Universidad Central “ Martha Abreu” de Las Villas
CUBA

Aplicación de la Multimedia en el Proceso Enseñanza – Aprendizaje de las Estructuras de Datos Lineales.

Tesis presentada en opción al Titulo Académico de Master en Computación Aplicada

Autor : **Liliana Serrano Zúñiga**.
Tutor (es): *Ana María García Pérez*.
Año en que se efectúa la defensa: **2002**.

Í n d i c e

Resumen

En este trabajo de tesis se organizaron los contenidos relativos a los tópicos de estructuras de datos lineales correspondientes a la materia de Estructuras de Datos de los programas de Licenciatura en Informática e Ingeniería en Computación del Centro Universitario de la Ciénega de Ocotlán, Jalisco.

Se muestra la implementación de un Software multimedia que tuvo la finalidad de mostrar algo vistoso e interesante a fin de mejorar los resultados de nuestro proceso enseñanza- aprendizaje.

Abstract

The contents of the topics related to linear data structures corresponding to the object of data structures of the programs of computer Engineering and BS in computing from the University of CUCIENEGA, Ocotlán, Jalisco, were organized.

The thesis shows the implementation of a multimedia software with the goal of show something beautiful and pleasant in order to improve our teaching process results.

Introducción

El vertiginoso desarrollo de las tecnologías informáticas y de comunicaciones cuestionan y reconfiguran cada día la forma en que percibimos nuestra realidad y la manera como nos apropiamos y relacionamos con el mundo. Los estudiantes de hoy han crecido en un mundo radicalmente diferente al de unas generaciones anteriores, las nuevas tecnologías son indudablemente fundamentales a la hora de decidir a donde iremos y como evolucionará la tecnología. La manera en que nosotros como usuarios, como maestros, como estudiantes, elijamos confrontar esos factores es un aspecto decisivo en el impacto y beneficio que pueden traer al proceso educativo y de aprendizaje.

El computador hoy en día es una plataforma capaz de actuar como meta-medio, por esto entre otras razones, se presenta como una alternativa eficaz para experimentar y desarrollar alternativas a procesos educativos tradicionales.

La Hipermedia. Se refiere usualmente al uso de una amplia variedad de medios dentro de una interfaz. La información es guardada en diferentes medios (voz-sonido, imágenes estáticas en movimiento, texto, etc.) y organizada de manera que pueda ser recuperada y mostrada de diversas formas de manera tal que el usuario final amplifica su significado y puede generar conexiones e interpretaciones diversas.

En el campo de la docencia, las transformaciones tecnológicas podrían llegar a imponer el reto, la necesidad y sobre todo; la posibilidad de renovar las técnicas de enseñanza y el tipo de material docente que se pone a disposición de los estudiantes (y maestros). Las condiciones actuales facilitan contar con herramientas de apoyo al proceso educativo que se encuentren más cercanas a la manera en como, niños, jóvenes (y adultos) perciben y entienden su mundo hoy, es decir de una manera mas dinámica llena de estímulos paralelos, preparados para el cambio constante, intercomunicados e integrados.

La multimedia representa una ventaja como soporte al proceso educativo, pues presenta y manipula la información en un lenguaje contemporáneo, que además permite a maestros y alumnos jugar con su estructura para lograr diferentes objetivos pedagógicos.

La enseñanza y el aprendizaje son procesos sociales por naturaleza con una dimensión comunicativa muy importante. La comunicación humana está basada en cierta medida en nuestras habilidades para procesar información multimodal, multimedia y de diferentes códigos.

Nuestras habilidades preceptuales hacen uso de los diferentes sentidos capaces de interpretar y razonar con diferentes códigos visuales, lingüísticos, etc. porque como humanos nos desempeñamos mas eficientemente bajo esos términos.

El computador, en el salón de clase, se convierte en un recurso que pone la multimedia al alcance de la vida diaria como herramienta para producirla, compartirla y explorarla. A través de un mismo canal, textos y material audiovisual se combinan generando nuevas posibilidades para aprender y explorar temas y contextos no solo a nivel individual sino con el apoyo y colaboración de compañeros y maestros.

Debido a los requerimientos de memoria se han tenido que buscar soluciones al problema de almacenamiento y distribución de contenidos; disquetes, discos ópticos, cd-rom, y más recientemente la red, son todos soportes para lograr este objetivo.

La ventaja que da la presentación en CD-ROM está definida por la portabilidad que se le imprime, lo que aumenta su flexibilidad como herramienta, a eso sumado el hecho de ofrecer una gran capacidad de distribución que no por ello reduce la calidad del material. Los contenidos multimedia han encontrado en este medio de almacenamiento un interesante canal para aportar al proceso de aprendizaje no solo en el salón de clases sino también en los hogares.

Los objetivos de esta Tesis son:

1. Estudiar la problemática de la enseñanza de las estructuras de datos en nuestras carreras.
2. Estudiar formas alternativas y complementarias que ayuden al proceso de enseñanza – aprendizaje.
3. Mostrar la factibilidad de la multimedia como medio de apoyo en el proceso de enseñanza – aprendizaje de los conceptos fundamentales de las estructuras de datos y su implementación.
4. Implementar un sistema multimedia que ayude al proceso de enseñanza-aprendizaje de las estructuras de datos lineales.

La tesis esta estructurada en tres capítulos, los cuales son:

Capitulo 1. Los sistemas de enseñanza asistidos por computadora.

Capitulo 2. La enseñanza de las Estructuras de Datos en nuestras carreras.

Capitulo 3. Diseño, características y uso de la Multimedia.

Capitulo 1.

Los sistemas de enseñanza asistida por computadoras

El propósito del presente capitulo es ofrecer una panorámica del desarrollo de los sistemas de enseñanza asistida por computadora, fundamentando el tipo de software educativo que hemos escogido para ayudar al proceso de aprendizaje de las principales estructuras de datos lineales y sus algoritmos de manipulación, que se imparten en el nivel de las carreras de Lic. en Informática e Ing. en Computación en el Centro Universitario de la Ciénega (CUCI) de Ocotlán, Jalisco.

El docente es quien sabe enseñar, conoce qué se enseña y puede facilitar el proceso de aprendizaje de sus alumnos con sus conocimientos, experiencias y metodologías. [GAR 2001]

Este aspecto es de gran relevancia a la hora de hablar de contenidos locales, contextualizarlos a las condiciones peculiares de una región especifica. Este contenido debe ser desarrollado de una manera local, por personas (docentes, investigadores, etc.) conocedoras de su entorno y de las necesidades particulares del área en que se trabajan, las inquietudes y capacidades de sus alumnos y las dinámicas que harán más efectivo este proceso. Solo de esta manera estos contenidos y su estructura multimedial tendrán sentido y razón de ser para su comunidad.

Los recursos tecnológicos como la multimedia informática son un apoyo para la comunicación en un lenguaje contemporáneo, pero su mayor aporte y su éxito como facilitador seguramente se encontrará en la medida en que seamos capaces de revertir un contenido que sea nuestro, que proponga nuestra visión y situación y que permita construir conocimiento de una manera colectiva.

Una de las ventajas de estos nuevos medios es que se dispone de métodos productivos que usan recursos diferentes, el acceso a material más localizado son, hoy en día, uno de las potencialidades de este medio que debemos aprovechar.

En los años 50 aparecieron los primeros sistemas de enseñanza, los llamados programas lineales. Estos se caracterizaban por mostrar el conocimiento de una manera lineal, ningún factor podía cambiar el orden de enseñanza establecido en su momento por el programador. [URR 2001]

La teoría de F.Skinner (1950) propugnaba que las personas funcionan por estímulos, esto es: a igual estimulo corresponde igual respuesta; no se debía permitir cometer errores a los alumnos. Por lo tanto en el desarrollo de una sesión de enseñanza no se tiene en cuenta para nada la aptitud del alumno. [URR 2001]

Los sucesores de los programas lineales fueron los programas ramificados [Cowed 1959]. Estos tenían un numero fijo de temas, se diferenciaban por la capacidad de actuar según la respuesta del alumno.

La mejora ofrecida por estos sistemas se consiguió gracias a la técnica de Pattern-matching y al diseño de lenguajes de autor. En cuanto a la técnica de Pattern-matching, ésta permitía tratar las respuestas del alumno como aceptables o parcialmente aceptables, en lugar de totalmente correctas o incorrectas como exige la propuesta de Skinner.

A finales de los sesenta y principios de los setenta surgieron los sistemas generativos o sistemas adaptativos. Estos van asociados a una nueva filosofía educativa que manifiesta que los alumnos aprenden mejor enfrentándose a problemas de dificultad adecuada, que atendiendo a explicaciones de generar un problema acorde al nivel de conocimiento del alumno, construir su solución y diagnosticar la respuesta del alumno.

Comenzaremos desde los primeros sistemas (los más básicos) hasta los sistemas tutores inteligentes (STI) surgidos hacia los años 80. Los sistemas de enseñanza tradicionales desarrollados antes de la aparición de los primeros STIs, se conocen con el nombre de CAI (Computer-Assisted Instruction – Enseñanza Asistida por Ordenadores). Las principales características de los mismos son:

- Los cursos son muy extensos.
- La comunicación entre el tutor y el alumno no está muy refinada.
- El conocimiento del cómo y por qué se ejecutan las tareas de enseñanza están fusionados.
- El diseño e implementación de los sistemas están hechos a medida.
- El conocimiento que incluye no se ve modificado con el tiempo, no evoluciona.

Como evolución de los sistemas CAI y con la incorporación de técnicas de I.A. aparecieron los Sistemas Tutores inteligentes (STI). Estos sistemas facilitan el proceso de enseñanza / aprendizaje haciéndolo más efectivo, correcto y también mas agradable. Sus características más importantes son: [URR 2001]

1) El conocimiento del dominio está acotado y claramente articulado.
2) Poseen conocimiento del estudiante que les permite dirigir y adaptar la enseñanza.
3) La secuencia de enseñanza no está predeterminada por el diseñador .
4) Realizan procesos de diagnóstico más adaptados al estudiante y más detallados.
5) La comunicación tutor–alumno mejora, permitiendo además que el alumno realice preguntas al Tutor.

Los STI se caracterizan por representar separadamente la materia que se enseña y las estrategias para enseñarla. Entiendo enseñanza como "transmisión de conocimiento que precisa de un seguimiento continuo del profesor hacia el alumno, especialmente en los procesos de resolución de problemas". [URR 2001]

El planteamiento "idílico" de los STI es el de un buen profesor, adaptándose al estudiante con el que interactúa. Los sistemas de entrenamiento inteligente o STI con vocación de entrenamiento tienen también este enfoque aunque en estos casos los procesos se dirigen más a actividades especificas de resolución de problemas. Inicialmente los STIs, aunque permiten una iniciativa mixta en las interacciones educativas y los estudiantes pueden hacer preguntas y tener algún control sobre el proceso de enseñanza, proponen una estrategia dirigida. [URR 2001]

El proceso de enseñanza va guiado sobre todo por el tutor según las prácticas tradicionales de la enseñanza. El diseño e implementación de los STI se ha venido realizando para un determinado dominio y bajo las estrategias pedagógicas de un determinado docente o grupo de docentes.

En los sistemas desarrollados según esta propuesta el estudiante lleva el control de la actividad docente, construyendo su propia sección de aprendizaje y asegurando sus propios objetivos de aprendizaje. Los sistemas hipermedia se adaptan perfectamente a las exigencias de este tipo de planteamientos. Estos sistemas permiten que el usuario: [URR 2001]

- Acceda a la información de la base de datos.

- Disponga de una gran variedad de formas de acceso a la información.
- Redefina la estructura y contenido del material a utilizar.

Tanto los STIs como los sistemas hipermedia presentados, centran el proceso de aprendizaje desde una perspectiva individualizada. Es decir, los estudiantes aprenden mediante un proceso de interacción sistema–estudiante. Sin embargo, existe un movimiento que propone nuevas estrategias didácticas en las que se defiende que el aprendizaje de calidad, especialmente el aprendizaje de las ciencias, requiere cooperación más que competición.

En este sentido el aprendizaje colaborativo ofrece un escenario de cooperación que permite al estudiante cuestionarse y construir su conocimiento. El estudiante aprende a dar explicaciones sobre los sucesos que experimenta, atiende a las explicaciones de sus compañeros, justifica los sucesos, etc.

Cada vez más está proliferando la enseñanza virtual sobre la red, que combina enseñanza presencial clásica (minimizada al máximo) y no presencial. Asi, la clase magistral desaparece como base de los cursos, desarrollándose documentación informatizada (libros digitales, documentos HTML, etc.).

Cuando se habla de nuevas formas de aprendizaje debemos analizar si se trata de cambios e innovaciones en términos de los procesos cognitivos del individuo o de nuevos procedimientos, metodología y modelos para promover el aprendizaje, aprovechando para ello diversos recursos y estrategias a nuestro alcance, en especial la introducción de las redes que en la educación ha venido a ampliar y acelerar el manejo e intercambio de información y de comunicación y en especial la educación a distancia. [FER 2000]

La educación a distancia se ha venido consolidando con el uso de los medios informáticos y de telecomunicaciones como modelos virtuales de educación el reto de la educación a distancia o educación virtual es favorecer las posibilidades de nuevas composiciones y creaciones a partir de las actuales condiciones del saber.

Los entornos de aprendizaje virtuales constituyen una forma totalmente nueva de tecnología educativa y ofrecen oportunidades y tareas a las instituciones de enseñanza de todo el mundo, el entorno de aprendizaje virtual se define como un programa informático interactivo de carácter pedagógico que posee una capacidad de comunicación integrada.

Los entornos de aprendizaje virtuales son una innovación relativamente reciente y fruto de la convergencia de las tecnologías informáticas y de telecomunicaciones que se han intensificado durante los últimos diez años.

Como innovaciones para el aprendizaje deben atenderse sus componentes: los asesores, tutores o monitores, los estudiantes, los contenidos y su tratamiento o metodología didáctica y los medios tecnológicos.

Hablar de nuevas tecnologías es referirse a la multimedia, la televisión por cable y satélite, al CD-ROM, y a los hipertextos donde su materia prima es la información, se consideran nuevas tecnologías esencialmente las computadoras y los programas informáticos que permiten el acceso a redes, básicamente porque los avances tecnológicos, han dado a la computadora un protagonismo como instrumento pedagógico ya que permite el acceso a grandes cantidades de información. [FER 2000]

La incorporación de medios por consiguiente, obliga a los usuarios a tener una alfabetización tecnológica, lo cual se logra teniendo acceso a lecturas e ideas relacionadas con el uso de la tecnología; adquiriendo un marco de referencia tecnológico amplio que le permita saber por qué esta haciendo lo que hace y por qué no hace otras cosas. Es importante que el estudiante y el docente se sientan seguros en su habilidad para apropiarse de la tecnología.

El trabajo en redes genera procesos de interacción y de diálogo entre personas donde la información adquiere nuevos significados mediante el intercambio de mensajes con otros, no debemos olvidar que el medio por sí mismo no hace de los estudiantes mejores aprendices, su incorporación requiere modelos de uso muy claros de manera que permitan la apropiación de los contenidos presentados.

Con Internet ha surgido una biblioteca mundial <<virtual>>, interconectada y diseminada geográficamente, cuyos documentos puede consultar cualquier persona que posea una computadora, un mecanismo de comunicación (MODEM) y un teléfono, Además, distintos usuarios pueden colaborar a distancia en la creación de documentos. Si no se dispone de instrumentos de navegación eficaces (mapas y brújulas, por así decirlo), los docentes y estudiantes se podrían pasar la vida navegando por Internet en busca de esa información.

Si bien es importante saber buscar y localizar bancos de información que enriquezcan y apoyen los procesos de aprendizaje, es necesario replantear las maneras en que los alumnos puedan adquirir conocimientos e informaciones sin perder de vista que en toda situación didáctica el centro deberá ser el estudiante. La función del profesor será la de un facilitador que presta asistencia cuando el estudiante busca conocimientos.

La herramienta utilizada es solo un medio para despertar el interés, mantener la motivación y la participación activa en el proceso de enseñanza aprendizaje. Es importante dejar en claro que la Internet informa, pero no transforma. Mediante las nuevas tecnologías, y de manera especial con Internet, se tiene acceso a información pero no al conocimiento.

El potencial que ofrecen las redes de computadoras especialmente Internet y WWW- en la educación, capacitación y entrenamiento, ha estimulado la investigación en sistemas integrados de enseñanza /aprendizaje que, además de proporcionar material educativo multimedia, permitan clasificar, planificar, evaluar y orientar las actividades de los alumnos, para que estos aprendan eficientemente. [NUÑ 1999]

Dentro de este campo, existe una línea bien definida denominada aprendizaje colaborativo soportado por computadora (Computer Supported Collaborative Learning - CSCL) dedicada a la creación de ambientes virtuales cooperativos y colaborativos para realizar a distancia y/o soportar las diversas actividades de enseñanza y aprendizaje que se realizan en las instituciones educativas.

El aprendizaje colaborativo es especialmente útil en dominios complejos, en los que es difícil asimilar conocimiento de manera individual. Los ambientes colaborativos de la enseñanza utilizan la inteligencia artificial, las tecnologías de groupware, workflow y agentes, las redes de computadoras y las telecomunicaciones, para generar ambientes virtuales de enseñanza y, en general, de trabajo en los que interactúan diversas personas.

Las tecnologías de información ofrecen excitantes oportunidades para replantear a fondo el proceso de transmisión y construcción del conocimiento y permiten lograr, entre otros, los siguientes beneficios: integración de medios (texto, audio, animación y vídeo), interactividad, acceso a grandes cantidades de información, planes y ritmos de trabajo individualizados y respuesta inmediata al progreso del aprendiz.

Los tipos de software Educativo pueden ser: [NUÑ 1999]

- Enseñanza asistida por Computadora basada en texto.
- Servicio de Información Hipermedia e Instrumentación basada en Internet.
- Clases en línea en tiempo real.
- Soluciones Multimedia.
- Sistemas Tutores Inteligentes.

Todas las tecnologías proporcionan ventajas estratégicas importantes para las instituciones educativas de todos los niveles, pero su utilización efectiva implica un replanteamiento o rediseño a fondo. [NUÑ 1999]

Hay un Proyecto de Espacios Virtuales de Aprendizaje, es un programa de investigación el área de CSCL, el cual se propone la generación, implantación y puesta en marcha de un ambiente de enseñanza aprendizaje personalizado y colaborativo.

Se eligió la multimedia como mecanismo complementario para la enseñanza de las Estructuras de Datos por el mundo que podemos mostrar y enseñar con ella: por ejemplo permite la **Ilusión de movimiento**. Se pueden hacer tales efectos con movimientos que podrán percibir lo que se quiere enseñar, se pueden dar formas muy vistosas y animadas para conseguir la atención que se quiere que se tenga al contenido. Permite también la **Visualización de resultados.** Se visualizará la forma en que se ejecutan las diferentes operaciones fundamentales de las Estructuras de Datos Lineales, es decir como se verían lógicamente las estructuras después de aplicarles los métodos de manipulación que conforman cada uno de estos.

Resumen del Capítulo

Dentro de las tendencias del software educativo, se eligió una multimedia como producto a obtener, con la finalidad de proporcionar interactividad, acceso a grandes cantidades de información, planes y ritmos de trabajo individualizados y respuesta inmediata al progreso del aprendiz.

Esta multimedia se quiso implementar para alumnos del Centro Universitario de la Ciénega de las carreras de Lic. en Informática e Ing. en Computación en la materia de Estructuras de Datos con la finalidad de brindarles otra forma más moderna y eficaz de aprendizaje.

Aquí el objetivo primordial es darles a conocer como una primera etapa las estructuras lineales básicas, donde se contempla desde los conceptos básicos, sus operaciones fundamentales, los códigos fuentes de los programas correspondientes a cada estructura, etc.; pero de una forma de mantener al alumno despierto e interesado para recibir el conocimiento, esto se logrará con animaciones, sonido, etc. Y lo más importante es que el alumno pueda aprender por su propia cuenta, sin necesidad de tener un maestro al frente.

Se podrá contemplar la implementación de cada una de las estructuras en programación estructurada y / o en programación orientada a objetos.

Actualmente los profesores en su curso utilizan solo la programación estructurada y con esta multimedia se les proporcionará otra opción en la programación que será la orientada a objetos. La finalidad es irlos induciendo a la programación de vanguardia, es decir actualizarlos en los nuevos métodos de la programación.

Capitulo 2

La enseñanza de las Estructuras de Datos en nuestras carreras.

El propósito del presente capitulo es mostrar las características de las Estructuras de datos lineales que vamos a implementar y el beneficio que tendrán los alumnos al utilizar una multimedia para el aprendizaje de esta materia, además de exponer la situación actual que se vive con los alumnos que cursan las materias de Estructuras de datos en nuestro Centro Universitario.

Uno de los aspectos mas importantes de la programación es la creación de nuevos tipos de datos que sean apropiados para resolver problemas específicos. Las estructuras de datos son construcciones de programación utilizadas para representar esos nuevos tipos. Éstas pueden ser aplicadas por medio del modelo de los tipos abstractos de datos (TAD) que permite definir la información en base a sus propiedades funcionales. En un TAD, los datos y las operaciones que los manipulan son "encapsulados" dentro de una interfaz de programación que oculta al usuario su implementación. Esto permite programar de forma modular además de facilitar el mantenimiento de los programas.

Ejemplos de estructuras de datos los podemos encontrar en muchos ámbitos, desde las matemáticas (estructuras algebraicas: grupo, anillo o cuerpo) hasta el mundo de los negocios (estructura de una empresa). Los elementos de una estructura de datos dependen del lenguaje de programación a través de los tipos de datos que los definen. Sin embargo, la estructura en sí, que está definida por el tipo de relación entre los elementos, no depende del lenguaje de programación empleado.

Estructuras de datos lineales.

Las estructuras de datos lineales se utilizan para almacenar secuencias. Y sus operaciones fundamentales son: crear, insertar un elemento, eliminar un elemento, consultar, entre otras.

Podemos considerar dentro de las estructuras de datos lineales las siguientes:

- Listas
- Pilas
- Colas

Estas a su vez son:

- **Estáticas:** hay un número máximo de elementos; implementadas con un vector.
- **Dinámicas:** sin número máximo de elementos, si suponemos memoria infinita.

Listas

Las listas constituyen una estructura flexible porque pueden crecer y acortarse según se requiera; los elementos son accesibles y se pueden insertar y suprimir en cualquier posición de la lista. Las listas también pueden concatenarse entre sí o dividirse en sublistas; Matemáticamente, una *lista* es una secuencia de cero o más elementos de un tipo determinado (que por lo general se denominará tipo-elemento). A menudo se representa una lista como una sucesión de elementos separados por comas.

$$a_1, a_2, ..., a_n$$

donde $n \geq 0$ y cada a_i es del tipo tipo-elemento. Al número n de elementos se le llama *longitud* de la lista. Al suponer que $n \geq 1$, se dice que a_1 es el *primer elemento* y a_n el *último elemento.* Si $n = 0$, se tiene una *lista vacía,* es decir, que no tiene elementos.

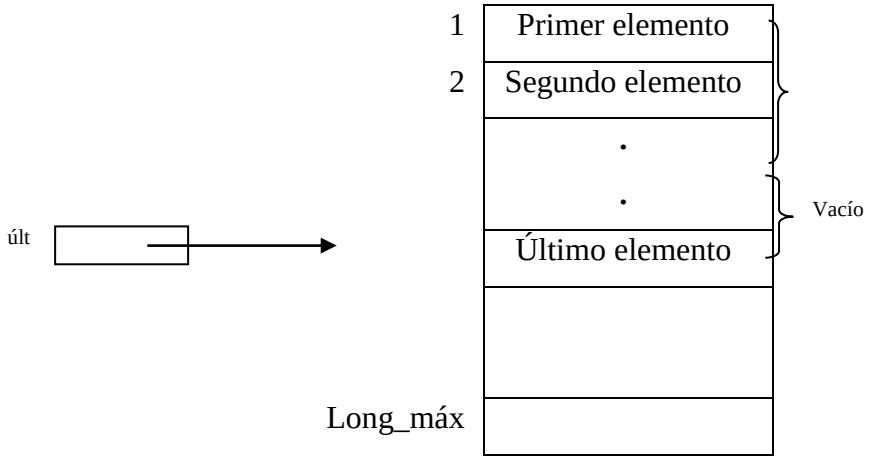
Una propiedad importante de una lista es que sus elementos pueden estar ordenados en forma lineal de acuerdo con sus posiciones en la misma. Se dice que a_i *precede* a a_{i+1} para i = 1,2,..., n-1, y que a_i *sucede* a a_{i-1} para i = 2,3,..., n. Se dice que el elemento a_i está en la *posición i.*

Listas mediante arreglos.

En la realización de una lista mediante arreglos, los elementos de ésta se almacenan en celdas contiguas de un arreglo. Esta representación permite recorrer con facilidad una lista y agregarle elementos nuevos al final. Pero insertar un elemento en la mitad de la lista obliga a desplazarse una posición dentro del arreglo a todos los elementos que siguen al nuevo elemento para concederle espacio. De la misma forma la eliminación de un elemento, excepto el último, requiere desplazamientos de elementos para llenar de nuevo el vacío formado.

En la realización con arreglos se define el tipo LISTA como un registro con dos campos; el primero es un arreglo de elementos que tiene la longitud adecuada para contener la lista de

mayor tamaño que se pueda presentar. El segundo campo es un entero *últ* que indica la posición del último elemento de la lista en el arreglo. El i-ésimo elemento de la lista está en la i-ésima celda del arreglo, para $1 \leq i \leq$ *últ*, como se muestra en la siguiente figura:



Las posiciones en la lista se representan mediante enteros; la i-ésima posición, mediante el entero i. La función *FIN(L)* sólo tiene que devolver *últ*. Las declaraciones más importantes son:

En Programación Estructurada

```
function localiza (x:tipo_elemento; var l:lista):posicion;
{LOCALIZA devuelve la posición de x en la lista L}
var
        q:posicion;
begin
        for q:=1 to l.ult do
                if l.elementos[q]= x then
                        return(q);
        return (l.ult+1) {si no se encuentra}
end;{LOCALIZA}
```

En Programación Orientada a Objetos

```
const
            long_max=100;
        vacio:=0;
            type
                        lista=class
                        private
                           elementos:[1..long_max] of tipo_elemento;
                           ult: integer;
                           function fin(var l:lista): integer;
                           procedure inserta(x:tipo_elemento;p:posicion;var l:lista);
                         public
                           constructor lista();
                    end;
constructor  lista();
begin
  for x:=1 to long_max do
            elementos[x]=vacio;
 ult:=0;
end;

function lista.fin( var l:lista): posicion;

begin
  result := l.ult + 1;
end;
```

```
procedure lista.inserta(x:tipo_elemento;p:posicion;var l:lista);

var
        q:posicion;

begin
        if l.ult >= long_max then
                error("la lista está llena")
        else if p> (l.ult+1)or (p<1) then
                error("la posicion no existe")
        else begin
                for q:=l.ult downto p do
                        l. elementos[q+1]:=l. elementos[q];
                l.ult:=l.ult+1;
                l.elementos[p]:=x;
            end;
end;

procedure lista.suprime(p:posicion;var l:lista);

var
        q:posicion;

begin
        if (p>ult) or (p<1)  then
                error("la posicion no existe")
        else begin
                l.ult:=l.ult -1;
                for q:=p to l.ult do
                        l. elementos[q]:=l. elementos[q+1];
            end;
end;

function lista.localiza(x:tipo_elemento;var l:lista):posicion;

var
        q:posicion;

begin
        for q:=1 to l.ult  do
           if l. elementos[q]=x then
                result:=q;
        result:=l.ult+1
end;
```

Los algoritmos anteriores realizan las siguientes tareas:

INSERTA pasa los elementos de las localidades *p, p* + 1, ...,*ult* a las localidades *p* + 1, *p* + 2,..., u*lt* + 1 y después inserta el nuevo elemento en la localidad p., Si no hay espacio en el arreglo para insertar un nuevo elemento, se invoca a la rutina *error*, que imprime su argumento y da por terminada la ejecución del programa.

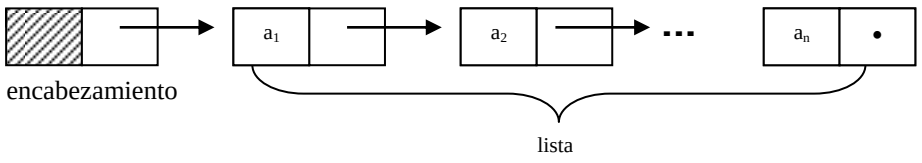
SUPRIME elimina el elemento de la posición *p* pasando los elementos de las posiciones *p* + 1, *p* +2,..., *ult*, a las posiciones *p, p* + 1 ,..., *ult* -1.

LOCALIZA revisa el arreglo secuencialmente, en busca de un elemento determinado; si no lo encuentra, LOCALIZA devuelve *ult* + 1.

Listas mediante apuntadores

Otra forma de realización de listas, es mediante celdas enlazadas sencillas, utilizando apuntadores para enlazar elementos consecutivos. Esta implantación permite eludir el empleo de memoria contigua para almacenar una lista y, por tanto, también elude los desplazamientos de elementos para hacer inserciones o rellenar vacíos creados por la eliminación de elementos. No obstante, por esto hay que pagar el precio de un espacio adicional para los apuntadores.

En esta representación, una lista está formada por celdas; cada celda contiene un elemento de la lista y un apuntador a la siguiente celda. Si la lista es a_1, a_2, ..., a_n, la celda que contiene a_i tiene un apuntador a la celda que contiene a a_{i+1} para i = 1, 2,..., n-1. La celda que contiene a_n posee un apuntador nil. Existe también una celda de *encabezamiento* que apunta a la celda que contiene a_1; esta celda de encabezamiento no tiene ningún elemento. En el caso de una lista vacía, el apuntador del encabezamiento es nil y no se tienen más celdas. Véase la siguiente figura donde se representa una lista mediante punteros:



Lista Enlazada

→

Para las listas enlazadas sencillas es conveniente usar una definición de posición ligeramente distinta de la que se empleó para las listas logradas mediante arreglos. Aquí, la posición *i* será un apuntador a la celda que contiene el apuntador a a_i para i = 2, 3, ..., *n*. La posición 1 es un apuntador al encabezamiento, y la posición *FIN(L)* es un apuntador a la última celda de L.

El tipo de una lista, en realidad, es el mismo que el de una posición; es un apuntador a una celda particular: el encabezamiento. La siguiente es una definición formal de las partes esenciales de una estructura de datos de lista enlazada.

En Programación Estructurada

```
type
    celda=record
    elemento: tipo-elemento;
    sig: ↑celda
end;
lista=↑celda;
posicion=↑celda;
```

En Programación Orientada a Objetos

```
type
                celda=class
                private
                   elemento: tipo_elemento;
                   sig: ^celda;
                end;
                lista=class
                private
                        apt:^celda;
                        posicion:^celda;
                end;
```

En seguida se muestra el algoritmo *FIN(L);* éste funciona moviendo el apuntador *q* por la lista, hasta que alcanza el final, el cual se detecta por el hecho de que *q* apunta a una celda con un apuntador nil. Obsérvese que esta realización de FIN es ineficiente, pues requiere revisar la

lista entera cada vez que se desea calcular *FIN(L).* Si es necesario hacer esto con frecuencia, puede optarse por cualquiera de las opciones siguientes:

1. Usar una representación de listas que incluya un apuntador a la última celda , o
2. Sustituir el uso de *FIN(L)* donde sea posible. Por ejemplo, la condición *p< > FIN(L)* de algoritmos anteriores se puede remplazar por p↑*.sig <> nil.*

En Programación Estructurada

```
function fin (l:lista): posicion;
{fin devuelve un apuntador a la celda l var
 q: posicion;
 begin
   q.=l;

 while q↑.sig <> nil do
   q := q↑ .sig;
 return (q)
 end; { fin }
```

En Programación Orientada a Objetos

```
function lista.fin(l:lista):posicion;
        var
                q: posicion;
        begin
                q:=l;
        while q^.sig <> nil do
                q:=q^.sig;
                result:=q;
        end;
```

Los algoritmos más elementales de listas con apuntadores son INSERTAR, SUPRIME, LOCALIZA; enseguida mostramos los algoritmos correspondientes:

En Programación Estructurada

```
Procedure INSERTA (x:tipo_elemento; p:posicion);
        Var
          temp:posicion;
        begin
          temp:=p↑.sig;
          new(p↑.sig);
          p↑.sig↑.elemento:=x;
          p↑.sig↑.sig:=temp
        end;{INSERTAR}

procedure SUPRIME (p:posicion);
        begin
          p↑.sig:= p↑.sig↑.sig
        end;{SUPRIME}

function LOCALIZA (x:tipo_elemento;l:lista):posicion;
    var
          p:posicion;
   begin
          p:=L;
          while p↑.sig <> nil do
                  if p↑.sig↑.elemento = x then
                          return (p)
                  else
                          p:=p↑.sig;
                  return (p) {si no es encontrado}
end;{LOCALIZA}
```

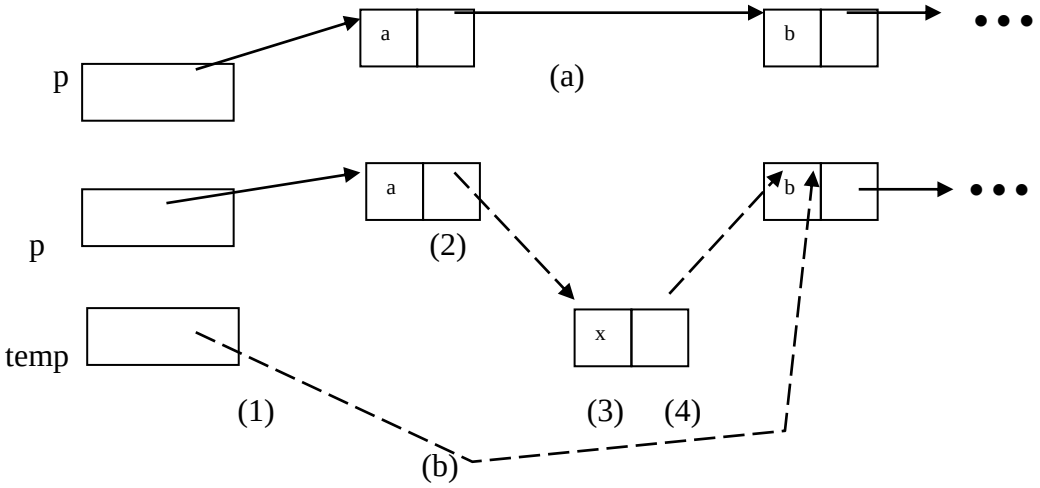
En Programación Orientada a Objetos

```
procedure Lista.Inserta(x:tipo_elemento;p:posicion);
var
        temp:posicion;
begin
        temp:=p^.sig;
        new(p^.sig);
        p^.sig^.elemento:=x;
        p^.sig^.sig:=temp;
end;
```

```
procedure lista.suprime(p:posicion;);
begin
        p^.sig^:=p^.sig^.sig;
end;

function lista.localiza(x:tipo_elemento; l:lista):posicion;
var
        p:posicion;
begin
      p:=l;
        while p^.sig <> nil do
          if  p^.sig^.elemento=x then
                result:=p
          else
                p:=p^.sig;
        result:=p;
end;
```

En el inciso (a) de la siguiente figura se muestra la situación antes de la ejecución de INSERTA. Se desea insertar un nuevo elemento antes de la celda que contiene b, por lo que p es un apuntador a la celda de la lista que contiene el apuntador a b.

En la línea (1), *temp* apunta a la celda que contiene *b.* En la línea (2), se crea una nueva celda de la lista y se hace que el campo *sig* de la celda que contiene *a* apunte hacia esta misma celda. En la línea (3) se almacena *x* en el campo *elemento* de la celda recién creada, y en la línea (4), el campo *sig* toma el valor de *temp* y, de esta manera, apunta a la celda que contiene *b.* La figura (b) muestra el resultado de la ejecución de INSERTA. Los apuntadores nuevos se representan con líneas punteadas y se marca el paso en el cual fueron creados.



El procedimiento SUPRIME es más sencillo. En la siguiente figura se ilustra la manipulación de los apuntadores que realiza este procedimiento. Los apuntadores antiguos se representan por medio de líneas de trazo continuo, y los nuevos por medio de líneas punteadas.

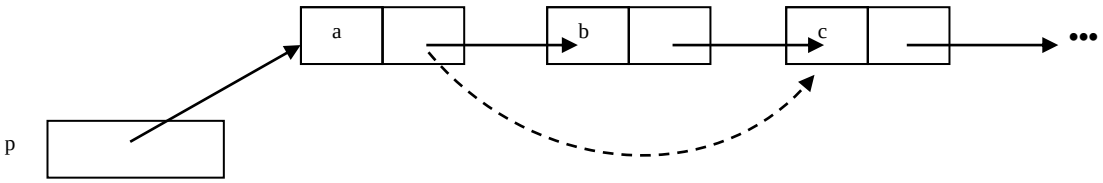


Diagrama de SUPRIME

Listas doblemente enzaladas

En algunas aplicaciones puede ser deseable recorrer eficientemente una lista, tanto hacia adelante como hacia atrás. O, dado un elemento, podría desearse determinar con rapidez el siguiente y el anterior. En tales situaciones, quizás se quisiera poner en cada celda de una lista un apuntador a la siguiente celda y otro a la anterior, como se sugiere en la lista doblemente enlazada de la siguiente figura.

Una lista doblemente enlazada.

Si se usan apuntadores se puede declarar que las celdas contienen un elemento y dos apuntadores, mediante

En Programación Estructurada

```
type
celda = record
      elemento: tipo_elemento;
       sig , ant: ↑celda
end;
 posicion = ↑ celda;
```

En Programación Orientada a Objetos

```
type
        celda=class
        private
           elemento: tipo_elemento;
           sig, ant: ^celda;
        end;
        lista=class;
                apt:^celda;
                posicion:^celda;
    end;
```

Enseguida mostramos un procedimiento para suprimir un elemento en la posición p en una lista doblemente enlazada.

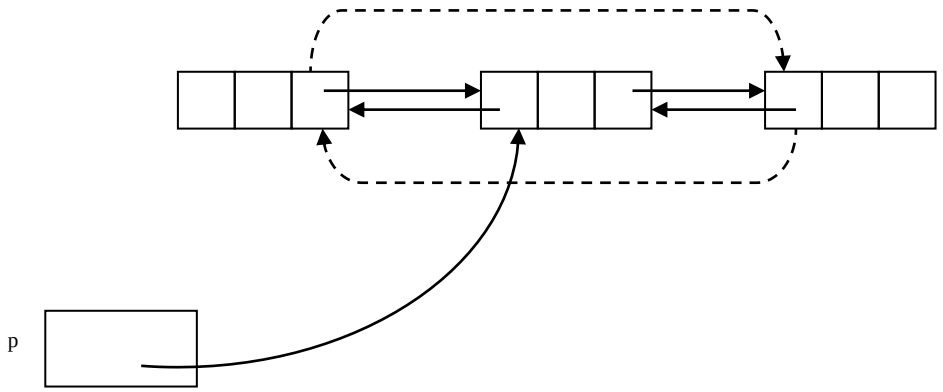
En Programación Estructurada

```
procedure  SUPRIME (var p:posicion );
begin
    if p↑.ant  <> nil then
        {la celda a suprimir no es la primera}
         p↑.ant ↑.sig:= p ↑.sig;
        if p↑.sig  <> nil then
                {la celda a suprimir no es la última}
                p↑.sig ↑.ant:= p ↑.ant;
end; {SUPRIME}
```

En Programación Orientada a Objetos

```
procedure Lista.Suprime(P:Posicion;);

begin
 if  p^.ant <> nil then
          P^.ant^.sig:=P^.Sig;
 if p^.sig <> nil then
          p^.sig^.ant:=p^.ant;
end;
```

En la siguiente imagen se muestra los cambios causados en los apuntadores por este procedimiento; los apuntadores anteriores se representan con líneas de trazo continuo, y los nuevos con líneas punteadas, en el supuesto de que la celda suprimida no es la primera ni la última. Primero se localiza la celda precedente, usando el campo *ant.* Se hace que el campo *sig* de esta celda apunte a la celda que sigue a la que ocupe la posición *p*. Después se hace que el que el campo *ant* de esta celda siguiente apunte a la celda que precede a la que ocupa la posición *p*. La celda apuntada por *p* queda sin usar y el sistema de ejecución dinámica de Pascal debe de usarla de nuevo automáticamente.



PILAS

Una *pila* es un tipo especial de lista en la que todas las inserciones y supresiones tienen lugar en un extremo denominado *tope.* A las pilas se les llama también <listas LIFO» *(last in first out)* o listas «último en entrar, primero en salir». El modelo intuitivo de una pila es precisamente una pila de fichas de póquer puesta sobre una mesa, o de libros sobre el piso, o de platos en una estantería, situaciones todas en las que sólo es conveniente quitar o agregar un objeto del extremo superior de la pila, al que se denominará en lo sucesivo «tope». Un tipo de datos abstracto de la familia PILA incluye a menudo las cinco operaciones siguientes:

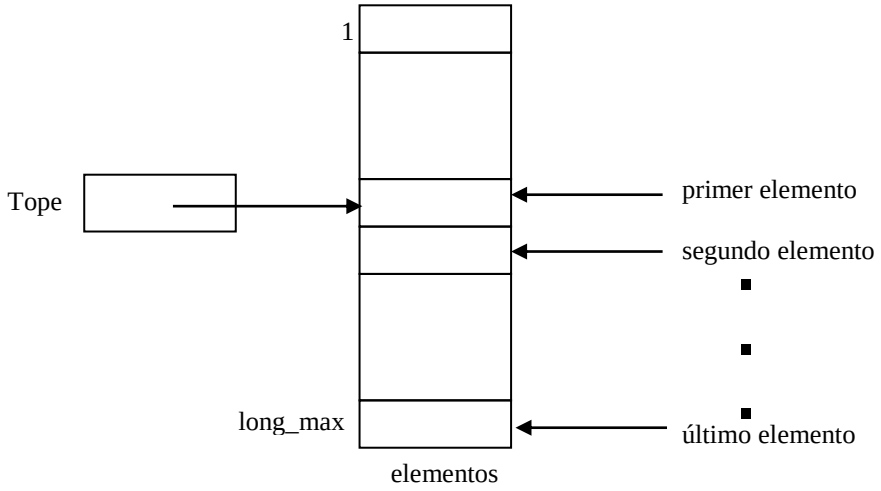
1. ANULA(P) convierte la pila *P* en una pila vacía. Esta operación es exactamente la misma que para las listas generales.
2. TOPE(P) devuelve el valor del elemento de la parte superior de la pila *P.* Si se identifica la parte superior de una pila con la posición 1, como suele hacerse, entonces TOPE(P) puede escribirse en función de operaciones con listas como RECUPERA(PRIMERO(P), *P).*
3. SACA(P), en inglés *POP*, suprime el elemento superior de la pila, es decir, equivale a SUPRIME(PRIMERO(P), *P).* Algunas veces resulta conveniente implantar SACA como una función que devuelve el elemento que acaba de suprimir, aunque aquí no se hará eso.
4. METE(x, *P),* en inglés *PUSH,* inserta el elemento *x* en la parte superior de la pila *P*. El anterior tope se convierte en el siguiente elemento, y así sucesivamente. En función de operaciones primitivas con listas, esta operación es INSERT*A(x,* PRIMERO *(P), P).*

5. VACIA(P) devuelve verdadero si la pila *P* está vacía, y falso en caso contrario.

Pilas basadas en arreglos.

Todas las realizaciones de listas descritas sirven para pilas, puesto que una pila con sus operaciones es un caso especial de lista. La representación de una pila como lista enlazada es sencilla, pues METE y SACA operan sólo con la celda de encabezamiento y con la primera celda de la lista. De hecho, los encabezamientos pueden ser apuntadores, más que celdas completas, puesto que para las pilas no existe la noción de «posición» que existe para las listas y no es necesario representar la posición 1, en forma análoga a las demás posiciones.

Un criterio para usar un arreglo es el que tiene en cuenta el hecho de que las inserciones y las supresiones ocurren sólo en la parte superior. Se puede anclar la base de la pila a la base del arreglo (el extremo de índice más alto) y dejar que la pila crezca hacia la parte superior del arreglo (el extremo de índice más bajo). Un cursor llamado *tope* indicará la posición actual del primer elemento de la pila. Obsérvese la siguiente imagen:



Realizacion de una pila mediante un arreglo.

Para esta realización de pilas basada en arreglos, el tipo de datos abstracto PILA se define:

En Programación Estructurada

```
type
   PILA = record
          tope: integer;
           elementos: array [1..long_max] of tipo_elemento
  end;
```

En Programación Orientada a Objetos

```
type
        pila=class
        private
            tope:integer;
            elementos:[1..long_max] of tipo_elemento;
    end;
```

Las cinco operaciones típicas con pilas se muestran enseguida:

En Programación Estructurada

```
procedure anula (var p:pila);
      begin
           p.tope := long_max +1
      end; {anula}

function vacia (p: pila): boolean;
      begin
          if p.tope > long_max  then
               return (true)
      else
               return  (false)
      end; {vacia}

function tope (var p:pila): tipo_elemento;
      begin
          if  vacia (p) then
             error  ('la pila está vacía')
          else
             return (p.elementos[p:tope])
       end; {tope}

procedure saca (var p:pila);
       begin
           if  vacia(p) then
                error ('la pila está vacía')
           else
                p.tope:= p.tope +1
         end; {saca}
```

```
procedure mete (x: tipo_elemento; p:pila);
        begin
            if p.tope := 1 then
                    error  ('la pila está llena')
            else begin
                 p.tope := p:tope-1;
                 p.elementos[p:tope]:= x
            end
          end {mete}
```

En Programación Orientada a Objetos

```
procedure pila.anula( var p:pila);
begin
  p.tope:= long_max + 1
end;

function pila.vacia( p:pila): boolean;
begin
  if p.tope> long_max then
    result := true;
  else
    result:=false;
end;

function pila. tope(var  p:pila): tipo_elemento;
begin
  if p.vacia then
    error("la pila está vacia);
  else
    result:=p.elementos[p.tope];
end;

procedure pila.saca(var p:pila);
begin
  if p.vacia then
    error("la pila está vacia");
  else
    p.tope:=p.tope+1;
end;

procedure pila.mete(x:tipo_elemento; p:pila);
begin
  if p.tope=1 then
    error("la pila está llena");
  else begin
            p.tope:=p.tope-1;
             p.elementos[p.tope]:=x
          end;
end;
```

COLAS.

Una *cola* es otro tipo especial de lista en el cual los elementos se insertan en un extremo (el *posterior)* y se suprimen en el otro (el *anterior* o *frente).* Las colas se conocen también como listas «FIFO» *(first-in first-out)* o listas «primero en entrar, primero en salir». Las operaciones para una cola son análogas a las de las pilas; las diferencias sustanciales consisten en que las inserciones se hacen al final de la lista, y no al principio, y en que la terminología tradicional para colas y listas no es la misma. Se usarán las siguientes operaciones con colas.

1. ANULA(C) convierte la cola C en una lista vacía.
2. FRENTE(C) es una función que devuelve el valor del primer elemento de la cola C. FRENTE(C) se puede escribir en función de operaciones con listas, como RECUPERA(PRIMERO(C), C).
3. PONE_EN_COLA(x, C) inserta el elemento *x* al final de la cola C. En función de operaciones con listas, PONE_EN_COLA(x, C) es INSERTA(x, FIN(C), C).
4. QUITA_DE_COLA(C) suprime el primer elemento de C; es decir; QUITA-DE-COLA(C) es SUPRIME(PRIMERO(C), C).
6. VACIA(C) devuelve verdadero si, y sólo si, C es una cola vacía.

Realización de Colas basadas en apuntadores

Para aumentar la eficacia de PONE_EN_COLA es posible aprovechar el hecho de que las inserciones se efectúan sólo en el extremo posterior. En lugar de recorrer la lista de principio a fin cada vez que se desea hacer una inserción, se puede mantener un apuntador al último elemento. Como en las listas de cualquier clase, también se mantiene un apuntador al frente de la lista; en las colas, ese apuntador es útil para ejecutar mandatos del tipo FRENTE o QUITA_DE_COLA. En Pascal, se utilizará una celda ficticia como encabezamiento y se tendrá el apuntador frontal dirigido a ella. Esta convención permite manejar convenientemente una cola vacía.

Para la realización basada en apuntadores, se definirán las celdas como antes:

```
type
    celda = record
          elemento: tipo_elemento;
           sig:↑celda
    end;
```

En Programación Orientada a Objetos

```
type
    celda = class
    private
          elemento: tipo_elemento;
           sig:^celda
    end;
```

Es posible definir una cola como una estructura que consiste en apuntadores al extremo anterior de la lista y al extremo posterior. La primera celda de una cola es una celda de encabezamiento cuyo campo elemento se ignora. Como se mencionó, esta convención permite representar de manera simple una cola vacía. Se define:

En Programación Estructurada

```
type
    cola = record
    ant, post: ↑celda
end;
```

En Programación Orientada a Objetos

```
type
                cola=Class
                private
                   ant,post:^celda;
               end;
```

Más adelante se muestran los programas para las cinco operaciones fundamentales de colas.

- En ANULA, la primera proposición *new(C.ant)* crea una variable de tipo *celda* y asigna su dirección a *C.ant.* La segunda proposición coloca **nil** en el campo *sig* de esa celda. La tercera proposición hace que el encabezamiento quede como primera y última celda de la cola.
- El procedimiento QUITA-DE-COLA(C) suprime el primer elemento de C desconectando el encabezado antiguo de la cola. El primer elemento de la lista se convierte en la nueva celda ficticia de encabezamiento.

En Programación Estructurada

```
procedure anula (var c:cola);
begin
        new(c.ant);    {crea la celda de encabezamiento}
        c.ant↑.sig:= nil;
        c.post:=c.ant {el encabezamiento es la primera  y la última celdas}
end; {anula}

function vacia (c:cola):boolean;
begin
        if c.ant=c.post then
                return (true)
        else
        return (false)
end {vacia}
```

```
function frente(c:cola):tipo_elemento;
begin
        if vacia( c) then
                error('la cola está vacia');
        else
                return (c.ant↑.sig↑.elemento)
end; {frente}

procedure pone_en_cola (x:tipo_elemento; var c:cola);
begin
        new(c.post↑.sig); {agrega una nueva celda en el extremo posterior de la cola}
        c.post:=c.post↑.sig;
        c.post↑.elemento:=x;
        c.post↑.sig:=nil
end; {pone_en_cola}

procedure quita_de_cola (var c:cola);
begin
        if vacia(c ) then
                error ('la cola esta vacia')
        else
                c.ant:=c.ant↑.sig
end; {quita_de_cola}
```

En Programación Orientada a Objetos

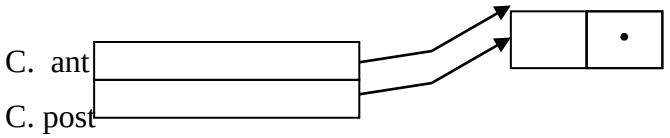
```
procedure cola.anula (var c:cola);
begin
        new(c.ant);
        c.ant^.sig:= nil;
        c.post:=c.ant
end; {anula}
```

```
function cola.frente(c:cola):tipo_elemento;
begin
        if cola.vacia( c) then
                error('la cola está vacia');
        else
                result:=c.ant^.sig^.elemento
end;

procedure cola.pone_en_cola (x:tipo_elemento; var c:cola);
begin
        new(c.post^.sig);
        c.post:=c.post^.sig;
        c.post^.elemento:=x;
        c.post^.sig:=nil
end;

procedure cola.quita_de_cola (var c:cola);
begin
        if vacia(c ) then
                error ('la cola esta vacia')
        else
                c.ant:=c.ant^.sig
end;
```

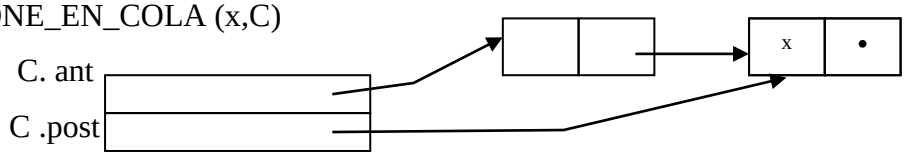
Implementación de operaciones para colas

En la siguiente imagen se muestran los resultados originados por la sucesión de mandatos ANULA(C), PONE_EN_COLA(x, C), PONE_EN_COLA(y, C), QUITA_DE_COLA(C). Obsérvese que después de la supresión, el elemento *x* ya no se considera parte de la cola, por estar en el campo elemento de la celda de encabezamiento.

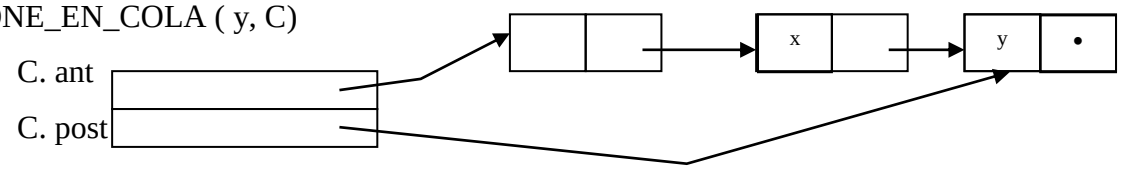
ANULA (C)



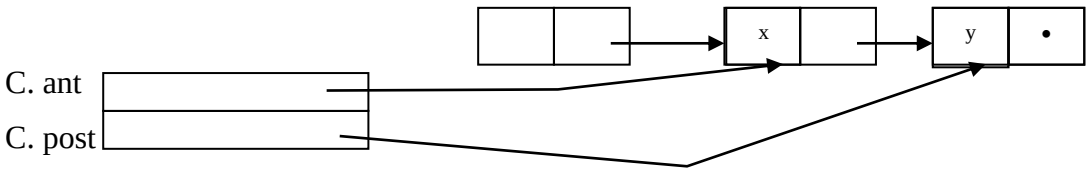
PONE_EN_COLA (x,C)



PONE_EN_COLA (y, C)



QUITA_DE_COLA (C)

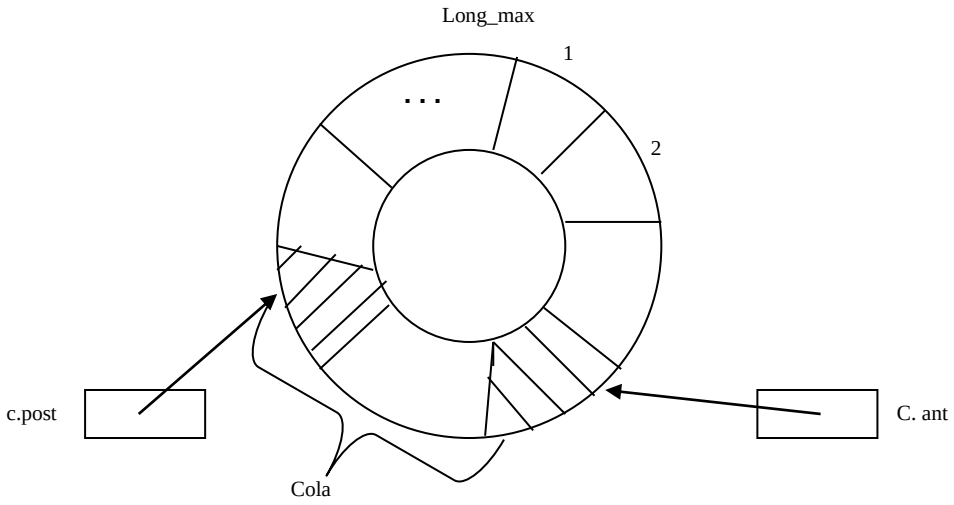


Sucesión de operaciones con colas

Realización de colas con arreglos circulares

Es cierto que con un apuntador al último elemento es posible ejecutar PONE-EN-COLA en un número fijo de pasos, pero QUITA-DE-COLA, que suprime el primer elemento, requiere que la cola completa ascienda una posición en el arreglo. Así pues, QUITA-DE-COLA lleva un tiempo $\Omega(n)$ si la cola tiene longitud n.

Para evitar este gasto, se debe adoptar un punto de vista diferente. Imagínese un arreglo como un círculo en el que la primera posición sigue a la última, en la forma sugerida en la siguiente imagen:



La cola se encuentra en alguna parte de ese círculo, ocupando t posiciones consecutivas, con el extremo posterior en algún lugar a la izquierda del extremo anterior. Para insertar un elemento en la cola, se mueve el apuntador *C.post* una posición en el sentido de las manecillas del reloj, y se escribe el elemento en esa posición. Para suprimir, simplemente se mueve *C.ant* una posición en el sentido de las manecillas del reloj. De esta manera, la cola se mueve en ese mismo sentido conforme se insertan y suprimen elementos.

Obsérvese que utilizando este modelo los procedimientos PONE-EN-COLA y QUITA-DE-COLA se pueden escribir de tal manera que su ejecución se realice en un número constante de pasos.

Hay una sutileza que surge en la representación de la imagen anterior y en cualquier variante menor de esta estrategia (es decir, si *C.post* apunta a una posición adelantada con respecto al

último elemento, en el sentido de las manecillas del reloj, y no a ese mismo elemento). El problema reside en que no hay manera de distinguir entre una cola vacía y una que ocupa el círculo completo, a menos que se mantenga un bit que sea verdadero si, y sólo si, la cola está vacía. Si no se está dispuesto a mantener ese bit, se debe evitar que la cola llegue a llenar todo el arreglo.

Para comprender la razón de esto, supóngase que la cola de la imagen anterior tiene *long-máx* elementos. Entonces *C.post* debería apuntar una posición adelante de *C.ant* en el sentido contrario al de las manecillas del reloj. ¿y qué ocurriría si la cola estuviera vacía? Para ver cómo se representa una cola vacía, considérese primero una que tenga un solo elemento. Entonces *C.ant* y *C.post* apuntan a la misma posición. Si en estas condiciones se suprime ese elemento, *C.ant* avanza una posición en el sentido de las manecillas del reloj, formando una cola vacía. Así, en una cola vacía, *C.post* está a una posición de *C.ant* en sentido contrario al de las manecillas del reloj; es decir, está exactamente en la misma posición relativa que ocuparía si la cola tuviera *long-máx* elementos. Resulta evidente, pues, que aun cuando el arreglo tenga *long-máx* lugares, no se puede permitir que la cola crezca más que *long-máx-1*, a menos que se introduzca un mecanismo para distinguir las colas vacías.

Formalmente, las colas las podemos definir de la siguiente manera:

En Programación Estructurada

```
type
    Cola = record
            elementos:array[1..long_max] of tipo_elemento;
            ant, post: integer;
    end;
```

En Programación Orientada a Objetos

```
type
    Cola=class
    private
       ant,pos:integer;
       elementos:[1..long_max] of tipo_elemento;
    end;
```

Los algoritmos más comunes para las colas los definiremos en seguida. La función suma_uno(i) suma uno a la posición i, en el sentido circular.

En Programación Estructurada

```
function suma_uno (i: integer): integer;
      begin
          return ((i mod long_max) +1)
      end; {suma_uno}

procedure vacia (var  c:cola): boolean;
        begin
            if  suma_uno(c.post) = c. ant then
                return (true)
            else
                return (false)
        end; {VACIA}

function frente (var c: cola ) : tipo_elemento;
     begin
         if vacia (c) then
              error ( ' la cola está vacía')
         else
              return (c.elementos [c.ant])
      end; {FRENTE}

procedure pone_en_cola (x: tipo_elemento; var c:cola);
        begin
             if suma_uno(suma_uno(c.post)) = c.ant then
                  error ('la cola está llena')
             else begin
                  c.post := suma_uno(c.post);
                  c.elementos [c.post] := x
             end
         end; {PONE_EN_COLA}

procedure quita_de_cola (var c: cola);
        begin
             if vacia (c) then
                  error (' la cola está vacía')
             else
                  c.ant := suma_uno(c.ant)
        end; {QUITA_DE_COLA}
```

En Programación Orientada a Objetos

```
function cola.suma_uno( i:integer): integer;
begin
   result := (i mod long_max)+1;
end;

procedure cola.vacia(var c:cola): boolean;
begin
  if c.suma_uno(c.post)=c.ant then
    result := true;
  else
    result:=false;
end;

function cola. frente(var  c:cola): boolean;
begin
  if c.vacia then
    error("la cola está vacia);
  else
    result:=c.elementos[c.ant];
end;

procedure cola.quita(var c:cola);
begin
  if c.vacia then
    error("la cola está vacia");
  else
    c.ant:=cola.suma_uno(c.ant);
end;

procedure cola.pone_en_cola(x:tipo_elemento; var c:cola);
begin
  if cola_suma_uno(c.post)=c.ant then
    error("la cola está llena");
  else begin
          c.post:=c.suma_uno(c.post);
              c.elementos[c.post]:=x
        end;
end;
```

SITUACION ACTUAL

Actualmente, en la impartición de la materia de Estructuras de Datos, los profesores no alcanzan a culminar todo el plan de estudio de la materia, debido a que se debe contemplar una infinidad de estructuras con todas sus derivaciones, implementaciones y operaciones que conlleva cada una. Esto ha ocasionado que el alumno no asimile y ejercite las estructuras de datos como debe ser. Se ha originado que el alumno no tenga la habilidad y la lógica bien desarrollada para resolver una problemática, y es por demás decir que esto se adquiere con la práctica y el buen entendimiento de los métodos de cada una de las estructuras de datos.

Esta situación afecta a semestres más adelante en materias que llevan como requisito la de estructuras de datos. A la hora de poner a los alumnos a desarrollar un determinado programa no lo saben resolver. La materia de Estructuras de Datos es elemental porque aquí refuerzan los conocimientos de la materia de Introducción a la programación y además toman cierta habilidad a la hora del diseño y la programación que más adelante necesitarán con mayor fuerza.

La materia de Estructuras de Datos hoy en día se está impartiendo la mayor parte teóricamente, y se les proporcionan pocos ejemplos prácticos de las operaciones elementales de las estructuras básicas de datos y es mínima la práctica en los laboratorios por parte de la clase o por parte de los mismos alumnos. No se les impulsa que ellos resuelvan en un gran porcentaje los problemas asociados a las estructuras de datos.

Una entrevista que se les hizo a alumnos de 3° de Lic. en Informática e Ing. en Computación y exalumnos sobre que les parecía(o) la materia de Estructuras de Datos; contenía las siguientes preguntas:

- Sí ellos creían que habían aprendido lo suficiente;
- Si creían que habían aprendido y ejercitado tanto en programación que cualquier problema que les pusiera lo resolverían;
- Si la materia se les hizo más teórica que práctica;
- Si practicaban lo suficiente, entre otras.

Todos a los que entrevistamos respondieron que las formas de impartir las clases siempre eran las mismas. Que la mayor parte de las clases eran teóricas; que veían muy pocos ejemplos de programas pero los que veían ya estaban resueltos y solo les explicaban como estaban hechos los programas; y a la hora que los ponían a programar, aunque fuera de otra materia ahí sí que se ponían a temblar porque no tenían la habilidad para desarrollar los programas. Los ejercicios y tareas de las estructuras de datos eran pocos por lo que no practicaban como debía ser todas las estructuras con sus diferentes implementaciones.

Esta situación nos obligó a diseñar formas de enseñanza alternativas que usaran el tiempo extraclase del alumno en las prácticas requeridas. Una de estas formas alternativas fue la elaboración de un software educativo que usó la multimedia como principal recurso para apoyar el aprendizaje.

Con este producto multimedia se pretende lograr los siguientes objetivos:

- Apoyar a los maestros para la impartición de la materia de Estructuras de Datos.
- Aminorar el tiempo de teoría con el uso del sistema multimedia que mostrará conceptos ejemplos en animaciones y códigos fuentes de las estructuras de datos que se contemplarán.
- El alumno podrá llevar un autoaprendizaje sin ayuda del profesor solo consultando la multimedia además podrá ser fuera del aula de clases.
- El alumno podrá tener una visión más amplia de los temas a tratar.
- Las clases serán más dinámicas, menos aburridas, interesantes al tiempo de la visualización de los resultados que serán con animaciones y representaran el funcionamiento de todas las operaciones de las estructuras de datos contempladas.

Resumen del Capitulo.

En este capitulo se describen las estructuras de datos lineales con sus implementaciones en arreglos o apuntadores y también las operaciones fundamentales de cada una de estas estructuras.

Al final se fundamenta la necesidad del uso de un software educativo al fin de solventar las dificultades que se tienen día con día en las clases de Estructuras de Datos dentro de la institución.

Capitulo 3.

Diseño, características y uso de la Multimedia.

En este capitulo se plantea la ventaja de la herramienta empleada para la construcción de la multimedia, así como una descripción de las principales características del producto obtenido.

Características de la herramienta de Construcción.

La multimedia se diseñó con Macromedia Flash, el fabricante de este software es Macromedia Inc.. Flash, que actualmente se encuentra en su versión 5.0, se ha convertido en el programa estándar para la creación de sitios web vectoriales en los que se aplica sonido, interactividad, gráficos y animaciones y que se pueden visualizar en cualquier plataforma o navegador.

Para visualizar esos contenidos y animaciones, el usuario necesita un reproductor, es decir, una herramienta que le permita visualizar contenidos Flash a través de Internet usando su navegador habitual.

Se suele usar el Flash Player para visualizar los contenidos creados con Flash. Esta herramienta forma parte de los principales navegadores y sistemas operativos como Netscape Navigator, Internet Explorer, Real Player G2, Web TV, entre otros. Macromedia ha licenciado, sin cargo alguno, el código fuente del popular reproductor Flash para que pueda ser incluido libremente en las diferentes soluciones de software y hardware que vayan apareciendo en el mercado. De esta forma, Macromedia se ha asegurado que todas las plataformas y aparatos incorporen la capacidad de ejecutar contenidos Flash en las futuras versiones de los navegadores, los sistemas operativos, los portales y otros dispositivos cualesquiera.

Datos de interés
Fabricante: Macromedia Inc.
Versión más actual: 5.0
Tamaño de archivo: 206 K

Se utilizó Macromedia Flash V 5.0 por lo fácil que se puede diseñar multimedias de alta calidad, podemos exportar archivos, agregar sonido, y las imágenes que se utilizan no generan tanto peso a la multimedia, además que con ellas podemos crear fabulosas animaciones dinámicas y sobre todo puede incluir un alto grado de interactividad.

Un recorrido por la Multimedia.

Para utilizar esta multimedia deberá explorar el CD y al encontrar el icono de la aplicación figura 3.1 dar doble clic para ejecutarlo.

Figura 3.1

Al ejecutarse la multimedia aparecerá la interfaz inicial figura 3.2 de la aplicación donde se mostrará las opciones que el usuario podrá elegir como es INTRODUCCIÓN, CONCEPTOS, PROBLEMAS COTIDIANOS, ACTIVIDADES SUGERIDAS, GLOSARIO, AYUDA O SALIDA.

Figura 3.2 Interfaz Inicial de la Multimedia

Para elegir alguna de las opciones Figura 3.3 solo posicione el cursor del mouse sobre la opción a elegir haga clic;

Figura 3.3

Si el usuario elige la opción de **INTRODUCCION** figura 3.4, aparecerá la siguiente pantalla:

Figura 3.4 Introducción.

Donde se mostrará una descripción general del contenido de la multimedia.

Nota: En la mayoría de las pantallas podremos observar herramientas auxiliares figura 3.5 para el desplazamiento de contenidos. A continuación se describen:

Herramienta	Función
	Está es una barra de desplazamiento vertical, la cual se utiliza para desplazarnos hacia arriba o hacia abajo, con el objetivo de poder ver el contenido de los textos; que en este caso son más largos que el espacio proporcionado para desplegar la información.
	Esta herramienta nos ayuda a mostrar la siguiente página de información.
	Esta herramienta nos ayuda a mostrar la página anterior de información.

	Esta herramienta nos ayuda a pasar a la página anterior o a la siguiente página. Al aparecer está significa que hay páginas antes y después de la actual.
	Esta herramienta nos llevará a otros sitios recomendados para la consulta de temas relacionados con los tratados en está multimedia.

Figura 3.5 Herramientas auxiliares.

Al elegir la opción de **CONCEPTOS** figura 3.6 se muestra la siguiente pantalla.

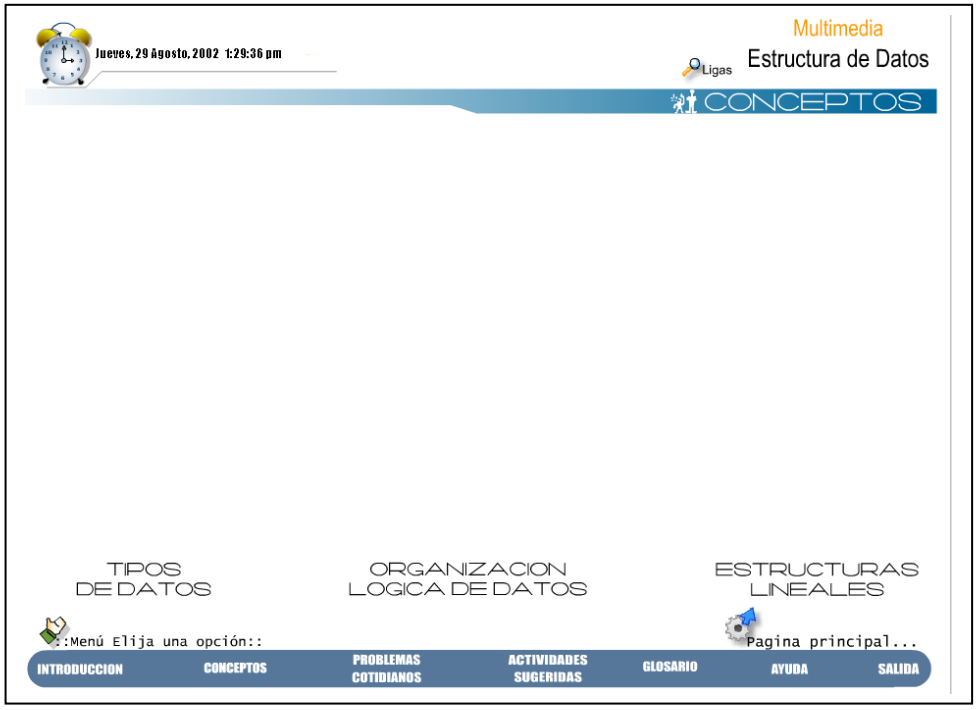


Figura 3.6 Conceptos.

Donde se mostrarán conceptos básicos relacionados a la materia de Estructuras de Datos. Aparecerán tres secciones como es **TIPOS DE DATOS, ORGANIZACIÓN LÓGICA DE DATOS Y ESTRUCTURAS LINEALES**.

Al seleccionar **TIPOS DE DATOS** figura 3.7 aparecerá la siguiente pantalla:

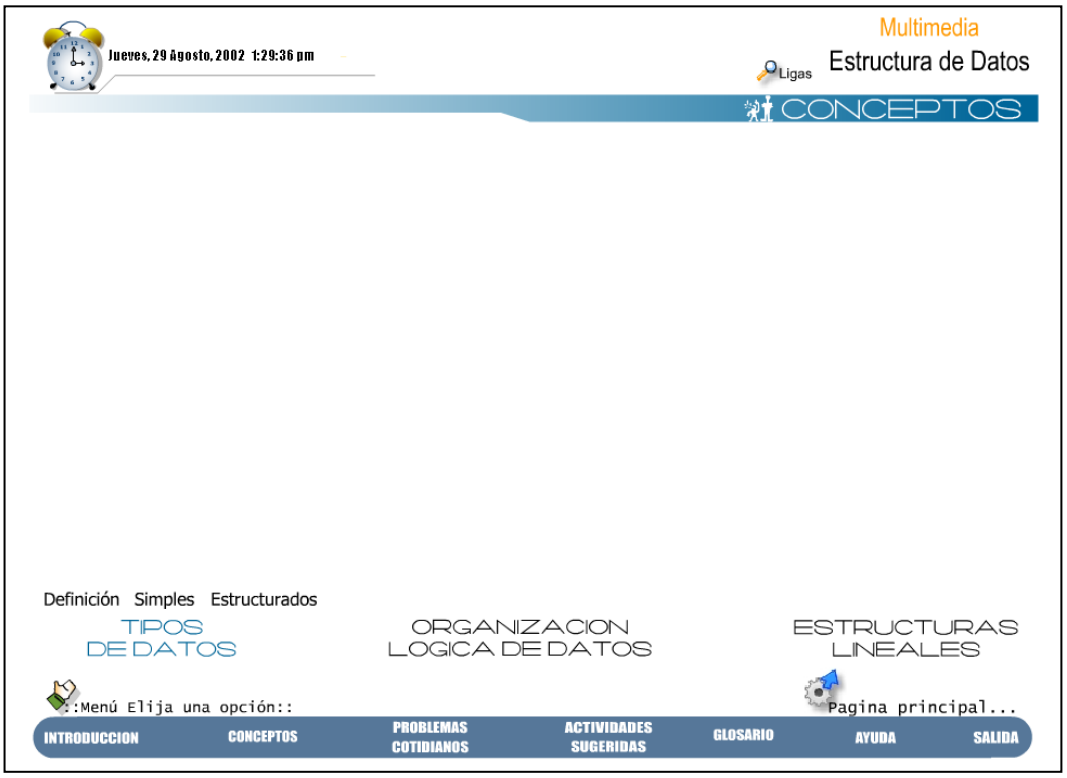


Figura 3.7 Tipos de Datos

De aquí se derivan tres opciones como es **Definición**, **Simples** y **Estructurados**, podrá elegir la opción deseada dando clic. Si el usuario elige accesar a la opción de **Definición** figura 3.8 se mostrará la siguiente pantalla:

Figura 3.8 Tipos de Datos / Definición

Aquí encontraremos conceptos como es la definición de una Estructura de Datos, información sobre los tipos de datos, ¿qué son?, su clasificación, etc.

En las siguientes opciones se describirán los tipos de datos **Simples y los Estructurados**.

Al seleccionar **ORGANIZACIÓN LÓGICA DE DATOS** figura 3.9 aparecerá la siguiente pantalla:

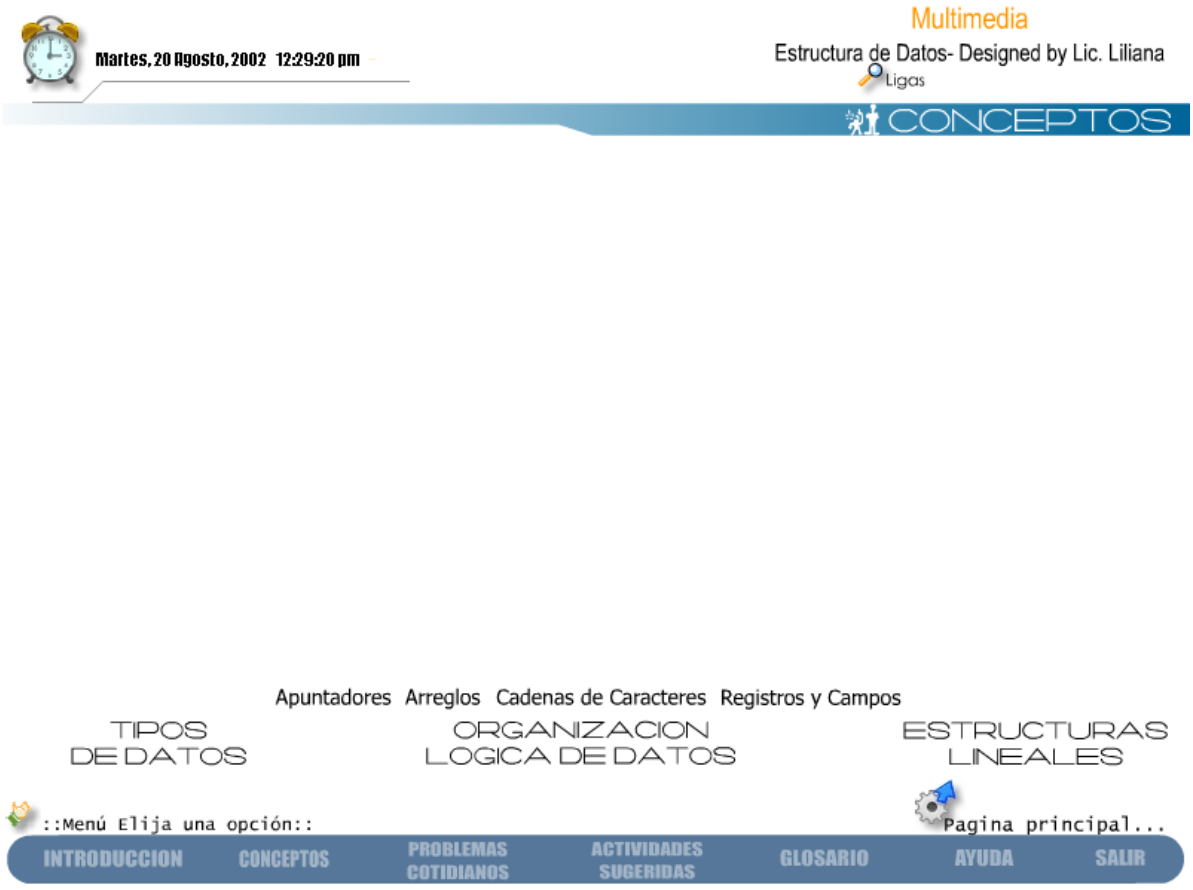


Figura 3.9. Selección Organización Lógica de Datos.

De aquí se derivan cuatro opciones como es **Apuntadores**, **Arreglos**, **Cadenas de Caracteres**, **Registros y Campos**, donde podrá seleccionar una, dando clic. Si por ejemplo el usuario elige accesar a la opción de **Apuntadores** (figura 3.10), se mostrará la siguiente ventana:

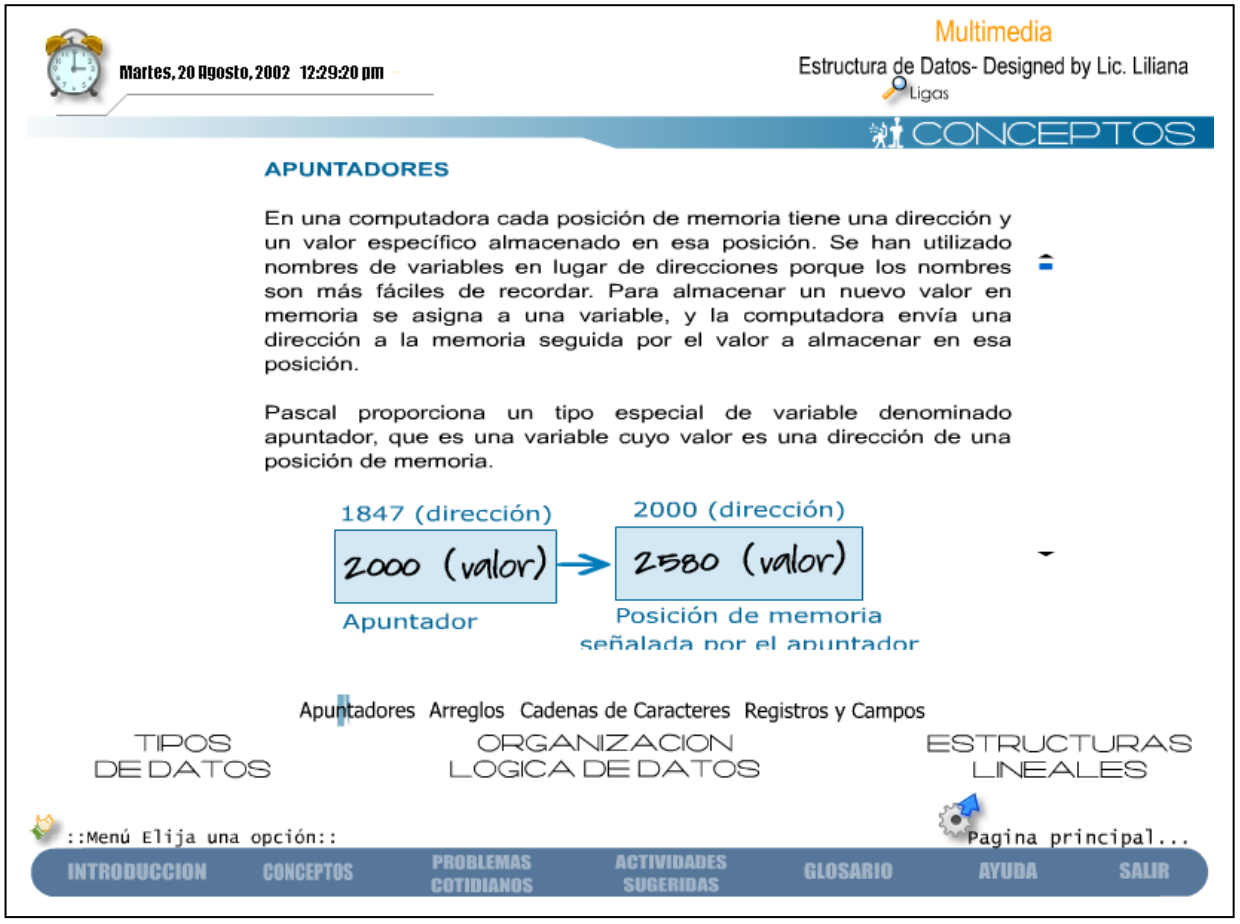


Figura 3.10 Apuntadores.

Aquí se muestran información sobre los *Apuntadores* como es su definición, ejemplos de programas que utilizan Apuntadores, ejemplos Gráficos, etc.

En las siguientes opciones se describirán los **Arreglos, Cadenas de Caracteres, Registro y Campos**, donde se describirán sus definiciones, características, ejemplos, etc.

Al seleccionar **ESTRUCTURAS LINEALES** y además la opción de **¿Qué son?** figura 3.11 aparecerá la siguiente pantalla:

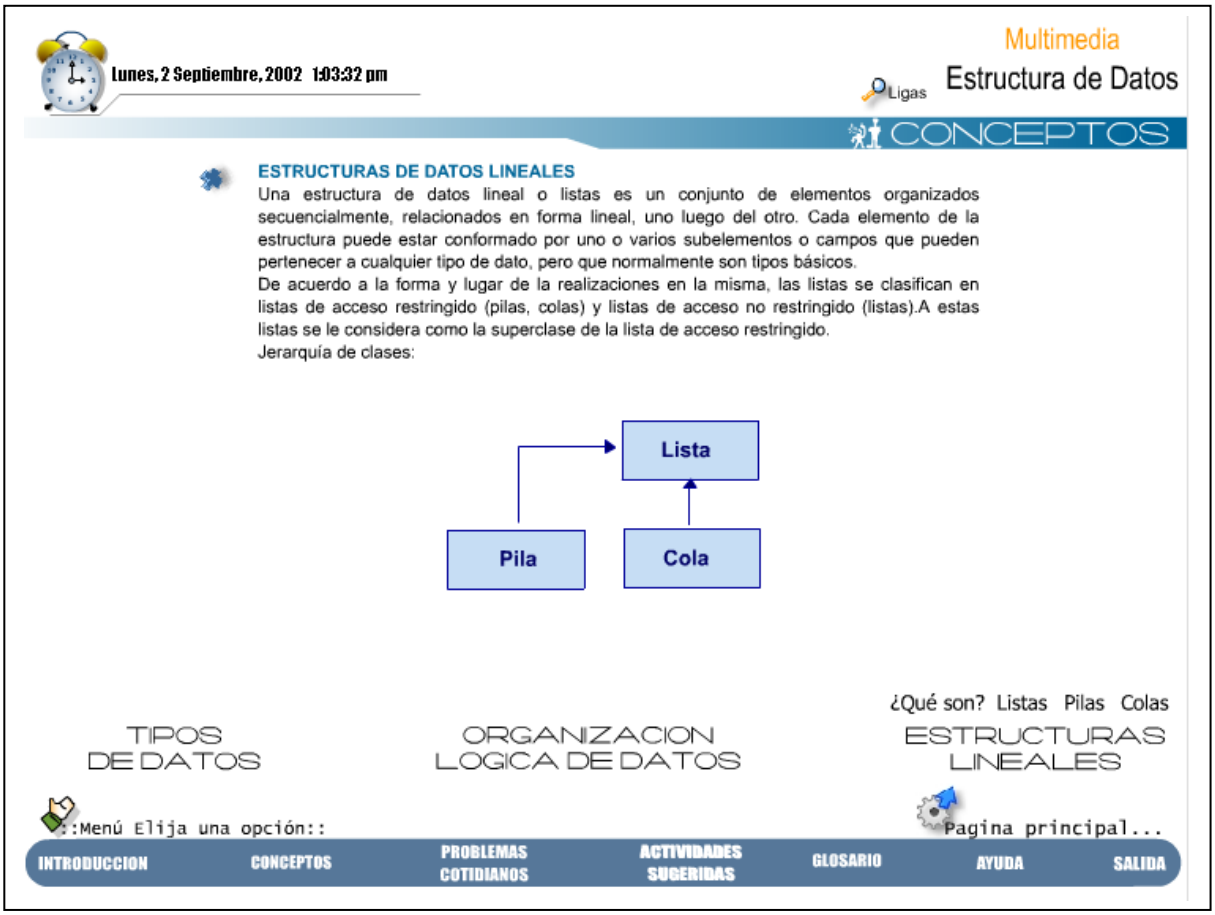


Figura 3.11. Estructuras Lineales.

También encontraremos otras opciones como es **Listas**, **Pilas** y **Colas**; donde se describen estas estructuras y se muestran algunos ejemplos.

Si el usuario elige la opción de **PROBLEMAS COTIDIANOS** figura 3.12, aparecerá la siguiente pantalla:

Figura 3.12 Problemas cotidianos.

De aquí se derivan tres opciones como es **Tráfico**, **Banco** y **Gallina**, podrá elegir la opción deseada dando clic. Si por ejemplo el usuario elige accesar a la opción de **Tráfico,** figura 3.13 se mostrará la siguiente pantalla:

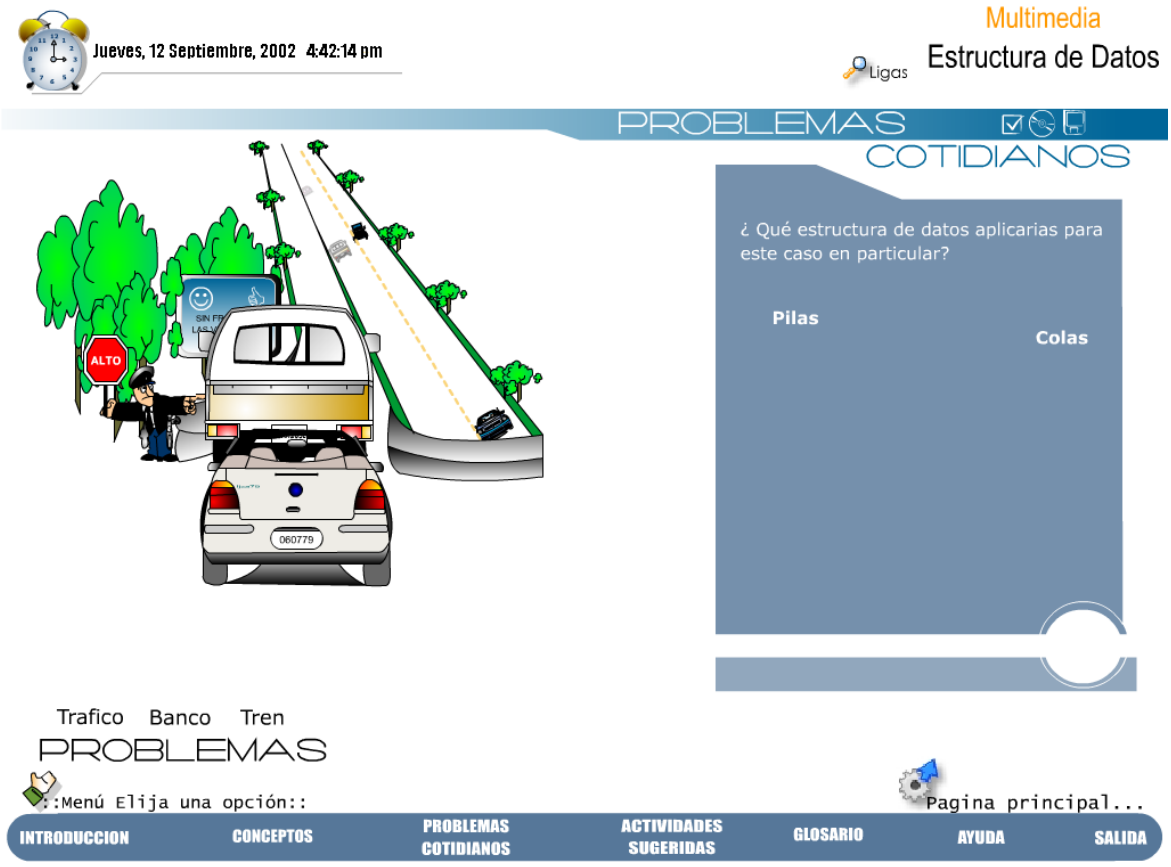


Figura 3.13 Problemas cotidianos/Tráfico.

Donde se muestra una animación referente al tráfico cotidiano. A un costado aparece una ventana con la pregunta ¿**Qué estructura de datos aplicarías para este caso en particular**? y da a elegir tres opciones **Listas Pilas o Colas**. Donde se debe seleccionar de acuerdo a la animación mostrada a que estructura corresponde. Una vez seleccionada la opción correcta aparecerá la pantalla siguiente Figura 3.14:

Figura 3.14. Problemas cotidianos/opciones de Programación

Una vez elegida la Estructura, da la opción para ver el código ya sea en **Orientada a Objetos o Estructurado**. Por ejemplo si el usuario elige **Orientada a objetos** figura 3.15 se mostrará la pantalla siguiente:

Figura 3.15 Código en Orientada a Objetos.

En la opción ACTIVIDADES SUGERIDAS (figura 3.16) se describirán ejercicios propuestos para que los alumnos ejerciten los conocimientos que adquirieron al utilizar la multimedia. Se puede elegir actividades de una de las tres opciones presentadas Listas Pilas o Colas.

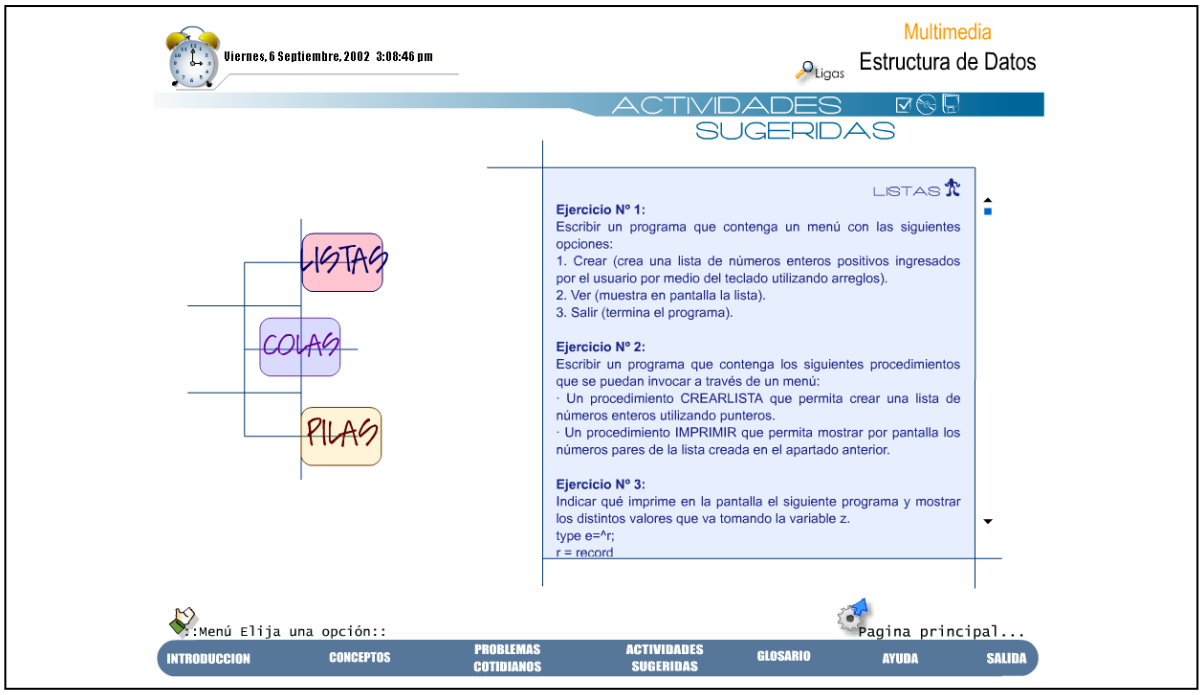


Figura 3.16. Actividades Sugeridas.

También encontraremos un GLOSARIO (figura 3.17) de términos, donde en la parte superior se encuentra un acceso a cada una de las Letras del abecedario sensibles al mouse, de forma que es muy sencillo e intuitivo encontrar algún concepto.

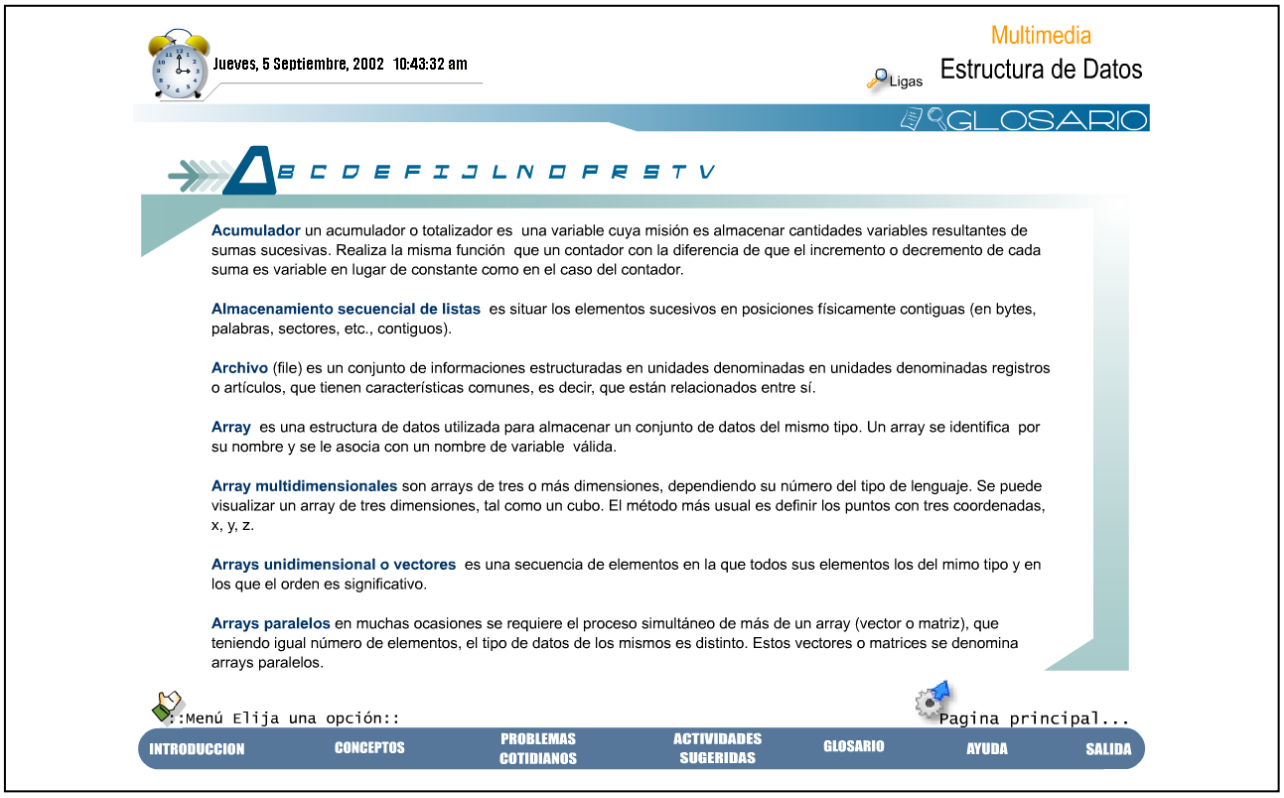


Figura 3.17 Glosario.

Otra opción del menú es la de AYUDA (figura 3.18) donde encontraremos tres opciones Manual de usuario, Bibliografía y Multimedias Auxiliares.

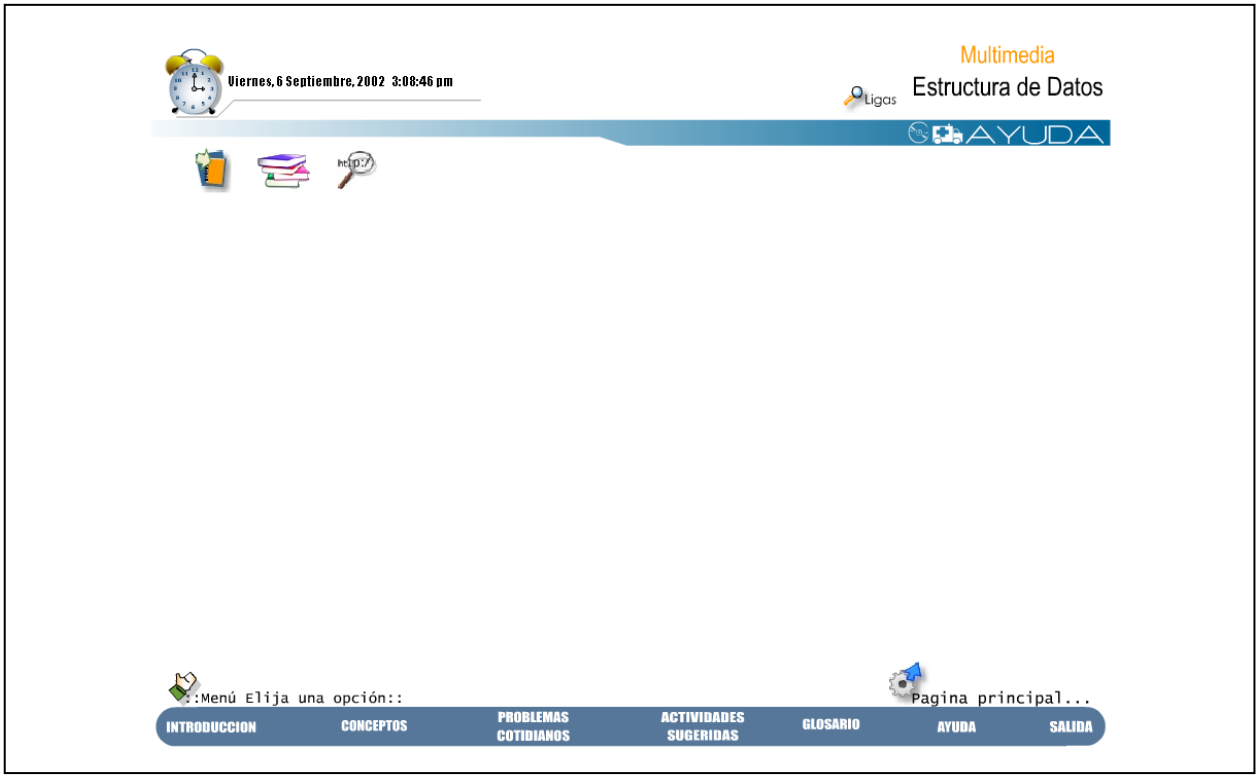


Figura 3.18. Ayuda.

Para accesar a cualquiera de las opciones pulse clic en la opción deseada. Al elegir Manual de Usuario encontraremos toda la información con la cual estamos definiendo la función de este sistema Multimedia. Documento que aparecerá en archivo con extensión HTML.

Al elegir Bibliografía (figura 3.19) nos mostrará información sobre los libros de donde se tomo información para formar esta multimedia.

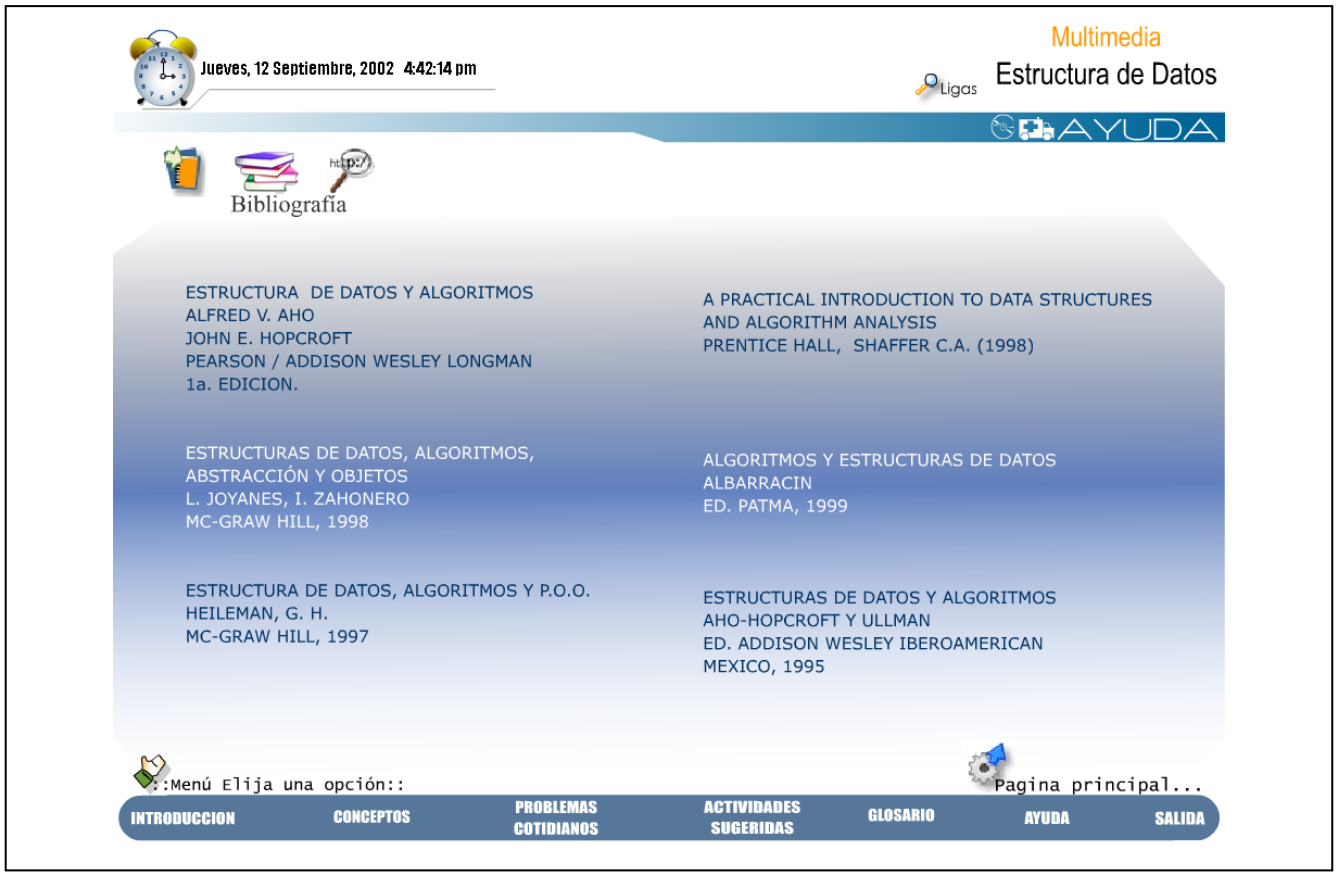


Figura 3.19. Bibliografía.

Al elegir Multimedias Auxiliares (figura 3.20) nos mostrará información referente a Multimedias que servirán de apoyo al alumnado que tenga dificultad para la programación estructurada y otra para los de Programación Orientada a Objetos. Estas Multimedias fueron diseñadas para la materia de Introducción a la Programación, que la materia de Estructuras de Datos la lleva como prerequisito y la de Programación Orientada a Objetos que va a la par de está asignatura. Y que para cursar la de Estructuras de Datos deben de tener en cuenta los conocimientos de estas multimedias.

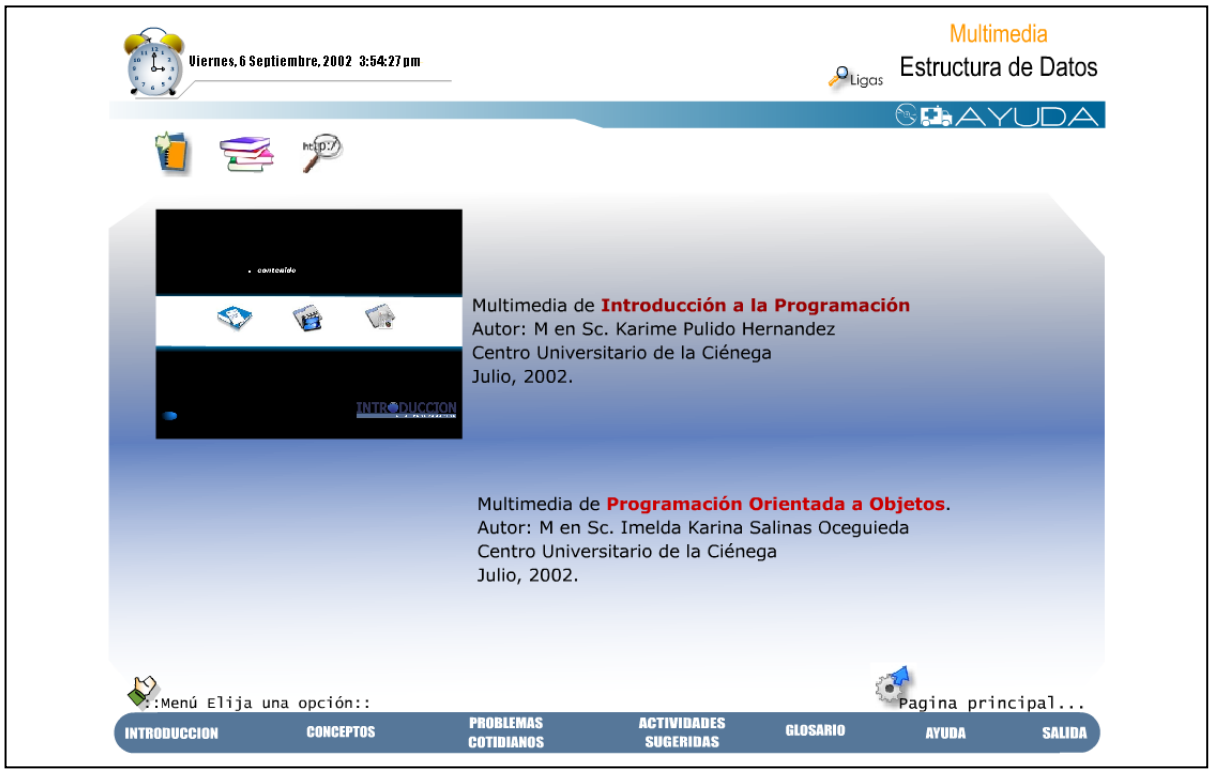


Figura 3.20. Multimedias Auxiliares.

Y por último tenemos la opción de SALIDA que con ella finalizamos la multimedia.

C O N C L U S I O N E S

La enseñanza de la asignatura de Estructura de Datos, en particular, la didáctica de los conceptos de Pilas y Colas, resulta compleja y presenta resultados deficientes en las carreras de Licenciatura en Informática e Ingeniería en Computación en nuestro Centro Universitario.

Existen en la actualidad numerosas formas basadas en el uso de la nuevas tecnologías que permiten apoyar el proceso de enseñanza-aprendizaje.

Se eligió la multimedia como herramienta dentro de esta forma alternativa y se logró implementar un producto el cual posee una organización de sus tópicos y que muestra un conjunto de ejemplos que facilitan el que el estudiante se apropie de forma más fácil de estos conocimientos.

R E C O M E N D A C I O N E S

- Complementar la multimedia con las Estructuras de Datos No Lineales.
- Incluir un módulo de autoevaluación para que se vayan evaluando los conocimientos por temas y también una evaluación general.
- Incluir una sección de voz donde, en vez de ver los contenidos, los escuchemos.
- Hacer una validación práctica de los resultados de esta tesis por medio de la aplicación de este tutorial a toda la matricula de nuestras carreras.

B I B L I O G R A F I A

[AHO 1995] Alfred V. Aho
Estructura de datos y algoritmos.
Pearson/ Addison Wesley Longman. 1a. Edición.

[AHO 1995] Aho-Hopcroft y Ullman.
Estructuras de Datos y Algoritmos.
Ed. Addison Wesley Iberoamerican.
México, 1995.

[ALB 1999] Albarracin.
Algoritmos y Estructuras de Datos.
ED. Patma, 1999.

[BOT 1999] © Andrea Botero - Kimera (Debogar & Cia)
La multimedia como facilitador en el proceso Educativo.
2do Encuentro Departamental de Informática Educativa
Villavicencio Sept. 30 - Oct. 1 1999

[FER 2000] Raúl R. Fernández Aedo; Pedro Mario Elianis Cepero Fadraga; El aprendizaje con el uso de las nuevas tecnologías de la información y las comunicaciones.
Universidad de Ciego de Avila.
OEI. Revista Iberoamericana de Educación. Año 2000

[GAR 2001] Juan Ignacio García Menendez. DIMAS.
Diseño de Módulos interactivos hipermediales para la explotación didáctica de los Mass Media.
Universidad de Oviedo. 2001

[HEI 1997] Heileman, G. H.
Estructura de datos, Algoritmos y Programación Orientada a Objetos.
Mc-Graw Hill, 1997.

[JOY 1998] L. Joyanes, I. Zanonero.
Estructuras de Datos Algoritmos Abstracción y objetos.
Mc-Graw Hill, 1998.

[NUÑ 1999] Gustavo Nuñez Esquel; Leonid Sheremetor.
Las Tecnologías avanzadas de información para la educación en el próximo Milenio.
Centro de Investigación en Computación. Instituto Politécnico Nacional Simposio Latinoamericano y del Caribe.
Aguascalientes, México 1999.

[SHA 1998] Shaffer C. A.
A Practical Introduction to data structures and algorithm Analysis. Prentice Hall, 1998.

[URR 2001] Maite Urretavizcaya Loinaz.
Sistemas Inteligentes en el ámbito de la Educación.
Revista Iberoamericana de I.A. No. 12 (2001)