



Entorns de Desenvolupament

Sopa de lletres exercici 2




AVESTRUZ


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




BUITRE



COBRA



CAMALEÓN



GACELA

Alumnos: José María Lagreca, Daniel Sánchez

Profesora: José Luís Luján



Índex

Enunciat.....	2
Solució.....	3



Enunciat

Utilitzant quan calgui els tipus definits en l'exercici anterior, declareu les accions i funcions especificades. No és necessari dissenyar-les.

- A. Declareu una acció/funció `getCharacter` que a partir d'una sopa `tSoup`, retorni el caràcter que es troba en una fila i columna donades.
- B. Declareu una acció/funció `setCharacter` que col·loqui un caràcter donat en una determinada posició d'una `tSoup`.
- C. Declareu una acció/funció `setWord` que col·loqui una `tWord` en una determinada posició d'una `tSoup`, i amb una orientació donada. A més, la paraula és marcada com a “no trobada” i inicialitzada amb la seva posició i orientació.
- D. Declareu una acció/funció `initSoup` que inicialitzi una sopa `tSoup` amb les dimensions donades.
- E. Declareu una acció/funció `readWord` que llegeixi i retorni una paraula `tWord` de l'entrada estàndard.

Solució

funcio getCharacter (sopa : tSoup, fila : sencer, columna : sencer) : caràcter

var

car : caràcter

fvar

si fila >= 1 i fila <= num_filas(sopa) i columna >= 1 i columna <= num_columnnes(sopa) llavors

car := sopa[fila, columna]

retorna car

sinó

retorna

fsi

ffuncio

acció setCharacter (sal sopa : tSoup, fila : sencer, columna : sencer, caràcter : caràcter)

fvar

si fila >= 1 i fila <= num_filas(sopa) i columna >= 1 i columna <= num_columnnes(sopa) llavors

sopa[fila, columna] := caràcter

finsi

facció

acció setWord (sal sopa : tSoup, paraula : tWord, fila : sencer, columna : sencer, orientació : tLloc)

var

i : sencer

longitudParaula : sencer

novaFila, novaColumna : sencer

fvar

longitudParaula := longitud(paraula.paraula)

per a := 1 fins a longitudParaula fer

novaFila := fila

novaColumna := columna

segons orientacio fer

cas horitzontal: novaColumna := columna + i - 1

cas vertical: novaFila := fila + i - 1

cas diagonal:

novaFila := fila + i - 1

novaColumna := columna + i - 1

fi_segon

```
si novaFila >= 1 i novaFila <= num_filas(sopa) i novaColumna >= 1 i novaColumna <=
num_columnnes(sopa) llavors
  caracter_de(paraula.paraula[i])
```

```
    paraula.paraula[i].horizontal := novaColumna
    paraula.paraula[i].vertical := novaFila
    paraula.paraula[i].diagonal := orientacion
finsi
fpara
```

```
    paraula.trobar := fals
    paraula.posicion.horizontal := columna
    paraula.posicion.vertical := fila
    paraula.posicion.diagonal := orientacion
```

facció

acció initSoup (sal sopa : tSoup, numFiles : sencer, numColumnnes : sencer)

var

i, j : sencer

fvar

redimensionar(sopa.sopa, numFiles, numColumnnes) // Funció hipotètica

```
    per a i := 1 fins numFiles fer
    per j := 1 fins numColumnnes fer
    sopa.sopa[i, j] := vacia
fpara
fpara
facció
```

funcio readWord () : tWord

var


```
    paraulaLeida : array [1..CARACTERS] DE caracter
    longitudLleida : sencer
    i : sencer
    novaParaula : tWord
```

fvar

llegir_cadena(paraulaLeida) // Funció hipotètica

longitudLeida := longitud(paraulaLeida)

per a i := 1 fins a CARACTERS fer



```
novaParaula.paraula[i].horitzontal := 0
novaParaula.paraula[i].vertical := 0
novaParaula.paraula[i].diagonal := horitzontal
fpara
novaParaula.posició.horitzontal := 0
novaParaula.posicion.vertical := 0
novaParaula.posicion.diagonal := horitzontal
novaParaula.trobar := FALS

torna novaParaula
ffuncio
```