ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION

"TUTOR DE ANALISIS Y DISEÑO DE SISTEMAS"

TESIS DE GRADO
Previa a la obtención del Título de
INGENIERO EN COMPUTACION

Presentada por
PATRICIA NARANJO GONZALEZ

GUAYAQUIL - ECUADOR
1996

A todas aquellas personas que colaboraron en la realización de este trabajo, y que confiaron, me apoyaron e incentivaron en todo momento.

Mil Gracias

DECANO DE LA FACULTAD DE
INGENIERIA ELECTRICA

DIRECTOR DE TESIS

MIEMBRO PRINCIPAL
DEL TRIBUNAL

MIEMBRO PRINCIPAL
DEL TRIBUNAL

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos profesionales de la ESPOL).

Patricia Naranjo González

A MIS PADRES

A MI ESPOSO

A MI HIJO

INDICE GENERAL

RESUMEN 7

INDICE GENERAL 5

INTRODUCCION 9

CAPITULO I

VISION GENERAL DE C++ Y WINDOWS

1.1 VISION GENERAL DE C++ 11

1.1.1 HISTORIA DE LOS LENGUAJES ORIENTADOS A OBJETOS 12

1.1.2 BORLAND C++ COMO LENGUAJE ORIENTADO A OBJETO 13

1.1.3 ESTRUCTURAS Y CLASES 14

1.1.4 HERENCIA MULTIPLE 15

1.1.5 SOBRECARGA DE FUNCIONES Y OPERADORES 16

1.1.6 CONSTRUCTORES Y DESTRUCTORES DE CLASES 16

1.1.7 FUNCIONES VIRTUALES Y CLASES VIRTUALES 18

1.2 VISION GENERAL DE WINDOWS

1.2.1 EL ADMINISTRADOR DE PROGRAMAS 19

1.2.2 EL ADMINISTRADOR DE ARCHIVOS 24

1.2.3 EL PANEL DE CONTROL 27

1.2.4 EL ADMINISTRADOR DE IMPRESION 29

1.2.5 AYUDA 32

CAPITULO II

SISTEMAS TUTORIALES

2.1 DEFINICION DE SISTEMAS TUTORIALES 34

2.2 SISTEMAS DE EJERCITACION Y PRACTICA 36

CAPITULO III

ANALISIS DEL SISTEMA TUTORIAL

3.1 BREVE EXPLICACION DEL SISTEMA

3.1.1 TEMAS DEL PROGRAMA DE ANALISIS Y DISEÑO 39

3.1.2 CARACTERISTICAS E INICIO DE LECCIONES 39

3.1.3 CARACTERISTICAS E INICIO DEL TEST 40

3.2 DETERMINACION DE REQUERIMIENTOS 41

3.3 JERARQUIA DE CLASES DE OBJECT WINDOWS 42

3.4 IDENTIFICACION DE CLASES Y OBJETOS DEFINIDOS 48

CAPITULO IV

DISEÑO E IMPLEMENTACION DEL TUTOR

4.1 ESTRUCTURA LOGICA DEL SISTEMA 69

4.2 DISEÑO MODULAR DEL SISTEMA 71

4.3 DISEÑO DE PANTALLAS Y MENUES DEL PROGRAMA 73

4.4 TIPO DE REPORTES 74

CONCLUSIONES Y RECOMENDACIONES 77

BIBLIOGRAFIA 80

ANEXOS 81

RESUMEN

En el presente proyecto ***"Tutor de Análisis y Diseño de Sistemas"*** se aplican técnicas que proporcionan una herramienta útil como lo constituyen los ***"Sistemas Tutoriales"***, los cuales pueden ser usados como complemento del material educativo en el que el usuario, en este caso los estudiantes de la materia de ***"Análisis y Diseño de Sistemas"*** puedan tener una visión global de lo que comprende y al mismo tiempo asimilen en forma clara y sencilla los conceptos a tratar. Las personas a las cuales va dirigido el Tutor es básicamente alumnos de la Facultad de Computación, así como a personas que deseen tener conocimientos del Análisis y Diseño de Sistemas.

El contenido de este proyecto ofrece en su parte inicial ***"Capítulo I"*** una visión general e historia de las técnicas y herramientas de programación que se usaron.

Los fundamentos, características, definiciones de los Sistemas Tutoriales fueron consideradas en el ***"Capítulo II"***.

En el ***"Capítulo III "*** se presenta un análisis del Sistema Tutorial desarrollado, los requerimientos de software y hardware, los temas del programa de la materia de Análisis y Diseño de Sistemas que se consideraron, las características de las lecciones y test presentadas por el Tutor, y así como la identificación de clases y objetos definidos.

El diseño e implementación del tutor son presentados en el ***"Capítulo IV"*** por medio de diagramas ilustrativos que muestran la estructura lógica, el diseño modular, las pantallas y menúes del programa y así como el tipo de reportes creados.

Además se incluye como anexo un **"Manual del Usuario"** el cual presenta de una forma ilustrativa todas las pantallas y menúes que contiene el tutor, explicando cada uno de los procedimientos necesarios para que el usuario puede navegar a través de él sin dificultad.

INTRODUCCION

El propósito para desarrollar el Tutor de Análisis y Diseño de Sistemas es proporcionar para el Sistema Educativo una herramienta de alto nivel, que provea la suficiente motivación al estudiante de manera que considere relevante lo que está aprendiendo, por medio de un material de estudio estimulante y por respectivas evaluaciones de rendimiento.

Este Tutor cubrirá puntos básicos y claves del programa del curso de **Análisis y Diseño de Sistemas**, actualmente esta materia dejó de dictarse y fue reemplazada por Sistemas de **Información**, pero tutor puede ser utilizado con éxito en este curso, ofreciendo información necesaria y de fácil acceso a los estudiantes.

El tutor de Análisis y Diseño de Sistemas se lo ha desarrollado basándose en el Análisis **Orientado a Objeto** tomando como herramienta **Borland C++ V.3.1**, el cual nos provee de vías para la modularización del Sistema y nos facilitará la utilización de librerías para el diseño de la interfaz gráfica más amigable al usuario.

El Tutor presentará un Menú Principal con los capítulos correspondientes sobre la materia de Análisis y Diseño de Sistemas, el usuario deberá escoger el tema y luego seleccionar cualquiera de las opciones presentadas, como tomar una lección, realizar un test o conocer algún concepto o definición sobre el tema seleccionado.

Las lecciones presentadas por el Tutor son de tipo diapositivas muy ilustrativas y con color las cuales fueron elaboradas utilizando el software Power Point, cada lección posee sus propios botones de control para el avance y retroceso.

Las preguntas presentadas en el Test por el Tutor, son de tipo verdadero o falso, además se presenta un reporte conteniendo las evaluaciones de las respuestas del estudiante asignándole el respectivo puntaje, así como las respuestas correctas a dichas preguntas para el respectivo estudio.

CAPITULO I

VISION GENERAL DE C++ Y WINDOWS

1.1 VISION GENERAL DE C++

C++ es simplemente un lenguaje de programación orientado objetos, las caracteristicas de orientación a objetos de C++ estan interrelacionadas, de modo que es importante tener un conocimiento general de ellas antes de tratar de aprenderlas con detalle. La primera parte de este capitulo trata del origen del C++ y describe la programación orientada a objetos, además de conocimientos generales de Windows.

C++ es una versión expandida de C. C es flexible y potente y ha sido usado para crear algunos de los productos más importantes de software de los últimos 15 años. Sin embargo, cuando un proyecto excede un cierto tamaño, C alcanza sus limites, dependiendo del proyecto, un programa de 25000 a 100000 lineas se vuelve dificil de manejar. En 1980, Bjarne Stroustrup cuando estaba trabajando en Bell Laboratories en Murray Hill, New Jersey, encaminó este problema añadiendo varias extensiones dal lenguaje C, inicialmente se llamó **" C con clase "** en 1983 el nombre se cambió a C++. La mayoria de los añadidos hechos a C por Stroustrup soportan la programación orientada a objetos, referida a veces como **"OOP"** (del inglés: object - oriented -programing), que fue inspirado por el lenguaje orientado a objetos el **"Simula67"**, por lo tanto C++ es el lenguaje representa la combinación de dos poderosos métodos de programación.

C++ ha sido revisado dos veces desde su invención, una en 1985 y de nuevo en 1990. Cuando inventó el C++, Stroustrup sabía que era importante mantener el espíritu original de C incluido su eficiencia, flexibilidad y la filosofia de que el programador y

no el lenguaje es el que manda añadiendo al mismo tiempo soporte para la programación orientada a objetos, usando las palabras de Stroustrup, **"permite programas estructurados con la claridad, extensibilidad y facilidad de mantenimiento sin perdida de eficiencia".**

Aunque C++ se diseñó inicialmente para ayudar en la gestión de procesos muy extensos, no está limitado a este uso. En realidad, los atributos de orientación a objetos de C++ pueden aplicarse con eficiencia virtualmente a cualquier tarea de programación. No es raro ver usar C++ para proyectos tales como un editor, bases de datos, sistemas de archivo personales y programas de comunicaciones. También pueden construirse sistemas de software de alto rendimiento, ya que C++ compa... la efectividad de C.

1.1.1 Historia de los lenguajes orientados a objetos

Son varios los lenguajes que han contribuido a la evolución de los lenguajes orientado... a objetos de hoy. En la década de los años 50 surge el LISP, en los años 60 el SIMULA y más tarde Pascal, C, Modula y Ada. Si bien, estos lenguajes no incluyen mecanismos para la programación orientada a objetos, sus características sirvieron de base para la construcción de estos mecanismos, por ejemplo el SIMULA contribuye con el concepto de clase.

En la década de los años 70, nace Smaltalk como un lenguaje orientado a objeto puro, con lo que queda introducido definitivamente este tipo de programación. En los años siguientes los avances experimentados en la programación orientada a objetos fueron pocos.

En la década de los 80 cuando los avances son mayores, debido fundamentalmente a la disponibilidad de extensiones orientadas a objetos en dos de los lenguajes más populares C y Pascal. Esto da lugar a la aparición de los lenguajes orientados a objetos híbridos, entre los que destacan C++ y Pascal Orientado a Objetos. Estos lenguajes tienen una caracteristicas extremadamente importante y es que guardan la compatibilidad con sus antecesores.

1.1.2 Borland C++ 3.1 como lenguaje orientado a objeto

Con la programación orientada a objetos las ventajas que ofrece un lenguaje de programación orientado a objetos, son muchas tales que beneficia el desarrollo de software, ofreciendo desarrollo de modelos, por medio de la utilización de clases, además los programas gozan de caracteristicas como menos lineas de códigos, y sentencias de bifurcación, y módulos que son más comprensibles porque reflejan de una forma clara la relación existente entre cada concepto a desarrollar y cada objeto que interviene en dicho desarrollo. Por otra parte la caracteristica herencia, exclusiva de este tipo de programación, es una de las claves más importantes para reducir código de programación.

Uno de los lenguajes de programación que soporta la programación orientada a objetos es: **Borland C++ 3.1**, él cual incluye las siguientes caracteristicas de Programación:

- **C y C++**

  Ofrece todo el poder de C y C++. También provee librerias de clases en C++, más la primera implementación de templetes comerciales con una eficiente colección de clases.

- **Optimización Global**

  Puede programar en el estilo que ud. encuentre más conveniente, produciendo un pequeño y rápido código altamente eficiente.

- **Velocidad de compilación rápida.**

- **Microsoft Windows Programing**:

  Provee completo soporte para desarrollar aplicaciones Windows incluyendo DLLs. Además incluye el Resource Compiler, el Help Compiler, y el Resource Workshop.

- **Windows-hosted IDE**:

  Borland C++ para Windows IDE (Integrated Development Enviroment) permite editar, compilar y correr sus programas bajo Windows, por lo tanto ud. no necesita switcharse entre Windows y Dos para crear sus programas WIndows. Esto incrementa grandemente la productividad ya que ud. puede hacer todo esto bajo el ambiente Windows.

- **Help**:

  Ayuda en línea, con programas ejemplos para casi todas las funciones.

1.1.3 Estructuras y Clases

En C++ cada estructura o clase puede contener una colección de objetos de datos los cuales pueden ser tipos de datos bases, uniones u otras estructuras y clases. Las estructuras y clases pueden contener también funciones miembros.

Todo esto en el nivel de codificación de C++ , se refleja utilizando la palabra clave class para definir nuevas clases. La siguiente declaración, por ejemplo define la clase "conjunto":

```
class conjunto {  struct conjunto?_member
                  {  int  member;
                     set_member  *next;
                      set_member (int m, set_member *n); }  };
```

A menudo una clase se define como una subclase de otra clase existente (clase base).

Por ejemplo, podemos escribir la siguiente definición de clase:

```
class competitor : public business {
    // .....
    };
```

La declaración anterior signifia la clase competitor es una clase base de la clase business.

1.1.4 Herencia Múltiple

Una de las caracteristicas más poderosas de C++ es que las clases pueden ser derivadas de una o más clases simultáneamente. Cuando se declara la nueva clase heredará datos encapsulados y funciones miembros.

La notación es la siguiente:

```
class competitor : public business, public adversary
        {
            // ....
        };
```

1.2.5 Sobrecarga de funciones y operadores

En C++ se permite a las funciones y operadores que utilizan la misma llamada para operar sobre una variedad de clases o tipos de datos. La sobrecarga de operadores es un rasgo integrado de C++ en el sentido que los operadores matemáticos como (+) pueden ser utilizados para operaciones con enteros, punteros, números de punto flotante. Para definir nuevas funciones sobrecargadas, la palabra clave de sobrecarga es utilizada para especificar explícitamente nuevos nombres de funciones a ser usados con diferentes tipos de datos.

Por ejemplo, si deseamos declarar una función de impresión universal, podemos escribir:

print (int) , print (double), print (long), print(char *);

Una características útiles de C++ es la oportunidad de definir funciones que pueden tomar un número específico de argumentos. Para hacer ésto, se usa una elipsis para representar una lista indefinida de argumentos.Por Ejemplo:

int read (**char** * ...);

1.2.6 Constructores y Destructores con Clases

Los constructores y destructores de la clase base no son heredados por sus clases derivadas.

Constructores . Cuando una clase base tiene un constructor y una clase derivada también, al crear un objeto, se llama primero al constructor de la clase base, y cuando la ejecución de este finaliza, entonces se llama al constructor de la clase derivada.

- Después son llamados los constructores para las respectivas clases miembro.

- Y por último se ejecuta el cuerpo de la clase derivada, este orden se aplica recursivamente por cada constructor.

Destructores. Las reglas relativas a destructores en clases derivadas y clases base, son inversas a las aplicadas a los constructores. Esto es, los objetos de una clase son destruidos de arriba a abajo; primeramente es llamado el destructor de la clase derivada, después son llamados los destructores de las clases miembro y, por último, el destructor de la clase base. Como los destructores no pasan argumentos, no requieren una sintaxis especial. Ejemplo:

```
~ficha( ){delete referencia; delete titulo; }          // destructor

~ficha_2( ) {delete autor; }                           // destructor
```

En la función **main( )** del programa, hemos definido tres objetos: **obj, obj1** y obj2. Cuando finaliza la ejecución de ésta, los destructores son llamados para destruir los objetos creados, debido a que salimos del ámbito donde ellos son visibles.

Para destruir los objetos obj y obj2, primeramente se llama al destructor de la clase derivada, **~ficha_2( )** y a continuación al destructor de la clase base,

~ficha( ) ; esto para cada uno de ellos.

Para destruir el objeto obj1, sólo se llama al destructor de la clase base,

~ficha( ), por ser este objeto de esta clase.

1.1.7 Funciones Virtuales y Clases Virtuales

Las funciones virtuales ofrecen una forma ilimitada de enlace en tiempo de ejecución. Cuando una función miembro se declara como virtual, luego esa función puede ser redefinida en cualquiera de sus clases derivadas. Esto no difiere de una función sobrecargada, se podría pensar, porque en efecto, permiten que las funciones sean redefinidas en otras clases. Sin embargo, existe una importante diferencia. Las funciones sobrecargadas son realmente nombres sobrecargados, las cuales el compilador las "corta" en nombres diferentes. Las funciones virtuales usan realmente estructuras de datos adicionales para proveer un enlace entre versiones diferentes de una función. Otra diferencia es que cuando una función virtual es redefinida, tiene que adherirse a la forma de la función original. Tiene que llevar el mismo número de argumentos del mismo tipo.

Las clases virtuales no tienen realmente nada que ver con las funciones virtuales.

Ellas son el resultado de una necesidad de poder alterar el mecanismo normal de herencia múltiple en C++. Ordinariamente, si una clase se hereda a partir de más de una clase que tienen un ancestro común, esa clase se comportará como si contuviera copias múltiples de las funciones miembros y datos comunes del ancestro. Sin embargo, si un ancestro es declarado como virtual, luego una sola copia de éste es referenciada en los descendientes, sin importar cuantas veces aparezca indirectamente en la derivación de clase.

1.2 VISION GENERAL DE WINDOWS

Windows se ha convertido en el ambiente usuario para la mayoría de aplicaciones importantes tales como Excel, Pagemaker, Designer, Ami, Corel Draw. La

programación de la interfase en un ambiente como Windows se ha convertido en una de las partes que más consume tiempo en la tarea de un programador. La programación orientada a objeto es particularmente valiosa para minimizar esta tendencia ya que tiene la habilidad para evitar el código redundante reutilizando las partes de un programa y personalizando código existente. Windows facilita la programación orientada a objeto.

Inicio de Windows

Una vez instalado en la computadora, Windows podrá iniciarse de la siguiente forma:

En el prompt del DOS después de **C>** ,escriba la palabra **win** y luego presione la tecla **<enter>:**

C> win (enter)

En el contexto de Windows, existen algunos términos básicos con los que conviene estar familiarizado. El trabajo se realiza en áreas rectangulares de la pantalla denominadas ventanas. Estas ventanas aparecen sobre un fondo denominado escritorio. Las aplicaciones con las cuales se trabaja (por ejemplo, el procesador de textos o la hoja de cálculo) se representan en Windows mediante pequeños símbolos gráficos denominados iconos.

En esta sección del capítulo se tratará de enfocar las características, funcionamiento de la versión de Windows 3.1.

1.2.1 El Administrador de Programas (Program Manager)

Cuando Windows se inicia, lo primero que se visualiza es el Administrador del Programa, el cual continúa activo durante el tiempo que se use Windows.

Inicialmente, la ventana del Administrador del Programas despliega los contenidos del grupo principal. Un grupo es una colección de aplicaciones que puede correr con Windows. Los nombres e iconos para las aplicaciones son desplegados en una ventana de grupos.

La ventana del Administrador de Programas también despliega iconos para otros grupos de aplicaciones como: Accesorios, Juegos e Inicio.

La ventana del Program Manager también puede contener otro grupo de iconos llamados Aplicaciones. Durante la inicialización de Windows, las aplicaciones ya instaladas en su computadora son inicializadas para usar con Windows. La ventana del grupo de Aplicaciones contiene iconos para estas aplicaciones.

Ud puede realizar muchas importantes tareas usando el Program Manager. Por ejemplo:

- Empezar aplicaciones
- Organizar sus aplicaciones en grupos para un fácil acceso.
- Salir de Windows

La siguiente ilustración muestra algunas de las características descritas anteriormente cuando se inicia Windows. (figura # 1)

Empezando una aplicación

Desde el Program Manager, ud puede fácilmente empezar una aplicación usando el mouse con un doble click en su icono de item-programa. Los iconos para muchas aplicaciones están en grupos mayores que el grupo principal., así que se puede

necesitar abrir otro grupo de ventana para encontrar el icono que se desea. Por ejemplo: se puede abrir el grupo de Accesorios para empezar el Bloc de Notas de Windows (figura # 2).

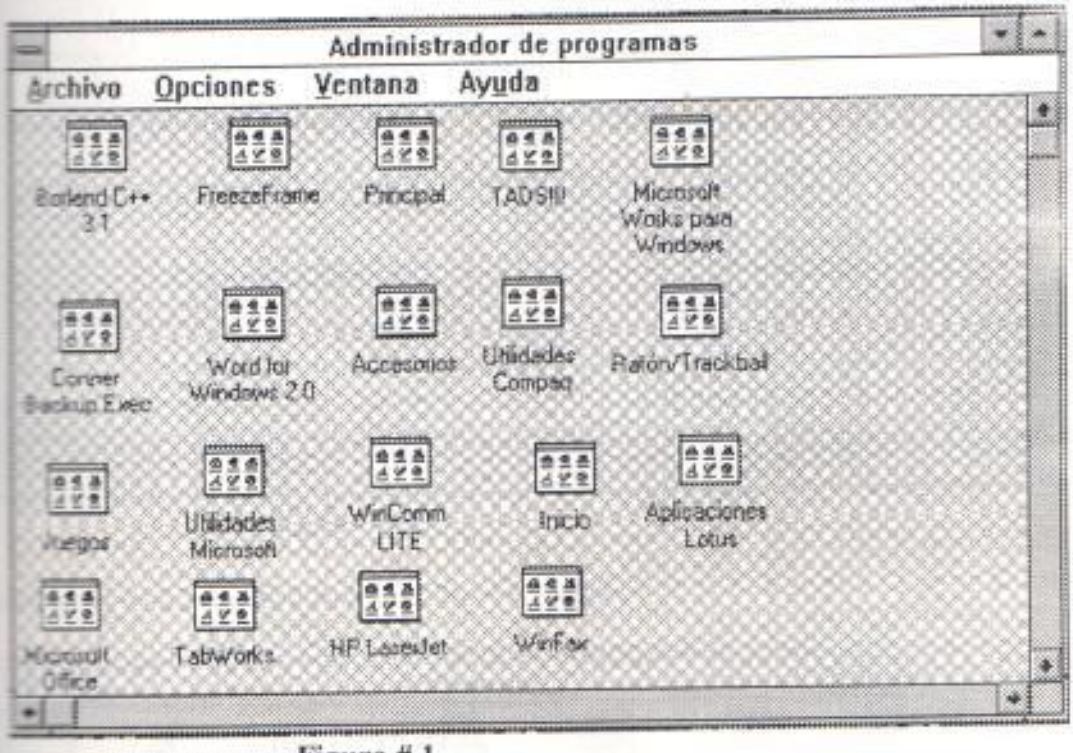


Figura # 1

La ventana de grupo de Accesorios despliega iconos para aplicaciones adicionales que ud puede correr.

Cuando se hace un doble clik en el icono elegido, en este caso Bloc de Notas, la aplicación aparece en una ventana, lista para trabajar con ella (figura # 3).

El Bloc de Notas de Windows es ahora la ventana activa. La ventana activa es generalmente la ventana que se encuentra superior a las demás. Para tener otra ventana activa, haga un click en cualquier parte de la ventana.

Figura # 2

Acceso al icono de Bloc de notas de Accesorios.

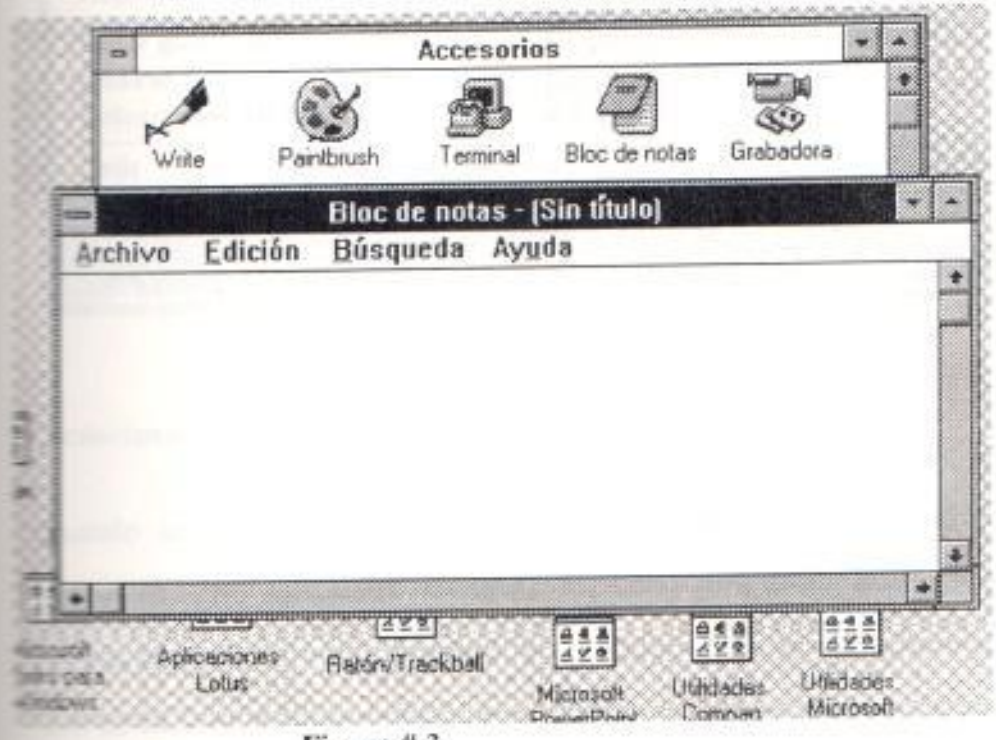


Figura # 3

Usando un Menú

En cada ventana de aplicación y también en otras ventanas, existe una lista de nombres de menús en la barra de títulos. Un menú contiene una lista de comandos, las cuales son acciones que se pueden llevar a cabo con Windows. Se elige un nombre de menú y un comando haciendo un click sobre cada uno de ellos (figura # 4).

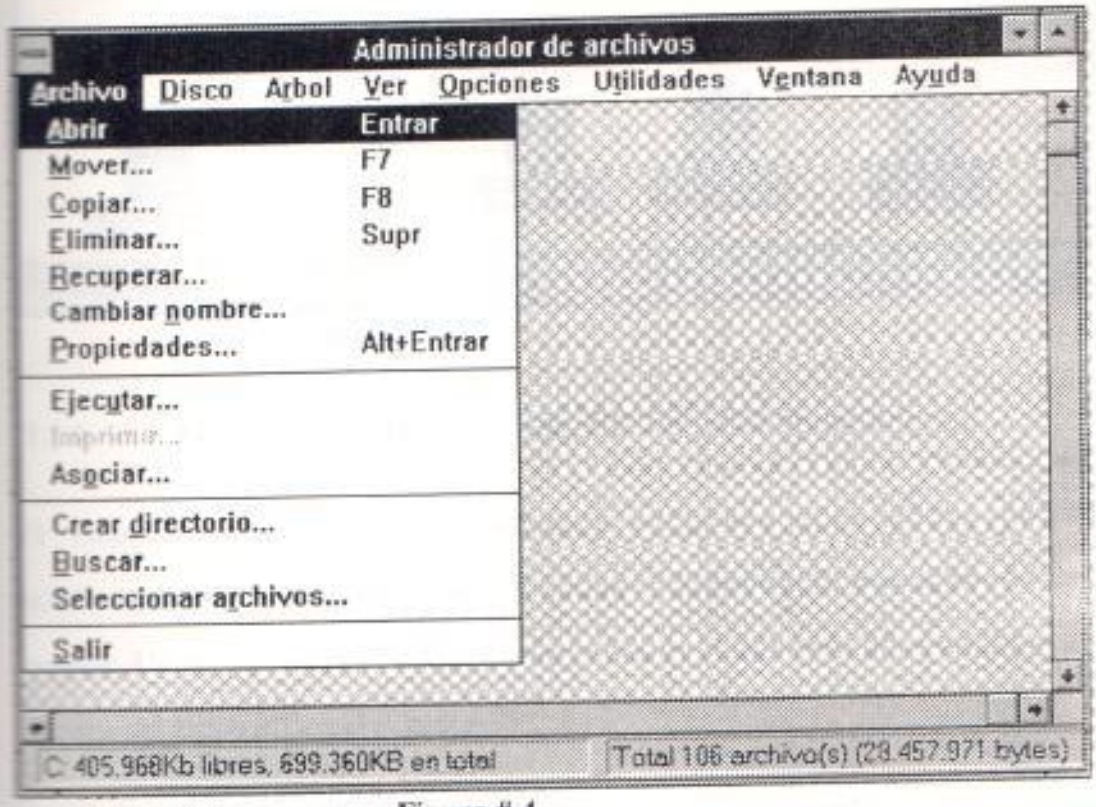


Figura # 4

Reduciendo una Ventana de Aplicación a un Icono

Cuando se trabaja con Windows, ud puede dejar a un lado temporalmente una aplicación, pero mantenerla fácilmente disponible para usarla luego. Se puede evitar sacarla de su pantalla por completo, si se reduce la ventana de aplicación a un icono con un click en el botón de Minimizar; el icono permanece en la ventana del escritorio

Figura # 5

1.2.2 EL ADMINISTRADOR DE ARCHIVOS (File Manager)

El Administrador de Archivos de Windows provee una representación gráfica de archivos y directorios que ayudarán a organizar y simplificar el mantenimiento de los mismos. Ud puede empezar el File Manager con un doble click sobre su icono en la ventana del grupo principal (figura # 2).

Puede realizar muchas tareas usandno el File Manager, por ejemplo:

- Ver el contenido de sus directorios.
- Mover, copiar y borrar archivos y directorios.
- Cambiar a otros disk drives, incluyendo drives de red.

Archivos y directorios

Un archivo es un documento electrónico o una aplicación, almacenandos en la computadora, a los cuales se ha asignado un nombre. Un archivo puede ser, por ejemplo, una carta o la propia aplicación de proceso de textos empleada para escribirla. Los archivos tienen nombres como CARTA.TXT o PROGRAMA.EXE y pueden almacenarse en la computadora dentro de directorios.

Un directorio es una colección de archivos y/o otros directorios (denominados subdirectorios) que están almacenados en el mismo lugar del disco. Dos directorios que seguramente encontrará en su sistema son WINDOWS y SYSTEM.

El directorio WINDOWS contiene el subdirectorio SYSTEM, así como la mayoría de los archivos del entorno Windows. El directorio SYSTEM contiene otros archivos adicionales (figura # 6).

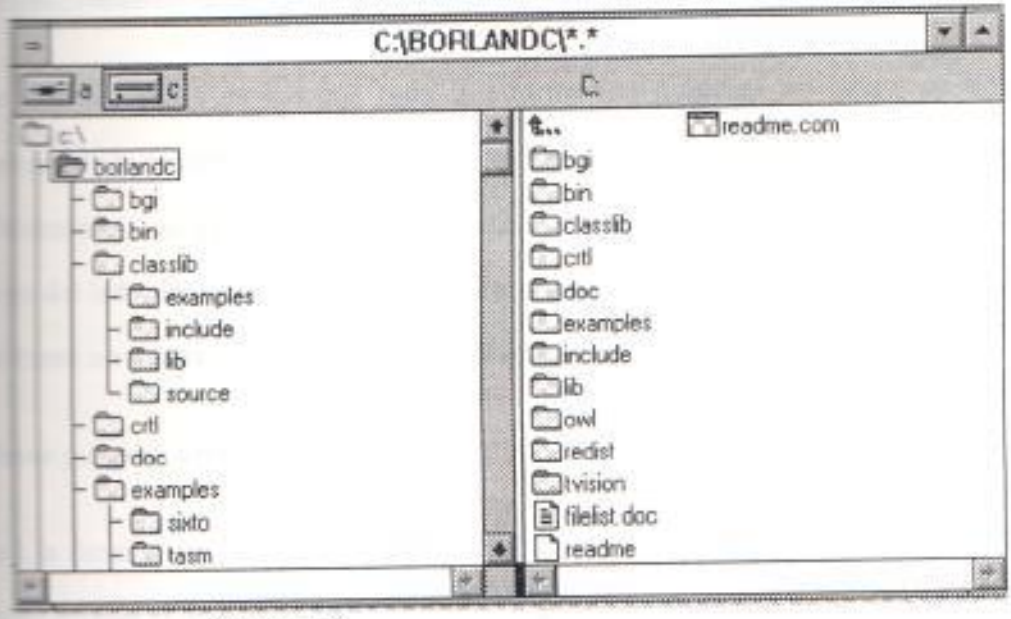


Figura # 6

Una ventana de directorio es una ventana secundaria que aparece en la ventana principal del Administrador de archivos. Las ventanas de directorio muestran gráficamente las relaciones existentes entre los directorios y archivos almacenados en una unidad de disco.

Ver el contenido de un directorio

El Administrador de Archivos permite examinar fácilmente el contenido de un directorio. Para ello basta con abrir el directorio deseado, haciendo click sobre su nombre o icono en la parte izquierda de la ventana.

Puede abrirse otro directorio, haciendo click sobre su icono o nombre. La información que aparece en la parte derecha de la ventana cambiará, para reflejar el contenido del directorio.

Desplazamiento en la pantalla para ver información

En aquellos casos en los que toda la información existente no quepa en una sola ventana de la pantalla, pueden utilizarse las barras de desplazamiento para mostrar más información. Las barras de desplazamiento aparecen automáticamente, dispuestas a lo largo del lado derecho y/o la parte inferior de una ventana, siempre que la información existente no pueda presentarse en una sola pantalla.

Mover y copiar archivos

Con la función de "arrastrar y dejar caer" que ofrece el Administrador de archivos, podrá utilizar el mouse para mover o copiar archivos rápidamente.

También es posible mover y copiar archivos de un directorio a otro, arrastrando los iconos de dichos archivos.

Para mover un archivo, arrastre el icono del mismo hasta el icono o ventana del directorio de destino.

Puede copiar un archivo manteniendo presionada la tecla CTRL mientras arrastra su icono hasta la ventana o icono del directorio de destino.

Otras formas de utilizar el Administrador de Archivos

Con el Administrador de Archivos también podrá hacer lo siguiente:

- Conectarse a unidades de red
- Dar formato a un disco o realizar tareas de mantenimiento sobre el mismo
- Imprimir documento
- Iniciar una aplicación
- Cambiar el tipo y la cantidad de información que se mostrará acerca de cada archivo

1.2.3 El Panel de Control

El Panel de Control de Windows es una herramienta que permite ajustar numerosas opciones del sistema como el aspecto del escritorio de Windows, o la configuración de determinados componentes hardware del sistema. Para iniciar el Panel de Control, haga doble click sobre su icono en la ventana del grupo principal (figura # 7)

Como posibilidades destacables son las siguientes:

- Agregar y eliminar fuentes.

- Cambiar los colores de pantalla y las opciones del escritorio

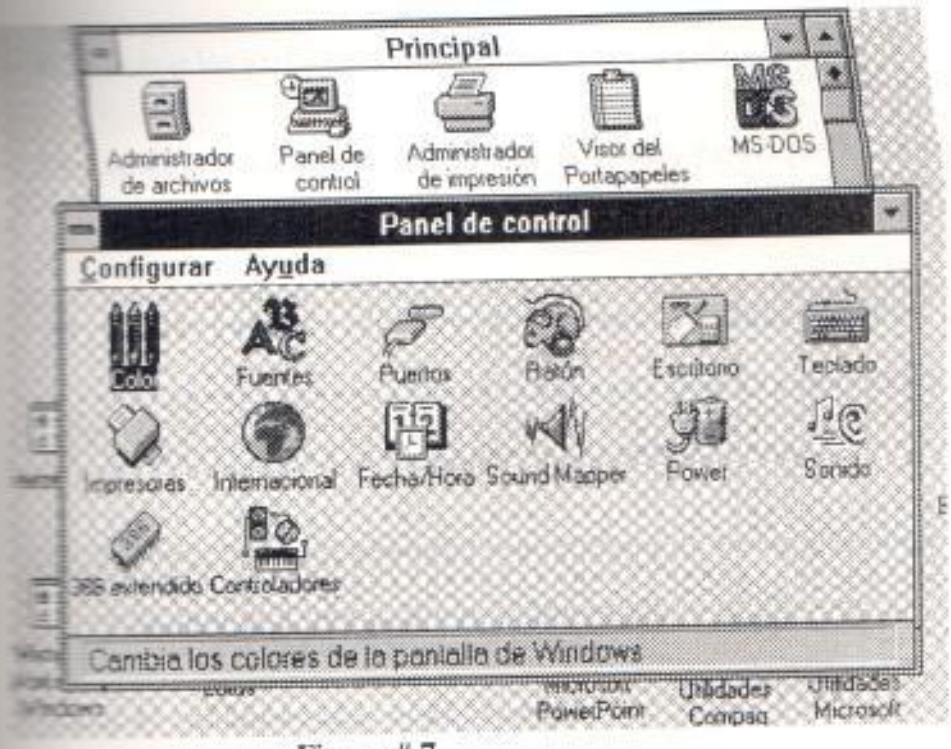


Figura # 7

Utilización de los cuadros de diálogo

Las ventanas que se utilizan para introducir modificaciones en las características del sistema se denominan cuadros de diálogo. En Windows, un cuadro de diálogo solicita información al usuario. Por ejemplo, en ocasiones es necesario seleccionar ciertas opciones, escribir texto o especificar determinados valores. Los cuadros de diálogo sirven para proporcionar al sistema la información concreta que necesitará para realizar las operaciones especificdas por el usuario al trabajar con Windows.

Información adicional sobre el Panel de Control

A continuación se indican otros tipos de características que pueden modificarse utilizando el Panel de Control.

- Cambiar las opciones relativas al mouse u otro dispositivo apuntador.
- Cambiar los parámetros de las conexiones a la red.
- Especificar las características internacionales, para aquellas aplicaciones que las utilizan.
- Asignar sonidos concretos a "sucesos del sistema". Por ejemplo, puede especificarse que Windows reproduzca una serie de notas musicales cada vez que se inicie el sistema, si se dispone de una tarjeta de sonido.
- Agregar,configurar o eliminar controladores de sonido y de dispositivos **MIDI**, si se dispone de una tarjeta de sonido MIDI.

1.2.4 El Administrador de Impresión

El Administrador de impresión controla las funciones de impresión de las aplicaciones de Windows y permite estar informado sobre el avance de los trabajos de impresión. La ventana del Administrador de Impresión muestra el estado de las impresoras que se están utilizando, como también los trabajos de impresión en curso o en espera de imprimirse (figura # 8) .

Para acceder a la ventana del Administrador de Impresiión, haga un doble_clic sobre el icono del Administrador de Impresión que aparecerá en la ventana del grupo principal.

Entre las tareas que pueden realizarse con el Administrador de Impresión se encuentran las siguientes:

- ✓ Comprobar el estado de un trabajo de impresión.
- Detener o reanudar un trabajo de impresión

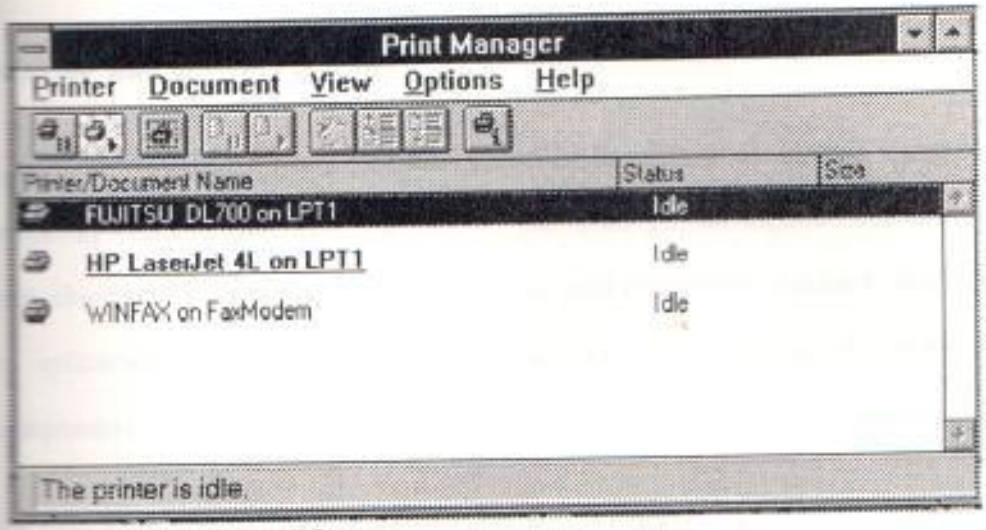


Figura # 8

- Cancelar un trabajo de Impresión

Comprobar el estado de un trabajo de impresión

Una aplicación habitual del Administrador de archivos consiste en mostrar el estado de las impresoras y de los trabajos de impresión del sistema. Esta posibilidad resulta especialmente útil cuando se necesita imprimir varios trabajos, o cuando se está utilizando una impresora de la red en un lugar no visible o al cual no se tenga acceso fácilmente.

Otras formas de utilizar el Administrador de Impresión

Mediante el Administrador de Impresión, también es posible:

- Configurar una impresora
- Cambiar las prioridades de los trabajos de impresión
- Mostrar los mensajes de Impresión
- Conectarse y desconectarse de Impresoras de la red

Pasar de una aplicación a otra

El uso de Windows para ejecutar varias aplicaciones al mismo tiempo, puede simplificar considerablemente su trabajo. Sin embargo, en algunas ocasiones en que estén abiertas varias ventanas de aplicación, probablemente resultará difícil localizar una aplicación específica, especialmente si está situada debajo de otras ventanas superpuestas.

Una forma muy sencilla de controlar las aplicaciones que se están utilizando es la función de conmutación rápida de Windows. Windows permite pasar rápidamente de una aplicación a otra con sólo mantener presionada la tecla **ALT** y presionar a continuación la tecla **TAB** repetidas veces. Cada vez que se presione la tecla **TAB**, irá apareciendo el título de cada una de las aplicaciones que estén ejecutando en ese momento en Windows, en un pequeño cuadro en el centro de la pantalla.

Lista de Tareas

Otro método para ejecutar aplicaciones, es la lista de tareas. Mediante la Lista de Tareas se podrá realizar lo siguiente:

- Mostrar una lista de las aplicaciones que se estén ejecutando
- Conmutar rápidamente a cualquier aplicación que se esté ejecutando
- Reorganizar las ventanas y los iconos del escritorio de Windows

Para acceder a la Lista de Tareas, puede hacer **doble_clic** en cualquier punto del escritorio de Windows. Otra forma de conseguirlo es presionar la combinación de teclas **CTRL+ESC**. O bien se conseguirá el mismo efecto escogiendo el comando Pasar a , en el menú Control de cualquier aplicación.

Los botones de la Lista de Tareas permiten realizar varias funciones:

" **Cambiar a** " conmuta a la aplicación seleccionada.

" **Finalizar tarea** " abandona la aplicación seleccionada.

" **Cascada** " muestra en pantalla las ventanas abiertas, una encima de otra, de tal modo que queda visible en pantalla, una junto a otra.

" **Cancelar** " cierra la Lista de tareas. También puede sonseguirse el mismo efecto presionando la tecla ESC.

" **Organizar icono**" reorganiza los iconos de aplicación, dejándolos en la parte inferior del escritorio.

1.2.5 Ayuda

Windows dispone de un sistema de Ayuda en línea, que le orientará sobre las distintas tareas que pueden realizarse con este programa. Existen varias formas de acceder a la Ayuda:

- Presionar la tecla **F1** para acceder al Indice de ayuda correspondiente a la aplicación con la cual se esté trabajando.

- En un cuadro de diálogo que incluya un botón **"Ayuda"**, hacer clic sobre el mismo, para acceder a la información en ese cuadro de diálogo.

- Abrir el **menú Ayuda** en cualquier aplicación, para acceder a una lista de los comandos Ayuda que pueden seleccionarse:

"Contenido": muestra una lista de los temas de Ayuda correspondientes a una aplicación.

"Buscar": muestra un cuadro de diálogo en el cual es posible especificar el tema que se desea localizar.

"Atrás": muestra el tema anterior.

"Historial": muestra una lista cronológica de todos los temas de Ayuda examinados durante la sesión actual de Windows.

"Glosario": muestra una lista de los términos de Windows, con sus correspondientes definiciones.

Salir de Windows

Para salir de Windows, se puede hacer doble_clic sobre el cuadro del menú Control del Administrador de Programas.

CAPITULO II

SISTEMAS TUTORIALES

2.1 DEFINICION DE SISTEMA TUTORIAL

Un Sistema Tutorial es un método de enseñanza que se basa en un material de tipo algoritmico en el que predomina el aprendizaje vía transmisión de conocimiento, desde quien sabe, hace quien lo desea aprender y donde el diseñador se encarga de encapsular secuencias bien diseñadas de actividades que conducen al aprendiz, desde donde está hasta donde desea llegar.

Los Sistemas Tutoriales guian al aprendiz a través de las distintas fases del aprendizaje, por medio del diálogo. Un Sistema Tutorial debe incluir cuatro fases para formar parte del proceso de enseñanza aprendizaje.

- La ***"Fase de Introducción"***, que se desea localizar, en la que se genera la motivación, se centra la atención y se favorece la percepción selectiva de lo que se desea que el alumno aprenda.
- La ***"Fase de Orientación Inicial"***, en la que se da la codificación, almacenaje y retención de lo aprendido.
- La ***"Fase de Aplicación"***, en la que hay evocación y transferencia de lo aprendido.
- La ***"Fase de retroalimentación"***, en la que se demuestra lo aprendido y se ofrece retroinformación y refuerzo.

El Sistema de motivación y de refuerzo que se emplee, depende en gran medida de la audiencia a la que se dirige el material y de lo que se desee enseñar, la secuencia que se sigue, depende en buena medida de la estructura de los aprendizajes que subyacen al objetivo terminal y del mayor o menor control que desee dar el diseñador a los usuarios.

Por ejemplo: En un tutorial con menú el aprendiz, puede decidir qué secuencia de instrucción sigue, mientras que cuando se lleva historia del desempeño del aprendiz, el diseñador puede conducir al usuario por rutas que ha prefijado en función del estado de la historia.

Las actividades y el entorno del aprendizaje también dependen de lo que se esté enseñando y de su nivel, así como de las personas a las que se dirige. Las oportunidades de práctica y la retroinformación asociada están directamente ligadas con lo que se esté enseñando y son parte muy importante del sistema tutorial.

Dependiendo de lo que el alumno demuestre que ha aprendido al resolver las situaciones que se les presenten, el sistema, deberá valorar lo hecho y tomar acciones que atiendan las deficiencias o logros obtenidos. Por Ejemplo; un grupo de aciertos puede hacer que el alumno pase a la siguiente unidad de instrucción remedial y complementaria con la ya obtenida.

La utilidad de los Sistemas Tutoriales radica en que el computador se vuelve particularmente útil cuando requiere alta motivación información de retorno diferencial e inmediata ritmo propio, secuencia controlable por el usuario parcial o totalmente. El computador está llamado a ofrecer un ambiente entretenido amigable y excitante que permite a los alumnos superar el desgano que la temática les genera y embarcarse en

una experiencia que les ayude a superar las limitaciones que tengan en el uso de sus destrezas.

2.2 SISTEMAS DE EJERCICIOS Y PRACTICAS

Los Sistemas de Ejercicios y Prácticas son aquellos que tratan de reforzar las fases finales del proceso de instrucción: aplicación y retroinformación. Se fundamentan en el uso de otro medio de enseñanza, el aprendiz ya adquirió los conceptos y destrezas que va a practicar. Por Ejemplo:

El profesor de Matemáticas explica las reglas básicas para efectuar operaciones con números decimales, dá algunos ejemplos y asigna ejercicios del texto de trabajo. Dependiendo de la cantidad de ejercicios del texto y del mayor o menor detalle que posea la retrorientación, el alumno podrá llevar a cabo suficiente aplicación de lo aprendido y obtebr información de retorno.

Sin embargo la retro-información estática que brinda un texto dificilmente puede ayudar al usuario a determinar en que parte del proceso cometió el error que le impidió obtener el resultado correcto. Por esto es conveniente complementar el trabajo del alumno usando un buen programa de ejercitación y práctica en el que pueda resolver variedad y cantidad de ejercicios, y según el proceso que siguió en su solución, obtener información de retorno diferencial.

En un Sistema de Ejercicios y Prácticas deben conjugarse tres condiciones: cantidad de ejercicios, variedad en los formatos con que se presentan y retro-información que oriente con luz indirecta la acción del aprendiz. No hay discusión de que la transferencia y la generalización de la destreza depende en buena medida de las dos primeras condiciones.

Respecto a la orientación, no tiene sentido dejar al estudiante sin ayuda o simplemente darle la respuesta al segundo o tercer intento, esto ocasiona pereza mental.

Debe imponerse la oportunidad de reprocesar la respuesta, dando pistas o criterios aplicables a la misma, cuando esto ya no es posible, cabe una solución guiada, pero no una respuesta directa.

Otros factores importantes en los Sistemas de Ejercicios y Prácticas son los sistemas de motivación y de refuerzo . Como de lo que se trata es de que el aprendiz logre destreza en lo que está practicando, y esto no se logra sino con amplia y variada ejercitación, es importante crear un gancho dentro del programa que promueva que el usuario haga una cantidad significativa de ejercicios que estén bien resueltos y sin ayuda.

La competencia puede ser un motivador efectivo (competencia contra otros estudiantes, contra el computador, contra uno mismo, o contra el tiempo). La variedad de despliegues de pantalla también es motivante, así como la fijación de metas y el suministro de recompensas relacionados (por ejemplo baila un muñeco, si logra tantos puntos, etc). También cabe administrar castigos (poe ejemplo pierde puntaje, etc.)

Los Sistemas de Ejercicios y Práctica, juegan un papel muy importante en el logro de habilidades y destrezas, sean éstas intelectuales o motoras en las que la ejercitación y reorientación son fundamentales.

ANALISIS DEL SISTEMA TUTORIAL

3.1 BREVE EXPLICACION DEL SISTEMA

Como principal objetivo tenemos el desarrollo de un TUTOR que proporcione al usuario los conocimientos fundamentales del Análisis y Diseño de Sistemas, de tal forma que constituya en una herramienta útil tanto para el profesor encargado de dictar la materia como para los estudiantes de la misma, ya que lo explicado en clase podría complementarse con lo enfocado en el tutor.

Además TUTOR debe tener la suficiente motivación de tal forma que el estudiante considere relevante lo que está aprendiendo, presentarle una explicación general del uso y contenido de cada tópico de estudio, de una forma fácil de moverse a través de menúes, para proveerle una buena interacción con el tutor, así como también evaluar el aprendizaje del estudiante en cada uno de estos tópicos y de animarlo a que refuerze en las áreas donde se le detectó errores.

El contenido de este Sistema Tutorial está de acuerdo con el plan académico de la materia de Análisis y Diseño de Sistemas y consistirá de varios tópicos a elección del usuario, una vez que el usuario escoga el tema deseado tutor presentará las siguientes opciones relacionadas con el tema como:

* ***Concepto Generales*:** Ofrecerá al usuario definiciones y conceptos sobre ciertos items relacionados con el tópico seleccionado.

* ***Tomar una Lección:*** Ofrecerá al usuario una breve explicación del tema relacionado con el tópico seleccionado mediante la presentación de pantallas atractivas con gráficos.

* ***Realizar un Test:*** Presentará al usuario la posibilidad de evaluar el conocimiento adquirido a través de preguntas lógicas V o F, mostrando reportes con calificaciones adquiridas en el tema seleccionado.

3.1.1 Temas del Programa de Análisis y Diseño de Sistemas

Los temas a elección del usuario que presentará el Tutor TADS está de acuerdo con el ***Plan Académico*** de la materia de ***Análisis y Diseño de Sistemas*** y consistirá de los siguientes capítulos:

1. Sistemas de Información.
2. Manejo del Desarrollo de Procesos.
3. Planificación y Técnicas.
4. Técnicas de Estimación.
5. Determinación de Requerimientos.
6. Contexto Organizacional.
7. Diagramas de Flujo de Datos y Modelo Conceptual de Datos.
8. Evaluación y Técnicas.

3.1.2 Características e Inicio de Lecciones

El tutor presenta en el menú principal 8 temas (capítulos) que pueden ser seleccionados por el estudiante. Una vez que se realiza la selección el tutor presentará un nuevo menú en donde aparecerá como una de las opciones **Lección** que consiste en una breve explicación del tema seleccionado.

El botón ejecuta una pantalla con la primera página de la lección, cada lección contiene un total de cinco o más páginas, con el resúmen del capítulo seleccionado y con pantallas atractivas e ilustradas para cautivar la atención del estudiante. En cada una de las páginas de las lecciones existe un botón titulado ***siguiente página*** que permite avanzar a la siguiente página de la lección, una vez que se ha recorrido todas las páginas encontrará un botón titulado ***retroceder*** que permitirá regresar a la lección anterior.

3.1.3 Características e Inicio de Test

El tutor presenta en el menú principal 8 temas (capítulos) que pueden ser seleccionados por el estudiante. Una vez que se realiza la selección el tutor presentara un nuevo menú en donde aparecerá como una de las opciones **Realizar un** Test que consiste en una prueba sobre el tema seleccionado.

Al seleccionar el botón (oscuro) se ejecuta la pantalla de prueba, que contiene preguntas relacionadas con la lección que se está revisando, usted debe contestar verdadero o falso en cada una de ellas, es posible seleccionar la respuesta con un click del ratón, o con el tabulador posicionado sobre la pregunta, avanzar con la flecha direcional y elegir ya sea el verdadero o falso. Una vez que se han contestado todas las preguntas, elija el botón que se encuentra en la parte inferior, el cual ejecuta una pequeña caja con la contestación correcta a las preguntas, al dar un enter o elegir el botón de la caja se ejecuta otra con el nombre del usuario si éste fue ingresado previamente en el menú principal del programa en el botón de usuario y su puntaje.

3.2 DETERMINACION DE REQUERIMIENTOS

El Sistema Tutorial es uno de los materiales educativos computacionales en el que el usuario, en este caso los estudiantes de la materia de Análisis y Diseño de Sistemas puedan tener una visión global de lo que comprende y al mismo tiempo asimilen en forma clara y sencilla los conceptos a tratar.

El Sistema Tutorial para que cumpla con los objetivos propuestos anteriormente fue desarrollado con una interfaz amigable al usuario como lo es Windows 3.1, para lo cual usamos una herramienta como el Lenguaje C++ de Borland, por su flexibilidad. Las especificaciones de Hardware y Software que se consideraron como requerimientos para el desarrollo de Tutor fueron las siguientes:

- **Personas:**

  Las personas a las cuales va dirigido el Tutorial de Informática es basicamente alumnos de la Facultad de Computación, ya que cubre un panorama muy general sobre conceptos informáticos, asi como a personas que deseen tener conocimientos del Análisis y Diseño de Sistemas.

- **Procedimientos:**

  El procedimiento seguido para el desarrollo del Tutorial, está basado en el desarrollo de lecciones que se encuentran dentro de un capítulo, en el cual el alumno va avanzando, encontrando en cada capitulo, un conjunto de preguntas que le ayudarán a repasar lo estudiado e irá reforzando sus conocimientos.

- **Datos:**

  - Archivos de Lectura con el contenido de cada lección e introducción.

- Archivo de escritura donde reside la información del usuario.

Componentes de Software:

- Sistema Operativo DOS versión 5.0
- Windows 3.1
- Borland C++ 3.1
- Power Point
- Microsoft Word

Componentes de Hardware:

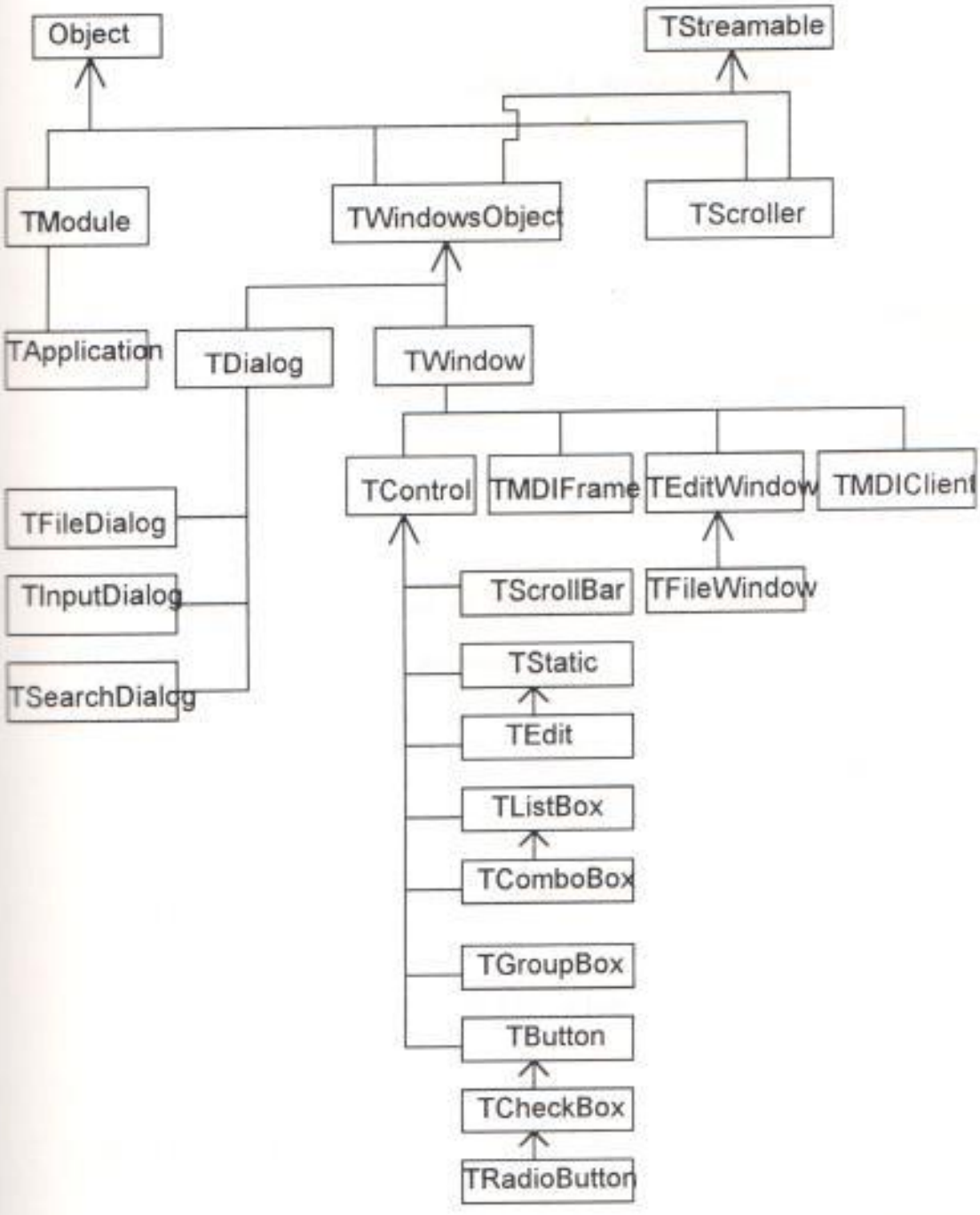
- 4 Mbytes de Memoria RAM
- Monitor VGA (color)
- Disco Duro de 120 Mbytes.
- Procesador 386 o superior.
- Disketera de 3 1/2.
- Mouse

3.3 JERARQUIA DE CLASES DE OBJECT WINDOWS

Object Windows es una librería que consiste en una jerarquía de clases que se pueden usar, modificar o añadir utilizando el concepto de herencia(diagrama # 1).

A continuación las principales:

DIAGRAMA # 1


Object
TStreamable
TModule
TWindowsObject
TScroller
TApplication
TDialog
TWindow
TControl
TMDIFrame
TEditWindow
TMDIClient
TFileDialog
TInputDialog
TSearchDialog
TScrollBar
TFileWindow
TStatic
TEdit
TListBox
TComboBox
TGroupBox
TButton
TCheckBox
TRadioButton

✓ **TWindows Object**

Es una clase base que unifica los tres tipos principales de objetos de interfaz de Windows: ventanas, cajs de diálogos y controles. Provee funciones miembros para manejar la creación, procesamiento de mensajes y destrucción de objetos tipo ventana.

✓ **Objetos ventana**

Los Objetos ventana representan no solamente las ventanas familiares del ambiente Windows, sino también la mayoría de los elementos visuales dentro de ese ambiente, tal como controles.

⇨ **TWindow**

Es una clase de ventana de propósito general cuayas instancias pueden representar ventanas principales, pop- up, o ventanas hijas (child windows). Las instancias de **TWindow** pueden desplegar gráficos, pero mayormente se deberá especializar el comportamiento de **TWindow** en clases que se deriven de ella.

⇨ **TEditWindow**

Derivada de **TWindow, TEditWindow** define una clase que permite editar texto en una ventana.

⇨ **TFileWindow**

Derivada de TWindow, TFileWindow define una clase que permite editar texto en una ventana, pero puede también cargar y grabar archivos.

✓ **Objetos Diálogo**

Los objetos diálogo sirven para facilitar grupos interactivos, particularmente grupos de controles tales como botones, listas y scrollbars.

⇨ **TDialog**

Esta clase pueden representar ventanas sirve como una clase base para clases derivadas que manejen cajas de diálogo de Windows. Las cajas de diálogos están asociadas con recursos y pueden ser ejecutadas como cajas de diálogo tipo modal o modeless. Se proveen funciones miembros para manejar comunicación entre un diálogo y sus controles.

⇨ **TFileDialog**

Es una clase diálogo que es útil inmediatamente en muchas aplicaciones. Define un diálogo que permite al usuario escoger un archivo para cualquier propósito, tal como abrir, editar o grabar.

⇨ **TInputDialog**

Esta clase define una caja de diálogo para que el usuario ingrese texto simple.

✓ **Objetos Control**

Dentro de los diálogos y algunas ventanas, los controles permiten a los usuarios ingresar datos y seleccionar opciones.Los objetos controles proveen un medio simple y consistente para manejar todos los tipos diferentes de controles definidos por Windows.

⇨ **TControl**

Es una clase abstracta que sirve como una base común para todos los objetos controles, incluyendo listas y botones. Define funciones miembros que crean controles y procesan mensajes para sus clases derivadas.

⇨ **TButton**

Las instancias de TButton representan botones tipo *push button.*

⇨ **TCheckBox**

Derivada de **TButton**, las instancias de **TCheckbox** representan los check boxes de Windows y proveen funciones para manejar su estado.

⇨ **TRadioButton**

Derivada de **TCheckBox,** las **instancias de TRadioButton** manejan la creación y estado para los radios buttons de Windows.

⇨ **TListBox**

Las instancias de **TListBox** representan un list box de Windows (Listas). Esta clase controla la creación y selección de un list box, y define funciones miembros para manipular los ítems de una lista.

⇨ **TComboBox**

Derivada de **TListBox**, TComboBox define el comportamiento de los Comboxes de Windows. Un combo box es una interdependente list box y un control tipo Edit.

⇨ **TGroupBox**

Las instancias de TGroupBox representan a los group boxes de Windows.

⇨ **TStatic**

Provee funciones miembros que setean, preguntas y limpian el texto de un control estático (sólo salida).

⇨ **TEdit**

Derivada de **TStatic, TEdit** provee las capacidades de procesamiento extensivo para un control edit de Windows.

⇨ **TScrollBars**

Define funciones miembros que manejan el rango y la posición de un control *scroll bar*.

⇨ **TInputDialog**

Esta clase define una caja de diálogo para que el usuario ingrese un texto simple.

✓ **Objetos MDI**

Windows implementa un stándar para manejar múltiples ventanas dentro de la plataforma o área de trabajo de una simple ventana. Este stándar se denomina Interfaz de Documento Múltiple (MDI). Object Windows provee un medio para inicializar y manipular ventanas MDI.

3.4 IDENTIFICACION DE CLASES Y OBJETOS DEFINIDOS

La Aplicación se basa en clases derivadas de OWL conocidas como: "Object Window Library", las clases que se identificaron son las siguientes:

Clase: **TMyApp**

Clase base: TApplication

Descripción:

Esta clase define el comportamento requerido por todas las aplicaciones de ObjectWindows. La clase Aplication que ud define en su programa, podria derivarse de TApplication. Esta es responsable entre, otras cosas de inicializar la ventana principal de la aplicación.

Esta clase contiene sus función constructor, con los parámetros que se indicaron anteriormente, estos parámetros son comunes para toda la aplicación realizada en Borland C++.

```
//************************************************************************
//                DEFINICION  DE  LA  CLASE  TMyApp
//************************************************************************

class TMyApp : public TApplication
{
 public:

 TMyApp(LPSTR AName, HINSTANCE hInstance, HINSTANCE hPrevInstance,
                LPSTR lpCmdLine, int nCmdShow)
      :TApplication(AName, hInstance, hPrevInstance, lpCmdLine, nCmdShow) {};
    virtual void InitMainWindow();
};
```

Clase: **TMyWindow**

Clase base: TWindow

Descripción:

Es una clase de ventana de propósito general, que puede representar ventanas principales, pop-up, ventanas hijas de una aplicación. Tipicamente, un programa crea un objeto TWindow en la función InitMainWindow, el constructor para esta clase fija las variables del objeto, como un Atrr, que contiene la posición y tamaño de la ventana, el estilo de la ventana, el estilo de la ventana, el control ID para el control de la ventana, y un puntero tipo far que usted lo puede usar cuando lo desee.

```
//************************************************************************
//                DEFINICION  DE LA CLASE TMyWindow
//************************************************************************
_CLASSDEF(TMyWindow)
class TMyWindow : public TWindow
{ private:
  HBITMAP GameOver;
  public:
  TTransferStruct  TransferStruct;
  HDC DragDC;
  BOOL ButtonDown;
  HINSTANCE BWCCMod;//HANDLE

 TMyWindow(PTWindowsObject AParent, LPSTR ATitle);
 virtual void WMPaint (RTMessage) = [WM_FIRST + WM_PAINT];
 virtual  void  Handle11ButtonMsg(RTMessage  Msg)  =  [ID_FIRST  +
ID_BUTTON11];
 virtual  void  Handle12ButtonMsg(RTMessage  Msg)  =  [ID_FIRST  +
ID_BUTTON12];
 virtual  void  Handle13ButtonMsg(RTMessage  Msg)  =  [ID_FIRST  +
ID_BUTTON13];
```

```
virtual void Handle15ButtonMsg(RTMessage Msg) =[ID_FIRST +ID_BUTTON15];
virtual  void Handle16ButtonMsg(RTMessageMsg) =[ID_FIRST + ID_BUTTON16];
virtual void Handle17ButtonMsg(RTMessage Msg) =[ID_FIRST + ID_BUTTON17];
virtual void Handle18ButtonMsg(RTMessage Msg) =[ID_FIRST + ID_BUTTON18];
virtual void Handle19ButtonMsg(RTMessageMsg) =[ID_FIRST + ID_BUTTON19];
virtual void Handle20ButtonMsg(RTMessageMsg) =[ID_FIRST + ID_BUTTON20];
virtual void Handle21ButtonMsg(RTMessageMsg) =[ID_FIRST + ID_BUTTON21];
virtual void CMSalir(RTMessage Msg)  =[ID_FIRST + ID_BUTTON22];
virtual void EmpInput(RTMessage Msg) = [CM_FIRST + CM_EMPINPUT];
~TMyWindow();
*****************************************************************
*****************************************************************
```

Clase: **TDialog_n**

Clase base: TDialog

Descripción:

Esta clase sirve como una base para las clases derivadas que manejan cajas de diálogo. Un objeto Dialog, está asociado con recursos Dialog, y puede correr tanto en forma modal o modeless. Funciones miembros son provistas para manejar la comunicación entre la caja de diálogo y sus controles.

Derivada de TDialog, esta clase se identificó para mostrar la información de las lecciones del tutorial.

Como atributos podemos mencionar lo siguiente:

Identificador de la caja de diálogo.

Identificador del recurso bitmap.

Identificador de aceptación.(avanzar).

Identificador de cancelación (retroceder).

Como servicios podemos mencionar lo siguiente:

Mostrar la información de la lección.

Controlar la secuencia de las lecciones.

```
*****************************************************************
```

DEFINICION DE LA CLASE TPRESENTACION

```
*****************************************************************
```

```
class TPresentacion : public TDialog
{

public:
TPresentacion(PTWindowsObject AParent, LPSTR AName);
virtual BOOL CanClose(void);
virtual void HandleButtonMsg(void)= [ID_FIRST + ID_BUTTON];
virtual void HandlePre1(void) = [ID_FIRST + ID_PRE1];
virtual void HandlePre2(void) = [ID_FIRST + ID_PRE2];
virtual void HandlePre3(void) = [ID_FIRST + ID_PRE3];
};
```

```
*****************************************************************
```

```
//*****************************************************************
//        DEFINICION  DE  LA  CLASE  Tint (INTRODUCCION)
//*****************************************************************

class TInt : public TDialog

{
public:
TInt(PTWindowsObject AParent, LPSTR AName);
virtual BOOL CanClose(void);
virtual void HandleButtonMsg(void)= [ID_FIRST + ID_BUTTON];
};


//*****************************************************************
//      DEFINICION DE LA CLASE VentUno_Dialogo
//*****************************************************************

_CLASSDEF(VentUno_Dialogo)
class VentUno_Dialogo : public TDialog

{
public:
int NumClicks;
HCURSOR CursorDown, CursorUp;
VentUno_Dialogo(PTWindowsObject AParent, LPSTR AName,int click);
virtual void SetupWindow(void);
virtual BOOL CanClose(void);
virtual void HandleButtonMsg(RTMessage Msg)   =[ID_FIRST+ID_BUTTON1];
virtual void HandleMenuMsg(RTMessage Msg)     //menu
=[ID_FIRST + ID_MENU1];
virtual LPSTR GetClassName(){return "MyClass";};
virtual void GetWindowClass(WNDCLASS&  AWndClass);
};


//*****************************************************************
//      DEFINICION DE LA CLASE VentDos_Dialogo
//*****************************************************************

_CLASSDEF(VentDos_Dialogo)
class VentDos_Dialogo : public TDialog

{
public:
int NumClicks;
```

```
VentDos_Dialogo(PTWindowsObject AParent, LPSTR AName, int click);
virtual void SetupWindow(void);
virtual BOOL CanClose(void);
virtual void HandleButtonMsg(RTMessage Msg) //avanzar
 = [ID_FIRST + ID_BUTTON2];
virtual void HandleRetroMsg(RTMessage Msg) //retroceder
 = [ID_FIRST + ID_RETRO2];
virtual void HandleMenuMsg(RTMessage Msg)   //menu
= [ID_FIRST + ID_MENU2];
virtual LPSTR GetClassName(){return "MyClass";};
virtual void GetWindowClass(WNDCLASS&  AWndClass);
};

//*****************************************************************
//      DEFINICION DE LA CLASE VentTres_Dialogo
//*****************************************************************

DEFINE_CLASSDEF(VentTres_Dialogo)

class VentTres_Dialogo : public TDialog
{
public:
int NumClicks;
VentTres_Dialogo(PTWindowsObject AParent, LPSTR AName, int click);
virtual void SetupWindow(void);
virtual BOOL CanClose(void);
virtual void HandleButtonMsg(RTMessage Msg)
  = [ID_FIRST + ID_BUTTON3];
virtual void HandleRetroMsg(RTMessage Msg)
  = [ID_FIRST + ID_RETRO3];
virtual void HandleMenuMsg(RTMessage Msg)
 = [ID_FIRST + ID_MENU3];
virtual LPSTR GetClassName(){return "MyClass";};
virtual void GetWindowClass(WNDCLASS&  AWndClass);
};

//*****************************************************************
//      DEFINICION DE LA CLASE VentCuatro_Dialogo
//*****************************************************************
```

```
ASSDEF(VentCuatro_Dialogo)
entCuatro_Dialogo : public TDialog

public:
 NumClicks;
entCuatro_Dialogo(PTWindowsObject AParent, LPSTR AName, int click);
virtual void SetupWindow(void);
virtual BOOL CanClose(void);
virtual void HandleRetroMsg(RTMessage Msg)
        = [ID_FIRST + ID_RETRO4];
virtual void HandleMenuMsg(RTMessage Msg)
        = [ID_FIRST + ID_MENU4];
virtual LPSTR GetClassName(){return "MyClass";};
virtual void GetWindowClass(WNDCLASS&  AWndClass);
};
```

DEFINICION DE LA CLASE VentInt_Dialogo

```
ASSDEF(VentInt_Dialogo)
entInt_Dialogo : public TDialog

public:
 NumClicks;
CURSOR CursorDown, CursorUp;
BeepButton *BeepButtonInt;
Beep1Button *Beep1ButtonInt;
entInt_Dialogo(PTWindowsObject AParent, LPSTR AName);
virtual void SetupWindow(void);
virtual BOOL CanClose(void);
virtual void HandleButtonInt(RTMessage Msg)
  = [ID_FIRST + ID_BUTTON1];
virtual void HandleRetroInt(RTMessage Msg)
  = [ID_FIRST + ID_RETRO1];
virtual void HandleMenuInt(RTMessage Msg)
  = [ID_FIRST + ID_MENU1];
};
```

DEFINICION DE LA CLASE TBitm

class TBitm : public TDialog

{

public:

TBitm(PTWindowsObject AParent, LPSTR AName);

virtual BOOL CanClose(void);

virtual void HandleButtonMsg(void)= [ID_FIRST + ID_BUTTON];

virtual void HandleBitm1(void)= [ID_FIRST + ID_AVANZ];

};

****

Clase **Registro:**

Derivada de TDialog, esta clase se identificó para mostrar la información registrada de los usuarios, con el fin de llevar un control de los alumnos que utilizan el tutorial.

Como atributos podemos mencionar lo siguiente:

Nombre del alumno.

Fecha en la que ingresa al tutor.

Como servicio podemos mencionar lo siguiente:

Mostrar el ingreso del usuario al tutor.

Controlar el ingreso del usuario.

Inicializar el archivo de usuarios.

****

DEFINICION DE LA CLASE TEmployeeDlg

****

class **TEmployeeDlg** : public TDialog

{

```
public:
  char EmpName[MAXNAMELEN] ;
  TEmployeeDlg(PTWindowsObject AParent, LPSTR name);
  virtual BOOL CanClose();
  virtual void HandleInfo(RTMessage Msg)= [ID_FIRST + IDINFO];
private:
  void FillBuffers();
  BOOL ValidName();
  BOOL ValidPassword();
};
```

```
*****************************************************************
      DEFINICION  DE LA CLASE TInfo
*****************************************************************

class TInfo : public TDialog
{
public:
  TInfo(PTWindowsObject AParent, LPSTR AName);
  TGroupBox *TestGroup;
  virtual void SetupWindow(void);
  virtual BOOL CanClose(void);
};
*****************************************************************
```

Clase Pregunta:

Derivada de TDialog, esta clase se identificó para poder realizr preguntas al alumno que está aprendiendo con el TADS, ya que las preguntas son parte importante dentro del Sistema Tutor.

Como atributos podemos mencionar lo siguiente:

Número de preguntas.

Texto de las preguntas.

Proposición verdadera o falsa.

Como servicio podemos mencionar lo siguiente:

Inicializar las preguntas.

Mostrar las peguntas.

Controlar la contestación del alumno a las preguntas.

```
*****************************************************************
      DEFINICION DE LA CLASE  TEST1 del Capítulo 1
*****************************************************************
class TTest1 : public TDialog
{
public:
TTest1(PTWindowsObject AParent, LPSTR Aname);
TBGroupBox,*TestGroup,*Test1Group,*Test2Group,*Test3Group,*Test4Group,
*Test5Group,*Test6Group,*Test7Group,*Test8Group,*Test9Group,
*Test10Group, *Test11Group,*Test12Group;
TBRadioButton
*P1,*X1,*X2,*P2,*X3,*X4,*P3,*X5,*X6,*OK,*P4,*X7,*X8,*P5,*X9,
*X10,*P6,*X11,*X12,*P7,*X13,*X14,*P8,*X15,*X16,*P9,*X17,*X18,*P10,*X19,
*X20,*P11,*X21,*X22;
virtual void SetupWindow(void);
virtual void HandleTestGroup(RTMessage Msg)  = [ID_FIRST + IDTESTGROUP];
virtual  void  Handle1TestGroup(RTMessage  Msg)  =  [ID_FIRST  +
ID1TESTGROUP];
virtual  void  Handle2TestGroup(RTMessage  Msg)  =  [ID_FIRST  +
ID2TESTGROUP];
virtual  void  Handle3TestGroup(RTMessage  Msg)  =  [ID_FIRST  +
ID3TESTGROUP];
virtual  void  Handle4TestGroup(RTMessage  Msg)  =  [ID_FIRST  +
ID4TESTGROUP];
virtual  void  Handle5TestGroup(RTMessage  Msg)  =  [ID_FIRST  +
ID5TESTGROUP];
virtual  void  Handle6TestGroup(RTMessage  Msg)  =  [ID_FIRST  +
ID6TESTGROUP];
virtual  void  Handle7TestGroup(RTMessage  Msg)  =  [ID_FIRST  +
ID7TESTGROUP];
```

```
virtual  void    Handle8TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual  void    Handle9TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual BOOL CanClose(void);
};

//---------------------------------------------------------------------------
//      DEFINICION DE LA CLASE TEST del Capítulo 2
//---------------------------------------------------------------------------

class TTest2 : public TDialog
{
public:
TTest2(PTWindowsObject AParent, LPSTR AName);
TGroupBox *TestGroup,*Test1Group,*Test2Group,*Test3Group,*Test4Group,
*Test5Group,*Test6Group,*Test7Group,*Test8Group,*Test9Group,
*Test10Group, *Test11Group,*Test12Group;
TRadioButton
*X1,*X2,*P2,*X3,*X4,*P3,*X5,*X6,*OK,*P4,*X7,*X8,*P5,*X9,
*P6,*X11,*X12,*P7,*X13,*X14,*P8,*X15,*X16,*P9,*X17,*X18,*P10,*X19,
*P11,*X21,*X22;
virtual void SetupWindow(void);
virtual void HandleTestGroup (RTMessage Msg) = [ID_FIRST + IDTESTGROUP];
virtual  void    Handle1TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual  void    Handle2TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual  void    Handle3TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual  void    Handle4TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual  void    Handle5TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual  void    Handle6TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual  void    Handle7TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
virtual  void    Handle8TestGroup(RTMessage    Msg)      =    [ID_FIRST   +
IDTESTGROUP];
```

```
virtual void Handle9TestGroup(RTMessage Msg) = [ID_FIRST +
ID9TESTGROUP];
virtual BOOL CanClose(void);
};

*****************************************************************
DEFINICION DE LA CLASE TEST DEL CAPÍTULO 3
*****************************************************************
class TTest3 : public TDialog
{
public:
TTest3(PTWindowsObject AParent, LPSTR AName);
TBGroupBox *TestGroup,*Test1Group,*Test2Group,*Test3Group,*Test4Group,
*Test5Group,*Test6Group,*Test7Group,*Test8Group,*Test9Group,
*Test10Group, *Test11Group,*Test12Group;
TBRadioButton
*P1,*X1,*X2,*P2,*X3,*X4,*P3,*X5,*X6,*OK,*P4,*X7,*X8,*P5,*X9,
*X10,*P6,*X11,*X12,*P7,*X13,*X14,*P8,*X15,*X16,*P9,*X17,*X18,*P10,*X19,
*X20,*P11,*X21,*X22;
virtual void SetupWindow(void);
virtual void HandleTestGroup (RTMessage Msg) = [ID_FIRST + IDTESTGROUP];
virtual void Handle1TestGroup(RTMessage Msg) = [ID_FIRST +
ID1TESTGROUP];
virtual void Handle2TestGroup(RTMessage Msg) = [ID_FIRST +
ID2TESTGROUP];
virtual void Handle3TestGroup(RTMessage Msg) = [ID_FIRST +
ID3TESTGROUP];
virtual void Handle4TestGroup(RTMessage Msg) = [ID_FIRST +
ID4TESTGROUP];
virtual void Handle5TestGroup(RTMessage Msg) = [ID_FIRST +
ID5TESTGROUP];
virtual void Handle6TestGroup(RTMessage Msg) = [ID_FIRST +
ID6TESTGROUP];
virtual void Handle7TestGroup(RTMessage Msg) = [ID_FIRST +
ID7TESTGROUP];
virtual void Handle8TestGroup(RTMessage Msg) = [ID_FIRST +
ID8TESTGROUP];
virtual void Handle9TestGroup(RTMessage Msg) = [ID_FIRST +
ID9TESTGROUP];
virtual BOOL CanClose(void);
};
```

```
*****************************************************************
DEFINICION DE LA CLASE TEST DEL CAPÍTULO 4
*****************************************************************
class TTest4 : public TDialog
{
public:
TTest4(PTWindowsObject AParent, LPSTR AName);
TBGroupBox *TestGroup,*Test1Group,*Test2Group,*Test3Group,*Test4Group,
*Test5Group,*Test6Group,*Test7Group,*Test8Group,*Test9Group,
*Test10Group, *Test11Group,*Test12Group;
TBRadioButton
*P1,*X1,*X2,*P2,*X3,*X4,*P3,*X5,*X6,*OK,*P4,*X7,*X8,*P5,*X9,
*X10,*P6,*X11,*X12,*P7,*X13,*X14,*P8,*X15,*X16,*P9,*X17,*X18,*P10,*X19,
*X20,*P11,*X21,*X22;
virtual void SetupWindow(void);
virtual void HandleTestGroup (RTMessage Msg) = [ID_FIRST + IDTESTGROUP];
virtual void Handle1TestGroup(RTMessage Msg) = [ID_FIRST +
ID1TESTGROUP];
virtual void Handle2TestGroup(RTMessage Msg) = [ID_FIRST +
ID2TESTGROUP];
virtual void Handle3TestGroup(RTMessage Msg) = [ID_FIRST +
ID3TESTGROUP];
virtual void Handle4TestGroup(RTMessage Msg) = [ID_FIRST +
ID4TESTGROUP];
virtual void Handle5TestGroup(RTMessage Msg) = [ID_FIRST +
ID5TESTGROUP];
virtual void Handle6TestGroup(RTMessage Msg) = [ID_FIRST +
ID6TESTGROUP];
virtual void Handle7TestGroup(RTMessage Msg) = [ID_FIRST +
ID7TESTGROUP];
virtual void Handle8TestGroup(RTMessage Msg) = [ID_FIRST +
ID8TESTGROUP];
virtual void Handle9TestGroup(RTMessage Msg) = [ID_FIRST +
ID9TESTGROUP];
virtual BOOL CanClose(void);
};

*****************************************************************
DEFINICION DE LA CLASE TEST DEL CAPÍTULO 5
*****************************************************************
class TTest5 : public TDialog
```

```
{
public:
 TTest5(PTWindowsObject AParent, LPSTR AName);
TBGroupBox *TestGroup,*Test1Group,*Test2Group,*Test3Group,*Test4Group,
*Test5Group,*Test6Group,*Test7Group,*Test8Group,*Test9Group,
*Test10Group, *Test11Group,*Test12Group;
TBRadioButton
*P1,*X1,*X2,*P2,*X3,*X4,*P3,*X5,*X6,*OK,*P4,*X7,*X8,*P5,*X9,
*X10,*P6,*X11,*X12,*P7,*X13,*X14,*P8,*X15,*X16,*P9,*X17,*X18,*P10,*X19,
*X20,*P11,*X21,*X22;
virtual void SetupWindow(void);
virtual void HandleTestGroup (RTMessage Msg) = [ID_FIRST + IDTESTGROUP];
virtual void Handle1TestGroup(RTMessage Msg) = [ID_FIRST +
ID1TESTGROUP];
virtual void Handle2TestGroup(RTMessage Msg) = [ID_FIRST +
ID2TESTGROUP];
virtual void Handle3TestGroup(RTMessage Msg) = [ID_FIRST +
ID3TESTGROUP];
virtual void Handle4TestGroup(RTMessage Msg) = [ID_FIRST +
ID4TESTGROUP];
virtual void Handle5TestGroup(RTMessage Msg) = [ID_FIRST +
ID5TESTGROUP];
virtual void Handle6TestGroup(RTMessage Msg) = [ID_FIRST +
ID6TESTGROUP];
virtual void Handle7TestGroup(RTMessage Msg) = [ID_FIRST +
ID7TESTGROUP];
virtual void Handle8TestGroup(RTMessage Msg) = [ID_FIRST +
ID8TESTGROUP];
virtual void Handle9TestGroup(RTMessage Msg) = [ID_FIRST +
ID9TESTGROUP];
virtual BOOL CanClose(void);
};

*****************************************************************
        DEFINICION DE LA CLASE TEST DEL CAPÍTULO 6
*****************************************************************

class TTest6 : public TDialog
{
public:
TTest6(PTWindowsObject AParent, LPSTR AName);
TBGroupBox *TestGroup,*Test1Group,*Test2Group,*Test3Group,*Test4Group,
```

```
*Test5Group,*Test6Group,*Test7Group,*Test8Group,*Test9Group,
*Test10Group,
 *Test11Group,*Test12Group;
TBRadioButton
*P1,*X1,*X2,*P2,*X3,*X4,*P3,*X5,*X6,*OK,*P4,*X7,*X8,*P5,*X9,
*X10,*P6,*X11,*X12,*P7,*X13,*X14,*P8,*X15,*X16,*P9,*X17,*X18,*P10,*X19,
*X20,*P11,*X21,*X22;
virtual void SetupWindow(void);
virtual void HandleTestGroup (RTMessage Msg)  = [ID_FIRST + IDTESTGROUP];
virtual   void   Handle1TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID1TESTGROUP];
virtual   void   Handle2TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID2TESTGROUP];
virtual   void   Handle3TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID3TESTGROUP];
virtual   void   Handle4TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID4TESTGROUP];
virtual   void   Handle5TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID5TESTGROUP];
virtual   void   Handle6TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID6TESTGROUP];
virtual   void   Handle7TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID7TESTGROUP];
virtual   void   Handle8TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID8TESTGROUP];
virtual   void   Handle9TestGroup(RTMessage   Msg)      =   [ID_FIRST   +
ID9TESTGROUP];
virtual BOOL CanClose(void);
};
```

```
*****************************************************************
```

DEFINICION DE LA CLASE TEST DEL CAPÍTULO 7

```
*****************************************************************

class TTest7 : public TDialog
{
public:
 TTest7(PTWindowsObject AParent, LPSTR AName);
TBGroupBox *TestGroup,*Test1Group,*Test2Group,*Test3Group,*Test4Group,
*Test5Group,*Test6Group,*Test7Group,*Test8Group,*Test9Group,
*Test10Group,
```

```
*Test11Group,*Test12Group;
TBRadioButton
*P1,*X1,*X2,*P2,*X3,*X4,*P3,*X5,*X6,*OK,*P4,*X7,*X8,*P5,*X9,
*X10,*P6,*X11,*X12,*P7,*X13,*X14,*P8,*X15,*X16,*P9,*X17,*X18,*P10,*X19,
*X20,*P11,*X21,*X22;
virtual void SetupWindow(void);
virtual void HandleTestGroup (RTMessage Msg) = [ID_FIRST + IDTESTGROUP];
virtual void Handle1TestGroup(RTMessage Msg) = [ID_FIRST +
ID1TESTGROUP];
virtual void Handle2TestGroup(RTMessage Msg) = [ID_FIRST +
ID2TESTGROUP];
virtual void Handle3TestGroup(RTMessage Msg) = [ID_FIRST +
ID3TESTGROUP];
virtual void Handle4TestGroup(RTMessage Msg) = [ID_FIRST +
ID4TESTGROUP];
virtual void Handle5TestGroup(RTMessage Msg) = [ID_FIRST +
ID5TESTGROUP];
virtual void Handle6TestGroup(RTMessage Msg) = [ID_FIRST +
ID6TESTGROUP];
virtual void Handle7TestGroup(RTMessage Msg) = [ID_FIRST +
ID7TESTGROUP];
virtual void Handle8TestGroup(RTMessage Msg) = [ID_FIRST +
ID8TESTGROUP];
virtual void Handle9TestGroup(RTMessage Msg) = [ID_FIRST +
ID9TESTGROUP];
virtual BOOL CanClose(void);
};
```

****

DEFINICION DE LA CLASE TEST DEL CAPÍTULO 8

****

```
class TTest8 : public TDialog
{
public:
TTest8(PTWindowsObject AParent, LPSTR AName);
TBGroupBox *TestGroup,*Test1Group,*Test2Group,*Test3Group,*Test4Group,
*Test5Group,*Test6Group,*Test7Group,*Test8Group,*Test9Group,
*Test10Group,*Test11Group,*Test12Group;
TBRadioButton
*P1,*X1,*X2,*P2,*X3,*X4,*P3,*X5,*X6,*OK,*P4,*X7,*X8,*P5,*X9,
```

```
*X10,*P6,*X11,*X12,*P7,*X13,*X14,*P8,*X15,*X16,*P9,*X17,*X18,*P10,*X19,
*X20,*P11,*X21,*X22;
virtual void SetupWindow(void);
virtual void HandleTestGroup (RTMessage Msg) = [ID_FIRST + IDTESTGROUP];
virtual void Handle1TestGroup(RTMessage Msg) = [ID_FIRST +
ID1TESTGROUP];
virtual void Handle2TestGroup(RTMessage Msg) = [ID_FIRST +
ID2TESTGROUP];
virtual void Handle3TestGroup(RTMessage Msg) = [ID_FIRST +
ID3TESTGROUP];
virtual void Handle4TestGroup(RTMessage Msg) = [ID_FIRST +
ID4TESTGROUP];
virtual void Handle5TestGroup(RTMessage Msg) = [ID_FIRST +
ID5TESTGROUP];
virtual void Handle6TestGroup(RTMessage Msg) = [ID_FIRST +
ID6TESTGROUP];
virtual void Handle7TestGroup(RTMessage Msg) = [ID_FIRST +
ID7TESTGROUP];
virtual void Handle8TestGroup(RTMessage Msg) = [ID_FIRST +
ID8TESTGROUP];
virtual void Handle9TestGroup(RTMessage Msg) = [ID_FIRST +
ID9TESTGROUP];
virtual BOOL CanClose(void); };
```

Clase Respuesta:

Derivada de TDialog, esta clase se identificó para poder mostrar las respuestas correctas a las preguntas realizadas, de manera que el alumno pueda ver en qué preguntas se equivocó y en cuales acertó.

Como atributo podemos mencionar lo siguiente:

Cantidad de las respuestas.

Texto de las respuestas.

Numeración de las respuestas.

Como servicios podemos mencionar lo siguiente:

Mostrar las respuestas.

Controlar la acción del usuario, es decir, que no pueda ver las respuestas hasta que haya contestado todas las preguntas.

Las clases definidas son las siguientes:

```
*****************************************************************
        DEFINICION DE LA CLASE TRESP DEL CAPÍTULO 1
*****************************************************************

class TResp1 : public TDialog
{
 public:
 TResp1(PTWindowsObject AParent, LPSTR AName);
 virtual void SetupWindow(void);
 virtual BOOL CanClose(void);
};

*****************************************************************
        DEFINICION DE LA CLASE TRESP DEL CAPÍTULO 2
*****************************************************************

class TResp2 : public TDialog
{
 public:
 TResp2(PTWindowsObject AParent, LPSTR AName);
 virtual void SetupWindow(void);
 virtual BOOL CanClose(void);
};
```

```
*****************************************************************
        DEFINICION DE LA CLASE TRESP DEL CAPÍTULO 3
*****************************************************************
class TResp3 : public TDialog
{
 public:
 TResp3(PTWindowsObject AParent, LPSTR AName);
 virtual void SetupWindow(void);
 virtual BOOL CanClose(void);
};

*****************************************************************
        DEFINICION DE LA CLASE TRESP DEL CAPÍTULO 4
*****************************************************************

class TResp4 : public TDialog
{
 public:
 TResp4(PTWindowsObject AParent, LPSTR AName);
 virtual void SetupWindow(void);
 virtual BOOL CanClose(void);
};

*****************************************************************
        DEFINICION DE LA CLASE TRESP DEL CAPÍTULO 5
*****************************************************************

class TResp5 : public TDialog
{
 public:
 TResp5(PTWindowsObject AParent, LPSTR AName);
 virtual void SetupWindow(void);
 virtual BOOL CanClose(void);
};
```

```
*****************************************************************
      DEFINICION DE LA CLASE TRESP DEL CAPÍTULO 6
*****************************************************************

class TResp6 : public TDialog
{
 public:
 TResp6(PTWindowsObject AParent, LPSTR AName);
 virtual void SetupWindow(void);
 virtual BOOL CanClose(void);
};

*****************************************************************
      DEFINICION DE LA CLASE TRESP DEL CAPÍTULO 7
*****************************************************************

class TResp7 : public TDialog
{
 public:
 TResp7(PTWindowsObject AParent, LPSTR AName);
 virtual void SetupWindow(void);
 virtual BOOL CanClose(void);
};

*****************************************************************
      DEFINICION DE LA CLASE TRESP DEL CAPÍTULO 8
*****************************************************************

class TResp8 : public TDialog
{
 public:
 TResp8(PTWindowsObject AParent, LPSTR AName);
 virtual void SetupWindow(void);
 virtual BOOL CanClose(void);
};
```

Clase Puntaje:

Derivada de TDialog, esta clase se identificó para poder mostrar el puntaje del alumno en la correspondiente lección, ya que se compara las respuestas correctas con las que el alumno ingresó, dando como resultado un puntaje.Como atributos podemos mencionar lo siguiente:

Nombre del alumno.

Puntaje obtenido.

Como servicios podemos mencionar lo siguiente:

Mostrar el puntaje

```
*****************************************************************
        DEFINICION DE LA CLASE TRESULTADO
*****************************************************************

class TResultado : public TDialog
{
public:
TResultado(PTWindowsObject AParent, LPSTR AName);
virtual void SetupWindow(void);
virtual BOOL CanClose(void);
};
```

Algunas de las clases TDialog_n que fueron definidas son las siguientes:

TPresentación, TInt, VentUno_Dialogo, VentDos_Dialogo, VentTres_Dialogo, TTest1, TTest2, TTest3, TTest4, TTest5, TTest6, TResp1, TResultado, TInfo, TBitm, TEmployeeDlg.

CAPITULO IV

DISEÑO E IMPLEMENTACION DEL TUTOR

4.1 ESTRUCTURA LOGICA DEL SISTEMA

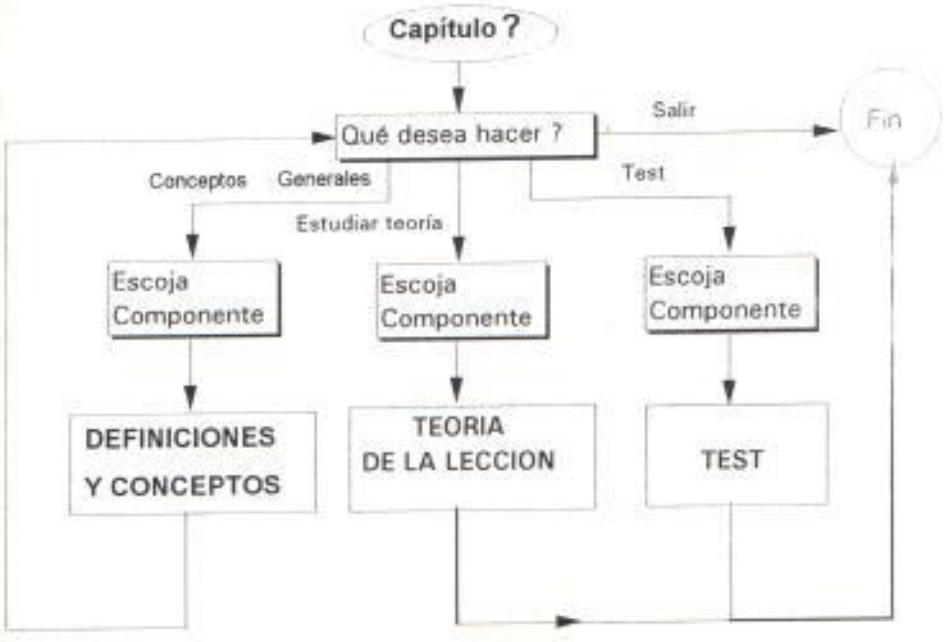
Las necesidades educativas a ser atendidas con el material computarizado sirven como base para operacionalizar las funciones que el software deberá cumplir. Debe considerarse las funciones hacia cada uno de los tipos de usuario a los que se dirigirá el material: al estudiante como base para aprender, al profesor como apoyo a la administración del proceso enseñanza- aprendizaje.

La estructura lógica de un programa de apoyo al aprendizaje expresa los procedimientos que el programa debe tener y sus interrelaciones, de modo que cumpla con las funciones definidas para sus usuarios y que permita al aprendiz recorrer la estructura de aprendizaje que subyace a los objetivos que se apoyan.

Siendo la estructura lógica una descripción macro, es deseable acompañar su diagramación con una explicación de cada una de las partes incluídas. Esto garantiza claridad en lo que abarca cada elemento y servirá de base más adelante para la especificación de macroalgoritmos para cada componente de la estructura lógica.

En el **Diagrama # 1** tenemos graficado la estructura lógica para la Selección de Capitulos del Sistema Tutorial manejado con la ayuda de un menú principal que comanda la acción del programa, en atención a las realizaciones y decisiones del usuario.

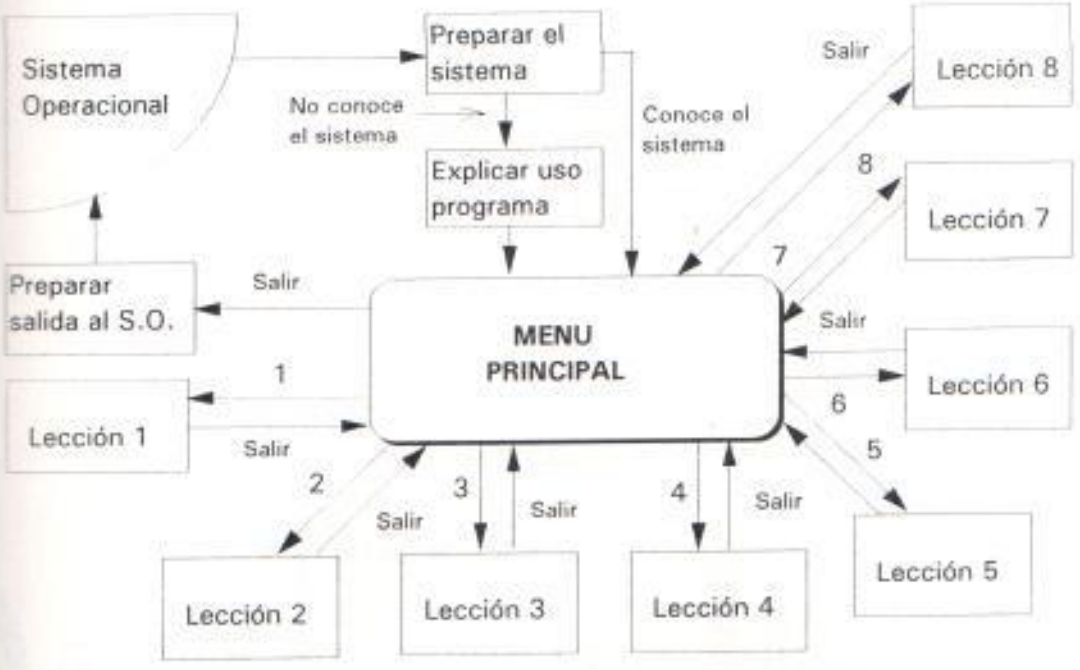
DIAGRAMA # 1



Es conveniente señalar que hay más de una forma de expresar la estructura lógica de interacción en un programa. El diagrama de flujo, expresa usandno propias convenciones, la secuencia de decisiones y procedimientos y así tenemos también los diagramas de transición que expresan diversos estados que pueden estar activos en un programa, y las condiciones que se deben cumplir para pasar de un estado a otro.

En el **diagrama # 2** presentamos la estructura lógica expresada en diagramas de transición para el caso de nuestro sistema TUTOR.

DIAGRAMA # 2



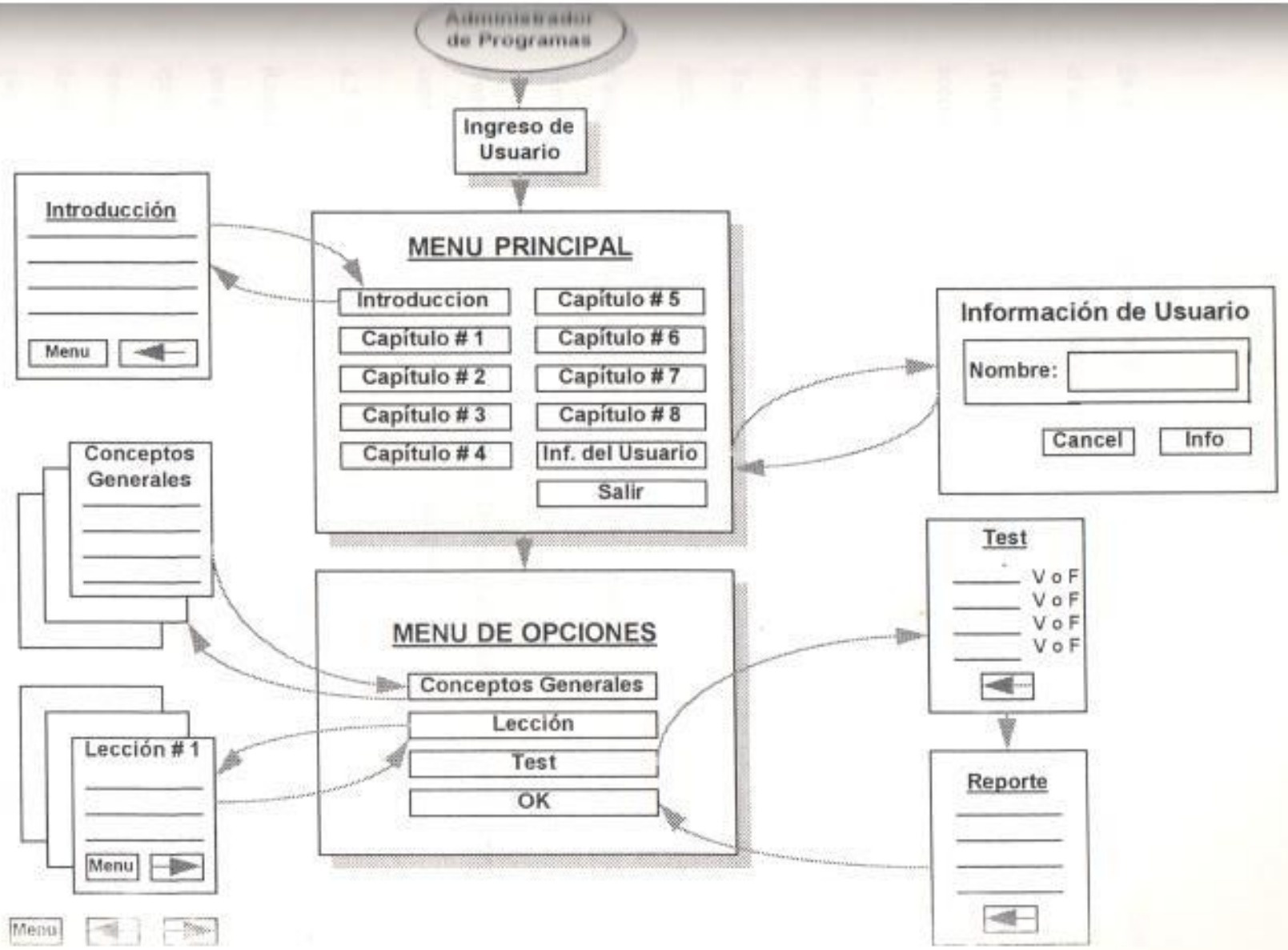
4.2 DISEÑO MODULAR DEL SISTEMA

El Tutor ha sido diseñado utilizando las clases Object Windows de la herramienta Orientada a Objetos que se identificaron en el análisis, además de utilizar estas clases, se utilizó un archivo cuya estructura permitía, guardar el nombre del alumno, la lección en la cual contestaba las preguntas, el puntaje que obtenía y la fecha del Sistema para saber en que fecha había realizado las preguntas.

Tutor TADS consta de dos programas de los siguientes programas:

Tutor00.cpp contiene las definiciones de las funciones para manejo del ingreso del usuario y de control de las cajas de diálogos para la presentación de los capítulos.

Tutor01.cpp contiene las definiciones de las funciones para el manejo y control de las pantallas principales, menú de opciones, test, lecciones, conceptos generales.


Administrador de Programas
Ingreso de Usuario
Introducción
Menu
MENU PRINCIPAL
Introduccion
Capítulo # 1
Capítulo # 2
Capítulo # 3
Capítulo # 4
Capítulo # 5
Capítulo # 6
Capítulo # 7
Capítulo # 8
Inf. del Usuario
Salir
Información de Usuario
Nombre:
Cancel
Info
Conceptos Generales
MENU DE OPCIONES
Conceptos Generales
Lección
Test
OK
Test
V o F
V o F
V o F
V o F
Lección # 1
Menu
Reporte
Menu

En el siguiente **diagrama # 3** se muestra el diseño modular en base al cual se realizó el desarrollo del Tutor de Análisis y Diseño de Sistemas.

Tutorw00.rc contiene las definiciones para el manejo y presentación de los bitmaps, botones y cajas de diálogos.

Tutocab1.h contiene las definiciones de las estructuras de los archivos de los registros de usuarios.

Tutorw00.h contiene definición de los identificadores de control utilizados en la definición del Tutorw00.rc

Tutorw01.h contiene las definiciones de las clases que controlan el manejo de la ventana principal.

Tutorw02.h contiene las definiciones de las clases para la presentación de los capítulos, introducción, test, lecciones y conceptos generales.

4.3 DISEÑO DE PANTALLAS Y MENUES DEL PROGRAMA

Respecto al diseño de pantallas, tenemos que tomar en cuenta las siguientes características: zona de identificación, en la que se muestra al usuario donde está en la ejecución del programa, también suele haber una zona de control donde, a través 1de menúes, funciones e iconos, se ofrece al usuario alterar el flujo de la acción, una zona de intercomunicación donde intercambian mensajes entre usuario y programa, en el que el usuario lleva a cabo las operaciones que requiere. La zona de retroinformación sirve para que el computador informe al usuario acerca de su desempeño, sea de una manera implícita o explícita.

Además de la definición de las áreas de trabajo, es necesario establecer las características de los elementos que se van a utilizar en ellos: menús, gráficos, mensajes, textos, animaciones, etc.

A pesar de aquellos procesos anteriores prácticamente determinan el diseño de comunicación y control del material, es prudente que completemos la especificación educativa a nivel de cada una de las pantallas.

Para nuestro caso, se utilizarán ventanas mostrando la información a manera de texto o gráfico de acuerdo al tema que se esté tratando. La interacción entre el tutor y el estudiante se realizará por medio de ventanas de diálogo o de entrada/salida de información desde/hacia el usuario.

La figura **# 1 -2** representan algunas de las pantallas principales que se presenta el Tutor, como por ejemplo la **pantalla de menú principal** con las diferentes opciones o tópicos a elegir, la pantalla de presentación de las lecciones, test y conocimientos generales.

4.4 TIPO DE REPORTES

Una vez seleccionada la opción de TEST, se procederá a la presentación de una pantalla de presentación de preguntas lógicas, en la que el usuario deberá establecer si sus respuestas son verdaderas o falsas, en base a las respuestas dadas por el usuario, se procederá a realizar una evaluación y presentar un reporte con el:

* Nombre del Estudiante.
* Capítulo al que ingresó
* Calificación.(puntaje) y fecha.

Menú Principal con los capítulos que presenta el tutor de Análisis y Diseño de Sistemas.

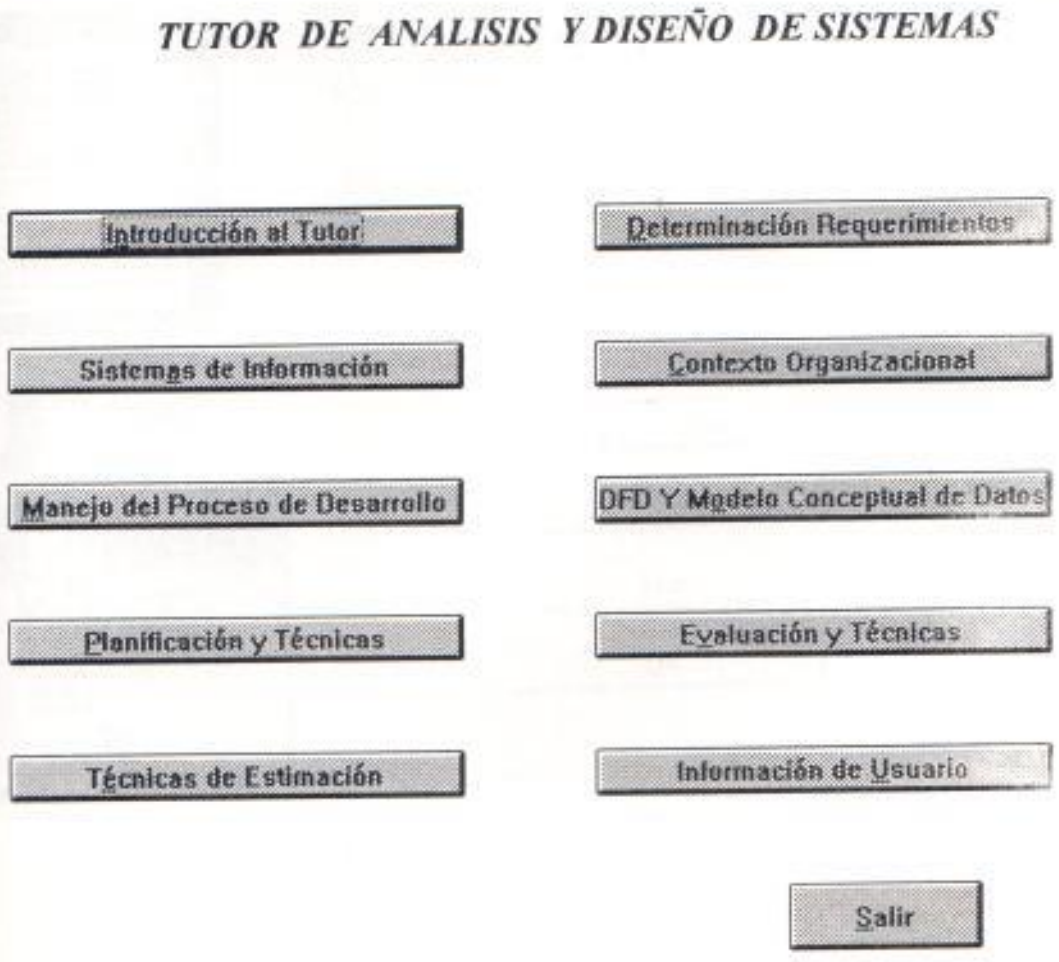


Figura # 1

Menú con las opciones que presenta cada uno de los capítulos del tutor

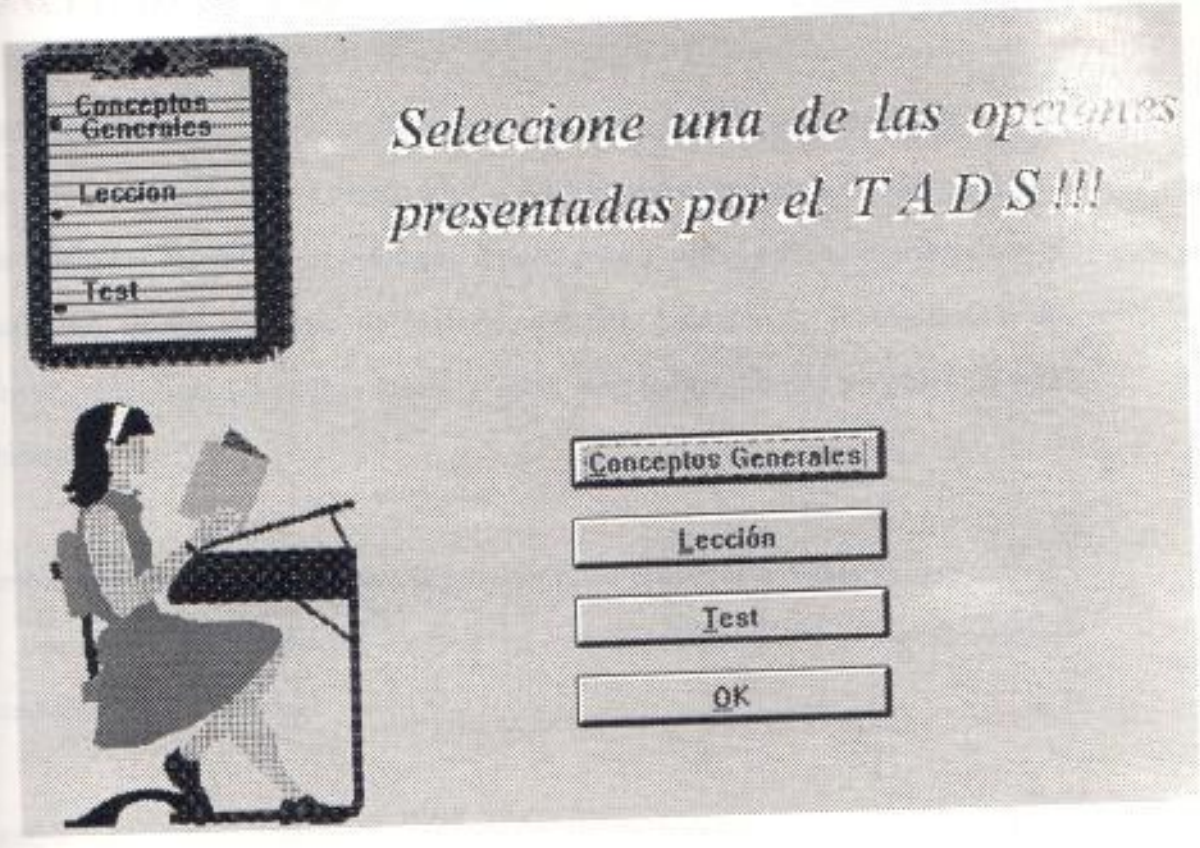


Figura # 2

CONCLUSIONES Y RECOMENDACIONES

El tutor de Análisis y Diseño de Sistemas ha sido desarrollado usando la metodología de programación orientada a objeto (Borland C++) bajo el ambiente Windows, debido a las múltiples ventajas y beneficios que soporta para el manejo de las interfases con el usuario, en cuánto a manejo de gráficos, pantallas interactivas, procesamiento de archivos, reducción del código fuente y otras facilidades que nos proporciona este ambiente.

El ambiente Windows es ideal para la programación orientada a objeto ya que todo lo que ocurre en el Sistema se traduce en mensajes que pueden ser fácilmente contestados por los objetos.

Se emplean conceptos de herencia, polimorfismo y encapsulación, los cuales permiten definir por ejemplo una clase derivada con datos y funciones que puede ser referenciada en varias partes del programa, y nos permite además agregar otras funciones miembros a la clase predefinidas, cambiar la forma en que el ratón aparece en pantalla, eventos tales como realizar doble-click en una determinada área de la ventana de una lección y se ejecuta otra, a manera de hipertexto, invocar archivos ejecutables externos, entre otras.

Una ventana en la pantalla trabaja de la misma manera que un objeto en Borland C++. Cada ventana encapsula todo sobre las rutinas y los datos que la ventana usa, justo como un objeto encapsula sus rutinas y datos, una ventana puede heredar y construir de las características de otras ventanas, tal como un objeto. Cada ventana maneja mensajes del Sistema Operativo en una manera que es similar al polimorfismo de

Borland C++, esto es que pueden recibir mensajes comunes y tomar acción apropiada como respuesta de la forma apropiada para cada ventana.

ObjectWindows permite el manejo de gráficos creados de diferentes formas, y la condición para que sean llamados en el programa es que sean grabados dichos archivos con la extensión ".bmp" e invocados desde el programa o que sean colocados en el programa código de binario. Estas y otras facilidades permitieron crear un programa amigable con pantallas gráficas que facilitan al estudiante su aprendizaje.

Un Sistema Tutorial es un método de enseñanza que se basa en un material de tipo algoritmico en el que predomina el aprendizaje vía transmisión de conocimiento, desde quien sabe, hacia quien lo desea aprender y donde el diseñador se encarga de encapsular secuencias bien diseñadas de actividades que conducen al aprendiz, desde donde está hasta donde desea llegar.

Los Sistemas Tutoriales guian al aprendiz a través de las distintas fases del aprendizaje, por medio del diálogo.

Las actividades y el entorno del aprendizaje también dependen de lo que se esté enseñando y de su nivel, así como de las personas a las que se dirige. Las oportunidades de práctica y de retroinformación asociadas están directamente ligadas con lo que se esté enseñando y son parte muy importante del sistema tutorial.

La utilidad de los Sistemas Tutoriales radica en que el computador se vuelve particularmente útil cuando está llamado a ofrecer un ambiente entretenido amigable y excitante que permite a los alumnos superar el desgano que la temática les genera y embarcarse en una experiencia que les ayude a superar las limitaciones que tengan en el uso de sus destrezas.

En cuánto a las recomendaciones que se puedan establecer se encuentran:

1. En caso de las evaluaciones se puede implementar un mecanismo que permita establecer los puntos débiles del estudiante de manera que proporcione de forma aleatoria más preguntas sobre el tema en el cuál está fallando, así que como ilustre en un reporte de resultados los tópicos en los que falló y las guías necesarias en donde puede encontrar las respuestas correctas.

Otra clase de evaluaciones que se recomienda está la implementación de frases incompletas en las que el estudiante deberá seleccionar la respuesta correcta de una serie de alternativas que se presenten, además podría incrementarse el número de las respuestas para así aumentar el grado de dificultad.

O podría el Tutor presentar varias preguntas al estudiante y éste seleccionar la respuesta correcta a cada una de ellas.

2. Involucrar animación y la utilización de multimedios (voz, imagen y datos), de esta forma el aprendizaje o estudio se vuelve más dinámico.

3. Incrementar el número de programas tutores para el aprendizaje en distintas áreas de interés.

4. Se sugiere la implementación de CASOS, de tal manera que el tutor muestre los pasos que deban seguirse y así como él o los métodos apropiados para el análisis y diseño del Sistema propuesto.

BIBLIOGRAFÍA

SYSTEMS DEVELOPMENT

Requerimientos, Evaluación, Diseño e Implementación.

Eleanor Jordan and Jefry J. Machesky, PWS-KENT Publishing Company, Edición 1990.

OBJECT ORIENTED ANALYSIS

Kelly Behr & Susan Brunke.

BORLANDC C++ 3.0 Tools and Utilities Guide

Borland International Corp. 1992

TEACH YOURSELF ...C++

Al Stevens, Publishing for the Twenty-first century.

WINDOWS PROGRAMMER'S GUIDE TO BORLAND C++ Tools

James McCord, Publishing SAMS, Edición 1992.

MICROSOFT WINDOWS 3.1

Manuales de referencia para el usuario, Microsoft Corp. Internacional 1992.