**Actividad**

**Ej. 15)** Para administrar una lista de personas se requiere hacer diversas acciones que permitan ese trabajo. Para almacenar los nombres se utilizará un registro que contiene dos campos, uno con un arreglo donde se guardan los nombres y otro con la cantidad actual de nombres almacenados en el arreglo (Utilice un registro con un campo arreglo y otro campo con la cantidad de datos que se han cargado al arreglo). La cantidad máxima que se podrá almacenar será de 1000 nombres. Para poder administrar la lista se deben desarrollar las siguientes acciones y/o funciones:

a) Una función llamada Vacia que reciba como parámetro el registro (con el arreglo de nombres y la cantidad de nombres cargados), y devuelva verdadero si la lista está vacía y sino debe devolver falso.

b) Una función llamada Llena que reciba como parámetro el registro (con el arreglo de nombres y la cantidad de nombres cargados), y devuelva Verdadero si el arreglo está completamente lleno (es decir si ya tiene 1000 nombres cargados) y sino debe devolver Falso.

c) Una acción que permita insertar un nombre en el arreglo. Para ello se pasará como parámetros el registro (con el arreglo de nombres y la cantidad de nombres cargados), y el nuevo nombre a insertar. El nuevo nombre se inserta siempre al final de la lista. Después de ejecutada la acción, la cantidad debe quedar incrementada en una unidad.

d) Una acción que permita suprimir al primer nombre de la lista, no importa cuál es. Simplemente suprime el primero cada vez que la acción es ejecutada. El parámetro que debe recibir es el registro (con el arreglo de nombres y la cantidad de nombres cargados). Después de ejecutada la acción, la cantidad debe quedar decrementada en una unidad.

e) Una acción que permita mostrar todos los nombres de la lista. Los parámetros que debe recibir son: el arreglo y la cantidad de nombres que tiene cargados.

Análisis

TNombres= arreglo[1..N] de Cadena

Arreglo de elementos de tipo cadena que va de 1 hasta N, para almacenar los nombres

TData: < nombres ∈ Tnombres , cant ∈ Z (0..N) >

Registro compuesto por el arreglo de nombres y un tipo entero para almacenar la cantidad de elementos cargados

N = 1000

Máximo de elementos propuesto en el problema

**Desarrollar**

**Función** Vacía(dato: registro)-> Lógico

Devuelve V si la lista esta vacía

**Función** Llena(dato: Tlista) -> Lógico

Devuelve V si la lista completamente llena (1000 elementos)

**Acción** Insertar( resultado: Tlista, dato: nombre)

Inserta un elemento siempre al final e incrementa cant.

**Acción** sumprimir( resultado: Tlista)

Borra primer elemento y disminuye cant.

**Acción** Mostrar ( dato: Tlista)

Muestra todos los elementos cargados de la lista

**Algoritmo**

**Léxico**

Max = 1000 ∈ Z

TNombres= arreglo[1.. Max] de Cadena

TData = < nombres ∈ TNombres, cant ∈ (0..Max) >

arreglo ∈ TData

nuevoNombre ∈ Cadena

**Función** Vacía(dato nom ∈ TData) → Lógico

**Inicio**

S i(nom.cant = 0 ) entonces

← verdadero

**Sino**

← falso

**Fsi**

**Ffuncion**

**Función** Llena(dato nom ∈ TData)→ Logico

**Inicio**

**Si** (nom.cant = 1000 ) **entonces**

← verdadero

**Sino**

← falso

**Fsi**

**ffuncion**

**Accion** cargarNuevoNombre (dato-resultado nom ∈ TData, dato nuevoNomb ∈ Cadena)

**Lexico Local**

msg ← Cadena

**Inicio**

**Si** (Llena(nom)) **entonces**

msg ← “el arreglo está lleno”

salida: msg

**sino**

nom.nombres[nom.cant] ← nuevoNomb

nom.cant ← nom.cant + 1

msg ← “Nuevo nombre cargado con éxito”

salida: msg

**fsi**

**faccion**

**Acción** SuprimirNombre(dato-resultado nom ∈ TData)

**Inicio**

nom.nombres[1] ← nom.nombres[nom.cant]

nom.cant ← nom.cant – 1

(ver acá: si es el primer elemento hay que correr la lista?)

O sea, está bien la solución, pero quedemos de acuerdo y documentemos.

**facción**

**Acción** Mostrar(dato nom ∈ TData )

**Léxico Local**

msg ← Cadena

**Inicio**

**Si** (Vacio(nom)) **entonces**

msg ← “el arreglo esta vacio”

**sino**

**Para**(i ← 1, i <= nom.cant, i ← i + 1) **hacer**

Salida: nom.nombres[i] // salida o mostrar?

**Fpara**

**faccion**