UNIVERSIDADE DE ÉVORA

CURSO DE ENGENHARIA INFORMÁTICA

- SOPA DE LETRAS -

PROGRAMAÇÃO I
2016/2017

[ANA SILVÉRIO Nº37561]
[JOÃO QUEIMADO Nº38176]
[VASCO VENTURA Nº14452]

- SOPA DE LETRAS -

FASE 2

1. INTRODUÇÃO

Como última fase do trabalho, considerou-se implementar o programa Gerar Matriz que irá gerar uma Matriz composta por letras e o Programa Principal deverá localizar as palavras, a sua direção e as suas coordenadas nessa mesma matriz.

Nesta fase temos a explicação em detalhe do Programa Principal e dos seus constituintes, das pequenas funções que o põem a "correr".

```
>>>
Operações:
 1-Encontrar
 2-Gerar
 3-sair
1
Nome do ficheiro:Test
-----------------
azul : F5 - I2 : nordeste
verde : I8 - I4 : norte
branco : H8 - C3 : nordoeste
preto : A2 - A6 : sul
vermelho : H1 - A1 : oeste
rosa : C6 - F3 : nordeste
amarelo : B2 - B8 : sul
cinza : C2 - G2 : este
lilas : I9 - E9 : oeste


Operações:
 1-Encontrar
 2-Gerar
 3-sair
```

Fig 1. *Output* do Programa Principal

2. PROGRAMA GERAR MATRIZ

Formado por 11 funções incluindo a função principal que tem como *output* a matriz final gerada. As funções recorrem ao método *random* para gerar posições, caracteres e direções aleatórias dando origem a uma nova matriz. Matriz esta, construída pelo utilizador.

Código Principal Do Programa Gerar Matriz:

```
def mainGerar():
    fileI=str(input('Nome do ficheiro a importar:'))+'.txt'
    fileO=str(input('Nome do ficheiro para exportar:'))+'.txt'
    size,lst=Import(fileI)
    while True:
        nxn=Position(size,lst)
        for l in nxn:
            print(l)
        resposta=input('Gerar novamente?(S \ N)').lower()
        while resposta!='s' and resposta!='n':
            resposta=input('Invalida:').lower()
        if resposta=='n':
            break
    Fill(nxn)
    Export(lst,nxn,fileO)
    print('Feito')
mainGerar()
```

3. MENU

Para uma melhor organização do pretendido, foi adicionado um controlo de comandos. Este controlo permite ao utilizador selecionar a opção pretendida. Gerar uma matriz ou encontrar palavras a partir de uma matriz previamente gerada.

```
while True:
    while True:
        print('Operações:'  )
        print(' 1-Encontrar')
        print(' 2-Gerar'    )
        print(' 3-sair'     )
        r=input()
        if r=='1'or r=='2' or r=='3':
            res=int(r)
            break
        else:
            print('Resposta invalida')
    if res==1:
        MainFind()
    elif res==2:
        MainGerar()
    else:
        break
```

```
>>>
Operações:
 1-Encontrar
 2-Gerar
 3-sair
```

4. IMPLETENTAÇÃO E TESTE DAS MINI FUNÇÕES

4.1 Função Extract

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
```

Output:

```
>>> Extract('Test.txt')
({'branco': [[-1, -1], [-1, -1]], 'verde': [[-1, -1], [-1, -1]], 'rosa': [[-1, -1], [-1, -1]],
'lilas': [[-1, -1], [-1, -1]], 'vermelho': [[-1, -1], [-1, -1]], 'preto': [[-1, -1], [-1, -1]],
'cinza': [[-1, -1], [-1, -1]], 'amarelo': [[-1, -1], [-1, -1]], 'azul': [[-1, -1], [-1, -1]]},
['ohlemrevw', 'pacinzail', 'rmouiazua', 'eaqcswzbe', 'trnonabid', 'oerpuaqyr', 'dlguwjrte', 'go
lcmuebv', 'upmvsalil'])
```

4.2 Função Reverse

Código:

```
def Reverse (st):
    ls=list()

    for c in st:
        ls.append(c)

    ls.reverse()

    return ''.join(ls)
```

Output:

```
>>> Reverse('Verde')
'edreV'
>>>
>>> Reverse('verde')
'edrev'
>>>
```

4.3 Função Find

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
```

Output:

```
>>>
>>> Find('azul', 'efwrqfwq')
-1
>>> Find('azul', 'ascceluza')
-1
>>> Find('azul', 'veegazul')
4
>>>
```

4.4 Função Transposta

Código:

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

Output:

```
>>>
>>> a,b=Extract('Test.txt')
>>> Transpos(b)
['opretodgu', 'hamarelop', 'lcoqnrglm', 'eiucopucv', 'mnisnuwms', 'rzawaajua', 'eazzbqrel', 'vi
ubiytbi', 'wlaedrevl']
>>>
```

4.5 Função DiagSE:

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
```

Output:

```
>>>
>>> DiagSE(b)
['u', 'gp', 'dom', 'ollv', 'tegcs', 'erruma', 'ranpwul', 'pmqoujei', 'oaocnarbl', 'hcusaqtv',
'liiwbye', 'enazir', 'mzzbd', 'raue', 'eia', 'vl', 'w']
>>>
```

4.6 Função DiagSW:

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
```

Output:

```
>>>
>>> DiagSW(b)
['o', 'ph', 'ral', 'emce', 'taoim', 'orqunr', 'dencize', 'glrosaav', 'uogpnwziw', 'pluuazul',
'mcwabba', 'vmjqie', 'suryd', 'aetr', 'lbe', 'iv', 'l']
>>>
```

4.7 Função DataForm:

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
```

Output:

```
>>>
>>> DataForm([[2,1],[4,5]])
['B1', 'D5', 'sudeste']
>>>
```

5. PROGRAMA PRINCIPAL

```
#FindMain
def MainFind():
    while True:
        filename=input('Nome do ficheiro:')+'.txt'
        chk=Extract(filename)
        if chk!=(dict(),list()):
            break
    palavras,matriz=chk
    matrizDiagSw=DiagSW(matriz)
    matrizDiagSe=DiagSE(matriz)
    matrizTransposta=Transpos(matriz)

    for p in palavras:
        reverseP=Reverse(p)
        cords=palavras[p]
        le=len(matriz)
        for c in range(1,le*2):
            if c<le+1:

                n=Find(p,matriz[c-1])+1
                if n!=0:
                    cords=[[n,c],[n+len(p)-1,c]]
                    break

                n=Find(reverseP,matriz[c-1])+1
                if n!=0:
                    cords=[[n+len(p)-1,c],[n,c]]
                    break

                n=Find(p,matrizTransposta[c-1])+1
                if n!=0:
                    cords=[[c,n],[c,n+len(p)-1]]
                    break
            if c<=le:
                x=n
                y=le-c+n
            else:
                x=c-le+n
                y=n
            cords=[[x+len(p)-1,y+len(p)-1],[x,y]]
            break

        n=Find(p,matrizDiagSw[c-1])+1
        if n!=0:
            if c<=le:
                x=n
                y=c-n+1
            else:
                x=c-le+n
                y=le-n+1
            cords=[[x,y],[x+len(p)-1,y-len(p)+1]]
            break

        n=Find(reverseP,matrizDiagSw[c-1])+1
        if n!=0:
            if c<=le:
                x=n
                y=c-n+1
            else:
                x=c-le+n
                y=le-n+1
            cords=[[x+len(p)-1,y-len(p)+1],[x,y]]
            break

    palavras[p]=cords

print('------------------')
for p in palavras:
    lst=DataForm(palavras[p])
    print(p,':',lst[0],'-',lst[1],':',lst[2])
print('\n')
```