div(FRAC, FRAC);
  ```

que retorna o quociente do primeiro argumento pelo segundo.

- Escreva um programa que permita testar as funções definidas anteriormente, da seguinte forma: Existe uma fração inicial **resultado**, com o valor 0. O utilizador dá um dos seguintes comandos:
  - **+**: o utilizador dá uma fração que é adicionada a `resultado`.
  - **-**: o utilizador dá uma fração que é subtraída a `resultado`.
  - *****: o utilizador dá uma fração que é multiplicada pela fração em `resultado`.
  - **/**: o utilizador dá uma fração pela qual a fração em `resultado` será dividido.

  Cada uma das operações anteriores deverá apresentar o resultado simplificado.

3. Um vector `lista[]` de nomes e telefones, tem a seguinte estrutura, e é inicializada da forma indicada:

```
#define MAX_LISTA 1000
struct pessoa{
    char* nome;
    char* telefone;
};
typedef struct pessoa PESSOA;

PESSOA lista[MAX_LISTA] = {{"rui", "226664441"},
              {"ana", "214444444"}};
int n = 2;  // Número de pessoas da lista
```

- Escreva uma função com o seguinte protótipo, que pesquise um nome **pal** na lista, indicando o índice onde se encontra, ou **-1** caso não exista.

```
int pesquisa(char *pal);
```

- Escreva uma função semelhante à da alínea anterior, mas que só procura nome **pal** apartir do índice **i** (parâmetro da função) até o fim da lista.

```
int pesqui(int i, char *pal);
```

- Escreva um programa que, utilizando a função `pesqui()` , leia repetidamente um nome e imprima os telefones correspondentes ou "**não existe!**".
- Altere o programa para aceitar os seguintes comandos:

```
p <nome> - imprime os telefones correspontes, ou "não existe!".
i <nome> <telefone> Insere um novo nome e telefone na lista (a não ser que
esse par já exista).
d <nome> Elimina da lista todas as ocorrências de pares (nome, telefone)
com o nome indicado.
```

```
Exemplo:
  p rui
    22666441
  i rui 888888
  p rui
    226664441
    888888
  d rui
  p rui
    não existe!
```