

UNIVERSIDAD NACIONAL DE RÍO CUARTO

FACULTAD DE CS. EXACTAS, FCO-QCAS Y NATURALES - DEPTO DE COMPUTACIÓN

ASIGNATURA:INTRODUCCIÓN A LA ALGORÍTMICA Y PROGRAMACIÓN(CÓD. 3300)

Año: 2021

PROYECTO FINAL

ALUMNOS:

BUCHIERI GIOVANNI
GIACHERO GABRIEL
PENNONE GENARO

Análisis

Al Empezar con la ejecución lo primero que se hace es invocar a la Acción “Cargar”

Análisis de la Acción Cargar

Dato: per //arreglo de Tipo TData pasado como tipo resultado.

g //ARCHIVO de TPersona con el que le daremos un nombre interno al archivo Personas.dat

Resultados: per

Relaciones y Subproblemas:

Esta acción carga la información que contiene el archivo “Personas.dat” en un arreglo.

En el caso de que algún registro del archivo tenga en su campo borrado un número distinto al 0, se ignorara ya que se lo tomara como que está eliminado.

Al terminar de ejecutar esta acción el arreglo tendrá la información de todas las personas que tengan información válida dentro del archivo.

Luego de esto se ejecutara la acción menu pasándole como parámetro el arreglo previamente cargado con los datos del Archivo:

Análisis de la Acción Menu

Datos: selec //Pedido como entrada para poder seleccionar una opcion del menu

Resultados: La opción que el usuario elija del menú

Relaciones y Subproblemas:

term ← falso

Usaremos la estructura iterativa “Mientras” con la condición de continuación “term = falso”, esto quiere decir que hasta que la variable lógica “term” no sea verdadera se continuará ciclando.

Dentro de esta estructura:

Primero se mostrará un menú (mediante la acción opciones()) la cual solo muestra un mensaje con las opciones y los correspondientes números). Luego se pedirá la entrada de la variable “selec” en la cual se va poder ingresar un valor entre el 0 al 7, ya que son 7 opciones las cuales el usuario puede elegir dentro del programa, estas son:

- 1 .Insertar al final.
- 2. Suprimir el primero.
- 3 .Mostrar todos.
- 4 .Mostrar menores.
- 5. Buscar por DNI.
- 6. Mayores al primero.
- 7. Edad mayor

- 0. Guardar y salir

Al Seleccionar la Opción que se quiera:

Usaremos la estructura “según” con las siguientes condiciones:

Cuando selección sea 1 (selec = 1): se invocará “InsertarAlFinal(personas)”

Análisis de la Acción InsetarAlFinal

Dato: per //pasado como tipo dato-resultado es un arreglo de Tipo TData

Resultado: per

Relaciones y Subproblemas:

Primero se fija que el arreglo no esté lleno mediante la función “Llena”. En caso que lo esté, se informará mediante un mensaje que no se podrá cargar la información de la persona.

En caso que no lo esté, se pedirá como entrada el nombre, DNI, edad de la persona y se la cargará al final del arreglo.

Cuando selección sea 2 (selec = 2): se invocará “SuprimirPersona(personas)”

Análisis de la Acción SuprimirPersona

Dato: per //arreglo de Tipo TData pasado como parámetro Dato-Resultado.

Resultados: per

Relaciones y Subproblemas:

Esta acción lo que nos permite es suprimir la información de la persona que se encuentra en el primer lugar del arreglo. Para hacerlo, primero:

Si el arreglo está vacío no se podrá eliminar nada, asíque se informará mediante un mensaje que no existe ningún elemento cargado. (para saber si está vacío utiliza la función “vacía” la cual devuelve verdadero si el arreglo no tiene información cargada)

Si el arreglo no está vacío, se ve la posibilidad de que per.cant = 1, si es este caso, solo se incrementara per.cant en 1 (per.cant-1)) osea que per.cant quedará igual a 0

En el caso de que per.cant sea mayor a 1 se pasa la información que existe de la última persona del arreglo al principio y decrementa el per.cant en 1 (per.cant-1)

Cuando selección sea 3 (selec = 3): Antes que nada se verificará si el arreglo se encuentra vacío (mediante la función vacía, la cual informa verdadero si lo está), en caso de estarlo se informará mediante un mensaje que no existe información cargada.

Si el arreglo no está vacío se invocará “Mostrar(personas)”

Análisis De la Acción Mostrar

Dato: per

Resultado: Mostrará la información de todas las personas que se encuentren en el arreglo.

Relaciones y Subproblemas:

Se mostrará en pantalla la información de todas las personas que se encuentran ingresadas en el arreglo “per”. Usando una estructura iterativa que vaya incrementando una variable de control hasta que se llegue al final del mismo.

Cuando selección sea 4 (selec = 4): Primero se verá si el arreglo “Personas” no está vacío, en caso de estarlo se indicará con un mensaje que no se puede ejecutar la opción ya que no existe información de personas cargados. En caso que no lo esté:

A la variable tipo puntero a TNode (que contiene 2 campos, info que será tipo TPers y next que será un campo puntero a TNode) se lo inicializará apuntando a “NIL”.

listaM ← nil

Luego se invocará a la función ListaMenores que el puntero que devuelva sera guardado en la variable listaM

listaM ← ListaMenores(personas, personas.cant, listaM)

Por último se ejecutará una acción que mostrará toda la lista previamente conseguida gracias a la función invocada.

MostrarMenores(listaM)

//MostrarMenores es una acción que da como salida la información almacenada en la lista creada anteriormente mediante la función recursiva ListaMenores

Análisis de la Función recursiva ListaMenores

Dato: per //parámetro tipo dato donde será el arreglo TData

cant //parámetro tipo dato, el cual tendrá la cantidad total de personas cargadas en el arreglo

lis //parámetro de tipo dato que apunta a la cabeza de la lista creada mediante la recursión de la acción

Resultado: puntero a TNode //la cabeza de la lista creada en la función

Relaciones y Subproblemas:

Esta función es recursiva:

- Como caso base, se tiene: (cant = 0) con lo cual devolverá “lis” que sera la variable tipo puntero que apuntará a la cabeza de la lista creada mediante la recursión.
- Como etapa inductiva se tiene (cant > 0)

Aquí existen 2 caminos, en caso que la edad de la persona que se encuentra en la posición “cant” del arreglo sea menor a 18 años: se ejecutara la acción “CargarLista” la cual recibe como parámetros tipo dato el arreglo “per” y el valor que tenga en el campo cant del arreglo se lo pasará como parámetro “cant”, también recibe como dato-resultado el puntero “lis”, que al terminar con la ejecución de la acción apuntara a la cabeza de la lista donde se almaceno la info de la persona menor de edad.

Al terminar la ejecución de la acción “CargarLista”, se volverá a invocar la función ListaMenores pero esta vez decrementándole el cant en 1 (cant-1)

En caso que la persona en la posición cant del arreglo tenga 18 años o sea mayor a 18, se invocará la función recursiva, pero decrementando el cant en 1 (cant-1)

Si selecciona la opción 5 (selec = 5), primero se comprueba si el arreglo “personas” está vacío, en caso de no estarlo, se realiza la llamada a la acción “OrdenarDNI” para ordenarlo, luego se solicita la entrada del dni a buscar, que será usado en la llamada de la función “BusquedaDNI”.

Análisis De la Acción OrdenarDNI

Dato-resultado: per

Resultado: Arreglo per ordenado

Relaciones y Subproblemas: Ordena mediante Bubblesort el arreglo, usando el campo dni

Análisis De la Acción BusquedaDNI

Dato: per y dni

Resultado: Tpersona

Relaciones y Subproblemas: Esquema de búsqueda dicotómica, el arreglo debe estar previamente ordenado.

Si el elemento NO existe lo informa, en caso de que el elemento exista en el arreglo, muestra todos los campos del elemento cuyo dni es igual al dni buscado.

Al seleccionar la opción 6 (selec = 6), se revisa que el arreglo no esté vacío y entonces es invocada la función “mayoresQueElPrimero” con el arreglo como parámetro tipo dato.

Se informa si hay 3 personas con edad mayor a la primera persona del arreglo o si no hay personas mayores al primero.

Análisis De la Acción mayoresQueElPrimero

Dato: per y dni

Resultado: Verdadero / Falso

Relaciones y Subproblemas: Se utiliza el esquema de Recorrido Parcial con marca final. recorre el arreglo de elementos incrementando un contador cuando el campo “edad” del elemento corriente es mayor que el primero. Si se encontraron 3 elementos con esta propiedad devuelve verdadero y sino devuelve falso

Si selecciona la opción 7 (selec = 7), primero se comprueba que el arreglo no esté vacío y luego se asigna en el elemento pri, los campos del primer elemento del arreglo personas, esto es necesario para invocar a la función edadMayor con los parámetros per, cant y aux.

Análisis De la Acción edadMayor

Dato: per, cant, aux

Resultado: TPers

Relaciones y Subproblemas: recursivamente compara el campo edad de aux con los elementos de la lista, hasta devolver el de mayor edad.

Por último Si selecciona la opción 0 (selec = 0):

Se ejecutara la acción “Guardar”:

Análisis De la Acción Guardar

Dato: per, f //donde per es el arreglo y f la variable con la que le daremos nombre interno al archivo.

Resultado: Almacenará en el archivo Personas.dat la información de las personas cargadas.

Relaciones y Subproblemas:

Primero se abrirá el archivo en modo escritura, osea se reescribirá todo el archivo "Personas.dat" borrando los elementos almacenados anteriormente. Dándole como nombre interno la variable "f" (Abrir("Personas.dat", f, e)).

Se inicializa una variable de control "i" en 1 ($i \leftarrow 1$) utilizada en un ciclo. Este irá ciclando hasta que la condición sea falsa, es decir hasta que ($i \leq \text{per.cant}$) sea falso. Dentro de esta estructura se irá almacenando en el archivo (mediante la primitiva "Escribir(f,aux)) la información de cada persona del arreglo. La variable que nos ayudará para esto será "aux" la cual es de tipo TPersona al igual que el archivo y en cada ciclo se le dará una nueva información de cada persona correspondiente, agregando en el campo borrado un 0 siempre ($\text{aux.borrado} \leftarrow 0$).

Definimos sobrecribir el mismo archivo.

y por último a la variable term se le asignará verdadero ($\text{term} \leftarrow \text{verdadero}$) con lo cual nos permitirá terminar con la estructura iterativa inicial culminando con el programa.

Diseño

Algoritmo TPrFinal

Léxico

Max = 1000

TPersona = <nombre \in Cadena, dni \in Z, edad \in (1..80), borrado \in Z>

TPers = <nombre \in Cadena, dni \in Z, edad \in (1..80)>

TArreglo = arreglo [1..Max] de TPers

TData = < info \in TArreglo, cant \in (0..Max) > // per.info[n].edad y per.cant

TNodo = < info \in TPers, next puntero a TNodo>

g \in archivo de TPersona //nombre interno que se le dará al archivo Personas.dat

// VARIABLES UTILIZADAS!

soporte \in TData //arreglo que tendrá la info

//Función Arreglo Vacío

Función Vacía (dato per \in TData) \rightarrow Lógico

Inicio

Si (per.cant = 0) entonces

← verdadero

Sino

← falso

fsi

ffunción

//Función Arreglo Lleno

Función Llena(dato per \in TData) \rightarrow Lógico

Inicio

Si (per.cant = 1000) entonces

← verdadero

Sino

← falso

fsi

ffuncion

//Acción que Inserta la información de una Persona en el arreglo

Acción InsetarAlFinal(dato-resultado per \in TData)

Léxico Local

persona \in TPers

msg \in Cadena

Inicio

Si (Llena(per)) entonces

msg \leftarrow "No es posible insertar la información de la persona ya que el arreglo está lleno"

Salida: msg

sino

//Ingresa la información de la persona que quieres agregar:

Entrada: persona.nombre

Entrada: persona.dni

Entrada: persona.edad

per.cant \leftarrow per.cant + 1

per.info[per.cant].nombre ← persona.nombre

per.info[per.cant].dni ← persona.dni

per.info[per.cant].edad ← persona.edad

msg ← "Información Cargada con éxito"

Salida: msg

fsi

facción

//Acción que Elimina la primera Persona del arreglo. last to first

Acción SuprimirPersona (dato-resultado per ∈ TData)

Inicio

Si (Vacía(arreglo)) entonces

msg ← "No se pudo suprimir ningún nombre, porque el arreglo está vacío"

Sino

Si (per.cant > 1)

per.info[1].nombre ← per.info[nom.cant].nombre

per.info[1].dni ← per.info[nom.cant].dni

per.info[1].edad ← per.info[nom.cant].edad

per.cant ← per.cant – 1

msg ← "Información suprimida con éxito!"

sino

per.cant <- per.cant-1 *//Si per.cant = 1 se quedara en 0*

fsi

fsi

Salida: msg

facción

//Acción que muestra la información de TODAS las personas del arreglo

Acción Mostrar (dato per ∈ TData)

Léxico Local

msg ∈ Cadena

Inicio

Para ($i \leftarrow 1, i \leq \text{per.cant}, i \leftarrow i + 1$) hacer

Salida: $\text{per.info}[i].\text{nombre}$

Salida: $\text{per.info}[i].\text{edad}$

Salida: $\text{per.info}[i].\text{dni}$

fpara

faccion

//INCISO g)

Acción Cargar (dato-resultado $\text{per} \in \text{TData}$, dato $g \in \text{archivo de TPersona}$)

Léxico Local

$\text{aux} \in \text{TPersona}$

Inicio

Abrir("Personas.dat", g, l) //Abre archivo "Personas.dat" en modo lectura dándole el nombre interno f.

$i \leftarrow 0$

Mientras ($\text{no EOF}(g)$) hacer

Leer(g, aux)

//verdadero si la info de la persona está borrada, falso no lo esta

Si ($\text{aux.borrado} = 0$) entonces

$i \leftarrow i + 1$

$\text{per.info}[i].\text{nombre} \leftarrow \text{aux.nombre}$

$\text{per.info}[i].\text{edad} \leftarrow \text{aux.edad}$

$\text{per.info}[i].\text{dni} \leftarrow \text{aux.dni}$

fsi

fmientras

Cerrar(g)

facción

//INCISO h)

Acción Guardar (dato $\text{per} \in \text{TData}$, dato $f \in \text{archivo de TPersona}$)

Léxico Local

$\text{aux} \in \text{TPersona}$

$i \in \mathbb{Z}$

Inicio

Abrir("Personas.dat", f, e) *//Reescribiendo el archivo Personas.dat*

$i \leftarrow 1$

Mientras ($i \leq \text{per.cant}$) hacer

$\text{aux.nombre} \leftarrow \text{per.info}[i].\text{nombre}$

$\text{aux.edad} \leftarrow \text{per.info}[i].\text{edad}$

$\text{aux.dni} \leftarrow \text{per.info}[i].\text{dni}$

$\text{aux.borrado} \leftarrow 0$ *// hardcoded*

 Escribir(f, aux)

$i \leftarrow i + 1$

fmientras

Cerrar(f) *//Cierra el archivo*

facción

//INCISO i)

//Esta acción la utilizara la función recursiva "ListaMenores" para poder cargar a la lista las personas mayores a 18 años

Acción CargaALista (dato $\text{per} \in \text{TData}$, $i \in (0..Max)$, dato-resultado $\text{aux} \in$ puntero a TNode)

Léxico Local

$x \in$ Puntero a TNode

Inicio

Obtener(x)

$(^x).\text{info.nombre} \leftarrow \text{per.info}[i].\text{nombre}$

$(^x).\text{info.edad} \leftarrow \text{per.info}[i].\text{edad}$

$(^x).\text{info.dni} \leftarrow \text{per.info}[i].\text{dni}$

$(^x).\text{next} \leftarrow \text{aux}$

$\text{aux} \leftarrow x$

faccion

//Antes de invocar a la función, aux debe apuntar a nil.

Función ListaMenores (dato $\text{per} \in \text{TData}$, $\text{cant} \in (0..Max)$, $\text{lis} \in$ puntero a TNode) \rightarrow puntero a TNode

Inicio

Según

(cant = 0): \leftarrow lis *//Caso base*

(cant > 0): *//Etapa inductiva*

Según

(per.info[cant].edad < 18): CargarLista(per, cant, lis)

\leftarrow ListaMenores (per, cant-1, lis)

(per.info[cant].edad >= 18):

\leftarrow ListaMenores (per, cant-1, lis)

fsegún

fsegún

ffunción

//Esta acción permitirá mostrar la información de la lista con nombre, edad y dni de los menores.

Acción MostrarMenores(dato lis \in puntero a TNode)

Léxico Local

Inicio

Según

(lis = nil): msg \leftarrow "La lista creada de menores se encuentra vacía"

Salida: msg

(lis \neq nil)

Mientras (lis \neq nil) hacer

Salida: lis.info.nombre

Salida: lis.info.edad

Salida: lis.info.dni

lis \leftarrow (^lis).next

fmientras

fsegun

faccion

//INCISO j)

//Acción utilizada en "bubblesort" para intercambiar los valores del arreglo al ordenarlo

Acción Swap (dato $x, y \in \mathbb{Z}$, dato-resultado $\text{per} \in \text{TData}$)

Léxico local

$\text{temp} \in \text{TPers}$ *//variable utilizada para poder hacer el intercambio*

Inicio

$\text{temp.nombre} \leftarrow \text{per.info}[x].\text{nombre}$

$\text{temp.edad} \leftarrow \text{per.info}[x].\text{edad}$

$\text{temp.dni} \leftarrow \text{per.info}[x].\text{dni}$

$\text{per.info}[x].\text{nombre} \leftarrow \text{per.info}[y].\text{nombre}$

$\text{per.info}[x].\text{edad} \leftarrow \text{per.info}[y].\text{edad}$

$\text{per.info}[x].\text{dni} \leftarrow \text{per.info}[y].\text{dni}$

$\text{per.info}[y].\text{nombre} \leftarrow \text{temp.nombre}$

$\text{per.info}[y].\text{edad} \leftarrow \text{temp.edad}$

$\text{per.info}[y].\text{dni} \leftarrow \text{temp.dni}$

faccion

// Acción que ordena el arreglo mediante "bubblesort"

Acción OrdenarDNI (dato-resultado $\text{per} \in \text{TData}$)

Léxico Local

$i, j \in (1..\text{per.cant}+1)$

$\text{term} \in \mathbb{Z}$ *//esta variable será utilizada para poder terminar el ciclo si ya no se necesita realizar ningún intercambio*

Inicio

$i \leftarrow \text{per.cant}$

$\text{term} \leftarrow 1$

Mientras ($i > 1$ y $\text{term} \neq 0$) hacer

$j \leftarrow 1$

$\text{term} \leftarrow 0$

Mientras ($j < i$) hacer

si ($\text{per.info}[j].\text{dni} > \text{per.info}[j+1].\text{dni}$) entonces

Swap($j, j+1, \text{per}$)

$\text{term} \leftarrow \text{term} + 1$

fsi

$j \leftarrow j+1$

fmientras

$i \leftarrow i-1$

fmientras

facción

//Acción de Búsqueda Dicotómica

Acción BusquedaDNI(dato per \in TData, dato dni \in Z)

Léxico local

k, inf, sup \in (1..Max)

msg \in Cadena

Inicio

según

//Si el dni es menor al 1er elemento o mayor al último, no se encuentra en el arreglo

((dni < per.info[1].dni) o (dni > per.info[per.cant].dni)):

msg \leftarrow "El Elemento no existe en el arreglo"

Salida: msg

//Si el dni es mayor o igual al primer elemento y

(per.info[1].dni <= dni y dni <= per.info[per.cant].dni):

inf \leftarrow 1

sup \leftarrow per.cant

mientras inf < sup hacer

k \leftarrow (inf+sup) div 2 //toma la mitad

según

(dni > per.info[k].dni): inf \leftarrow k+1 // derecha

(dni <= per.info[k].dni): sup \leftarrow k //izquierda

fsegún

fmientras

según

(per.info[inf].dni = dni): Salida: per.info[inf].nombre //elem encontrado

Salida: per.info[inf].edad

Salida: per.info[inf].dni

(per.info[inf].dni<>dni): msg \leftarrow "No se ha encontrado ninguna Persona con ese DNI"

Salida: msg // elemento no encontrado

fsegún

fsegún

ffaccion

//INCISO K)

//Antes de invocar a la función chequear que el arreglo no esté vacío.

Funcion mayoresQueElPrimero (dato per \in TData) \rightarrow Lógico

Léxico Local

i \in Z

cantMayor, edad \in Z

Inicio

i \leftarrow 1

cantMayor \leftarrow 0

pri \leftarrow per.info[i].edad

i \leftarrow i + 1

Mientras (i < per.cant+1 y cantMayor < 3) hacer

Si (per.info[i].edad > pri) entonces

cantMayor \leftarrow cantMayor + 1

fsi

i \leftarrow i + 1

fmientras

Según

(i < per.cant+1): \leftarrow verdadero // cantMayor = 3

(i = per.cant+1): // fin sec

Si (cantMayores < 3) entonces

\leftarrow falso

sino

\leftarrow verdadero

fsi

fsegun

ffuncion

//INCISO L)

{pre-cond: aux = per.info[1]}

{pre-cond: per.cant = cant}

Función edadMayor (dato per \in TData, cant \in (0..Max) ,dato aux \in TPers) \rightarrow TPers

Inicio

Según

(cant = 1): \leftarrow aux //Caso base

(cant > 1): // Etapa Inductiva

si (aux.edad < per.info[cant].edad) entonces

aux.nombre \leftarrow per.info[cant].nombre

aux.dni \leftarrow per.info[cant].dni

aux.edad \leftarrow per.info[cant].edad

fsi

\leftarrow edadMayor(per, cant-1 ,aux)

fsegun

ffuncion

// Muestra las opciones del menú, estática

Acción Opciones ()

Léxico local

msg \in cadena

Inicio

msg \leftarrow

“ MENÚ DE OPCIONES

1 .Insertar al final.

2. Suprimir el primero.

3 .Mostrar todos.

4 .Mostrar menores.

5. Buscar por DNI.

6. Mayores al primero.

7. Edad mayor

0. Guardar y salir “

Salida: msg // **Mostrar**

facción

//Accion del Menu de Opciones que permitirá al usuario elegir qué operación hacer

Acción Menu (dato: personas \in TData)

Léxico Local

term \in Lógico //variable utilizada para salir del ciclo

lista \in puntero a TNode //puntero a la cabeza de lista

msg \in Cadena //variable que utilizaremos para informar mensajes

listaM \in Puntero a TNode //variable que utilizada para lista de menores de edad

dni \in Z //variable utilizada para la entrada del DNI a buscar

mayor \in Lógico //variable utilizada para guardar lo que devolverá la función mayoresAlPrimero

pri \in TPers //var utilizada para guardar el primer campo del arreglo y comparar las edades

f \in archivo de TPersona //nombre interno que se le dará al archivo Personas.dat

selec \in (0..7) //Variable utilizada para la selección de opciones

dni \in Z //variable utilizada para la entrada del dni de la opción 5

mostrarPersona \in TPersona //variable utilizada para mostrar la información de la persona mayor

Inicio

term \leftarrow falso

opciones()

Mientras(no(term)) **hacer**

Entrada : selec

Según

(seleccion = 0): //Opción De Guardar y Salir

Guardar(personas, f)

term \leftarrow verdadero

(seleccion = 1): //Insertar Nombre Al Final del arreglo

InsertarAlFinal(personas)

(seleccion = 2): [*//Eliminar primera persona del arreglo*](#)

SuprimirPersona(personas)

(seleccion = 3): [*//Mostrar todo el arreglo*](#)

Si (no Vacía(personas)) entonces

Mostrar(personas)

sino

msg ← “No se puede Mostrar nada ya que El Arreglo Está Vacío”

Salida: msg

fsi

(seleccion = 4): [*//Mostrar info de personas menores de edad*](#)

Si (no Vacía(personas)) entonces

listaM ← nil

listaM ← ListaMenores(personas, personas.cant, listaM)

MostrarMenores(listaM)

sino

msg ← “El Arreglo Está Vacío”

Salida: msg

fsi

(seleccion = 5): [*//Buscar por DNI*](#)

Si (no Vacía(personas)) entonces

OrdenarDNI(personas)

Entrada: dni

BusquedaDNI(personas, dni)

sino

msg ← “El Arreglo Está Vacío”

Salida: msg

fsi

(seleccion = 6): [*//Informar si hay por lo menos 3 personas mayores al primero*](#)

Si (no Vacía(personas)) entonces

si (mayoresQueElPrimero(personas)) entonces

msg ← “hay 3 personas mayores al primer individuo del arreglo”

sino

msg ← “no hay 3 personas mayores al primer individuo del arreglo”

fsi

sino

msg ← “El Arreglo Está Vacío”

fsi

Salida: msg

(seleccion = 7): [//Mostrar la información de la persona con mayor edad del arreglo](#)

Si (no Vacía(personas)) entonces

pri.nombre ← personas.info[1].nombre

pri.edad ← personas.info[1].edad

pri.dni ← personas.info[1].dni

mostrarPersona <- edadMayor(personas,personas.cant, pri)

Salida: mostrarPersona.nombre

Salida: mostrarPersona.dni

Salida: mostrarPersona.edad

sino

msg ← “El Arreglo Está Vacío”

Salida: msg

fsi

fsegún

fmientras

facción

INICIO

Cargar (soporte, g)

Menú(soporte)

FIN