Ficha 4

Programação Imperativa

Strings e Structs

1. Defina uma função `int minusculas (char s[])` que substitui na string `s` todas as letras maiúsculas pela correspondente letra minúscula.

   A função deverá retornar quantas substituições foram efectuadas.

2. Defina uma função `int contalinhas (char s[])` que calcula quantas linhas tem uma string (assuma que o caracter que separa duas linhas é `'\n'`).

3. Defina uma função `int contaPal (char s[])` que calcula quantas palavras tem uma string. Assuma que uma palavra é um conjunto de caracteres *não brancos* terminada por um *caracter branco*. Use para isso a função standard (`ctype.h`) `int isspace(int c)`.

4. Defina uma função `int procura (char *p, char *ps[], int N)` que procura uma string `p` num array de strings `ps`).

5. Considere o seguinte tipo para representar *stacks* de números inteiros.

   ```
   #define MAX 100
   typedef struct stack {
       int sp;
       int valores [MAX];
   } STACK;
   ```

   Defina as seguintes funções sobre este tipo:

   (a) `void initStack (STACK *s)` que inicializa uma stack (passa a representar uma stack vazia)

   (b) `int isEmptyS (STACK *s)` que testa se uma stack é vazia

   (c) `int push (STACK *s, int x)` que acrescenta `x` ao topo de `s`; a função deve retornar `0` se a operação fôr feita com sucesso (i.e., se a stack ainda não estiver cheia) e `1` se a operação não fôr possível (i.e., se a stack estiver cheia).

   (d) `int pop (STACK *s, int *x)` que remove de uma stack o elemento que está no topo. A função deverá colocar no endereço `x` o elemento removido. A função deverá retornar `0` se a operação for possível (i.e. a stack não está vazia) e `1` em caso de erro (stack vazia).

   (e) `int top (STACK *s, int *x)` que coloca no endereço `x` o elemento que está no topo da stack (sem modificar a stack). A função deverá retornar `0` se a operação for possível (i.e. a stack não está vazia) e `1` em caso de erro (stack vazia).