

# Programação Imperativa 2022/2023 (CC1003), DCC/FCUP

## Folha 7

**7.1** Escreva um programa que lê uma sequência de valores inteiros positivos terminada por  $-1$  e no final escreve toda a sequência sem repetidos.

**7.2** Escreva uma função `int calc(char str[])` que implementa uma mini-calculadora: a cadeia de caracteres dada tem sempre caracteres correspondente a algarismos decimais de `'0'` até `'9'` (pelo menos um), seguidos por um sinal de operação de adição, subtração ou multiplicação (`'+'`, `'-'`, `'*'`), seguido por mais caracteres correspondente a algarismos decimais. A função deve calcular o valor da expressão e retornar o inteiro correspondente. Exemplos: `calc("15-3")` dá 12; `calc("2*3")` dá 6.

**7.3** Defina uma função `void eliminar(char str[], char ch)` que elimina a primeira ocorrência de um carácter `ch` de uma cadeia de caracteres. Exemplo: se `str = "ABBA"`, então depois de executar `eliminar(str, 'B')` devemos ter `str = "ABA"`. Tenha o cuidado de colocar corretamente o terminador `'\0'`.

**7.4** Considere a seguinte função para inserir um valor num vetor de  $n$  elementos baseada no algoritmo usado na *ordenação por inserção*; assumimos que os valores do vetor estão por ordem crescente à entrada da função; no final, o vetor terá mais um valor e mantém a ordem ascendente.

```
void inserir(int vec[], int n, int x) {
    int j = n-1;
    while(j >= 0 && vec[j] > x) {
        vec[j+1] = vec[j];
        j--;
    }
    vec[j+1] = x;
}
```

Acrescente asserções à função para exprimir as seguintes condições de correção:

- Uma *pré-condição* à entrada da função: o vetor está ordenado.
- Uma *pós-condição* à saída da função: o vetor continua ordenado e o comprimento aumentou de 1 unidade.

**7.5** Implemente uma função `void ordenar(char str[])` que ordena os caracteres numa cadeia pelos seus códigos. Por exemplo: se `str = "ALGORITMO"` então após execução devemos ter `str = "AGILMOORT"`.

**7.6** Escreva uma função `int anagramas(char str1[], char str2[])` que determina se duas cadeias de caracteres são *anagramas*, isto é, se se escrevem com os mesmos caracteres. O resultado deve ser 1 em caso afirmativo e 0 caso contrário. Por exemplo, "deposit" e "topside" são anagramas.

**7.7** O conceito de anagrama pode ser aplicado mais geralmente a uma *frase* com várias palavras, ignorando os espaços, sinais de pontuação e a distinção entre maiúsculas e minúsculas. Por exemplo, a pergunta em Latim "Quid est veritas?" (*O que é a verdade?*) é um anagrama de "Est vir qui adest" (*É o homen que aqui está*). Modifique o programa do exercício anterior para testar anagramas neste sentido mais lato.

*Sugestão:* comece por escrever uma função auxiliar `void normalizar(char str[])` para “normalizar” uma cadeia de caracteres eliminando todos os caracteres não-letra e convertendo todas as letras em minúsculas.

**7.8** O programa para *Quicksort* apresentado na aula teórica escolhe sempre o primeiro valor da sub-sequência para *pivot*. Isto causa que a complexidade no caso em que a sequência está ordenada seja quadrática. Pretende-se melhorar esta escolha usando como *pivot* a mediana do primeiro, último e do ponto-médio da sub-sequência. Para tal, basta trocar a mediana com o valor no início da partição. Implemente esta modificação no programa do *Quicksort*.