**Recursividad**

**Problema:**

Implementar un algoritmo recursivo que encuentre la salida de un laberinto: La idea es la de encontrar un camino que nos permita salir de un laberinto definido en una matriz NxN. Para movernos por el laberinto, sólo podemos pasar de una casilla a otra que sea adyacente (a la derecha, a la izquierda, atrás, adelante) y no esté marcada como una casilla prohibida (esto es, las casillas prohibidas determinan las paredes que forman el laberinto).

Algoritmo:

- Se comienza en la casilla (1,1) y se termina en la casilla (N-1,N-1)

- Nos movemos a una celda adyacente si esto es posible.

- Cuando llegamos a una situación en la que no podemos realizar ningún movimiento que nos lleve a una celda que no hayamos visitado ya, retrocedemos sobre nuestros pasos y buscamos un camino alternativo.

**Solución:**

Función resolucionLaberinto(nFila : Entero, nColumna: Entero): Logico

Var laberinto:Laberinto

Var adelante: Entero

adelante <- nFila + 1

Var atras:Entero

atras <- nFila - 1

Var derecha:Entero

derecha <- nColumna + 1

Var izquierda:Entero

izquierda <- nColumna - 1

Si laberinto[nFila][nColumna] == ‘F’ haga

Retornar verdadero

FinSi

Si laberinto[nFila][nColumna] == ‘#’ || laberinto[nFila][nColumna] == ‘\*’ haga

Retornar falso

FinSi

laberinto[nFila][nColumna] <- ‘\*’

imprimirLaberinto()

Si resolucionLaberinto(nFila, derecha)|| resolucionLaberinto(atrás, nColumna)||

resolucionLaberinto(nFila, izquierda)||resolucionLaberinto(adelante, nColumna) haga

Retornar verdadero

FinSi

imprimirLaberinto()

laberinto[nFila][nColumna] <- ‘ ‘

Retornar falso

Fin Funcion resolucionLaberinto