# Programação Imperativa 2021/2022 (CC1003), DCC/FCUP

Folha 9

**9.1** Considere a função apresentada na aula teórica 9 para contar espaços de uma cadeia de carateres. Pretende-se que escreva uma versão alternativa desta função usando apontadores em vez de índices. A função dever ter a declaração

```
int contar_espacos(char *str)
```

O resultado deve ser o número de espaços na cadeia.

**9.2** Considere a função apresentada na aula teórica 9 para inverter a ordem de carateres de uma cadeia. Pretende-se que escreva uma versão alternativa desta função usando apontadores em vez de índices. A função dever ter a declaração

```
void inverter(char *str)
```

**9.3** (**Plataforma codex**) Considere a função apresentada na aula teórica 9 para procurar um carater numa cadeia. Pretende-se que escreva uma versão alternativa desta função usando apontadores em vez de índices. A função dever ter a declaração

```
char *procurar(char *str, char ch);
```

O resultado deve ser um apontador para a primeira ocorrência do carater `ch` (se este ocorrer) ou `NULL` caso contrário.

**9.4** Considere a função apresentada na aula teórica 9 para comparar igualdade de cadeias de carateres. Pretende-se que escreva uma versão alternativa desta função usando apontadores em vez de índices. A função dever ter a declaração

```
int comparar(char *str1, char *str2);
```

O resultado deve ser 1 se as cadeias são iguais e 0 se são diferentes.

**9.5** Encontre os erros nas seguintes declarações de estruturas.

1. `struct ponto ( double x, y )`
2. `struct ponto { double x, double y };`
3. `struct ponto { double x; double y }`
4. `struct ponto { double x; double y; }`

**9.6** Encontre os erros nas seguintes declarações que envolvem o uso da instrução `typedef`.

1. `typedef struct { double x; double y } Ponto;`
2. `typedef { double x; double y; } Ponto;`
3. `typedef struct { double x; double y; };`

**9.7** Considere a seguinte estrutura que representa pontos em $\mathbb{R}^2$ (no plano):

```
typedef struct ponto{
   double x, y;
} Ponto;
```

1. Escreva uma função `double distancia(Ponto a, Ponto b)` que calcula a distância euclidiana entre dois pontos.
2. Escreva uma função `int mesmo_ponto(Ponto a, Ponto b)` que retorna "1" se dois pontos são iguais e "0" em caso contrário. Sendo que os pontos são definidos através de número vírgula flutuante com precisão finita, dois pontos são considerados iguais se a distância euclidiana entre eles é menor que 0.000001.
3. Escreva um programa que lê uma lista de 10 pontos da entrada padrão (dados pelos valores x e y) e que imprime na saída padrão o par de pontos com distância maior entre eles.
4. Defina um novo tipo de dado `Rect` para representar rectângulos com lados paralelos aos eixos em um sistema de coordenadas cartesianas. Represente os rectângulos mediante os pontos inferior esquerdo e superior direito, usando o tipo de dado `Ponto`.
5. Escreva uma função `double rect_area(Rect r)` que calcula a area de um rectângulo.
6. Escreva uma função `int ponto_dentro(Rect r, Ponto p)` que retorna "1" se o ponto `p` se encontra dentro do rectângulo `r` e "0" em caso contrário.
7. Escreva uma função `int rect_dentro(Rect r1, Rect r2)` que retorna "1" se o rectângulo `r2` está complemente contido dentro do rectângulo `r1` e "0" em caso contrário.

**9.8** Considere a seguinte estrutura que representa datas:

```
typedef struct data{
   int dia, mes, ano;
} Data;
```

Escreva uma programa que lê duas datas da entrada padrão. No seu programa, escreva também as seguintes funções:

1. Escreva uma função `int comparar(Data d1, Data d2)` que compare duas datas. A função deve retornar "0" se as datas forem iguais, "-1" se a primeira for anterior à segunda e "1" se a primeira for posterior à segunda.
2. Escreva uma função `Data diff(Data d1, Data d2)` que calcula a diferença entre duas datas (em dias, meses e anos). O resultados da função é representado com uma variável de tipo `Data`.

**9.9** Escreva um programa que lê uma lista de datas da entrada padrão. A introdução das datas é terminada quando for inserido o valor "-1" para o dia e são introduzidas no máximo 10 datas. As datas são armazenadas num vetor. No seu programa, escreva também as seguintes funções:

1. Escreva uma função `void ordena_datas(Data vec_datas[], int size)` que ordena o vetor de datas.
2. Escreva uma função `Data menor_data(Data vec_datas[], int size)` que retorna a menor data presente no vetor.
3. Escreva uma função `void datas_ano(Data vec_datas[], int size, int ano)` que imprime na saída padrão todas as datas do vetor que correspondem a um ano passado como parâmetro à função.