Universidad Nacional de Río Cuarto
Facultad de Cs. Exactas, Fco-Qcas y Naturales - Dpto de Computación
Asignatura: **INTRODUCCIÓN A LA ALGORÍTMICA Y PROGRAMACIÓN**
Código: 3300
Año: 2022

Proyecto integrador de mitad de año

Alumnos: Celina Gillo Mayer, Valentìn Cardozo, Joaquín Pablo Tissera

Sea presente el siguiente proyecto (dictado por los profesores de la materia):

Para administrar la nómina de empleados de una sucursal de un banco se requiere hacer un algoritmo (y luego un programa) con un menú de opciones que permite realizar estas tareas:

1. Alta de empleado (se carga apellido).
2. Baja de empleado (se elimina un empleado, se lo busca por apellido).
3. Modificar datos de empleado (se lo busca por apellido y se ofrece modificarlo).
4. Listar (muestra todos los apellidos almacenados).
5. Buscar a un empleado (si el apellido existe en la lista lo muestra, sino informa apellido inexistente).
6. Salir.

La cantidad máxima que se podrá almacenar será de 1000 empleados (apellidos).

Para poder administrar la lista se deben desarrollar las siguientes acciones y/o funciones:

A. Una función llamada Vacía que reciba como parámetro el registro (con el arreglo de apellidos y la cantidad de apellidos cargados), y devuelva verdadero si la lista está vacía y si no debe devolver Falso.
B. Una función llamada Llena que reciba como parámetro el registro (con el arreglo de apellidos y la cantidad de apellidos cargados), y devuelva Verdadero si el arreglo está completamente lleno (es decir si ya tiene 1000 apellidos cargados) y sino debe devolver Falso.
C. Una acción que permita insertar un apellido en el arreglo. Para ello se pasarán como parámetros el registro (con el arreglo de apellidos y la cantidad de apellidos cargados), y el nuevo apellido a insertar. El nuevo apellido se inserta siempre al final. Después de ejecutada la acción, la cantidad debe quedar incrementada en una unidad.
D. Una acción que permita suprimir un apellido de la lista. Debe ubicarlo por el apellido y eliminarlo (desplazando los otros registros). Después de ejecutada la acción, la cantidad debe quedar decrementada en una unidad.
E. Una acción que permita mostrar todos los apellidos. Los parámetros que debe recibir son: el arreglo y la cantidad de apellidos que tiene cargados.

F. Una función que permita encontrar un empleado por apellido y que devuelva la ubicación en el arreglo.
G. Una función que dados dos apellidos, devuelve Verdadero si están repetidos.

Nota: no debe haber apellidos repetidos.

Bajo el compromiso de realizar las consignas:

Consignas a entregar

La resolución del Trabajo Práctico Integrador debe incluir:

1) Un documento en formato PDF con:

a) Análisis de cada módulo (incisos a hasta g). Identificar los datos de prueba y relaciones.

Aclaración: no hace falta el análisis del problema general.

b) Diseño (algoritmo en notación algorítmica)

c) Pruebas (completar el resultado de cada uno de los 6 casos de prueba)

2) Un archivo de texto con el Programa fuente en C.

Se incluye a continuación la resolución del Trabajo Práctico, dando las respuestas a las consignas:

1) a) Análisis de cada módulo:

Inciso A (Función Vacía):

Datos: arr (TData) *// Recibe el registro como parámetro*
Resultados: (Lógico) *// Retorna un valor lógico*
Relación: La función devuelve verdadero si el campo del registro que corresponde a la cantidad de apellidos cargados es cero. En otro caso devuelve falso.

Inciso B (Función Llena):

Datos: arr (TData) *// Recibe el registro como parámetro*
Resultados: (Lógico) *// Retorna un valor lógico*
Relación: La función devuelve verdadero si el campo del registro que corresponde a la cantidad de apellidos cargados es igual al valor de la constante AMax (que vale mil). En otro caso devuelve falso.

Inciso C (Acción InsertarApellido):

Datos: apellido(Cadena), empleados (TData) // Recibe el apellido del nuevo empleado que se desea agregar y el arreglo de empleados previamente ya cargados

Resultados: empleados (TData) // Devuelve el arreglo de empleados con uno nuevo registro.
Relación: Se toma la variable **apellido** y se la coloca al final del arreglo **empleados** como una nueva componente.

Inciso D (Acción SuprimirApellido):

Datos: apellido (Cadena), empleados (TData) // Recibe el arreglo empleados y el apellido del empleado que se desea eliminar del arreglo.
Resultados: empleados(TData)
Relación: Se recorre el arreglo **empleados** y de haber coincidencia con la variable **apellido** se solicita una validación del usuario y luego se elimina.

Inciso E (Accion MostrarApellidos):

Datos: empleados (TData)
Resultados: // No tiene parámetros de tipo resultado
Relación: Se recorre el arreglo **empleados** y hace la salida de todas sus componentes

Inciso F (Función UbicarApellido):

Datos: apellido (Cadena), arr (TData) *// Recibe un apellido y el registro del arreglo (con campos; el arreglo y la cantidad cargada)*
Resultados: i (Z) *// Devuelve un entero, en particular, el iterador*
Relación: Se recorre todo el arreglo; va comparando “apellido” con los componentes del arreglo. Ni bien coincidan con algún elemento del arreglo, devuelve el índice en el cuál coincidió. Si no coincide con ningún componente, devuelve -1

Inciso G (Función ApellidosRepetidos):

Datos: apellido1 (Cadena), apellido2 (Cadena) *// Recibe dos apellidos*
Resultados: (Lógico) *// Retorna un valor lógico*
Relación: Se compara “apellido1” con “apellido2”, si son iguales devuelve Verdadero. Si no son iguales, devuelve Falso.

b) Diseño del proyecto:

Algoritmo NóminaEmpleadosBanco
Léxico

```
AMax = 1000                          // Constante; límite para que la nómina se llene
TArreglo = arreglo [1..AMax] de Cadena
TData = < arreglo ϵ TArreglo, cant ϵ (0..AMax) >

Función Vacía (dato arr ϵ TData) → Lógico
Inicio
  ← ( arr.cant = 0 )
Ffunción
```

```
Función Llena (dato arr ϵ TData) → Lógico
Inicio
  ← ( arr.cant = AMax )
Ffunción

Acción InsertarApellido (dato apellido ϵ Cadena, dato/resultado empleados ϵ TData)
Inicio
  empleados.cant ← empleados.cant + 1
  empleados.arreglo[empleados.cant] ← apellido
Facción

Acción SuprimirApellido (dato apellido ϵ Cadena, dato/resultado empleados ϵ
TData)
Léxico local
  msg ϵ Cadena
  rta ϵ (Si, No)
  i, j ϵ Z
Inicio
  i ← 1
  msg ← “Desea dar de baja el empleado “ + apellido + “? Ingrese “Si” o “No”: ”
  Salida: msg
  Entrada: rta
  Si ( rta = Si ) entonces
    Mientras ( i ≤ empleados.cant ) hacer
      Si ( apellido = empleados.arreglo[i] ) entonces
        Para ( j ← i , j ≤ empleados.cant , j ← j+1 ) hacer
          empleados.arreglo[j] ← empleados.arreglo[j+1]
        Fpara
          empleados.cant ← empleados.cant - 1
        Fsi
      Fsi
    i ← i + 1
    Fmientras
    msg ← “Baja del empleado “ + apellido + “ exitosa.”
    Salida: msg
  sino
    msg ← “Volviendo al menú principal.”
    Salida: msg
  Fsi
Facción

Acción MostrarApellidos (dato arr ϵ TData)
Léxico local
  msg ϵ Cadena
  i ϵ Z
Inicio
  msg ← “Inicio de la nómina de empleados…”
  Salida: msg
```

```
    Para ( i ← 1 , i ≤ arr.cant , i ← i+1 ) hacer
      Salida: arr.a[i]
    Fpara
    msg ← “Fin de la nómina de empleados…”
    Salida: msg
  Facción

  Acción ModificarApellido (dato posicion ϵ Z, apellidoViejo ϵ Cadena,
apellidoNuevoMod ϵ Cadena, dato/resultado lista ϵ TData)
// No forma parte de los incisos del trabajo práctico. Ayuda en la modularización del
algoritmo.
  Léxico local
    auxPosicion ϵ Z
    msg ϵ Cadena
  Inicio
    Si ( apellidoViejo = apellidoNuevoMod ) entonces          // Si se quiere modificar
un apellido por el mismo no hace la modificación y avisa que es el mismo.
      msg ← “La modificación del apellido debe ser distinta al que se quiere modificar.”
      Salida: msg
    sino
      auxPosicion ← UbicarApellido(apellidoNuevoMod, lista)
      Si ( auxPosicion = -1 ) entonces  // Siempre y cuando no se haya ubicado ningún
apellido, lo podremos modificar.
        lista.arreglo[posicion] ← apellidoNuevoMod
        msg ← “Modificación exitosa.”
        Salida: msg
      sino          // Si se encuentra que hay alguno, entonces no podemos poner
apellidos repetidos.
        msg ← “ERROR. El apellido no se puede modificar por uno que ya existe.”
        Salida: msg
      Fsi
    Fsi
  Facción

  Función UbicarApellido (dato apellido ϵ Cadena, arr ϵ TData) → Z
  Léxico local
    i ϵ Z
  Inicio
    Para ( i ← 1 , i ≤ arr.cant , i ← i + 1 ) hacer
      Si ( apellido = arr.arreglo[i] ) entonces
        ← i
      Fsi
    Fpara
    ← -1          // Si devuelve -1 es porque no hubo coincidencia. Usado como una
guarda por ser una función incompleta.
  Ffunción

  Función ApellidosRepetidos (dato apellido1, apellido2 ϵ Cadena) → Lógico
```

```
Inicio
    ← ( apellido1 = apellido2 )
Ffunción

Acción DesplegarMenuDeOpciones () // Creada y usada para el programa específicamente. No forma parte de los incisos del trabajo práctico.
Léxico local
  msg ϵ Cadena
Inicio
  msg ← "· Menú principal
          Seleccione una opción para continuar:
          1. Dar de alta un empleado (por apellido)
          2. Dar de baja un empleado (buscando por apellido)
          3. Modificar apellido
          4. Listar empleados
          5. Buscar un empleado (por apellido)
          6. Salir"
  Salida: msg
Facción

Acción DefensivaEntrada(dato/resultado entr ϵ Z)  // Resguardo contra negativos o mayores a 6. Creada y usada para el programa específicamente. No forma parte de los incisos del trabajo práctico.
Léxico local
  msg ϵ Cadena
Inicio
  Si ( entr < 1 o entr > 6 ) entonces
    Repetir
      msg ← "Debe ingresar un digito del 1 al 6"
      Salida: msg
      Entrada: entr
    Hasta que ( entr ≥ 1 o entr ≤ 6 )
  Fsi
Facción

nominaEmpleados ϵ TData          // Almacenador de la nómina
opcionEntry ϵ Z                  // Entrada y selección del menú
apellidoNuevo, bajaDeApellido, apellidoModificarViejo, apellidoModificarNuevo, apellidoBuscar ϵ Cadena   // Variables para cada opción del menú
auxRepetido ϵ Lógico             // Auxiliar/flag
posicionArreglo ϵ Z              // Auxiliar
msg ϵ Cadena                     // Variable para mensajes al usuario
i ϵ Z                            // Iterador

Inicio
opcionEntry ← 0
nominaEmpleados.cant ← 0
auxRepetido ← 0
```

```
Repetir
  DesplegarMenuDeOpciones()
  Entrada: opcionEntry
  DefensivaEntrada(opcionEntry)

  Según
    ( opcionEntry = 1 ) : msg ← “Opción 1: Dar de alta un empleado"
                          Salida: msg
                          Si ( Llena(nominaEmpleados) ) entonces
                            msg ← “No se puede agregar, la lista está llena”
                            Salida: msg
                          sino
                            msg ← “Ingrese el apellido del nuevo empleado: "
                            Salida: msg
                            Entrada: apellidoNuevo
                            Para ( i ← 1 , i ≤ nominaEmpleados.cant , i ← i + 1 ) hacer
                              auxRepetido ← ApellidosRepetidos(apellidoNuevo,
nominaEmpleados.arreglo[i])
                            Fpara
                            Si ( auxRepetido ) entonces
                              msg ← “ERROR. El apellido ya existe"
                              Salida: msg
                            sino
                              InsertarApellido(apellidoNuevo, nominaEmpleados)
                              msg ← “Alta del empleado " + apellidoNuevo + “exitosa.”
                              Salida: msg
                            Fsi

    ( opcionEntry = 2 ) : msg ← “Opción 2: Dar de baja un empleado"
                          Salida: msg
                          Si ( Vacia(nominaEmpleados) ) entonces
                            msg ← “La nómina está vacía.”
                            Salida: msg
                          sino
                            msg ← “Ingrese el apellido del empleado a dar de baja: ”
                            Salida: msg
                            Entrada: bajaDeApellido
                            posicionArreglo ← UbicarApellido(bajaDeApellido,
nominaEmpleados)
                            Si ( posicionArreglo = -1 ) entonces
                              msg ← “ERROR. El apellido no existe.”
                              Salida: msg
                            sino
                              SuprimirApellido(bajaDeApellido, nominaEmpleados)
                            Fsi
                          Fsi
```

```
    ( opcionEntry = 3 ) : msg ← “Opción 3: Modificar apellido"
                          Salida: msg
                          Si ( Vacia(nominaEmpleados) ) entonces
                            msg ← “La nómina está vacía.”
                            Salida: msg
                          sino
                            msg ← “Ingrese el apellido del empleado que quiere
modificar: "
                            Salida: msg
                            Entrada: apellidoModificarViejo
                            posicionArreglo ← UbicarApellido(apellidoModificarViejo,
nominaEmpleados)
                            Si ( posicionArreglo = -1 ) entonces
                              msg ← “ERROR. No hay coincidencia de apellidos.”
                              Salida: msg
                            sino
                              msg ← “Va a cambiar el apellido ” +
nominaEmpleados.arreglo[posicionArreglo]
                              Salida: msg
                              msg ← “Ingrese el nuevo apellido: ”
                              Salida: msg
                              Entrada: apellidoModificarNuevo
                              ModificarApellido(posicionArreglo, apellidoModificarViejo,
apellidoModificarNuevo, nominaEmpleados)
                            Fsi
                          Fsi

    ( opcionEntry = 4 ) : msg ← “Opción 4: Listar empleados"
                          Salida: msg
                          Si ( Vacia(nominaEmpleados) ) entonces
                            msg ← “La nómina está vacía.”
                            Salida: msg
                          sino
                            MostrarApellidos(nominaEmpleados)
                          Fsi

    ( opcionEntry = 5 ) : msg ← “Opción 5: Buscar un empleado"
                          Salida: msg
                          Si ( Vacia(nominaEmpleados) ) entonces
                            msg ← “La nómina está vacía.”
                            Salida: msg
                          sino
                            msg ← “Escriba el apellido que quiere buscar: "
                            Salida: msg
                            Entrada: apellidoBuscar
                            posicionArreglo ← UbicarApellido(apellidoBuscar,
nominaEmpleados)
                            Si ( posicionArreglo = -1 ) entonces
```

```
                                msg ← “El apellido " + apellidoBuscar + “no está en la
nómina.”
                                Salida: msg
                               sino
                                msg ← “El apellido " +
nominaEmpleados.arreglo[posicionArreglo] + “ está en la nómina.”
                                Salida: msg
                                msg ← “Además, corresponde a la posición ”
                                Salida: msg posicionArreglo
                               Fsi
                              Fsi

    ( opcionEntry = 6 ) : msg ← “Opción 6: Salir"
                              Salida: msg

  Fsegún

 Hasta que ( opcionEntry = 6 )
 msg ← “Salida exitosa."
 Salida: msg
Fin
```

c) Pruebas:

Modelamos las siguientes pruebas con base en lo visto en el teórico, como pruebas de escritorio. Las mismas corresponden a pruebas para el algoritmo y tienen un funcionamiento similar en el programa fuente entregado.

Inciso A:

Vacía(nominaEmpleados)

arr.cant	arr.cant = 0
0	V

arr.cant	arr.cant = 0
1	F

Inciso B:

Llena(nominaEmpleados)

arr.cant	arr.cant = AMax
1000	V

arr.cant	arr.cant = AMax
70	F

Inciso C:

apellidoNuevo	nominaEmpleados	InsertarApellido(apellidoNuevo, nominaEmpleados)
Sion	{ [Vas, Bob], 2 }	<table><tr><td>apellido</td><td>empleados</td><td>empleados.cantidad ← empleados.cantidad +1 empleados.apellido[empleados .cantidad] ← apellido</td></tr><tr><td>Sion</td><td>{ [Vas, Bob], 2 }</td><td>{ [Vas, Bob, Sion], 3 }</td></tr></table> { [Vas, Bob, Sion], 3 }

apellidoNuevo	nominaEmpleados	InsertarApellido(apellidoNuevo, nominaEmpleados)
Plop	{ [ ], 0 }	<table><tr><td>apellido</td><td>empleados</td><td>empleados.cantidad ← empleados.cantidad + 1 empleados.apellido[empleados .cantidad] ← apellido</td></tr><tr><td>Plop</td><td>{ [ ], 0 }</td><td>{ [Plop], 1 }</td></tr></table> { [Plop], 1 }

Inciso D:

<table>
<tr><th>bajaDeApellido</th><th>nominaEmpleados</th><th>SuprimirApellido(bajaDeApellido, nominaEmpleados)</th></tr>
<tr><td>Plop</td><td>{ [Vas, Plop, Bob, Sion], 4 }</td><td>
<table>
<tr><td>apellido</td><td>empleados</td><td>empleado.apellido = apellido.
empleados.apellido[empleados.cantidad] ⟵ apellido.
empleados.cantidad ⟵ empleados.cantidad - 1.</td></tr>
<tr><td>Plop</td><td>{ [Vas, Plop, Bob, Sion], 4 }</td><td>{ [Vas, Bob, Sion], 3 }</td></tr>
</table>
{ [Vas, Bob, Sion], 3 }
</td></tr>
</table>

<table>
<tr><th>bajaDeApellido</th><th>nominaEmpleados</th><th>SuprimirApellido(bajaDeApellido, nominaEmpleados)</th></tr>
<tr><td>Vas</td><td>{ [Vas], 1 }</td><td>
<table>
<tr><td>apellido</td><td>empleados</td><td>empleado.apellido = apellido.
empleados.apellido[empleados.cantidad] ⟵ apellido.
empleados.cantidad ⟵ empleados.cantidad - 1.</td></tr>
<tr><td>Vas</td><td>{ [Vas], 1 }</td><td>{ [ ], 0 }</td></tr>
</table>
{ [ ], 0 }
</td></tr>
</table>

Inciso E:

<table>
<tr><th>MostrarApellido(nominaEmpleados)</th></tr>
<tr><td>
<table>
<tr><th>nominaEmpleados</th></tr>
<tr><td>
<table>
<tr><th>nominaEmpleados.arreglo</th><th>nominaEmpleados.cant</th></tr>
<tr><td>[Bob, Plop, Vas]</td><td>3</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

c	i	i < arr.cant	Salida: apellido[i]
1 (1) 2 (5) 3 (9)	1 (2) 2 (6) 3 (10)	V (3) V (7) V (11)	Bob (4) Plop (8) Vas (12)

MostrarApellido(nominaEmpleados)

nominaEmpleados

nominaEmpleados.arreglo	nominaEmpleados.cant
[Bob, Sion]	2

c	i	i < arr.cant	Salida: apellido[i]
1 (1) 2 (5)	1 (2) 2 (6)	V (3) V (7)	Bob (4) Sion (8)

Inciso F:

UbicarApellido(apellidoUbicar, nominaEmpleados)

apellido	arr
Sion	arr.arreglo: [1] : Vas [2] : Bop [3] : Sion arr.cant: 3

c	i	i ≤ arr.cant	apellido=arr.arreglo[i]
- 1 (3) 2 (7)	1 (1) 2 (5) 3 (9)	V (2) V (6) V (10)	- F (4) F (8) V (11) ←i (12)

3 (13)

UbicarApellido(apellidoUbicar, nominaEmpleados)

apellido	arr
Sion	arr.arreglo: [1] : Vas [2] : Sion [3] : Bop arr.cant: 3

c	i	i ≤ arr.cant	apellido=arr.arreglo[i]
- 1 (3) 2 (7)	1 (1) 2 (5)	V (2) V (6)	- F (4) V (8) ←i (9)

2 (10)

UbicarApellido(apellidoUbicar, nominaEmpleados)

apellido	arr
Sion	arr.arreglo: [1] : Vas [2] : Fen [3] : Bop arr.cant: 3

c	i	i ≤ arr.cant	apellido=arr.arreglo[i]
- 1 (3) 2 (7) 3 (11)	1 (1) 2 (5) 3 (9) 4 (13)	V (2) V (6) V (10) F (14) ←-1 (15)	- F (4) F (8) F (12)

-1 (16)

Inciso G:

apellidoNuevo	nominaEmpleados.arreglo[1]	ApellidosRepetidos(apellidoNuevo, nominaEmpleados.arreglo[1])
Sion	Vas	apellido 1 | apellido 2 | apellido1 = apellido2 Sion | Vas | ←F F

apellidoNuevo	nominaEmpleados.arreglo[1]	ApellidosRepetidos(apellidoNuevo, nominaEmpleados.arreglo[1])
Bop	Bop	apellido 1 | apellido 2 | apellido1 = apellido2 Bop | Bop | ←V V

Siendo este el fin del trabajo, damos un cierre comentando que lo que se trabajó se vió afianzado por lo visto en prácticos, teóricos y laboratorios de la materia; no fue necesario utilizar bibliografía extra para hacer este trabajo práctico.

Por último, queremos aportar un comentario para próximos trabajos similares: Dentro de lo posible, y si los conocimientos que se hayan llegado a adquirir ese mismo año son suficientes, estaría muy bueno poder almacenar la lista que se haya creado en algún archivo de texto para que no se pierda.