PRÁCTICA

CURSO	:	***Algorítmica II***
No.	:	*Práctica Dirigida No.06*
TEMA	:	*Lista Enlazada Simples – Alumnos.*
DURACIÓN ESTIMADA	:	*01:40 horas.*

I. OBJETIVOS

La presente práctica tiene por objetivos:

- Utilizar oLoop para la especificación de pseudocódigo.
- Desarrollar una solución aplicando lista enlazada simple.

II. RESUMEN

Debe desarrollar una aplicación en oLoop que se encargue de administrar una lista enlazada simple de alumnos. Considere que debe la clase **CNodo** que contendrá la información y el apuntador del siguiente elemento de la lista.

III. CONSTRUCCIÓN DE LA APLICACIÓN

1. Crear la Clase CNodo

Iniciamos la solución declarando la clase CNodo que representa la información relevante del nodo que contiene los significados.

```
Clase CNodo
   Atributos
      codigo
      nombre
      siguiente     // variable de tipo CNodo
   Metodos
      constructor( nCodigo, sNombre )
      destructor
      colocarCodigo( nCodigo )
      colocarNombre( sNombre )
      colocarSiguiente( oSiguiente )
      obtenerCodigo→
      obtenerNombre→
      obtenerSiguiente→
fClase
```

2. Especificar el cuerpo de los métodos de CNodo

Una vez que hemos declarado la clase CNodo, especificamos el cuerpo de cada método. A continuación ilustramos el uso de oLoop para algunos de los principales métodos.

```
Metodo CNodo.constructor( nCodigo, sNombre )
   codigo ← nCodigo
   nombre ← sNombre
   siguiente ← nulo
fMetodo

Metodo CNodo.colocarCodigo( nCodigo )
```

```
   codigo ← nCodigo
fMetodo

Metodo CNodo.obtenerCodigo→
   retornar codigo
fMetodo

Metodo CNodo.colocarNombre( sNombre )
   nombre ← sNombre
fMetodo

Metodo CNodo.obtenerNombre→
   retornar nombre
fMetodo

Metodo CNodo.colocarSiguiente( oSiguiente )
   siguiente ← oSiguiente
fMetodo

Metodo CNodo.obtenerSiguiente→
   retornar siguiente
fMetodo
```

3. Crear la clase CLista

A continuación especificamos la clase CLista, que especificará los métodos para agregar un nodo alumno a la lista.

```
Clase CLista
   Atributos
      inicio     // variable de tipo CNodo
   Metodos
      constructor
      colocarInicio( oInicio )
      obtenerInicio →
      longitud→
      existe( nCodigo )→
      agregarInicio( nCodigo, sNombre )
      agregarFinal( nCodigo, sNombre )
      insertarNodo( nCodigo, sNombre, posición)→
      eliminarNodo( posición )→
      mostrarLista →
      liberarLista →
fClase
```

4. Especificar el cuerpo de los métodos de CLista

Una vez que hemos declarado la clase CLista especificamos el cuerpo de cada método. A continuación ilustramos el uso de oLoop para algunos de los principales métodos.

```
Metodo CLisda.constructor
   inicio ← nulo
fMetodo

Metodo CLista.colocarInicio( oInicio )
   inicio ← oInicio
fMetodo

Metodo CLista.obtenerInicio() →
   retornar inicio
fMetodo

Metodo CLista.longitud →
```

```
   ptr ← inicio
   num ← 0
   Mientras ( ptr <> nulo) Hacer
      num ← num + 1
      ptr ← ptr.obtenerSiguiente
   fMientras
   retornar num
fMetodo

Metodo CLista.existe( nCodigo ) →
   ptr ← inicio
   Mientras (ptr <> nulo y ptr.obtenerCodigo <> nCodigo) Hacer
      ptr ← ptr.obtenerSiguiente
   fMientras
   Si (ptr = nulo) entonces
      retornar falso
   Sino
      retornar verdadero
   fSi
fMetodo

Metodo CLista.agregaInicio( nCodigo, sNombre )
   Si (No existe( nCodigo ) ) entonces
      objeto nuevo ejemplar-de CNodo
      nuevo.constructor( nCodigo, sNombre )
      nuevo.colocarSiguiente( inicio )
      inicio ← nuevo
   fSi
fMetodo

Metodo CLista.agregaFinal(nCodigo, sNombre )
   Si (No existe( nCodigo ) ) entonces
      objeto nuevo ejemplar-de CNodo
      nuevo.constructor(nCodigo, sNombre )
      Si (inicio = nulo) entonces
         inicio ← nuevo
      Sino
         ptr ← inicio
         Mientras (ptr.obtenerSiguiente <> nulo) Hacer
            ptr ← ptr.obtenerSiguiente
         fMientras
         ptr.colocarSiguiente( nuevo )
      fSi
   fSi
fMetodo

Metodo CLista.insertaNodo(nCodigo, sNombre, posicion )→
   num ← longitud
   Si (posicion <= num) entonces
      Si (No existe( nCodigo ) ) entonces
         objeto nuevo ejemplar-de CNodo
         nuevo.constructor(nCodigo, sNombre )
         ptr ← inicio
         qtr ← ptr
         k ← 1
         Mientras (k < posicion) Hacer
            qtr ← ptr
            ptr ← ptr.obtenerSiguiente
            k ← k + 1
         fMientras
         Si ( ptr = inicio ) entonces
            nuevo.colocarSiguiente( inicio )
            inicio ← nuevo
         sino
            nuevo.colocarSiguiente( ptr )
```

```
            qtr.colocarSiguiente( nuevo )
         fSi
         retornar verdadero
      sino
         retornar falso
      fSi
   sino
      retornar falso
   fSi
fMetodo

Metodo CLista.eliminaNodo( nCodigo )→
   Si (existe( nCodigo ) ) entonces
      ptr ← inicio
      qtr ← ptr
      Mientras (ptr.obtenerCodigo <> nCodigo) Hacer
         qtr ← ptr
         ptr ← ptr.obtenerSiguiente
      fMientras
      Si (ptr = inicio) entonces
         inicio ← inicio.obtenerSiguiente
      sino
         qtr.colocarSiguiente( ptr.obtenerSiguiente )
      fSi
      liberar ptr
      retornar verdadero
   sino
      retornar falso
   fSi
fMetodo

Metodo Clista.muestraLista() →
   cadena ← ""
   ptr ← inicio
   Mientras (ptr <> nulo) hacer
       cadena ← cadena + "Codigo: " + ptr.obtenerCodigo
                + "Nombre " + ptr.obtenerNombre + "\n"
       ptr ← ptr.obtenerSiguiente
   fMientras
   retornar cadena
fMetodo

Metodo Clista.liberaLista →
   ptr ← inicio
   Mientras (ptr <> nulo) hacer
      aux ← ptr
      ptr ← ptr.obtenerSiguiente
      liberar aux
   fMientras
   retornar verdadero
fMetodo
```

5. Especificar la clase de la Interfaz

En las etapas anteriores, hemos concluido la especificación de las clases de la aplicación. En esta etapa de la especificación de la Interfaz del usuario, nos corresponde crear la clase de la Interfaz, la cual utilizará objetos de las clases de la jerarquía de clases de desarrollada.

```
Clase CInstituto viene-de CFormulario
   Atributos
      // objeto de la clase CLista
      objeto alumno ejemplar-de CLista

      // Botones de Acción
```

```
      Objeto agregaInicioBotón   ejemplar-de CBotónAcción
      Objeto agregaFinalBotón    ejemplar-de CbotónAcción
      Objeto insertaBotón        ejemplar-de CBotónAcción
      Objeto eliminaInicioBotón ejemplar-de CBotónAcción
      Objeto muestraBotón        ejemplar-de CBotónAcción
      Objeto liberaBotón         ejemplar-de CBotónAcción

      // Campos de Texto
      Objeto codigoTexto   ejemplar-de CTexto
      Objeto nombreTexto   ejemplar-de CTexto
      Objeto posicionTexto ejemplar-de Ctexto

   Métodos
      Constructor
      // botones de acción para el arreglo
      agregaInicioBotónAlClic
      agregaFinalBotónAlClic
      insertaBotónAlClic
      eliminaBotónAlClic
      muestraBotónAlClic
      liberaBotónAlclic

fClase
```

6. Especificar los métodos de la clase Interfaz

En esta etapa, especificamos el código de aquellos métodos que se ejecutarán en respuesta a los eventos de la interfaz del usuario. A continuación ilustramos algunos de ellos.

```
// En el constructor
Método CInstituto.Constructor
     alumno.construtor
fMétodo

// En el método de respuesta a la acción del botón Agrega Inicio
Método CInstituto.agregaInicioBotónAlClic()
   codigo ← codigoTexto.obtenerTexto()
   nombre ← nombreTexto.obtenerTexto()
   estado ← alumno.agregaInicio(codigo, nombre)
   MostrarMensaje("Agrega inicio: "+estado)
fMétodo

// En el método de respuesta a la acción del botón Agrega Final
Método CInstituto.agregaFinalBotónAlClic()
   codigo ← codigoTexto.obtenerTexto()
   nombre ← nombreTexto.obtenerTexto()
   estado ← alumno.agregaFinal(codigo, nombre)
   MostrarMensaje("Agrega final: "+estado)
fMétodo

// En el método de respuesta a la acción del botón Inserta Nodo
Método CInstituto.insertaBotónAlClic()
   codigo ← codigoTexto.obtenerTexto()
   nombre ← nombreTexto.obtenerTexto()
   posicion ← posicionTexto.obtenerTexto()
   estado ← alumno.insertaNodo(codigo, nombre, posicion)
   MostrarMensaje("Inserta nodo: "+estado)
fMétodo

// En el método de respuesta a la acción del botón Elimina Nodo
Método CInstituto.eliminaBotónAlClic()
   codigo ← codigoTexto.obtenerTexto()
   estado ← alumno.eliminaNodo(codigo)
```

```
   MostrarMensaje("Elimina nodo: "+estado)
fMétodo

// En el método de respuesta a la acción del botón Muestra
Método CInstituto.muestraBotónAlClic()
   cadena ← alumno.muestraLista
   MostrarMensaje("muestra lista: "+cadena)
fMétodo

// En el método de respuesta a la acción del botón Liberar
Método CInstituto.liberaBotónAlClic()
   estado ← alumno.libera
   MostrarMensaje("Libera lista: "+estado)
fMétodo
```

7. Tarea para completar

En esta etapa, desarrollaremos el código de aquellos métodos que se requieran como respuesta a lo siguiente:

- Elaborar un método para buscar un nodo por código y mostrar el nombre asociado
- Elaborar un método para buscar un nodo por nombre y mostrar el código asociado
- Elaborar un método para modificar un nombre que ya se encuentra en la lista, para ello debo de buscarlo por código.
- Elaborar un método para eliminar un alumno indicando la posición lógica en la lista enlazada.