



# Entorns de Desenvolupament


## Sopa de lletres exercici 4



AVESTRUZ




BITRE




CAMALEÓN


O	C	A	M	E	L	L	O	K	M
S	T	V	I	N	T	W	S	U	P
C	I	E	Y	B	U	I	T	R	E
A	F	S	H	E	J	I	K	O	M
O	U	T	A	Q	P	E	R	Z	I
Ñ	W	R	B	T	C	O	B	R	A
O	L	U	G	U	C	F	A	N	Y
H	E	Z	S	G	A	C	E	L	A
I	V	D	O	J	X	U	K	L	A
G	E	C	A	M	A	L	E	O	N



CAMELLO



COBRA



GACELA



## Índex

<b>Enunciat.....</b>	<b>2</b>
<b>Solució.....</b>	<b>3</b>
Acció reverseWord().....	3
Modificació exercici 3.....	3

## Enunciat

En els exercicis anteriors, s'han centrat els esforços en definir la sopa de lletres: omplir la quadrícula i amagar-hi les paraules. Un cop feta aquesta feina, és el moment d'abordar la resolució del joc pròpiament dit, és a dir, trobar-hi les paraules. El següent algorisme implementa una possible solució. Per a això, llegeix de l'entrada estàndard una seqüència que conté la "sopa", seguida de la llista de paraules a trobar.

El format és:

```
< N M
    c11 c12 c13 c14 ... c1M
    c21 c22 c23 c24 ... c2M
    ...
    cN1 cN2 cN3 cN4 ... cNM
W word1 word2 word3 ... wordw
>
```

On

- $N$  i  $M$  són les dimensions de la quadrícula que conté la sopa de lletres.
- $c_{ij}$  és el caràcter  $i$ -èsim de la sopa de lletres, en majúscules.
- $W$  és el nombre de paraules a buscar dins la sopa.
- $word_i$  és la paraula  $i$ -èsima que cal cercar a la sopa de lletres, **en majúscules**.

L'algorisme escriu la quadrícula original, però substituint les lletres que no pertanyen a cap paraula per un punt. El format de sortida, doncs, seria el següent:

```
< N M
    s11 s12 s13 s14 ... s1M
    s21 s22 s23 s24 ... s2M
    ...
    sN1 sN2 sN3 sN4 ... sNM
>
```

On:

- $N$  i  $M$  són les dimensions de la quadrícula, els mateixos valors llegits a l'entrada.
- $s_{ij}$  és el caràcter  $ij$ -èsim de la sopa de lletres (un punt o una lletra d'una wordi).

Per exemple, amb la següent entrada:

```
< 3 3
    U R T
    M O A
```

```

W E C
1
UOC >
La sortida serà:
< 3 3
U . .
. O .
. . C >

```

Es demana que codifiqueu en llenguatge C l'algorisme següent:

```

const
    MAXROWS: enter = 50;
    MAXCOLUMNS: enter = 50;
    MAXLENGTH: enter = 15;
    POINT: caracter = '.';

fconst
tipus
    tLetterSoup = tupla
        letters: taula [MAXROWS,MAXCOLUMNS] de caracter;
        nRows: enter;
        nColumns: enter;

        ftupla
    tString = tupla
        letters: taula [MAXLENGTH] de caracter;
        length: enter;

        ftupla
ftipus

algorisme letterSoup
    var
        i,n,m,w, x, y, dirX, dirY: enter;
        inputSoup: tLetterSoup;
        outputSoup: tLetterSoup;
        currentWord: tString;
        found: boolea;

    fvar
        inputSoup.nRows:= readInteger();
        inputSoup.nColumns:= readInteger();
        readInputSoup( inputSoup );outputSoup.nRows:= inputSoup.nRows;
        outputSoup.nColumns:= inputSoup. nColumns;
        initOutputSoup( outputSoup );
        w:= readInteger();

```

```

per i:= 1 fins w fer
    currentWord:= readString();
    found:= fals;
    x:= 1;
    mentre x <= inputSoup.nRows i no found fer
        y:= 1;
        mentre y <= inputSoup.nColumns i no found fer
            lookForWord(inputSoup,currentWord,x,y,dirX,dirY,found);
            y:=y+1;
        fmentre
            x:=x+1;
    fmentre
    si found llavors
        putWord( outputSoup, currentWord, x-1, y-1, dirX, dirY );
    fsi
fper
    writeSoup(outputSoup);
falgorisme

```

```

accio readInputSoup( entsor soup: tLetterSoup )
    var
        i,j: enter;
        c: caracter;
    fvar
    per i:= 1 fins soup.nRows fer
        per j:= 1 fins soup.nColumns fer
            c:= readCharacter();
            mentre no isUpperCaseLetter(c) fer
                c:= readCharacter();
            fmentre
                soup.letters[i,j]:= c;
        fper
    fper
faccio

```

```

accio initOutputSoup( entsor soup: tLetterSoup )
    var
        i,j: enter;
    fvar
    per i:= 1 fins soup.nRows fer
        per j:= 1 fins soup.nColumns fer
            soup.letters[i,j]:= POINT;

```

```

fper
fper
faccio

funcio isUpperCaseLetter( c: caracter )
    retorna c >= 'A' i c <= 'Z';
ffuncio

funcio readString(): tString
    var
        string: tString;
        c: caracter;
    fvar
        c:= readCharacter();
        mentre no isUpperCaseLetter(c) fer
            c:= readCharacter();
        fmentre
            string.length:= 0;
            mentre isUpperCaseLetter(c) fer
                string.length:= string.length+1;
                string.letters[string.length]:= c;
                c:= readCharacter();
            fmentre
                retorna string;
ffuncio

accio lookForWord( ent inputSoup: tLetterSoup, ent word: tString, sor x,y: enter, sor dirX,dirY:
enter, sor found: boolea )
    found:= fals;
    dirX:= -1;
    mentre dirX <= 1 i no found fer
        dirY:= -1;
        mentre dirY <= 1 i no found fer
            checkWord(inputSoup, word, x, y, dirX, dirY, found);
            dirY:= dirY + 1;
        fmentre
            dirX:= dirX + 1;
        fmentre
            dirX:= dirX - 1;
            dirY:= dirY - 1;
faccio

```

**accio** checkWord( **ent** soup: tLetterSoup, **ent** currentWord: tString, **ent** x: **enter**, **ent** y: **enter**,  
**ent** dirX: **enter**, **ent** dirY: **enter**, **sor** found: **boolea**)

```
var
    i: enter;
    match, inBounds: boolea;
fvar
match:= cert;
inBounds:= (x >= 1) i (x <= soup.nRows) i (y >=1) i (y <= soup.nColumns);
i:= 1;
mentre i <= currentWord.length i inBounds i match fer
    match:= currentWord.letters[i] = soup.letters[x,y];
    x:= x+dirX;
    y:= y+dirY;
    inBounds:= (x >= 1) i (x <= soup.nRows) i (y >=1) i (y <= soup.nColumns);
    i:= i + 1;
fmentre
found:= match i i >currentWord.length;
```

**facció**

**accio** putWord(**entsor** soup: tLetterSoup, **ent** currentWord: tString, **ent** x: **enter**, **ent** y: **enter**,  
**ent** dirX: **enter**, **ent** dirY: **enter** )

```
var
    i: enter;
fvar
i:= 1;
mentre i <= currentWord.length fer
    soup.letters[x,y]:= currentWord.letters[i];
    x:= x+dirX;
    y:= y+dirY;
    i:= i + 1;
fmentre
```

**faccio**

**accio** writeSoup( **ent** soup: tLetterSoup )

```
var
    i,j: enter;
fvar
writeInteger( soup.nRows );
writeInteger( soup.nColumns );
writeCharacter( CarriageReturn ); { Traduiu com a printf("\n"); }
per i:= 1 fins soup.nRows fer
    per j:= 1 fins soup.nColumns fer
```

```

writeChar( soup.letters[i,j] );
fper
writeCharacter( CarriageReturn ); { Traduiiu com a printf("\n"); }
fper
faccio
A.
```





## Solució

L'arxiu **main.c** amb el codi de la sopa de lletres es pot trobar en el repositori de github dins el [feature/ej5](#).