



UNIVERSIDADE DE ÉVORA  
CURSO DE ENGENHARIA INFORMÁTICA

---

# - SOPA DE LETRAS -

---

PROGRAMAÇÃO I  
2016/2017

[ANA SILVÉRIO N°37561]  
[JOÃO QUEIMADO N°38176]  
[VASCO VENTURA N°14452]



# - SOPA DE LETRAS -

## FASE 1: LOCALIZAÇÃO

---

### 1. INTRODUÇÃO

---

Dada uma matriz previamente gerada com letras, *input* do programa, pretende-se criar um programa que analisa a “Sopa de Letras”, localizando quaisquer palavras nela contidas. No final, o programa deverá apresentar um *display* de todas as palavras encontradas, *output* do programa, com base nas coordenadas da grelha dada. É assim que o jogo “sopa de Letras” se processa.

No entanto, o programa em si tem de ser repartido em secções, mini - funções, de forma a se obter o produto final. Sendo o objetivo da fase 1 visar essas mesmas funções, o seu funcionamento, bem como os seus argumentos e *outputs* esperados.

- Direção: [Horizontal, Vertical, Oblíqua]
- Sentido: [Norte – Sul, Oeste – Este, Noroeste – Sudeste, Nordeste – Sudoeste]

	a	b	c	d	e	f	g	h	i
1	O	H	L	E	M	R	E	V	U
2	B	B	R	A	N	C	O	Z	A
3	S	R	A	M	S	U	P	A	O
4	A	A	B	A	P	O	T	Z	Z
5	L	N	Z	R	O	E	R	N	U
6	I	U	I	E	P	D	O	I	I
7	L	O	L	L	O	R	A	C	A
8	I	T	M	O	T	E	R	P	P
9	L	I	E	A	Z	V	Y	U	U

Fig 1. *Input* do programa

---

## 2. MINI FUNÇÕES DO PROGRAMA PRINCIPAL

---

### 2.1 Função Extract

#### Funcionamento:

Extrai os valores de um ficheiro e coloca-os em listas para que possam ser manipulados mais facilmente pelo próprio programa.

#### Argumentos:

São esperados argumentos do tipo String. O argumento é **fname (FileName)**, que será o nome do ficheiro de onde os valores serão extraídos.

#### Output (Return):

É esperado um Tuplo formado por um Dicionário e uma Lista. O Dicionário contém: chaves (palavras a procurar) e valores (Lista com a posição inicial e final da palavra, sendo que inicialmente é uma lista vazia). Por outro lado, a Lista contém uma lista com as linhas do ficheiro que correspondem à Sopa de Letras (Matriz).

#### Código:

```
def Extract (fname='file.txt') :
    dic=dict()
    lst=list()
    f=open(fname)
    f.seek(0)
    for l in f:
        if l!='\n' and l!='':
            if l[:len(l)-1].isdigit():
                l=l[:len(l)-1]
                n=int(l)
            elif n>0:
                l=l[:len(l)-1]
                dic[l]=[[ -1,-1],[ -1,-1]]
                n-=1
            else:
                if l[len(l)-1].isalpha():
                    l=l[:len(l)]
                else:
                    l=l[:len(l)-1]
                lst.append(l)
    f.close()
    return dic,lst
```

## 2.2 Função Reverse

### Funcionamento:

Inverte a ordem dos caracteres de uma String.

### Argumentos:

Os argumentos esperados são do tipo String. O argumento é **st** (**String**), que corresponde à String que será invertida.

### Output (Return):

É esperada uma String inversa à String que serviu de argumento, ou seja, String inversa a **st**.

### Código:

```
def Reverse (st):  
    ls=list()  
  
    for c in st:  
        ls.append(c)  
  
    ls.reverse()  
  
    return ''.join(ls)
```

## 2.3 Função Find

### Funcionamento:

Procura uma palavra num conjunto de caracteres, ou seja, numa String.

### Argumentos:

Os argumentos da função são do tipo String. O argumento é **wrd** (**Word**), e é a palavra a procurar, que tinha sido inserida na função anterior de argumento **st**.

### Output (Return):

É esperado um valor inteiro que corresponde à posição inicial da palavra encontrada na String (**wrd**), **c**. No caso de a palavra não ser encontrada, o programa deve fazer *return -1*.

### Código:

```
def Find (wrd,st):  
    for c in range(len(st)):  
        if len(st)-c >= len(wrd):  
            if wrd == st[c:c+len(wrd)]:  
                return c  
  
    return -1
```

## 2.4 Função Transpos

### Funcionamento:

Transforma uma matriz na sua transposta. Ou seja, troca as linhas com as colunas das posições de uma matriz inicial.

### Argumentos:

A função tem argumentos do tipo Lista. O argumento é **lst (Lista)**, que corresponde a uma matriz gerada por listas de Strings a transpor.

### Output (Return):

É esperada uma lista de Strings onde cada coluna da matriz inicial corresponde a cada linha da nova matriz.

### Código:

```
def Transpos (lst):  
    rl=len(lst)*['']  
    for c in range(len(lst)):  
        for l in lst:  
            rl[c]+=l[c]  
    return rl
```

## 2.5 Função DiagSE:

### Funcionamento:

Retorna uma matriz onde as linhas correspondem as linhas correspondem ás linhas diagonais de outra matriz (do canto superior esquerdo para o canto inferior direito), da matriz transposta.

### Argumentos:

São esperados argumentos do tipo Lista. O argumento é **lst (Lista)**, que corresponde a uma matriz de Strings a converter em palavras.

### Output (Return):

Espera-se uma lista de Strings onde cada linha corresponde á sequencia diagonal da lista (matriz) inserida.

### Código:

```
def DiagSE(lst):
    rlst=list()
    for c in range(len(lst)):
        st=''
        x=0
        y=c
        while True:
            l=lst[y]
            st+=l[x]
            x+=1
            y+=1
            if x>=len(lst[y-1]) or y>=len(lst):
                break
        rlst.append(st)
    rlst.reverse()

    for c in range(1,len(lst)):
        st=''
        x=c
        y=0
        while True:
            l=lst[y]
            st+=l[x]
            x+=1
            y+=1
            if x>=len(lst[y-1]) or y>=len(lst):
                break
        rlst.append(st)
    return rlst
```

## 2.6 Função DiagSW:

### Funcionamento:

Retorna uma matriz onde as linhas correspondem as linhas correspondem às linhas diagonais de outra matriz (do canto superior esquerdo para o canto inferior esquerdo), da matriz transposta.

### Argumentos:

São esperados argumentos do tipo Lista. O argumento é **lst (Lista)**, que corresponde a uma matriz de Strings a converter em palavras.

### Output (Return):

É de esperar uma lista de Strings onde cada linha corresponde á sequencia diagonal da lista (matriz) inserida.

### Código:

```
def DiagSW(lst):
    rlst=list()
    for c in range(len(lst)):
        st=''
        x=0
        y=c
        while True:
            l=lst[y]
            st+=l[x]
            x+=1
            y-=1
            if x>=len(lst[y-1]) or y<0:
                break
        rlst.append(st)

    for c in range(1,len(lst)):
        st=''
        x=c
        y=len(lst)-1
        while True:
            l=lst[y]
            st+=l[x]
            x+=1
            y-=1
            if x>=len(lst[y-1]) or y<0:
                break
        rlst.append(st)
    return rlst
```



## 2.7 Função DataForm:

### Funcionamento:

Formata um conjunto de coordenadas para que a sua leitura seja fácil e direta pelo utilizador. Convertendo, depois, essas mesmas coordenadas em direções que têm como referencia os Pontos Cardeais.

### Argumentos:

São esperados argumentos do tipo Lista. O argumento é **data (Data)**, que corresponde a uma lista de coordenadas fornecidas pela grelha em que a sua nomenclatura se baseia nas colunas por Letras e as linhas por números inteiros positivos.

### Output (Return):

Espera-se uma lista com as coordenadas formatadas e a direção correspondente ao Ponto Cardinal respetivo.

### Código:

```
def DataForm(data):
    lst=list()

    for c in data:
        lst.append(chr(c[0]+64)+str(c[1]))

    c1=data[0]
    c2=data[1]
    x=c1[0]-c2[0]
    y=c1[1]-c2[1]

    if x==0 and y>0:
        lst.append('norte')
    elif x==0 and y<0:
        lst.append('sul')
    elif x<0 and y==0:
        lst.append('este')
    elif x>0 and y==0:
        lst.append('oeste')
    elif x<0 and y>0:
        lst.append('nordeste')
    elif x<0 and y<0:
        lst.append('sudeste')
    elif x>0 and y>0:
        lst.append('nordoeste')
    else:
        lst.append('sudoeste')

    return lst
```