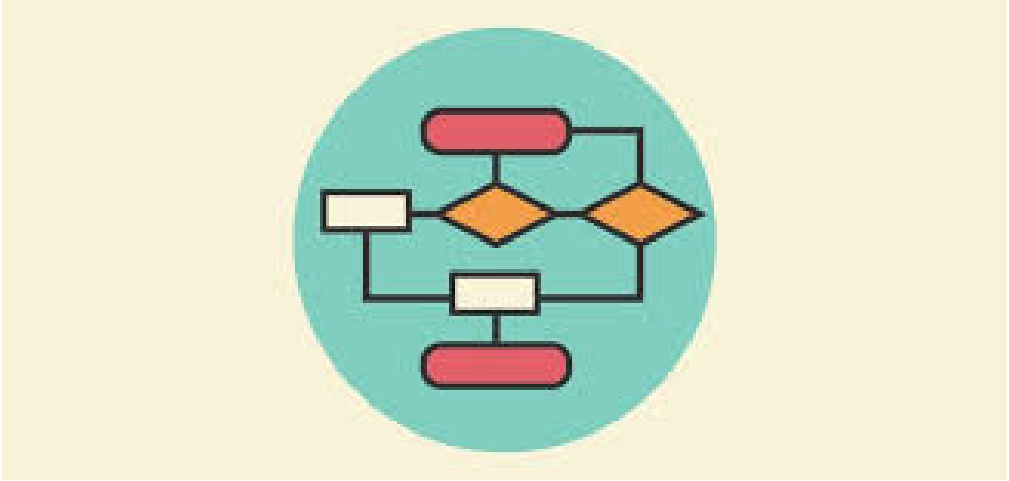
ALGORITMO



ÍNDICE

EJERCICIO 1

a.

```
tipus
        tLugar { HORIZONTAL, VERTICAL, DIAGONAL}
        tPosition: tupla
                        horizontal:  entero
                        vertical:  entero
                        diagonal: tLugar
                ftupla
ftipus
```

b.

```
const
        CARACTERS: char= 15;
fconst
tipus
        tWord: tupla
                        paraula: taula [CARACTERS] de tPosition;
                        posicion: tPosition
                        encontrar: boolean
                ftupla
ftipus
```

c.

```
tipus
        tEstat {BUIDA, OMPLERTA, PARAULA};
        tSoup: tupla
                sopa: tEstat
                paraula: tWord
                ftupla
ftipus
```

d.

```
const
        PARAULES: char[] = 15;
fconst
tipus
        tSearch:  taula[PARAULES] de tWord;
ftipus
```

e.

```
tipus
        tGame: tupla
                        sopa: tSoup
                        paraula_cercar: tWord
                ftupla
ftipus
```

EJERCICIO 2

```
FUNCION getCharacter (sopa : tSoup, fila : ENTERO, columna : ENTERO) : CARACTER
VAR
   car : CARACTER
FVAR
   SI fila >= 1 Y fila <= NUM_FILAS(sopa) Y columna >= 1 Y columna <=
NUM_COLUMNES(sopa) ENTONCES
      car := sopa[fila, columna] // Suponiendo acceso directo al carácter
      RETORNA car
   SINO
      RETORNA '' // Carácter nulo para indicar error
   FSI
FFUNCION

ACCION setCharacter (sal sopa : tSoup, fila : ENTERO, columna : ENTERO, caracter :
CARACTER)
FVAR
   SI fila >= 1 Y fila <= NUM_FILAS(sopa) Y columna >= 1 Y columna <=
NUM_COLUMNES(sopa) ENTONCES
      sopa[fila, columna] := caracter // Suponiendo acceso directo al carácter
   FINSI
FACCIÓN

ACCION setWord (sal sopa : tSoup, paraula : tWord, fila : ENTERO, columna : ENTERO,
orientacion : tLugar)
VAR
   i : ENTERO
   longitudParaula : ENTERO
   nuevaFila, nuevaColumna : ENTERO
FVAR
   longitudParaula := LONGITUD(paraula.paraula)

   PARA i := 1 HASTA longitudParaula HACER
      nuevaFila := fila
      nuevaColumna := columna
      SEGUN orientacion HACER
         CASO HORIZONTAL: nuevaColumna := columna + i - 1
         CASO VERTICAL: nuevaFila := fila + i - 1
         CASO DIAGONAL:
            nuevaFila := fila + i - 1
            nuevaColumna := columna + i - 1
      FIN_SEGUN

      SI nuevaFila >= 1 Y nuevaFila <= NUM_FILAS(sopa) Y nuevaColumna >= 1 Y
nuevaColumna <= NUM_COLUMNES(sopa) ENTONCES
```

```
            CARACTER_DE(paraula.paraula[i])

             paraula.paraula[i].horizontal := nuevaColumna
            paraula.paraula[i].vertical := nuevaFila
            paraula.paraula[i].diagonal := orientacion
        FINSI
    FPARA

    paraula.encontrar := FALSO
    paraula.posicion.horizontal := columna
    paraula.posicion.vertical := fila
    paraula.posicion.diagonal := orientacion

FACCIÓN

ACCION initSoup (sal sopa : tSoup, numFilas : ENTERO, numColumnas : ENTERO)
VAR
    i, j : ENTERO
FVAR
    REDIMENSIONAR(sopa.sopa, numFilas, numColumnas) // Función hipotética

    PARA i := 1 HASTA numFilas HACER
        PARA j := 1 HASTA numColumnas HACER
            sopa.sopa[i, j] := BUIDA
        FPARA
    FPARA
FACCIÓN

FUNCION readWord () : tWord
VAR
    palabraLeida : ARRAY [1..CARACTERS] DE CARACTER
    longitudLeida : ENTERO
    i : ENTERO
    nuevaPalabra : tWord
FVAR
    LEER_CADENA(palabraLeida) // Función hipotética

    longitudLeida := LONGITUD(palabraLeida)

        PARA i := 1 HASTA CARACTERS HACER
         nuevaPalabra.paraula[i].horizontal := 0
         nuevaPalabra.paraula[i].vertical := 0
         nuevaPalabra.paraula[i].diagonal := HORIZONTAL
    FPARA
    nuevaPalabra.posicion.horizontal := 0
    nuevaPalabra.posicion.vertical := 0
    nuevaPalabra.posicion.diagonal := HORIZONTAL
    nuevaPalabra.encontrar := FALSO
```

```
    RETORNA nuevaPalabra
FFUNCION
```