UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Campus Aragón

Análisis sobre lenguajes de programación basándose en la creación y manejo de sistemas de Base de Datos

T E S I S

Que para obtener el título de

INGENIERO EN COMPUTACION

p r e s e n t a

GABRIEL JAIMES GONZALEZ

Asesor de Tesis:

Ing. Juan Gastaldi Pérez

San Juan de Aragón 1999

Gracias:

A Dios: Porque siempre me da fuerzas para salir adelante y me da la oportunidad de vivir y realizarme como ingeniero.
A Mis Padres: Porque siempre cuento con su apoyo incondicional y esta tesis es realmente de ellos.
A Adriana Tlatelpa: Por todo lo que significa en mi vida y porque en este trabajo hay mucho de ella.

INDICE

INTRODUCCIÓN.

La realización de sistemas de base de datos es una de las principales tareas a las que se dedica el ingeniero en computación, ya que los sistemas forman parte esencial en todas las empresas.

La importancia que tienen los sistemas de base de datos se basa en que por medio de estos se realizan de manera automática muchas de las tareas que las empresas tenían que realizar de forma manual ahorrando con esto tiempo y esfuerzo para las empresas y esta ahorro da beneficios a la empresa que trabaja con sistemas automatizados.

Existen varios problemas que se generan cuando un programador tiene la responsabilidad de crear un sistema de base de datos, pero el principal radica en cual será el lenguaje que se utilizara para realizar el sistema. El lenguaje que se seleccione deberá ser un lenguaje que ofrezca las herramientas que el programador necesita para poder cumplir satisfactoriamente con las condiciones propuestas por la empresa para la realización del sistema, sobre todo porque de esta elección dependerá el buen manejo de los datos que se trabajen en cualquier sistema.

Algunas empresas deciden realizar el sistema en el lenguaje que el programador considere mas optimo para poder trabajar adecuadamente, pero el programador necesita realizar un análisis acerca de los requerimientos y recursos que la empresa tenga para poder determinar el lenguaje en el cual se trabajara.

El programador tiene que revisar los recursos técnicos y económicos con las que cuenta la empresa para la cual se realizara el sistema, es decir que para la realización del sistema el programador deberá de adecuarse a las condiciones que imponga la empresa para poder trabajar, aunque como se menciono anteriormente algunas empresas dejan en manos del programador la elección del lenguaje de programación.

INTRODUCCIÓN.

La realización de sistemas de base de datos es una de las principales tareas a las que se dedica el ingeniero en computación, ya que los sistemas forman parte esencial en todas las empresas.

La importancia que tienen los sistemas de base de datos se basa en que por medio de estos se realizan de manera automática muchas de las tareas que las empresas tenían que realizar de forma manual ahorrando con esto tiempo y esfuerzo para las empresas y esta ahorro da beneficios a la empresa que trabaja con sistemas automatizados.

Existen varios problemas que se generan cuando un programador tiene la responsabilidad de crear un sistema de base de datos, pero el principal radica en cual será el lenguaje que se utilizara para realizar el sistema. El lenguaje que se seleccione deberá ser un lenguaje que ofrezca las herramientas que el programador necesita para poder cumplir satisfactoriamente con las condiciones propuestas por la empresa para la realización del sistema, sobre todo porque de esta elección dependerá el buen manejo de los datos que se trabajen en cualquier sistema.

Algunas empresas deciden realizar el sistema en el lenguaje que el programador considere mas optimo para poder trabajar adecuadamente, pero el programador necesita realizar un análisis acerca de los requerimientos y recursos que la empresa tenga para poder determinar el lenguaje en el cual se trabajara.

El programador tiene que revisar los recursos técnicos y económicos con las que cuenta la empresa para la cual se realizara el sistema, es decir que para la realización del sistema el programador deberá de adecuarse a las condiciones que imponga la empresa para poder trabajar, aunque como se menciono anteriormente algunas empresas dejan en manos del programador la elección del lenguaje de programación.

También se debe de hacer un análisis acerca de las tareas que se necesita que realice el sistema, asi como también de la presentación que este ofrecerá. Ademas de que se debe de considerar la forma en la que se manejan las tablas ya que esta contienen todos los datos que el sistema necesita para trabajan y del buen manejo de la tabla dependerá mucho el buen manejo del sistema.

El trabajo que se esta realizando puede ofrecer a la gente que este relacionada con la creación o manejo de algún sistema una herramienta para poder elegir el software adecuado y que dicho sistema trabaje de la manera mas optima.

Esta trabajo tratara de mostrar al lector un análisis detallado de algunos de los lenguajes de programación que cuentan con herramientas para el trabajo con las bases de datos. Aunque existe mucho software en el mercado para el manejo de las bases de datos se contempla que se esta revisando el software mas actual y mas comercial dentro del campo de las bases de datos.

En el capitulo I se analizaran los principales conceptos de las bases de datos y de sistemas de bases de datos. En la revisión de los conceptos de bases de datos analizaremos también los diferentes tipos de bases de datos que existen , por otro lado dentro de los sistemas de bases de datos revisaremos como se debe de realizar un análisis del sistema. Este análisis es muy importante ya que de ahí comenzaremos a definir cual es el software que mejor se adecua al sistema en cuestión ademas de que siempre es recomendable que se realice un análisis previo para poder definir todo lo que necesitamos para la realización del sistema.

En los cinco capitulos siguientes se revisara detalladamente las herramientas que ofrecen los lenguajes de programación Clipper, FoxPro, Visual Basic, Delphi, y Visual C del capitulo II al VI respectivamente.

El hecho de haber tomado estos lenguajes para trabajar no significa que se consideren los mejores o los mas potentes, se han seleccionado estos ya que se piensa que con ellos se

abarca gran parte de los ambientes que se tienen para realizar sistemas de bases de datos las causas de porque se eligieron los sistemas se explican a fondo en los respectivos capítulos.

La revisión de los lenguajes se hará en base a cuatro puntos que se consideran esenciales para el adecuado análisis del software en cuestión. Estos puntos se revisaran en todos los lenguajes que se mencionaron anteriormente:

Introducción: que es donde se analizara cuales son las características básicas que tiene el software es decir los conceptos generales acerca del lenguaje y cuales son los requerimientos técnicos que solicita este software para un buen funcionamiento. Hablaremos en esta parte también de los costos que se generarían al realizar un sistema en el software que se este revisando.

Herramientas: este es el punto mas importante ya que en este se revisaran todas las herramientas que ofrecen estos lenguajes, herramientas para presentación contemplando menús y gráficos, herramientas que faciliten la programación tales como etiquetas e informes y también como se crean y manejan las tablas en dicho lenguaje. Ademas también se hará un análisis acerca de como es la programación en el lenguaje que se este revisando, es decir se va a analizar que tan dificil resulta programar en los lenguajes revisados. Y se contemplaran ademas otras herramientas que puedan ofrecer los lenguajes.

Desventajas y ventajas: en este punto se comparara el lenguaje en cuestión con los otros incluso con otro software que no se este revisando en este trabajo. Estas ventajas o desventajas se basaran en los análisis que se están realizando. También se revisara en esta parte la migración, es decir que tan fácil será convertir algún programa o sistema realizado en un software a otro, por ejemplo si se tiene un sistema realizado en Visual Basic y se quiere cambiar a Visual C se tendrá que verificar si lo que se tiene programado puede servir o no, si las tablas ya realizadas se pueden volver a utilizar o no, etc

Ejemplo: para poder realizar un análisis amplio de los lenguajes en cuestión no solo basta con tomar datos de libros o de las mismas ayudas de los lenguajes si no que es necesario tener un sistema para poder compararlo en los diferentes lenguajes, es por ello que en este trabajo se realizara un sistema consistente en un recetario medico el cual permitirá utilizar la mayoría de las herramientas que el software que se esta analizando maneja.

Cabe mencionar que este sistema será el mismo en cada uno de los lenguajes revisados. El sistema será utilizado de ejemplo en todo el trabajo ya que para el análisis de capitulo I se requiere también de algunos ejemplos, por lo que será en este parte donde se explique detalladamente en que consiste este sistema.

Este sistema de ejemplo se englobara al final de cada uno de los capítulos pero dentro del desarrollo del tema se irán mostrando ejemplos de como se van utilizando las herramientas para poder llegar al resultado final que en este caso será el sistema medico.

1 BASES DE DATOS.

1.1 Conceptos generales

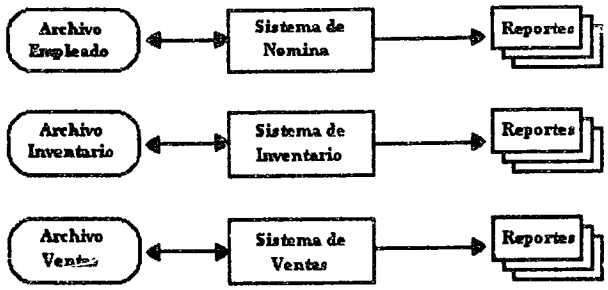
A diario todos los seres humano hacemos uso de las bases de datos por ejemplo cuando consultamos un diccionario, un directorio telefónico o vamos a una biblioteca. Debido a los grandes avances de la tecnología el manejo de bases de datos ha pasado a ser parte importante de la informática.

Cuando se habla de las bases de datos lo primero que se nos viene a la cabeza es un conjunto de datos acomodados de alguna forma, pero hablando de una manera mas formal una base de datos es una colección de datos relacionados y estructurados de tal manera que nos permitan consultarlos y de esta manera poder trabajar con ellos.

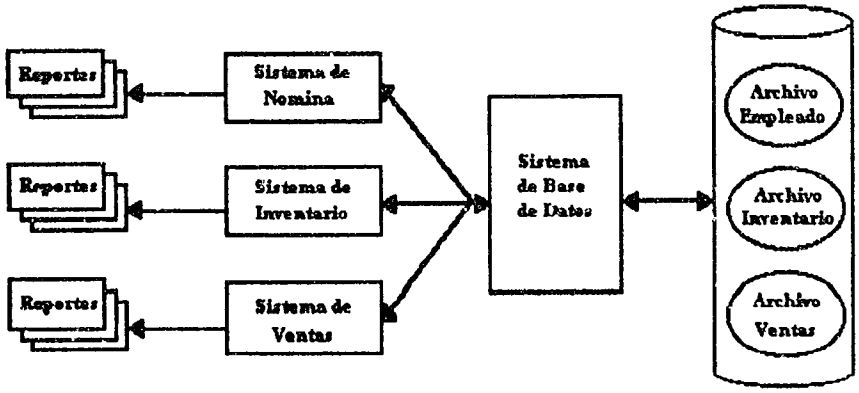
Desde el punto de vista informático podemos decir que las bases de datos son programas especializados en el manejo de información que nos permiten manejar cantidades de datos, ordenándolos de tal manera que su consulta sea cómoda, practica y productiva.

Mucha gente a comprobado a través de los años que con el manejo de las bases de datos se da solución a todos los problemas de administración de la información, porque de hecho cualquier área que se pueda mencionar requiere de manejar mucha información para múltiples tareas.

En algunos casos la información se maneja de forma separada para los procesos que de alguna forma están ligados. Observe la figura siguiente.



De ahí que se considere tener los datos en una base que sea la encargada de almacenar toda la información que nuestra empresa requiera. Observe la figura.



Las bases de datos son actualmente la herramienta mas valiosa con la que cuentan algunas empresas, por ejemplo una escuela tiene registrados a todos sus alumnos en bases de datos, una tienda tiene registrados todos sus productos en una base de datos, etc. de ahí la gran importancia que tiene las bases de datos.

Es importante indicar que la información que se encuentran en las bases de datos se encuentra acomodada en archivos por medio de campos y registros, mas adelante veremos mas a fondo estos conceptos.

Cabe mencionar que para el manejo de las bases de datos la información por si sola no puede manejarse o relacionarse, es decir los datos por si solos no sirven de mucho, de ahí que se requiera una herramienta para poder administrar estos datos. El sistema administrador de bases de datos (DBMS) realiza precisamente esta tarea.

El DBMS en un concepto mas exacto se puede considerar como un conjunto de programas desarrollados para ejecutarse en una aplicación administrativa de bases de datos.[1]

En el mercado actual existen muchos administradores de bases de datos (Dbase, Access, etc. por mencionar algunos), la mayoría de estos paquetes nos permiten trabajar los datos de forma sencilla aunque para poder trabajar con ellos es necesario tener conocimientos básicos en computación, estos también llamados manejadores de bases de datos ofrecen muchas ventajas y desventajas pero también existen en el mercado lenguajes que permiten crear sistemas cuya tarea principal sea la de administrar la información de las bases de datos, entiéndase por administrar el uso y manejo de los datos que se tienen, y como se puede ver creando un sistema también se van a crear un conjunto de programas que ejecutándose en una aplicación van a administrar bases de datos.

La principal ventaja que nos ofrece tener nuestros propios sistemas es que los sistemas se deber realizar orientados a las actividades que nuestra información requiera, es decir los sistemas son mas personalizados al área que nosotros estemos manejando, de ahí que sea mas fácil de aprender a manejar un sistema que un gran manejador de bases de datos. Ademas el sistema nos permitirá agregarle o quitarle algunas tareas que se requieran.

En términos generales las operaciones básicas que se deben realizar con los programas de administración son:

- Elaborar formularios que nos permitan recolectar información dentro de un proceso de trabajo

[1] Cesar E. Rose, Archivos, organización y procedimientos, Edit. Computec, 1993.

- Introducir nueva información de manera sencilla y rápida con el objeto de que esta se pueda clasificar para una posterior búsqueda.
- Elaborar listas ya sea ordenadas alfabéticamente o por forma numérica de acuerdo a las necesidades que el usuario tenga.
- Agregar, borrar y poder modificar datos de manera que la base siempre se encuentre actualizada.
- Imprimir de fácil manera la información que la base contenga.
- Posibilidad de ajustar los programas a las necesidades de los usuarios.

En este trabajo utilizaremos una base de datos que contiene 500 medicamentos escogidos con la finalidad de que se pueda mostrar la flexibilidad que tendrán los programas de aplicación. La forma en que se encuentran organizados estos medicamentos se explicar en el punto siguiente.

I.2. Definiciones.

En el punto anterior quedo mas que definido el concepto de bases de datos y de sistema administrador de las bases de datos, pero para poder entender mejor estos conceptos analizaremos en este parte diferentes temas que están ligados al concepto de bases de datos.

Comenzaremos la definición de términos analizando la diferencia que existe entre los conceptos de información y datos. Los datos han nacido para la humanidad debido a que los seres humanos han tenido la necesidad de registrar o de referir diversos sucesos.

Formalmente se puede decir que los datos vienen a ser los testimonios de la información que tenemos acerca de un hecho, es decir son el reflejo de las condiciones en que se encuentra una situación dada.[2]

[2] Cesar E. Rose, Archivos, organización y procedimientos, Edit. Computec, 1993

La información por otro lado son los datos sometidos a un proceso, es decir los datos son la materia prima que sirve para crear la información.

El concepto que se acaba de ver acerca de los datos se ha referido para que el lector pueda entender que es la estructura de datos.

Se menciono anteriormente que las bases de datos se encuentran organizada de alguna forma. Esta forma de organizar los datos este definida por una estructura. Esta estructura es muy importante ya que de ella depende la flexibilidad que tengan los datos para su manejo. La estructura de datos dará la pauta para que los programas aplicación accesen a los datos de forma correcta.

Al crear una estructura de base de datos es necesario que se conozca cuales son las aplicaciones que se le van a dar al sistema que va a manejar estos datos ya que en esta estructura se va a definir la manera en que se van a almacenar los datos. Por ejemplo en algunas bases de datos se requiere el nombre de la persona de forma completa para lo cual en la estructura se elige definir si el nombre lo vamos a almacenar en un solo campo o en varios, es decir podemos poner en la estructura un campo único para el nombre completo o tres campos uno para el apellido paterno, otro para el apellido materno y otro para el nombre.

I.2.1 Campos y Registros.

Para poder crear la estructura de una base de datos es necesario que conozcamos el concepto de campo. Los datos se pueden representar por medio de caracteres alfanuméricos y por números de tal modo que un dato como por ejemplo el nombre de una medicina, un numero de identificación, etc. puedan ser especificados por una entidad que se llama campo, entonces entendamos por campo una unidad de almacenamiento para guardar un dato.

Para especificar una estructura de base de datos se requieren tres datos para los campo: nombre, tipo y tamaño.

Los campos son muy importantes porque van a establecer que tipo de dato se va a almacenar. Básicamente existen cinco tipos de campos que se pueden utilizar:

- Numérico: aquel que solo aceptara números.
- Carácter o alfanumérico: es aquel que aceptar todo tipo de caracter ya sea números, letras, espacios y caracteres especiales, aunque solo acepta almacenar 255 caracteres por registro.
- Fecha: es un campo que almacena únicamente fechas con un formato establecido.
- Memo: es un campo que también permite todo tipo de caracteres pero la diferencia es que este permite mas de 255 caracteres.
- Lógico: es un campo que solo permite dos entradas ya sea falso o verdadero

Se acaba de revisar los tipos de campo que basicamente existen aunque depende del lenguaje los tipos de campos que se manejaran, algunos manejan mas tipos, otros menos, etc. En este trabajo se revisara mas adelante los tipos de campo que trabajan nuestros lenguajes

Al realizar el análisis de los campos también tenemos que tomar en cuenta el tamaño que van a tener estos. El tamaño básicamente dependerá del usuario o del programador según las necesidades que el sistema tenga, pero cabe mencionar unos puntos importantes.

- El tamaño de los campos tipos caracter es abierto permitiendo hasta 255 caracteres.
- En los campos de tipo numérico se debe de declarar también un tamaño para los decimales en caso de que se vayan a utilizar estos, cuando no se utilicen decimales el tamaño de estos sera 0. Cuando se ocupen decimales el tamaño del campo deberá comtemplarlos a estos y al punto decimal, es decir si se fuera a utilizar un formato como el que sigue 999.99 el tamaño del campo deberá de ser 6 y 2 decimales
- Los campos de tipo fecha tienen un tamaño predeterminado de 8.

- Los campos de tipo memo se utilizaran según las necesidades de usuario.
- Por supuesto como los campos lógicos solo permiten una entrada el tamaño de estos sera de 1.

Finalmente el nombre del campo sera responsabilidad también del usuario pero la mayoría de los manejadores de bases de datos no permiten mas de 10 caracteres para dicho nombre.

Nos hemos tomado una parte para explicar los campos detalladamente ya que estos serán utilizados como variables dentro de los programas de aplicación.

Por otro lado los registros serán la información que el sistema va a tener, es decir que una vez que se ha formado la estructura para la base de datos se podran almacenar los registros siguiendo la estructura ya creada

Observemos ahora como funcionan los conceptos que acabamos de ver en el ejemplo que para este trabajo tenemos destinado La estructura de la base que se presenta sera para la base de datos de medicamentos.

Nombre del campo	Tipo del Campo	Tam,año del campo
Medicament	Caracter	50
Generico	Caracter	60
Registro	Caracter	8
Grupo	Caracter	60
laboratori	Caracter	60
Presentac	Caracter	30
reacsecun	Memo	4
indicacion	Memo	4
contraindi	Memo	4
Dosis	Caracter	50

I.2.2 Archivos

Obviamente cuando se esta generando una base de datos es porque la información que esta contenga estará almacenada en archivos para poder ser utilizada por los programas de aplicación.
La definición de archivos puede resultar muy simple pero para el tema que estamos tratando es algo compleja ya que tenemos que tener en cuenta que de el correcto almacenamiento de la información sera el correcto funcionamiento del DBMS.

Una definición muy clara de archivo seria "un archivo se define como un conjunto de registros semejantes conservado en dispositivos de computadora de almacenamiento secundario.[3]

Al manejar un sistema de administrador de bases de datos tendremos que manejar varios tipos de archivos :

Archivo maestro: es el archivo que guardará toda la información, es decir sera el que contrendrá la base de datos, mas adelante analizaremos mas a detalle este archivo.

Archivos de programa: son los que contienen el código que permitirá realizar las actividades con la base de datos, es decir este es el que hemos llamado programa de aplicación.

Archivo de reporte: es el que se encargara de tener los formatos que el programador establezca para la presentación de la información al usuario.

Archivo de respaldo: se trata de archivos que son copias de seguridad de los anteriores.

[3] S.M. Deen, Fundamentos de los sistemas de BD., Edit. Gustavo Hili S.A. 1987

Aunque se pueden añadir mas tipos de archivos a la lista se toman los anteriores ya que son los nos encontramos en todos los manejadores, algunos cuentan con mas archivos para sus tareas pero eso ya se vera cuando se analicen los lenguajes.

Archivo Maestro

Como ya se menciono el archivo maestro guarda toda la información por lo que es muy importante revisarlo un poco mas.

Existen ciertas tareas que los archivos de bases de datos deben de considerar como las posibles tareas a realizar.
Inserción de un nuevo registro
Modificación de un registro existente
Borrado de algún registro.
Ordenamiento de los datos que la base de datos tiene.
Reestructuración de la base de datos, es decir agregar o quitar campos.

Aunque se acaba de mencionar algunas tareas, la tarea principal de este archivo sera la de entrada/salida de información por lo que este archivo tiene que cumplir con ciertas condiciones que se presentan a continuación:

Tamaño: aunque no es muy relevante el tamaño de las bases de datos ya que con la gran capacidad de las computadoras estas pueden manejar millones de datos si es importante tener en cuenta que no se debe de tener archivo de gran tamaño sobre todo por cuestiones de almacenamiento en disco. Ademas recordemos que dependiendo del tamaño del archivo sera la velocidad en la que se realicen búsquedas en el.

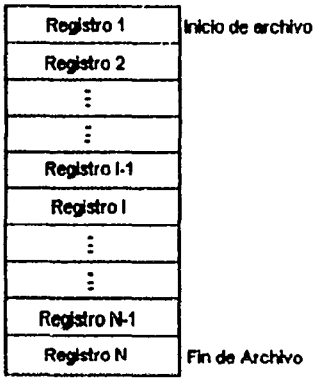
Flexibilidad: aunque un archivo de base de datos debe de ser seguro también debe de ofrecer mucha flexibilidad para poder compartir (entrada/salida) la información que este contiene ya

que de ahí dependerá el buen manejo del sistema. Ademas también deben de tener la facilidad de que la información pueda ser ordenada de alguna forma

Recuperación: el archivo maestro debe de tener la posibilidad de recuperar información perdida, esto por lo regular se hace por medio de las copias de seguridad. Para este fin ademas de las copias de seguridad, cuando se borra un registro no se hace físicamente sino que se marca de manera lógica, es decir que no desaparece de la base de datos sino que únicamente el registro se oculta.

Ya que se han definido las condiciones que debe de tener un archivo para trabajar adecuadamente estudiemos ahora la forma de almacenamiento que presentan los archivos maestros. Básicamente hablaremos de tres formas de almacenamiento: secuencial, con índice y multillave.

La forma de almacenamiento secuencial es aquella en que los registros son guardados uno detrás del otro sin importar un orden del almacenamiento, es decir que el registro siempre se va a agregar al final de las base de datos. Observe la figura siguiente:



El almacenamiento como se puede ver en la figura es secuencial física y lógicamente, al hablar físicamente nos referimos a como la base de datos guarda la información el la unidad

de almacenamiento y lógicamente es como de manera interna también almacena la información igual.

Las ventajas y desventajas de esta forma de almacenamiento son claras, por un lado al agregar un registro, es decir en la entrada de información como el registro se acomoda al final sin importar algún orden, la velocidad a la que hace la entrada de información es mayor pero cuando se va a utilizar un registro que ya se contiene en la base para consulta o modificación, este proceso resulta algo lento ya que al realizar la búsqueda debe de recorrer todos los registros antes del que estemos buscando. Por ejemplo regresando a la figura revisemos que si quisiéramos llegar al registro I tendríamos que recorrer I-1 registros.

En los archivos de almacenamiento secuencial se ha buscado la manera de ordenar la información pero se cae en un problema. Al ordenar la información se genera el nuevo archivo con la información ordenada pero al agregar nuevos registros a las base se tiene que hacer nuevamente la ordenación, esta ordenación se hace de manera física por lo que el proceso es lento

Los archivos secuenciales son ocupados para poder trabajar los archivos de reporte ya que dentro de estos no se modifica la información sino que únicamente se presenta al usuario ya sea en forma de etiquetas o en forma de informes. Ademas de que en los archivos secuenciales su tamaño en menor que en los demás archivos de almacenamiento.

Ahora hablaremos de la forma de almacenamiento indexada. El index es una llave que se asigna a los registros con la finalidad de que por medio de esta llave se organicen los registros en la base de datos.

En la tabla siguiente podemos ver una base de datos almacenada en forma secuencial y como se vería en forma indexada.

1	Propirina
2	Isodine
3	Tesofin
4	Alin
5	Dorbanex
6	Vincapan
7	Aspirina
8	Doriol
9	Enfamil

4	Alin
7	Aspirina
5	Dorbanex
8	Doriol
9	Enfamil
2	Isodine
1	Propirina
3	Tesofin
6	Vincapan

Como se puede ver en la tabla el ordenamiento indexado de los registros se hace de forma física pero no de forma lógica por que se puede ver que el registro 1 de la base secuencial sigue siendo el 1 pero ubicado en otra posición, es importante mencionar que los datos se introdujeron a las bases de datos en el mismo orden.

La llave se puede elegir según las necesidades del usuario, es decir se puede elegir cualquier campo como llave para que se ordene según ese campo. Al almacenar una dato este se acomoda automáticamente dentro de la base, el registro será el registro N pero se añadirá en la posición que le corresponda según la llave. Sunpóngase que se van a agregar a la base que tenemos dos registros Bactrim y Oncovin, obviamente el numero de registro será 10 y 11 consecutivamente pero observemos la posición física en la que se acomodarían. Observe la figura siguiente.

4	Alin
7	Aspirina
10	Bactrim
5	Dorbanex
8	Doriol
9	Enfamil
11	Oncovin
2	Isodine

1	Propirina
3	Tesofin
6	Vincapan

Al añadir la información de esta forma la entrada de datos es rápida pero para lo que mas nos ayuda es para los procesos de salida de información ya que los registros se encuentran ordenados según una llave lo que facilita las búsquedas haciéndolas mas rápidas y eficientes.

En el proyecto que se va a realizar en este trabajo la forma de almacenamiento será indexada ya que proporciona mayor velocidad en los procesos de búsqueda de información.

Por ultimo la forma de almacenamiento multillave es muy parecida a la forma indexada ya que también funciona por llaves, la diferencia es que no solo existirá una llave sino que pueden existir varias llaves secundarias.

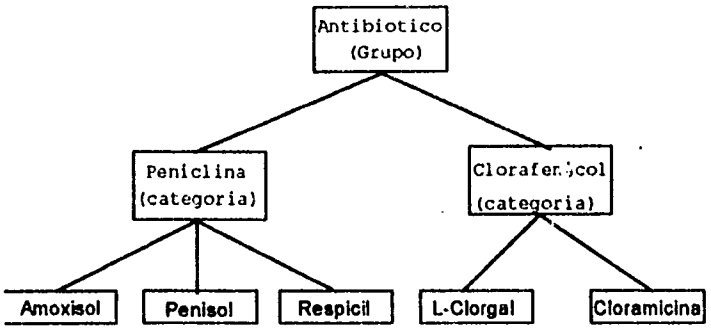
La finalidad de estas llaves será la de hacer un poco mas rápido el manejo y consulta de los registro.

El manejo del almacenamiento es multillave es de la siguiente manera, se cuenta con una llave primaria que es la principal y la que se encarga del primer ordenamiento pero supongamos el caso de la base de datos del IFE y tomando como llave principal el nombre de pila, como existen muchas personas con el mismo nombre se cae en un problema al ordenar estos registros, pero con las llaves secundarias evitamos este problema al tener otras llaves que, una vez ordenado en base al nombre tomaran los grupos de nombres y los ordenaran en base a otro campo, este método puede tener una o mas llaves secundarias que serán las que determinaran otros criterios de ordenación pero siempre respetando que el primer ordenamiento lo hace la llave primaria.

I.3 Tipos de Bases de Datos.

Aunque existen varios tipos de bases de datos en este parte analizaremos solo dos debido a que realizar un análisis de todos nos llevaría un capítulo por cada tipo. Revisaremos dos tipos, las bases de datos jerarquizadas y relacionales.

Las bases de datos jerarquizadas se basan en el modelo del árbol jerárquico es decir que no solo identifica los datos de la base sino que además define relaciones entre estos. Para estas bases de datos existen tres formas diferentes El modelo uno-a-uno, uno-a-varios y varios-a-varios (este modelo también conocido como de red). Para entender mejor esto revisemos la figura siguiente.



Al revisar este árbol podemos encontrar los modelos que acabamos de mencionar. Si revisamos la parte del medicamento veremos que para el medicamento Amoxisol pertenece únicamente la categoría penicilina (modelo uno-a-uno), pero si revisamos al revés veremos que para la categoría penicilina pertenecen 3 medicamentos (modelo uno-a-varios). En este modelos no se puede representar el modelo de red ya que necesitaríamos que algún medicamento perteneciera a las dos categorías y entonces podríamos relacionar a dicho medicamento con la otra categoría.

Al trabajar con un modelo jerárquico se tienen muchos problemas y las relaciones son muy complejas por lo que el manejo con la base de datos resulta también complejo. De los tres tipos de modelos presentados el que presenta mayores ventajas es el modelo de red ya que en este los datos son únicos y no se tiene redundancia de la información, pero no existe una cabeza que permanezca como única, es decir a pesar de que el modelo es jerárquico no se pueden señalar muy claras las jerarquías por lo que las relaciones son todavía mas complicadas .

El tipo de bases de datos mas utilizado y eficiente es el de bases de datos relacionales. En este tipo de bases de datos los no programadores son capaces de poder manipular la información que la base de datos contiene. Este tipo de base de datos son las mas populares debido a la sencillez de la estructura que manejan.

El modelo básico de este tipo es una tabla, es decir se trabaja con columnas y filas. Cada fila contiene información que pertenece a una entrada, es decir cada fila representa un registro. Por su parte las columnas serán las que indicaran el tipo de entrada que se hará, es decir representan a los campos.

En este modelo los valores que la tabla tiene son sencillos y como se encuentra acomodada por campos (columnas) esto hace que los datos de las columnas sean de la misma clase. El modelo relacional permite el almacenamiento secuencial o por índice.

Este modelo como su nombre lo dice permite las relaciones entre dos o mas bases de datos. Las relaciones son una herramienta para evitar que las bases de datos sean muy grandes, por ejemplo en una escuela que se encarga de dar cursos es necesario tener dos bases de datos una que controle los cursos y otra que se encargue de llevar el control de los alumnos.

Supongamos que las bases de datos cuentan con las siguientes campos:

Base de cursos	Base de alumnos
Clave de curso	Clave del curso
Nombre del curso	Nombre del alumno
Duración	Dirección
Profesor	Fecha de incripción
Costo	Forma de pago

Nótese que las dos bases de datos tienen un campo en común que es la clave del curso, esto se hace porque por medio de este campo crearemos una relación entre las dos bases Si un alumno llegara a inscribirse al curso se utilizaría la base de alumnos solicitando los datos que se presentaron en la tabla, al imprimir la ficha de inscripción se hace la relación con la otra base de datos es decir ya que se registro al alumno se verifica en la otra base los datos del curso para que en la ficha salgan impresos todos los datos de las dos bases.

Las aplicaciones que se pueden realizar con las bases de datos son muchas por ello se ha formado el concepto de álgebra relacional que contiene varias tareas para las relaciones de las base de datos, tareas como selección, producto, división, etc.

I.4 Sistemas de bases de datos

Se menciono en el punto uno de este capitulo el concepto de DBMS, se dijo en esa parte que los sistemas administradores de bases de datos se encargan de manipular la información para lograr algunas aplicaciones con estos datos. Los sistemas de base de datos son herramientas de programación que se realizan de acuerdo a las necesidades que los usuarios tengan.

Un sistema de base de datos provee:

- Representación y almacenamiento de datos para que puedan ser accesados en otras actividades.
- Organización de estos datos para que puedan ser accesados eficientemente.

- Una eficiente interfaz entre el usuario y el sistema para que la toma de desiciones sea la correcta.
- Protección de los datos es decir la seguridad de la información

Al crear o trabajar con un sistema de base de datos se tiene que contemplar ciertas condiciones que estos deben cumplir para un mejor manejo de la información que contengan:

Independencia de los datos: es decir que al estar trabajando con un sistema administrador de bases de datos este no debe alterar los datos que se tienen en la base o cuando cambiamos los datos de la base este cambio no debe de alterar al sistema administrador.

Compartición: las bases de datos deben de tener la posibilidad de compartir su información con varios programas de aplicación, esto permitirá que una sola base de datos sea utilizada para varias aplicaciones y evitemos la redundancia de los datos.

Seguridad: un sistema debe de contener ciertas medidas de seguridad como lo puede ser asignar ciertos derechos de acceso a algunas aplicaciones de nuestro sistema, por ejemplo para poder eliminar algún dato de la base se podría colocar un password en este modulo.

Flexibilidad: es muy importante ya que la flexibilidad de acceso a los datos hace que nuestro sistema trabaje de manera mas fácil y eficiente. Es correcto indicar que se debe de tener ciertas restricciones para algunos datos de la base pero la flexibilidad se refiere a que se pueda accesar a los datos de forma directa y rápida.

I.4.1 Análisis del sistema

Los sistemas de bases de datos se realizan de acuerdo a las necesidades que el usuario tenga. Para lograr que el sistema cumpla con lo anterior es necesario que antes de realizar cualquier

actividad de programación se realice un análisis de lo que el usuario quiere para el sistema. Este análisis debe englobar todas las actividades que el usuario quiere en el sistema.

El análisis de sistema tiene los siguientes objetivos:

a) Identificar las necesidades del usuario
b) Revisar la viabilidad del sistema
c) Realizar un análisis económico.
d) Analizar las funciones del software, del hardware y del usuario final.
e) Establecer restricciones de tiempo.
f) Forma una estructura de la base de datos adecuada al sistema.

Los objetivos que se muestran están muy ligados unos con otros ya que del buen análisis de estos lograremos formar un sistema de acuerdo a la que se necesite y que además trabaje de la mejor manera posible. Al identificar las necesidades del usuario se tiene que contemplar que estas necesidades puedan ser complementadas con los otros puntos, es decir que tenemos que ver que el usuario tenga los suficientes recursos técnicos y económicos para que el sistema sea viable.

De acuerdo también a las necesidades que tenga el usuario podremos definir un tiempo en el que se llevará a cabo la realización del sistema, desde el inicio del análisis hasta la puesta en marcha de sistema final.

También tenemos que poner en claro los alcances que nuestro sistema va a tener es decir, definir que funciones van a tener el software y el hardware y que funciones va a tener que desempeñar el usuario de forma manual.

Al realizar este análisis es necesario también que se defina de manera clara la estructura de base de datos que se utilizará en la realización del sistema. Como ya se menciono la estructura se va a crear de acuerdo a las necesidades que el usuario tenga.

Para lograr los objetivos del análisis del sistema se puede hacer de manera sencilla, haciéndose las siguientes preguntas:

¿ Que es lo que se quiere hacer?
¿Para que se quiere hacer?
¿Se puede hacer?
¿Contamos con los recursos técnicos y económicos para hacerlo?
¿Como se puede hacer?
¿Cuanto tiempo tardaríamos en hacerlo?

Al responder estas preguntas de manera clara y bien definida podremos decir que hemos logrado un buen avance en la realización del análisis del sistema.

Para ejemplificar lo anterior revisemos de manera sencilla el análisis para el sistema que en este trabajo se va a presentar.

Se requiere hacer un recetario medico que contenga en la base de datos gran cantidad de medicamentos y que al utilizarlo un doctor pueda crear una receta, es decir que el usuario final será un doctor que tendrá la posibilidad de consultar los medicamentos con la finalidad de recetarlos.

El doctor quiere primero teclear los datos del paciente para que la receta también contenga estos datos. Una vez que el doctor ha tecleado los datos del paciente comenzara a recetar medicamentos, es decir que el doctor realizara una consulta tecleando el nombre del medicamento que quiere, en pantalla se quiere que se presente toda la información del medicamento es decir las indicaciones, las dosis, etc. si al doctor le interesa recetar este medicamento podrá hacerlo. Además se debe de tener la posibilidad de imprimir las recetas o imprimir fichas técnicas acerca de los medicamentos.

También se quiere tener la posibilidad de actualizar los medicamentos, es decir agregar, borrar o modificar los medicamentos.

El sistema se quiere hacer con la finalidad de que se automaticen los procesos en los consultorios médicos ya que ahora el doctor únicamente tecleara el nombre del medicamento y en la impresión aparecerán todos los datos que se requieren en la receta.

El sistema se va a implementar en varios consultorios pero en este caso el doctor sugirió un software en el que se tenia que hacer el sistema y como los equipos de computo con los que se cuentan soportan trabajar con este software se adecuara el sistema en el mismo.

Se ha destinado un tiempo aproximado de dos meses para la realización de este sistema sobre todo para la captura de los datos de los medicamentos.

El análisis anterior es únicamente un pequeño ejemplo para poder visualizar como se presenta un análisis del sistema.

I.4.2 Realización del sistema.

Durante la realización del sistema es necesario que se contemplen los siguientes puntos.

- Establecer un tiempo para la elaboración de cada uno de los módulos del sistema
- Nunca hacer lo contrario a lo que el análisis del sistema estableció.
- Realizar pruebas a los módulos con datos verídicos.
- Determinar la correcta presentación es decir el manejo de menús y pantallas.
- Realizar la programación de manera sencilla y clara.
- Guardar copias de seguridad de todos los archivos ya sean de lenguaje o de bases de datos.
- Establecer la forma de instalación del sistema en los equipos.

Al revisar los puntos anteriores es necesario detenernos un poco en la presentación de las pantallas del sistema, algunas veces durante el análisis se establece también la presentación del sistema, pero otras veces queda a criterio del programador como será la presentación del mismo. En este caso el programador debe de acomodar las pantallas de acuerdo a las necesidades que el usuario tiene aunque tomando su propio criterio.

En la realización del sistema se debe de llevar un orden acerca del sistema que estamos haciendo lo correcto es primero llenar la base de datos con toda la información que tenemos, definir las pantallas y menús que se utilizaran para la presentación, programar los módulos de acuerdo a lo que el sistema requiere.

Al realizar el sistema se debe de contemplar los posibles errores que los usuarios finales pudieran cometer para evitar que se presenten y en el caso de que se presenten colocar mensajes preventivos y de error para que durante la ejecución del sistema no se tengan problemas de errores internos.

Lo anterior únicamente son recomendaciones que se puede seguir para la realización del sistema pero dicha realización queda a criterio del programador.

I.4.3 Pruebas y puesta en marcha.

Cuando un sistema se ha terminado por el programador este tiene la responsabilidad de hacer pruebas al sistema antes de ponerlo en marcha. El programador al entregar un sistema tiene la obligación de probarlo al usuario final para que se compruebe que el sistema no presenta errores y que esta funcionando de acuerdo a lo que se estableció dentro del análisis del sistema.

Durante las pruebas que se hacen al sistema se deben utilizar datos reales, es recomendable que se deje a un usuario que desconozca por completo el manejo del sistema para ver como se comporta el mismo ante situaciones difíciles.

También se deben de hacer pruebas para ver cuanta información logra manejar el sistema y así poder determinar sus alcances. Otra prueba que se tiene que hacer es de tiempo para poder verificar que tan rápido es nuestro sistema, esta prueba se realiza tomando el tiempo de las actividades que se realizan es decir cuanto tiempo tarda en imprimir, etc.

Se recomienda que las pruebas se realicen en un equipo que sea similar al equipo en el que será instalado el sistema para que las pruebas sean reales y no se tenga problemas durante la puesta en marcha.

La pruebas se deben de hacer de manera clara y definida para poder encontrar algún error que se haya pasado durante la realización del sistema.

La puesta en marcha como su nombre lo dice es poner a trabajar el sistema ya en el área real. En caso de encontrar algún error en la puesta en marcha es obligación del programador corregirlo aunque esto no se debe presentar ya que todos los errores deben salir durante las pruebas.

Para la puesta en marcha del sistema es necesario que el programador entregue al usuario final un manual de usuario que explicara el manejo dicho sistema desde la instalación hasta los posibles errores que se pudieran cometer durante el manejo del mismo.

Se espera que con lo presentado en este capitulo se entiendan los conceptos mas básicos acerca de las bases de datos y los sistemas que se encargan de manejar estas, algunos conceptos no se abarcaron tanto ya que de este tema se puede hacer un libro completo y nos faltarían conceptos. La finalidad de este capitulo es dar una introducción a los sistemas de bases de datos ya que en los capítulos siguientes revisaremos lenguajes para realizar estos sistemas.

II. Clipper en las bases de datos

II.1 Conceptos Generales.

En programación de sistemas es muy indispensable que se tengan lenguajes que trabajen bajo el ambiente DOS y no únicamente con Windows en cualquiera de sus versiones.

Se podría pensar que al trabajar con Clipper en la actualidad resulta ser obsoleto ya que Clipper es un lenguaje con aplicaciones para DOS y no para Windows. Pero la realidad es que en muchos lugares basan los sistemas en DOS ya que requiere de equipos no tan actuales, es decir un programa en Clipper se puede ejecutar hasta en un equipo con procesador 386.

La idea de tener los sistemas en DOS es también debido a que algunos de estos sistemas requieren de equipos dedicados al sistema, es decir que no se ocupen en otras actividades si no que únicamente trabajen con el sistema en cuestión. Cuando esto sucede las empresas no pueden tener un equipo muy actual ya que se consideraría como un desperdicio del equipo.

Ahora también pensemos, aunque sea en ambiente DOS requerimos que los sistemas tengan un gran alcance y que trabajen de una manera óptima. Clipper es el lenguaje de programación que ofrece mas ventajas para el ambiente en DOS. Clipper nace mejorando a Dbase, de hecho Clipper nace únicamente como compilador de Dbase y por ello es que tiene gran relevancia ya que Dbase dominaba el mercado en cuestiones de sistemas para DOS.

La primera versión de Clipper fue denominada Summer 87, esta versión fue básicamente una copia de Dbase aunque ya generaba archivos ejecutables. Esa fue la principal razón por la que nace Clipper, Dbase era solamente un interprete es decir no genera archivos exe, únicamente interpreta el lenguaje y realizaba las tareas pero dentro del mismo Dbase. Por otro lado Clipper nace como un compilador, es decir que genera un archivo exe

interpretando y traduciendo el lenguaje para que pueda ser ejecutado desde el DOS y no sea necesario estar en Clipper para hacerlo.

También existen otros lenguajes en el mercado para el manejo de sistemas en DOS como lo pueden ser C, Pascal, Foxpro, etc. pero Clipper es un lenguaje especializado en el manejo de datos, aunque C y Pascal son lenguajes con gran potencia, en el manejo de datos tienen varias deficiencias ya que no cuentan con herramientas para trabajar los datos y uno tiene que crearlas y eso requiere de mucho tiempo para el programador. Por otro lado Foxpro es un lenguaje también especializado en las bases de datos pero Clipper ofrece mas ventajas de programación que Foxpro.

Clipper es un lenguaje que cuenta con mas de 350 ordenes y funciones con posibilidades amplias y diversas. La mayoría de estas instrucciones están obviamente dedicadas al manejos de datos. Además también cuenta con varias herramientas que ya harían de Clipper un manejador de bases de datos sin necesidad de programar nada, un ejemplo es esto seria el DBU.

En este trabajo se va a programar con Clipper 5.2, esta versión se elige ya que es una de las mas actuales, no las mas actual ya que la versión mas actual de Clipper es la versión 5.3 pero esta versión tiene ya aplicaciones para Windows y en este trabajo se quiere mostrar un lenguaje en DOS 100 %. La versión 5.2 de Clipper es una versión muy completa que ofrece mas de 400 comandos y funciones para trabajar.

Utilizar Clipper es sencillo ya que únicamente es necesario conocer las ordenes mas comunes del sistema operativo tales como copiar, borrar, etc., La programación tampoco es muy compleja ya que todas las funciones y comandos son sencillas de manejar con una sintaxis no tan compleja como otros lenguajes de programación.

"Los prerequisitos mas valiosos a la hora de aprender a programar en Clipper son, probablemente, la aptitud y el deseo, no el conocimiento del DOS"

II.1.1 Características Técnicas.

Como se menciono en el punto anterior ya que el manejo de Clipper es en el entorno de DOS, es necesario tener conocimientos básicos en sistema operativo,

Con lo anterior es importante mencionar que cuando se quiera programar en Clipper únicamente necesitamos instalado el sistema operativo y Clipper. Para poder programar en Clipper se necesita de un editor de textos, aunque Clipper ofrece uno este no es muy potente y confiable como se pudiera pensar por lo cual es mas recomendable tener un editor de textos aparte al que ofrece Clipper por ejemplo el Edit del sistema operativo.

Cuando se instala Clipper es necesario modificar el archivo autoexec.bat con las siguientes líneas.

```
PATH C:\DOS;\C.\CLIPPER5\BIN
SET LIB=C:\CLIPPER5\LIB
FILES=55
BUFFERS=8
```

Estas líneas por lo regular son generadas al instalar Clipper pero es recomendable verificar que realmente se encuentren en el autoexec.bat. Al contener estas lineas permitiremos que el lenguajes trabaje sin problemas en cuestión con el sistema operativo permitiendo que el manejo de Clipper sea sin problemas.

II.1.1.1 Requerimientos en hardware.

El hardware es el equipamiento fisico que conforma el sistema de la computadora. Una buena combinación en el hardware puede reducir considerablemente el tiempo de compilación y enlazado de programas.

Ya se menciono lo que se necesita en software para poder trabajar con Clipper adecuadamente, de esto podemos deducir que el hardware que se necesita para poder trabajar con Clipper no es muy complejo y tampoco muy costoso.

El equipo mas simple para poder trabajar con Clipper deberá cubrir los siguientes requerimientos:

Procesador 386 o superior.
4Mb en memoria Ram
Monitor Color Vga
Disco duro de 80 Mb o mayor.
Unidad de disquete 3 ½
Tarjeta de vídeo Vga
Mouse y Teclado

En la actualidad ya no se consiguen estos equipos nuevos pero para implementar los sistemas en Clipper se puede utilizar un equipo que ya se tenga en la empresa que cumpla con las características que se acaban de presentar.

El equipo que se presento es el que se requiere para trabajar con Clipper de manera óptima pero si se pudiera tener un equipo mas potente ofrecería mas rapidez sobre todo cuando el sistema ocupa gran cantidad de datos. La velocidad es importante no solo para trabajar ya con el sistema en si sino también para poder programar ya que la compilaciones y enlaces son un proceso que requiere de que exista una buena velocidad en nuestra computadora y una buena capacidad en memoria.

La capacidad de almacenamiento que se da en disco duro es porque Clipper maneja muchos archivos no solo los de bases de datos sino los que contiene el código, los índices, los

ejecutables y sobre todo también hay que asignar un buen espacio para las copias de seguridad de nuestro sistema.

Para los sistemas de Clipper, el monitor se recomienda de color para poder dar una buena presentación en cuestión de colores de pantalla, pero si se tiene un monitor monocromático también se puede trabajar ya que las aplicaciones de Clipper pueden o no llevar color.

II.2 Principales Herramientas.

Clipper como ya se mencionó es un lenguaje que trabaja bajo el entorno DOS, de ahí que todas sus herramientas se basen en sistema operativo. La principal herramienta de Clipper es la de poder generar archivos ejecutables, pero no es la única ya que este lenguaje contiene una gran cantidad de herramientas.

Clipper contiene herramientas para el manejo de los sistemas de bases de datos sin necesidad de programar nada. Por ejemplo contiene un editor de textos que sirve para poder redactar ahí los programas fuentes que contendrán el código del sistema, además en este editor se pueden hacer otras tareas. Contiene también un manejador de bases de datos conocido como DBU. Este permite realizar tareas en las bases de datos como búsquedas, ordenación, añadir, borrar o modificar registros, etc., el DBU puede ser trabajado no solo por el programador sino también por el usuario final aunque es un poco complejo no es tan dificultoso aprender a usarlo.

Además al instalar Clipper, se genera un subdirectorio de ayuda que permite consultar todos los comandos y funciones. La ventaja que ofrece esta ayuda es que cada uno de los comandos contiene un programa de ejemplo.

II.2.1 Manejo de Archivos

Clipper es un lenguaje que genera varios tipos de archivos, desde los archivos de base de datos hasta los archivos ejecutables. Básicamente Clipper maneja los siguientes tipos de archivos.

Tablas (DBF)
Programa Fuente (PRG)
Indices (NDX)
Etiquetas
Informes
Copias de seguridad (BAT)
Programas ejecutables (EXE)

Existen otros archivos pero estos se generan según las aplicaciones de programación que se den.

Los archivos que se generan en Clipper ya sean de código o ejecutables no ocupan mucho espacio en el disco, el único archivo que puede ser muy grande es el de base de datos y eso depende de la información que se almacene.

Los índices son otro tipo de archivo muy importante en Clipper ya que de los índices puede basarse el adecuado manejo de la información. Los archivos de índice no ocupan mucho espacio en disco pero son los que guardan la base de datos ordenada de forma lógica, es por ello que su tamaño es menor que si ordenáramos la base de datos de forma física.

II.2.1.1 Tablas

Las tablas son la base de todo sistema, las tablas se refieren a la base de datos que se esta trabajando. En Clipper la mayoría de los comandos se refieren al manejo de tablas. Las

tablas en Clipper como ya se menciono se puede trabajar desde el DBU. Pero para poder trabajar con las tablas es mejor generar programas que trabajen directamente con ellas.

Con lo anterior podemos definir una cosa, en Clipper las tablas se pueden manipular de forma manual y por medio de los programas de aplicación. El hecho de manejarlas de forma manual implica que el usuario conozca el manejo de Clipper y con los programas de aplicación no, ya que las aplicaciones de estos son directamente enfocadas a las necesidades del usuario.

Clipper es un lenguaje que como ya se menciono cuenta con muchos comandos que se basan en el manejo de las tablas, este tiene la capacidad de manejar los tipos de datos que se mencionaron en el capitulo anterior, es decir los campos numéricos, alfanuméricos, lógicos y campos memo.

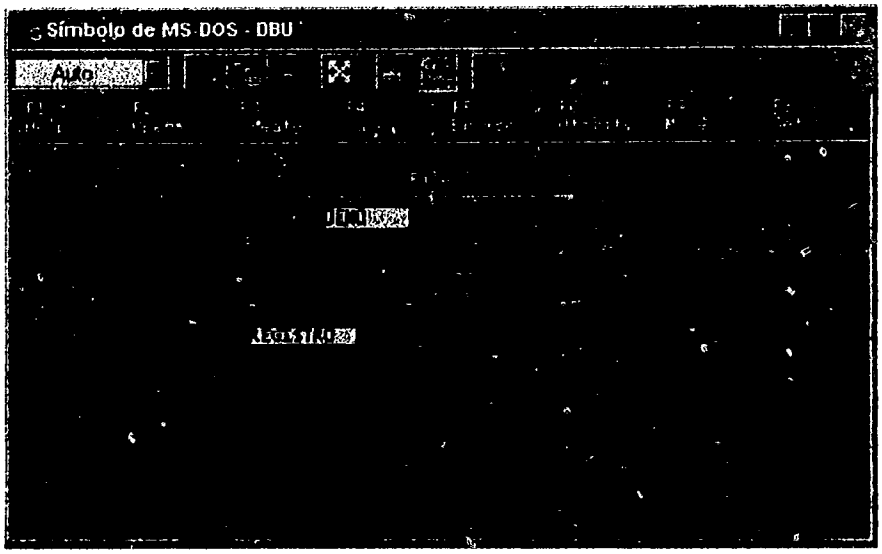
Los campos memo son muy importantes para las tablas en Clipper ya que permiten guardar gran cantidad de información en un solo campo, para los campos memo, este lenguaje cuenta con una gran variedad de comandos para poder trabajar con ellos, en el ejemplo que se realizara al final se verán ejemplos de estos.

Además también es importante mencionar que Clipper es un lenguaje que permite los manejos de la información de manera simple y sencilla, es decir que para agregar, modificar, consultar y borrar la información se cuenta con comandos definidos muy claramente para poder realizar estas tareas y evitar mucho lenguaje de programación.

Las tablas basan su funcionamiento en la estructura que se genera. En Clipper existe el comando Create que permite crear una tabla desde el programa de aplicación. Lo anterior se menciona ya que Clipper es el único lenguaje que permite crear la base de datos directamente desde el programa de aplicación, es decir que el comando create contiene una sintaxis de manera que se pudiera crear la base de datos si esta o existiera, esa es una de sus grandes ventajas.

También se puede crear la tabla desde el DBU que es como lo hicimos para el sistema de ejemplo de este trabajo

En la figura que se muestra a continuación se vera como es que se ha creado la tabla que utilizaremos en nuestro sistema desde el DBU

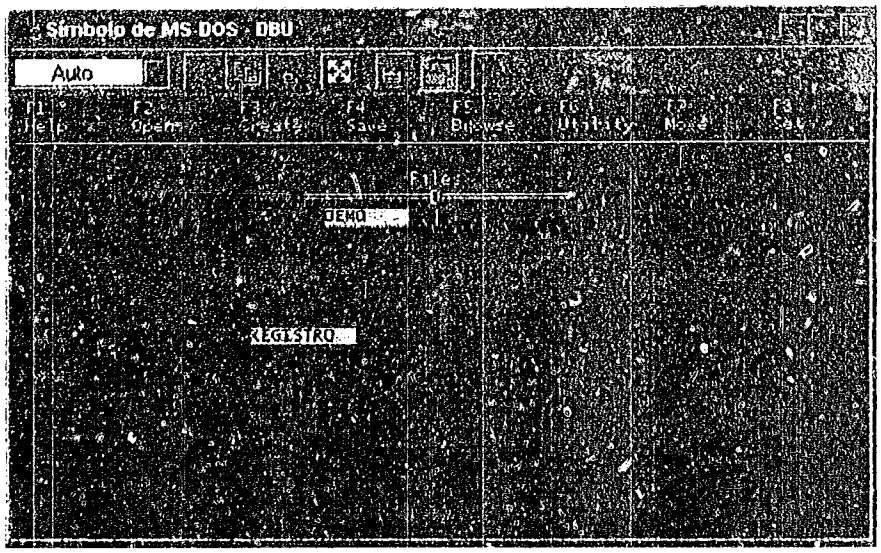


El crear un tabla y almacenar datos en ella el almacenamiento que hace Clipper es de tipo secuencial, le asigna a los registro un numero de registro que será la llave que se ocupara cuando no se tenga creado un índice para la base. En Clipper no es recomendable trabajar con la tabla sin haber creado un índice antes ya que las búsquedas de información en la tabla será mas tardadas y el sistema será algo lento. Los índices se pueden crear también desde el DBU o desde un programa, para ambos casos se debe de indicar el campo y el nombre que tendrá el índice, vea como es la sintaxis para un ejemplo de programación:

Index on <campo> to <nombre_indice>

Tambien se puede crear la tabla desde el DBU que es como lo hicimos para el sistema de ejemplo de este trabajo

En la figura que se muestra a continuación se vera como es que se ha creado la tabla que utilizaremos en nuestro sistema desde el DBU



El crear un tabla y almacenar datos en ella el almacenamiento que hace Clipper es de tipo secuencial, le asigna a los registro un numero de registro que será la llave que se ocupara cuando no se tenga creado un indice para la base En Clipper no es recomendable trabajar con la tabla sin haber creado un indice antes ya que las búsquedas de informacion en la tabla será mas tardadas y el sistema será algo lento Los indices se pueden crear también desde el DBU o desde un programa, para ambos casos se debe de indicar el campo y el nombre que tendrá el indice, vea como es la sintaxis para un ejemplo de programación

Index on <campo> to <nombre_indice>

Cuando se crea un índice el almacenamiento será indexado por lo que al agregarse un registro este se acomodara de acuerdo al índice que ya se estableció. Es importante mencionar que para manejar las tablas Clipper permite crear varios índices, lo que permite que el manejo de información sea de manera mas directa. Para poder abrir un índice dentro de un programa se debe de indicar el nombre del índice que se ha de utilizar, se puede hacer al abrir la base de datos o se puede abrir un índice en otra instrucción. El comando Use se utiliza para abrir una tabla.

Use <nom_base> Index <nom_indice>

O bien

Set Index to <nom_indice>

Clipper es un lenguaje que permite el manejo de bases relacionales por ello es que el indexado es muy importante para poder trabajar la información ya que el indexado en este lenguaje es la forma de trabajar con dichas bases . Es decir que el índice no solo sirve para hacer mas rápidas las tareas de búsqueda sino que además en Clipper es la forma de trabajar los datos relacionados.

Supongamos que se tiene dos tablas que se quieren relacionar por medio de un campo, para que la relación sea mas directa, es necesario que en ambas bases de datos ya se haya creado un índice basado en este campo común.

Por otro lado la forma de visualizar el contenido de las tablas se hace por medio de el DBU o por medio de un programa de aplicación. En el DBU la información se muestra en forma de lista (browse) permitiendo modificar la información que la base de datos contenga.

En el programa de aplicación se utiliza el comando @ ... Say para poder mostrar la información y @.. get para poder modificar la información de la base, en el ejemplo que se muestra al final de este capitulo se puede ver la forma en que se utilizan estos comandos para realizar las tareas que el sistema tiene que manejar.

Clipper también permite compartir las tablas dentro de una red. Si se agrega la palabra Exclusive cuando se intenta abrir una base de datos se indicara que esta bases de datos únicamente se podra abrir por medio de ese sistema y no se podra accesar a ella mediante la red.

Las funciones mas útiles que Clipper da para las tablas de bases de datos son:

1) Mantenimiento, puesta a punto y actualización.
2) Clasificación de los datos por un orden definido.
3) Localización de los datos, corrección y borrado de los mismos.
4) Posibilidad de relacionar varios archivos entre si.

II.2.1.2 Etiquetas

Las etiquetas son una forma de presentar el contenido de la base de datos en pequeñas fichas que pueden servir al usuario para ver el contenido de la base de datos. Estas se pueden presentar en papel o en la pantalla.

En Clipper el hacer etiquetas es de forma muy sencilla, existen dos formas para hacerlo programando o con una herramienta conocida como RL.

Cuando se van a hacer etiquetas dentro de programación se comienza mandando el puntero al inicio de la base de datos. Después se va haciendo el formato que se quiera dar a la etiqueta indicando la posición de cada uno de los campos.

Esto se debe de hacer en un ciclo según las necesidades del usuario, es decir se puede hacer un ciclo que imprima ya sea en pantalla o en papel todos los registro de la base de datos o únicamente algunos que cumplan con una condición dada.

En el ejemplo siguiente se puede ver esto en programación de Clipper.

```
CLEAR
USE DEMO.DBF
R=SPACE(1)
MEDICA=SPACE(50)

@4,4 SAY "¨IMPRIMIR TODAS LAS FICHAS O POR NOMBRE?F/N" GET R;
PICTURE "!" VALID (R$"FN")
IF R="F"
GO TOP
DO WHILE .NOT. EOF()
      SET DEVICE TO ?RINT
      CLEAR
      @2,2 SAY TRIM(REGISTRO)+TRIM(MEDICAMENT)
      @3,2 SAY TRIM(GENERICO)
      @4,2 SAY TRIM(GRUPO)
      @5,2 SAY TRIM(CATEGORIA)
      @6,2 SAY TRIM(LABORATORI)
      @7,2 SAY MEMOLINE(PRESENTACION,60,1)
      SKIP
      SET DEVICE TO SCREEN
      @9,2 SAY "INGRESE LA SIGUIENTE FICHA Y PULSE CUALQUIER TECLA"
      WAIT ""
      EJECT
      LOOP
   ENDDO

ELSE
   @6,2 SAY "NOMBRE DEL MEDICAMENTO" GET MEDICA
   READ
   GO TOP
   LOCATE FOR MEDICAMENT=TRIM(MEDICA)
   DO WHILE .NOT. EOF()
      SET DEVICE TO PRINT
      CLEAR
      @2,2 SAY TRIM(REGISTRO)+TRIM(MEDICAMENT)
      @3,2 SAY TRIM(GENERICO)
      @4,2 SAY TRIM(GRUPO)
```

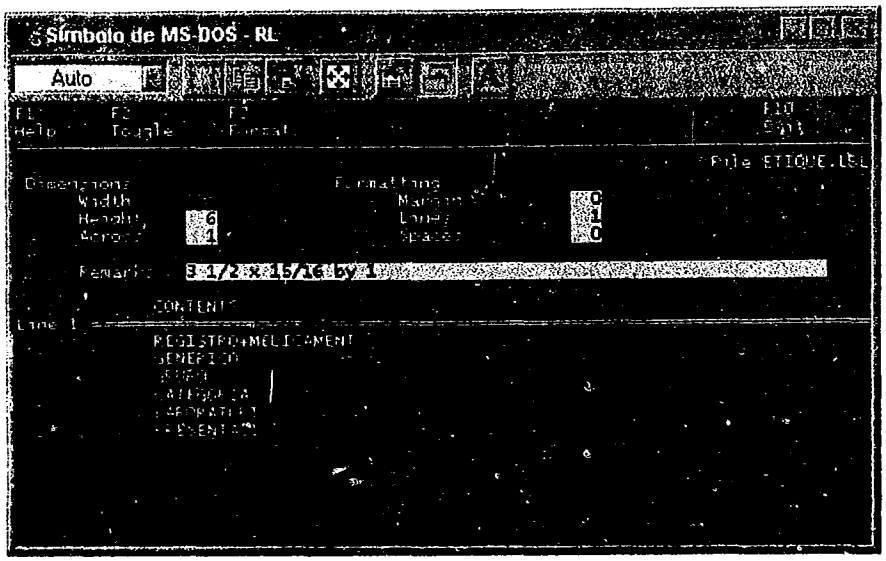
```
      @5,2 SAY TRIM(CATEGORIA)
      @6,2 SAY TRIM(LABORATORI)
      @7,2 SAY MEMOLINE(PRESENTACION,60,1)
      SKIP
      SET DEVICE TO SCREEN
      @9,2 SAY "INGRESE LA SIGUIENTE FICHA Y PULSE CUALQUIER TECLA"
      WAIT ""
      CONTINUE
      EJECT
      LOOP

   ENDDO

ENDIF
```

En este programa se puede ver como es que se imprimen etiquetas ya sea seleccionando que se impriman todas o únicamente por medio del nombre del medicamento.

Por otro lado existe un editor para crear un formato establecido de etiquetas. Este editor permite al usuario definir el tamaño que tendrá la etiqueta y los campos que se añadirán a esta.

El editor es un poco obsoleto ya que no permite agregar algunos detalles de presentación como lo pudieran ser líneas o marcos que pudieran dar mejor presentación a la etiqueta únicamente muestra información de la base de datos.

En la figura siguiente se muestra cual es el formato de etiqueta que estamos utilizando para nuestro ejemplo, recuerde que el sistema que estamos haciendo es para un recetario medico por lo que esta ficha contiene unicamente la informacion que un doctor considera necesaria en un medicamento:



En este formato se establece el tamaño que tendrá la etiqueta y también los campos que se mostraran en las etiquetas. El tamaño de las etiquetas esta definido por Clipper pero el programador puede definir el ancho y los renglones que esta contendrá. Como se puede ver en esta etiqueta en numero de renglones será 6 y el ancho 60.

Además de crear el formato obviamente este se debe de llamar desde nuestro programa. Existe un comando en Clipper que llama al formato de etiquetas para que se puedan imprimir. Este comando es Label Form <nombre>. En el ejemplo final se ve como es que se llama al formato de etiquetas desde nuestro sistema. Este comando además permite también crear ciclos que impriman únicamente las etiquetas que cumplan con ciertas condiciones dadas.

Una de las principales desventajas que se tienen al trabajar con el editor de etiquetas es que este no permite visualizar los campos de tipo Memo, esta desventaja se vera según el tipo de sistema que estemos manejando pero como en el sistema de ejemplo de este libro se utilizan varios campos este tipo, no es muy conveniente utilizarlo. Pero por cuestiones de ejemplo si se utilizo.

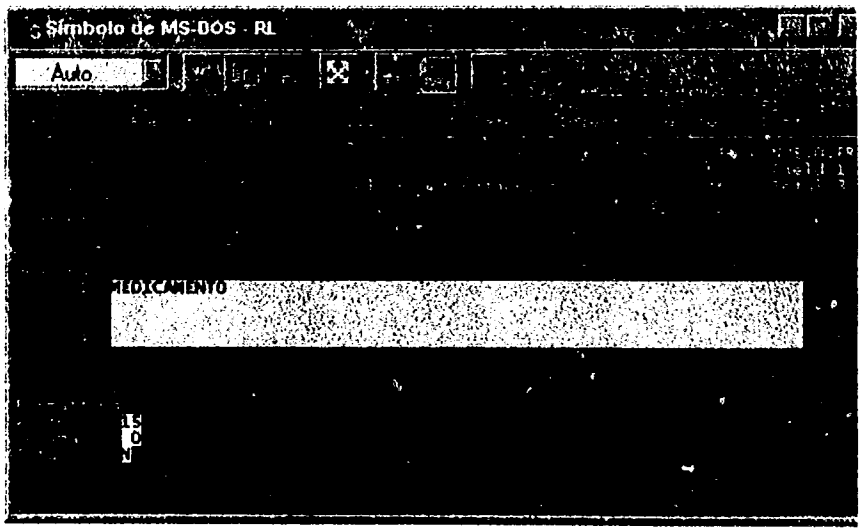
II.2.1.3 Informes

Los informes al igual que las etiquetas presentan la información ya sea en papel o en pantalla. El informe es en realidad un compendio de los datos de una base de datos, que adopta en cada caso un formato especial y que consiste en unos títulos y subtítulos que resumen los resultados habidos en un período de tiempo determinado.

La información de la base de datos se puede presentar visualizando los campos, combinando el contenido de los mismos, es decir realizar operaciones con ellos o totalizando los resultados.

Como se puede ver el lo anterior el manejo de informes es parecido al de las etiquetas, es decir tendremos dos formas de poder manejar los informes, por programación o por editor.

El editor nos muestra la siguiente pantalla.



Cuando se trabaja por medio del editor se debe de indicar en este varios datos:

- El titulo que llevara el informe. Por medio de la tecla F2 nos llevara a una pantalla en la cual se debe teclear el titulo que llevara el informe además de el numero de renglones

que tendrán las hojas del informe, es decir en esta parte se configurara la presentación del informe

- Los campos que se imprimirán en el informe. Por medio de la tecla F4 nos llevara la pantalla en la cual podemos configurar los campos indicando el nombre del campo y el nombre de la columna que llevara dicho campo. Además también se configurar el ancho que se quiere en el campo.
- Los grupos que se quieren formar. Por medio de la tecla F3 nos muestra la pantalla en la cual se puede indicar los grupos por medio de los cuales se imprimirá el informe. Para los grupos se declara un campo en particular. Supongamos que se declara el campo de genérico. Al hacer esto la impresión del informe se realizara agrupando los registro que cumplan con un mismo campo en este caso el nombre genérico del medicamento. Es importante mencionar que para que se vea claramente la impresión en base a grupos es necesario que la base de datos se encuentre indexada por medio del campo que seleccionaremos como criterio de grupo. Veamos en el siguiente ejemplo para entender esto. Este ejemplo es una parte de un informe de la base de datos de medicamentos pero por medio del campo Grupo.

```
Page No.      1
        11/25/98
           RELACION DE MEDICAMENTOS POR GRUPO TERAPEUTICO

        MEDICAMENTO      GENERICO          CATEGORIA

** GRUPO Cancerología
        Fareston         Toremifeno        891
        Leukeran         Clorambucilo      Alquilantes
        Myleran          Busulfano         Alquilantes
        Purinethol *     Mercaptupurina    Antimetabolitos

** GRUPO Cardiovascular
        Acylanid         Acetildigitoxin   Glucósidos
                         a                 cardiácos
        Adalat           Nifedipina        Antianginosos
        Decme            Metanosulfonato   Vasodilatadores
```

	de dihidroergocristina	periféricos y/o activadorescerebrales
Decme	Metanosulfonato de dihidroergocristina	Vasodilatadores periféricos y/o activadorescerebrales

- Existen otras herramientas dentro de los informes que sirven para borrar (F5), añadir (F6) o ir a un registro en particular (F7), además para guardar y salir se tecleara F10.

Para hacer un informe dentro de la programación se hace de manera parecida a las etiquetas, es decir dentro del programa se van poniendo las coordenadas dentro de las cuales se imprimirán los campos que se requieren. La ventaja de la programación es que el informe puede relacionar varias bases de datos, es decir que dentro de la impresión se pueden combinar varios campos de varias bases de datos y en el editor solo se hará la impresión con una sola base de datos. Además de que por medio de la programación se puede dar un mejor formato a la presentación de la información, aunque por medio de los informes se pueden sacar los datos de la base de una manera organizada.

En el ejemplo que se muestra al final de este capitulo se podrá ver como es que se utiliza el programa para generar un informe que en este caso será la impresión de la receta pero trabajando con tres bases de datos.

II.2.2. Manejo de Presentaciones

La presentaciones una de las principales partes en las cuales se debe de detener un programador al realizar un sistema, ya que de ello depende que tan fácil de manejar sea el sistema.

Para dar una buena presentación se deben de utilizar todas las herramientas que el lenguaje nos proporciona. Clipper no maneja gráficos pero se pueden utilizar algunos comandos que

manejando una sintaxis en particular permitiran dar una buena presentación al sistema. En la presentación también es importante el manejo de menús, Clipper es un lenguaje que contiene varios comandos que permiten trabajar los menús de una manera óptima. Esto lo veremos mas explicado en unos puntos mas adelante

Las presentaciones deben de contemplar no solo las herramientas que se puedan ofrecer sino que ademas se debe de tener muy en cuenta la combinación de colores que se manejara para la presentación.

En Clipper se utiliza un solo comando que permite trabajar los colores:

Set color to <standar>,<resaltado>,<borde>,<fondo>,<resaltado actual>

Este comando indica los colores con los cuales se hará la presentación. Debido a que Clipper es un lenguaje bajo el entorno de sistema operativo no contiene muchos colores para poder trabajar Ademas de poder manejar los colores para realizar la presentación, se cuenta con varios atributos que pueden servir para dar una mejor presentación. La forma de trabajar los colores y los atributos en Clipper es por medio de las letras. Los colores y atributos que se manejan son los siguientes.

Color	Numero	Letra
Negro	0	N
Azul	1	B
Verde	2	G
Cian	3	Bg
Rojo	4	R
Magenta	5	Rb
Marron	6	Gr
Blanco	7	W
Amarillo		Gr+

Subrayado		U
Brillo		+
Parpadeo		*

Una vez que conocemos como se manejan los colores es importante tomar en cuenta también como es que acomodara la infirmación dentro de la pantalla. En los lenguajes que veremos mas adelante veremos una herramienta que permite organizar la presentación de la pantalla denominada formularios. Clipper no contiene esta herramienta pero por medio de programación se puede organizar la forma en la que se presentaran los datos que necesitamos se muestran en pantalla. Por ejemplo en nuestro sistema utilizaremos una pantalla dividida en tres secciones. La sección superior es la que presentara el titulo de la pantalla, la parte de en medio sera la parte en la que se realizaran todas las tareas que nuestro sistema tiene encomendadas, y la parte inferior sera la que mostrar los mensajes ya sean de error o algunos otros mensajes necesarios durante la ejecución del sistema. Vea el siguiente programa:

```
CLEAR     /* limpia la pantalla */
@1,6 TO 3,70 DOUBLE   /*Crea el cuadro superior  */
@2,7 CLEAR TO 2,69    /*Limpia el área de cuadro superior */

@5,6 TO 19,70 DOUBLE  /*Crea cuadro de área de trabajo */
@6,7 CLEAR TO 18,69   /*Limpia área de trabajo */

@21,6 TO 23,70 DOUBLE /*Crea cuadro inferior */
@22,7 CLEAR TO 22,69  /*Limpia área de cuadro inferior */

@2,27 SAY "AGREGANDO MEDICAMENTO A RECETA"  /* Titulo de pantalla */
SET COLOR TO W+/N*  /* Colores para mensaje letra blanca con brillo
                       Con fondo negro y parpadenado */
@22,20 SAY "--INGRESE LAS RECOMENDACIONES TERAPEUTICAS!!" /* mensaje */
SET COLOR TO   /* regresar a los colores originales */
```

En el ejemplo se puede ver el comando @... say , como ya se menciono este comando se utiliza para mostrar mensajes en pantalla, como se puede ver se indican unas coordenadas

que serán el renglón y la columna en la cual se mostrara el mensaje. Este comando es muy importante ya que todos los mensajes que se pueden utilizar para dar la presentación adecuada al sistema se basan en este comando..

En cuestión de presentación también es importante que se contemple la forma en la cual se acomode la infirmación para las etiquetas y los informes ya que es otra forma de presentar la infirmación al usuario final.

II.2.2.1. Menús

En Clipper se pueden crear varios tipos de menús dependiendo de las necesidades del usuario pero como ya se ha visto en Clipper las presentaciones se tienen que programar. Para programar los menús en Clipper se cuenta con varios comandos que con la sintaxis correcta pueden formar los distintos tipos de menús.

Básicamente la forma de hacer los menús es la misma, la diferencia radica en cual sera la forma en la que se presentaran estos, es decir se puede presentar un menú horizontal con varias opciones y al seleccionar una de estas se genere un submenu vertical con otras opciones, es decir como los menús tradicionales manejados en Windows. Otra forma de hacer un menú es de forma vertical todo, es decir que se genera un menú con las opciones una debajo de la otra y al seleccionar una de estas mostrara el submenu de forma parecida.

La forma de crear los menús que se acaban de mencionar se hace únicamente cambiando la posición de las coordenadas en la que se mostraran las opciones del menú, es decir que se puede crear un menú horizontal y si se desea cambiar a forma vertical únicamente se cambiaran las coordenadas de las opciones.

En el ejemplo siguiente se muestra una parte del menú que utilizaremos en nuestro sistema:

```
SET WRAP ON /*Permite pasar en un menu de la ultima a la primera
opcion*/DO WHILE OP # 0CLEAR          /* Limpia la Pantalla /*OP2=1OP=1@1,3
```

```
TO 3,73 DOUBLE /*Crea el cuadro para el menu principal /*@2,4 CLEAR TO
2,72   /* Limpia el area /*SET MESSAGE TO 22 CENTER /*Indica la
coordenada de los mensajes */
@2,5 PROMPT "HERRAMIENTAS" MESSAGE "HERRAMIENTAS BASICAS DE LA BASE"
@2,21 PROMPT "MEDICAMENTO" MESSAGE "DATOS DE LOS MEDICAMENTOS"
@2,37 PROMPT "RECETA" MESSAGE "HERRAMIENTAS DE RECETA"
@2,47 PROMPT "AYUDA" MESSAGE "AYUDA ACERCA DEL SISTEMA"
MENU TO OP /*Asigna el valor de la opcion /*
DO CASE

        CASE OP=1  /* Lo que se hara cuando se seleccione la opcion 1 */
           DO WHILE OP2 # 0
             @3,3 TO 10,20 DOUBLE  /* Cuadro para el submenu */
             @22,7 CLEAR TO 22,70
             @4,4 CLEAR TO 9,19
             @4,5 PROMPT "NUEVO" MESSAGE "AVADIR UN MEDICAMENTO"
             @5,5 PROMPT "BORRAR" MESSAGE "ELIMINAR MEDICAMENTO"
             @6,5 PROMPT "MODIFICAR" MESSAGE "MODIFICAR MEDICAMENTO"
             @7,5 PROMPT "IMPRIMIR FICHA" MESSAGE "IMPRIMIR FICHAS DE
MEDICAMENTO"
             @8,5 PROMPT "CONSULTAR" MESSAGE "CONSULTAR MEDICAMENTO"
             @9,5 PROMPT "SALIR" MESSAGE "SALIR DEL SISTEMA"
             MENU TO OP2
             DO CASE
               CASE OP2=1   /* Primera opcion del submenu */
               DO NUEVOMED
               CASE OP2=2   /* Seguna opcion del submenu */
               DO ELIMIMED
               CASE OP2=3   /* Tercera opcion del submenu */
               DO MODIMED
               CASE OP2=4   /* Cuarta opcion del submenu */
               DO FICHAS1
               CASE OP2=5   /* Quinta opcion del submenu */
               DO CONSULT4
               CASE OP2=6   /* Sexta opcion del submenu */
               CLEAR
               QUIT
               OTHERWISE   /* Cualquier otra opcion */
               OP=1
               EXIT        /* Regresar al menu principal */
```

```
            ENDCASE
            OP2=1
            ENDDO
ENDCASE
```

Como se puede ver para mostrar en pantalla las opciones del menú no se utiliza el comando @...say sino el comando @...Prompt. En los menús es el comando que se utiliza para mostrar las opciones, ademas este comando permite mostrar un mensaje cuando se encuentra seleccionada esta opción. Como se puede ver al estar declarando las opciones de menú principal no cambia el renglón sino la columna esto es que las opciones se mostraran en forma horizontal y en el submenu se puede ver que lo que cambia es el renglón estos es que el submenu sera vertical.

II.2.2.2. Gráficos y botones

En Clipper como ya se menciono no se pueden trabajar gráficos ya que el entorno bajo el que trabaja es DOS. Pero en el sistema que estamos haciendo utilizamos unos comandos de programación con la finalidad de dar un poco mas de presentación a nuestro sistema .

```
CLEAR       /*Limpiar pantalla */I:=0        DO WHILE I<25  @I,1 SAY
REPLICATE(CHR(178),80)    I=I+1ENDDO
```

En este programa que se presenta se utiliza el comando Replicate que repite un caracter cualquiera un cierto numero de veces. En este ejemplo se crea un ciclo que permite repetir un caracter 80 veces en todos los renglones, es decir que se cubrira toda la pantalla con este caracter. Notese que el tamaño de la pantalla es de 25 renglones por 80 columnas. El caracter es el CHR(178) que en clipper se obtiene por medio de codigo Ascii

Creando una combinacion de ciclos es posible hacer varias cosas con este comando, por ejemplo en el programa que estamos haciendo se trata de hacer una pantalla que muestre este fondo pero que se vaya creando desde adentro hacia afuera.

Otra forma de trabajar graficos es creando programas en Microsoft C y que se puedan compilar con Clipper pero la desventaja de esto es que Microsoft C tampoco maneja los graficos de un manera muy optima y los enlaces no son sencillos.

Dentro de Clipper unicamente se permiten crear cuadros y lineas. Esto, que ya se ha visto en los ejemplos anteriores, se hace mediante el comando:

@<coordenada_inicio> to <coordenada_final>

Al hacer un cuadro con este comando se puede agregar la palabra double al final si queremos que nuestro cuadro sea de linea doble. Con este comando tambien se pueden crear lineas dejando el renglon o la comuna de inicio igual que el renglon o la columna final. Tambien se pueden hacer las lineas dobles.

Como conclusion de presentacion mencionaremos que al combinar los menus, las cuadros, los mensajes, las coordenadas y los colores podemos llegar a presentar el Clipper un sistema que resulte amigable al usuario final.

II.2.3. Programación

Ya se menciono al principio Clipper en un lenguaje enfocado 100 % a la creacion de sistemas de bases de datos, es por ello que todos los comandos son enfocados al manejo de las bases de datos. Pero en esta parte no hablaremos de los comandos ya que la finalidad de este trabajo no es la de enseñar a programar pero si la de mostrar cuales son las herramientas de programación que tiene el lenguaje.

La forma de programar en Clipper es recomendable hacerla por medio de subprogramas, es decir que se genera un programa principal y este manda a llamar a los subprogramas para que realicen las tareas que se requieran . Para mandar llamar a un subprograma se hace por

medio del comando Do <Programa>. En Clipper también existen los procedimientos y funciones, que serán pequeños programas que se mandaran llamar desde el programa principal pero estarán contenidas dentro del mismo archivo, a diferencia que con los subprogramas estos serán otros archivos de programa.

La base para crear un programa es el manejo de las variables. Clipper maneja dos tipos de variables publicas y privadas. Como ya se explico la forma de programar en Clipper se hace llamando a subprogramas, las variables privadas serán variables que perderán su valor al terminar un subprograma, pero las variables publicas serán variables que solo perderán su valor cuando se de por terminado el programa completo. Cuando se programa manejando las bases de datos los campos también serán variables que se podran manejar dentro del programa.

Es importante mencionar un detalle que se debe de tomar en cuenta cuando se programa. Las letras mayúsculas y minúsculas son diferentes, es decir que si se declara una variable con una letra mayúscula, esta de debe de mandar llamar de la misma forma ya que si no Clipper la tomara como una variable completamente diferente.

Clipper como todos los lenguajes cuenta con funciones especiales para crear ciclos en la siguiente tabla se muestra cual es la sintaxis que utilizan las dos funciones para crear estos.

Do while <condicion>	For <var_inicio> TO <var_fin>
<Conjunto de instrucciones<	<Conjunto de Instrucciones>
Enddo	Next

Según sea el tipo de ciclo que se necesite hacer sera el comando que se utilizara para hacerlo , pero para crear ciclos con las bases de datos es mas recomendable utilizar Do while. Obviamente estos dos comandos tienen sus diferencias pero se toman dentro del mismo entorno ya que los dos crean ciclos. Depende del programador la forma de utilizar estos dos comandos.

También se cuenta en Clipper con comandos para la toma de desiciones, estos comandos manejan sintaxis claramente definidas:

If <condicion>	Do case
<Conjunto de instrucciones>	Case <condicion>
Else	<conjunto de instrucciones>
<Conjunto de Instrucciones	Case <Opcion>
Endif	<Conjunto de instrucciones>
	Otherwise
	<Conjunto de instrucciones
	Endcase

Al igual que con los anteriores estos dos comandos son diferentes pero los dos tiene la tarea de la toma de desiciones, básicamente Do Case es utilizado para el manejo de menús como ya se vio en el punto de menús. El comando If es utilizado para crear condiciones que son necesarias dentro de la programación.

Clipper contiene comandos para el manejo de fechas, de campos memo, de manejo de memoria, de manejo de arreglos, para menús, para impresión, etc. En el sistema final se podra ver como es que se realizo la programación para nuestro proyecto.

II.2.4. Otras herramientas

Clipper cuenta con un editor que ya se explico es un poco obsoleto pero aun así es importante mencionarlo. Este editor se conoce como PE.EXE que es utilizado para realizar los programas.

Clipper dentro de la programación utiliza dos herramientas que son muy eficientes para el manejo de las bases de datos aunque no son utilizadas debido a la dificultad que presentan en su sintaxis, los Codeblocks y los arrays.

Los Codeblocks también se conocen como bloques y pueden simular lo que se conoce como una macro, es decir que contiene un programa código ejecutable. A los codeblocks se les puede asignar cualquier valor como si fuera una variable y se ejecutan mediante el comando Eval( ). Su sintaxis es:

Nombre = { | [Lista de argumentos] | <comandos y funciones> }

Por ejemplo:

```
Suma = { | A,B | A+B+B}
?Eval (Suma,2,3)      /* El resultado seria 8
```

Otra forma de crear funciones como los Codeblocks es por medio de la instrucción #Define que permite asignar una valor a una variable, este valor se conservara durante toda la ejecución del programa y nunca se perderá el valor que se le asigno.

Por otro lado los arrays son matrices de tamaño n*n que se utilizan sobre todo para el manejo con las bases de datos, algunas veces no se quiere alterar directamente la infirmación que se tiene en las bases de datos, una buena forma de proteger esta infirmación seria crear un array y después copiar la infirmación de la base de datos a este array con la finalidad de tener una copia de seguridad o con la finalidad de realizar los cambios desde el array y no desde la base de datos.

Los array se pueden declarar así:

Declare nom[2,3] o Declare nom[2][3]

De cualquiera de estas dos formas se esta declarando un arreglo de 2*3 es decir una matriz del mismo tamaño. Como los arreglos también son variables se pueden declarar como públicos o privados según se vayan a utilizar.

Clipper es un lenguaje que cuenta con muchos comandos para el manejo de los arrays. En el ejemplo siguiente se ve como es que se copia la información de una base de datos a un array.

```
fil = Reccount()    /*Numero de registros*/
col = Fcount()      /*Numero de Campos */
Declare base[fil,col] /Declaración de array */
I=0
Do while .not. eof()  /*Ciclo para registros */
I=I+1
  For j=1 to Fcount() /* Ciclo para campos */
  base[i,j]:=&(Field(j)) /* Copia valor del campo a array*/
  Next
Skip
Enddo
```

En este ejemplo se generan dos ciclos, el principal que llevara un contador para copiar todos los registros, se detendra de uno por uno para que el ciclo que esta dentro copie la informacion que contienen los campos de todos los registros

II.3 Ventajas y desventajas ante los demás paquetes

Clipper es un lenguaje que no se puede comparar de lleno con los otros que se manejaran en este trabajo ya que los otros lenguajes trabajan bajo el entorno de Windows. Al principio de este capitulo se explico el porque se reviso Clipper. Por lo cual se puede pensar que una de las ventajas que ofrece este es que es un lenguaje que trabajara bajo un ambiente DOS y los requerimientos técnicos que requiere no son muy costosos y la implantación del sistema resultaría económica. Ademas de que este lenguaje genera archivos ejecutables.

Tal vez la que puede ser ventaja también puede ser la principal desventaja de Clipper, es decir los sistemas mas actuales requieren trabajar bajo el ambiente de Windows y la version 5.2 que es la que estamos revisando no lo permite, aunque cuenta con ciertas herramientas para trabajar el Mouse pero bajo el ambiente DOS.

Otra ventaja que puede ofrecer Clipper ante todos los demás lenguajes es que, como ya se menciono, todos sus comandos son referidos al manejo de bases de datos, es por ello que la programación en Clipper resulta sencilla. Se podría pensar que Foxpro para DOS es un lenguaje que podría competir con Clipper en este aspecto, pero FoxPro para DOS es un lenguaje que tiene algunas reglas que no se tienen en Clipper, y estas reglas traban un poco la programación y no la hacen tan sencilla.

Por otro lado otra de las principales desventajas que tiene Clipper es la falta de comandos para presentación, ya se vio en este capitulo como es que se tiene que utilizar algunos trucos para poder obtener una presentación agradable pero no llega a ser como las presentaciones que veremos en los demás lenguajes.

También es importante mencionar que aunque se tienen herramientas para informes y etiquetas son obsoletas y no permiten tampoco trabajar mucho con ellas.

Otro detalle que también se debe de tomar en cuenta es que Clipper posee comandos y funciones para archivos compartidos e impresoras comunes que trabajen en redes de área local.

Entonces resumiendo, Clipper es un lenguaje que permite generar sistemas para DOS muy eficientes pero que contaran con una presentación deficiente.

II.3.1 Compatibilidad

Por lo que se acaba de mencionar en el punto anterior es importante tomar en cuenta que la compatibilidad que pueda tener Clipper con los demás lenguajes revisados en este trabajo es casi nula.

La forma de trabajar con los archivos que tiene Clipper es muy diferente a otros lenguajes por lo que en si el único archivo que se podría compartir con otros lenguajes es el archivo

DBF, es decir la tabla, ya que los índices, informes y etiquetas, manejan formatos especiales de archivos por lo que no es recomendable compartirlos con otros lenguajes. Y aun así Clipper no puede abrir tablas que hayan sido modificadas con algún lenguaje de Windows.

Un lenguaje con el que se puede trabajar en conjunto con Clipper es Microsoft C, pero este es un lenguaje algo complicado, aun así se pueden crear enlaces para que Clipper mande llamar a funciones hechas en C.

Entonces se puede observar que Clipper es un lenguaje que no permite fácilmente la compartición de sus archivos por lo que no es recomendable utilizar archivos de otros lenguajes en Clipper ni utilizar los archivos de Clipper en otros lenguajes, ni de programación ni de otras herramientas.

II.3.2 Migración.

Se acaba de mencionar que no se recomienda utilizar los archivos de Clipper en otros lenguajes por lo cual, si se quiere hacer una migración de un sistema realizado bajo el entorno de este a un lenguaje de Windows esto resultara bastante complicado ya que la forma de programar es completamente diferente, tal vez el lenguaje que pudiera aceptar algunas partes del programa de Clipper seria FoxPro para Windows, pero solo se tomarían unas partes del programa hecho en nuestro lenguaje ya que no es posible utilizar todo el sistema.

Ahora bien si se desea migrar el sistema a una versión mas actual de Clipper esto resulta fácil ya que los comandos de este no cambian mucho con las versiones.

Clipper solo se podría migrar a lenguajes de DOS tales como Dbase o Foxpro y también se podría migrar de estos lenguajes a Clipper pero con los lenguajes de windows no se puede recuperar mucho ni de Clipper hacia los lenguajes ni viceversa.

Por esto concluimos con nuestro lenguaje los siguiente: Clipper es el lenguaje bajo DOS mas eficiente por todo lo que se acaba de mostrar en este capitulo, pero si se tienen las posibilidades económicas de crear un sistema en un lenguaje bajo Windows es mas recomendable, no solo por las desventajas que pudiera ofrecer Clipper sino por las desventajas que ofrece DOS.

III. FoxPro para Windows en las bases de datos

III.1 Conceptos Generales.

FoxPro es un lenguaje de programación que al igual que Clipper es un lenguaje destinado al manejo de las bases de datos, de ahí que sea sencilla la forma de trabajar con. FoxPro es un lenguaje creado para trabajar en el ambiente de Windows 3.X, aunque también trabaja bajo Windows 95 o 98.

En este trabajo se toma en cuenta a FoxPro ya que existen algunas empresas que aun tienen equipos que operan bajo el ambiente Windows 3.X. Este ambiente es muy agradable para trabajar con sistemas de bases de datos sobre todo por la misma facilidad que ofrece Windows.

FoxPro nace como competencia de Clipper, es decir nace siendo un lenguaje para DOS, pero nacen nuevas versiones tales como la 2.5 que permite trabajar en los dos entornos o la versión 2.6 que trabaja directamente con Windows. Esta ultima será la que nosotros utilizaremos para trabajar.

FoxPro es un lenguaje que es muy similar a Dbase o sea que la programación en Foxpro, Dbase y Clipper es muy parecida, pero hablando de programación, en el entorno de windows se tienen que realizar algunos cambios ya que como Windows maneja un ambiente gráfico distinto se tienen que acoplar los comandos a este ambiente.

FoxPro es un paquete que se pudiera utilizar como un manejador de bases de datos ya que ofrece un ambiente muy sencillo de manejar pero también es un lenguaje que ofrece muchas herramientas para realizar sistemas de bases de datos. FoxPro es un lenguaje que cuenta con muchas herramientas para dar una presentación al estilo de Windows y esto hace que los sistemas sean más agradables.

En este trabajo nos detendremos un poco a verificar cuales son las herramientas con las que cuenta FoxPro pero nos enfocaremos mas a ver como es que estas herramientas se unen para poder crear un sistema, lo anterior se menciona ya que como ya se dijo anteriormente FoxPro se puede utilizar como un manejador de bases de datos, de hecho muchas personas lo utilizan así y no como lenguaje de programación, pero la potencia que tiene FoxPro para poder generar sistemas es muy grande ya que se basa en Dbase Y Clipper pero los comandos que se manejen ya cuentan con muchas mas aplicaciones de las que vimos en el capitulo anterior.

De lo anterior es importante mencionar que FoxPro 2.6 es un paquete que basa todas sus herramientas en programación aunque para la realización del sistema ofrece los ambiente tradicionales de windows, FoxPro genera un código siempre para casi todas las herramientas, por ejemplo cuando creamos un menú, este se crea con un asistente que hace que la realización del menú resulte hasta cierto punto sencillo, pero FoxPro crea un archivo de lenguaje en el cual se puede ver la programación utilizada para hacer el menú, este archivo de lenguaje se crea ya que FoxPro lo utilizara para crear los archivos de aplicación.

En el párrafo anterior se menciono que para crear un menú se utiliza un asistente, FoxPro cuenta con varios asistentes no solo para menús, que ayudan a que las aplicaciones se generen de manera más sencilla. Esto se hace con la finalidad de que sea mas practico trabajar con FoxPro.

Existe una definición que a mi punto de vista se encuentra incompleta es "Microsoft FoxPro para Windows es una potente aplicación para la gestión de base de datos que se puede usar para almacenar, elaborar informes, u organizar la información que se utiliza cada día"[1]. Esta definición yo la siento incompleta ya que se deja fuera la potente herramienta de programación con la que cuenta FoxPro además de las herramientas gráficas y de presentación que el paquete ofrece.

[1] Microsoft FoxPro para Windows Paso a Paso, Edit Mc Graw Hill 1994

Aunque en el mercado también existen otros paquetes parecidos a FoxPro, como lo podría ser Dbase 5.0, se ha elegido FoxPro ya que este paquete se hace a mi juicio un poco más potente que los otros.

III.1.1 Características Técnicas.

FoxPro 2.6 es un lenguaje que trabaja bajo el entorno de Windows por lo que es necesario que para poder trabajar con se tengan los conocimientos básicos de Windows, es decir el manejo de ventanas y el conocimiento de la organización de archivo.

Como FoxPro 2.6 es un paquete de Windows 3.X las ventanas de abrir, guardar, etc. serán mostradas en este mismo entorno.

FoxPro es un lenguaje que instala todos los componentes necesarios para su adecuado funcionamiento por lo que se recomienda instalar completa la versión 2.6 de FoxPro. Durante la instalación el programa pregunta que tipo de pulsaciones se utilizaran, si serán para DOS o para Windows, al registrar las pulsaciones para DOS se indicara que el lenguaje se instalara para poder trabajar en DOS y Windows, pero lo más recomendable es trabajar con las pulsaciones para Windows para que el lenguaje ofrezca toda la potencia bajo Windows.

Un problema que presenta FoxPro es que no genera archivos ejecutables, por lo que para poder ejecutar un programa hecho bajo FoxPro es necesario que se encuentre instalado el paquete dentro del equipo.

III.1.1.1 Requerimientos en hardware.

En el punto anterior mencionamos que FoxPro trabaja con Windows 3.x por lo que el hardware que se requiere para poder trabajar sistemas bajo FoxPro no necesita ser muy

potente, al igual que con Clipper se podría utilizar un equipo que no fuera muy complejo y por lo mismo no resultara costoso.
El hardware básico recomendado para poder trabajar con sistemas basados en FoxPro seria:

Procesador 386 o superior.
2 Mb en memoria Ram
Monitor Color Vga
Disco duro de 200 Mb o mayor.
Unidad de disquete 3 ½
Tarjeta de vídeo Vga
Mouse y Teclado

Al igual que en Clipper ya no se encuentran nuevos equipos con estas características pero se puede utilizar cualquier equipo que soporte Windows 3.X ya que al soportar este software soportara también FoxPro.

La velocidad del equipo es importante para trabajar con los sistemas de FoxPro ya que como se maneja gran cantidad de información, la velocidad del procesador indicara la velocidad a la cual el sistema realizara sus tareas.

Es importante mencionar que se requiere la menos de 200 Mb de espacio en disco duro ya que se tiene que tener en cuenta que est espacio es destinado para instalar DOS, Windows 3.x y Foxpro, además FoxPro al trabajar las tablas genera archivos con un tamaño grande por lo que ocupa bastante espacio en disco, además también es sabido que las imágenes ocupan mucho espacio en disco duro y en Clipper se pueden añadir imágenes para dar una mejor presentación al sistema.

Por las cuestiones de presentación es importante que se maneje con un monitor de color ya que FoxPro permite el manejo de muchos colores (todos los de Windows) y gráficos por lo

que al trabajar con un monitor monocromático se perdería un poco la potencia que FoxPro ofrece para la presentación.

En FoxPro es muy necesario el Mouse ya que esta bajo el ambiente de Windows 3.X pero no solamente por eso sino porque cuando trabajemos con los formularios veremos que el Mouse es una herramienta básica en el manejo de sistemas creados en FoxPro.

FoxPro es una aplicación que puede trabajar con redes por lo que si se trabajara un sistema bajo red es necesario que se encuentre instalado FoxPro en el servidor, obviamente el servidor debe de cumplir las características de hardware que se mencionaron anteriormente.

III.2 Principales Herramientas.

FoxPro es un paquete que como ya se menciono es utilizado por mucha gente únicamente como manejador de bases de datos, esto se debe precisamente a las herramientas que ofrece el paquete, es decir que contiene muchas herramientas para el manejo de bases de datos, ya sea desde manejador o desde un programa.

La principal característica que tiene FoxPro es la herramienta para crear aplicaciones, debido a que no crea archivos ejecutables, estas aplicaciones son parecidas a los archivos ejecutables, con la diferencia que los archivos ejecutables se puede ejecutar desde el administrador de archivos o desde el prompt de DOS, y las aplicaciones se tienen que ejecutar en este caso desde FoxPro. Pero la forma de crear estas aplicaciones es muy sencilla.

FoxPro también cuenta con herramientas para trabajar con las bases de datos relacionales, este es el caso de las consultas y de las vistas, estas dos herramientas permiten trabajar con una o más bases de datos, por medio de estas relaciones también es que se generan los informes con varias bases de datos.

Ya se ha mencionado la potencia que tiene FoxPro en cuestión de gráficos, esto no solo es dado por ser un lenguaje de Windows si no por las mismas herramientas de FoxPro, dentro de las herramientas de FoxPro para gráficos genera menús, formularios, botones, etc., estas se verán mas adelante.
Estos gráficos combinados con otras herramientas de programación permitirán que el usuario genere sistemas amigables y que además trabajen de manera adecuada.

Además FoxPro también cuenta con una ayuda muy potente que contiene todos los comandos, posibles errores, aplicaciones, etc. Los importante de esta ayuda es que la forma de hacer consultas es de manera muy sencilla y simple.

III.2.1 Manejo de Archivos

FoxPro genera una gran cantidad de archivos, cada uno de estos archivos tiene una tarea asignada para que el sistema pueda trabajar de forma adecuada. Básicamente genera los siguientes archivos.

Tablas (DBF)
Programa Fuente (PRG, SPR, MPR)
Informes (FRM, FRX)
Etiquetas (LBX)
Pantalla (SCX)
Menú (MNX)
Consulta (QPR)
Proyecto (PJX)
Aplicación (APP)
Indice (IDX)

Ya se menciono que FoxPro todo lo trabaja por medio de programación, entonces hablando de archivos al generar un menú se genera un archivo con extensión MPR que contiene el

código y un archivo MNX que contiene el formato del menú. Por esto se dice que los archivos trabajan de manera compartida y casi todos generan archivos de código. Las únicas herramientas que no generan archivo de código son los informes y las etiquetas que serán formatos establecidos.

El hecho de generar estos archivos de código hacen que el programador no trabaje tanto ya que es mucho más sencillo generar tareas desde un asistente que programarlas.

También es importante mencionar que cuando se añaden o quitan registros a las bases de datos no solo existe el archivo DBF sino que se genera un archivo con extensión DBT que será como una copia de seguridad que guarda las tareas que se van realizando, también genera un archivo CDX que será un índice interno que utiliza FoxPro para poder trabajar las tablas de manera adecuada.

Los archivos de aplicación (APP) no se pueden modificar, estos archivos se generan al generar el proyecto (mas adelante sé vera como se generan los proyectos), por ello es que los archivos APP y PJX en un sistema contienen el mismo nombre.

Se recomienda que no se borre ningún archivo de los que genera FoxPro aunque tengan un tamaño grande, ya que algunas veces aunque no se vea que se utilicen todos los archivos internamente FoxPro los enlaza para poder trabajar.

III.2.1.1 Tablas

Las tablas en FoxPro se guardan con la extensión DBF, aunque como ya se menciono también se genera un archivo DBT.

Las tablas se trabajan de manera muy parecida a Clipper, por ejemplo para crear una base de datos se puede hacer desde el menú o desde la línea de comando, el comando utilizado para hacerlo desde la línea de comando será:

Create <nombre_tabla>

Es decir que será el mismo que se utilizaba en Clipper. Por menú se tendrá que seleccionar del menú Archivo la opción Nuevo, se mostraran en pantalla las diferentes herramientas que podemos crear, en este caso daremos un Clic en la parte de Tabla/DBF y se tecleara Enter. En cualquiera de las dos formas que lo hagamos aparecerá una pantalla en la que se indicaran el nombre, el tamaño y el tipo de los campos que utilizaremos.

En FoxPro nos detendremos un poco a revisar los tipos de campos. En él capitulo 1 se menciono que según fuera el lenguaje será el tipo de campos que se maneje, los tipos de campo que conocemos son carácter, numérico, memo, lógico, aunque en el sistema de Clipper también utilizamos el campo tipo fecha. En FoxPro se añaden otros dos tipos de campo: float (flotante) y general.

El campo tipo flotante es muy parecido al campo numérico, es decir se tiene que declarar un tamaño para los enteros y un tamaño para los decimales (esto ya se explico antes). El campo flotante es recomendado para cuando los datos que se vayan a utilizar se consideren científicos, es decir que se vayan a realizar cálculos que necesiten mas de 16 dígitos.

Por otro lado el campo general es un campo que nos ofrece un tamaño preestablecido de 10 espacios, aunque como en los campos memo este tamaño únicamente sirve para identificarlo, en realidad el campo general ocupa mucho mas espacio que los 10 caracteres que se dicen. Este campo es utilizado para agregar imágenes y sonidos a la tabla, por ejemplo supóngase una empresa en la cual se realiza una base de datos que contiene los datos generales de los empleados pero además queremos agregar la fotografía de cada uno de ellos, para hacer esto se declara un campo de tipo general en el cual se agrega la foto.

Para agregar ya sea un sonido o una imagen se realiza de la siguiente manera: cuando se estén agregando los datos a la base y queremos agregar un gráfico, bastara con editar en

cualquier paquete (paintbrush, corel, reproductor de sonidos, etc.) la imagen o sonido que queremos añadir, se selecciona, se copia y se pega en Foxpro en el área que corresponda al campo general.

Una vez que ya tenemos nuestra base de datos creara podemos manipularla desde el menú o desde la línea de comandos, pero para hacerlo por medio de la línea de comando es necesario conocer claramente los comandos utilizados para añadir, editar, eliminar, cambiar, consultar, etc.

Para realizar las tareas anteriores por medio del menú es muy sencillo ya que FoxPro nos va indicando los datos que tenemos que introducir para realizar la actividad que deseamos. Para abrir una tabla se hará de una manera muy sencilla ya que como ya se menciono las ventanas que se trabajan serán las mismas de windows, por lo que abrir una tabla es tan sencillo como abrir un documento en Word.

Una vez abierta la tabla, para poder trabajar con ella, sé eligirá la opción examinar del menú tabla, y se mostrara en pantalla la tabla que abrimos en forma parecida a Excel, es decir por medio de filas y columnas, en la parte superior se mostraran los nombres de los campos. Los menús de tabla y registro contienen todas las herramientas para poder trabajar con las tablas. Además de mostrar la información en forma de tabla, también se puede mostrar los campos en forma de lista, a esta forma se le llama modo añadir y se encuentra en el menú Tabla.

Es importante mencionar que, como ya se ha explicado, en los lenguajes de base de datos no se borran completamente los registros únicamente se marcan, cuando estamos trabajando con la tabla, los registros marcados aparecen con una marca negra en la parte izquierda de la tabla. Para eliminar los registros marcados se utiliza la opción empaquetar del menú tabla.

Otra forma de visualizar y trabajar con el contenido de la tabla es utilizando los comandos que ya vimos en Clipper, @ ... Say y @ ... Get, estos comando trabajan de igual manera que en Clipper pero con la diferencia que en FoxPro contienen otras aplicaciones como la de

poder definir el tipo y el tamaño de letra con el que se mostrara la información. (Esto sé vera a más grandes rasgos en las presentaciones)

Otro punto importante en las tablas son los índices, en FoxPro los índices se generan también de una manera muy sencilla, se recomienda que los índices se generan cuando se esta generando la estructura de la base de datos.

Para crear los índices se hace al igual que en Clipper se elige un campo y este trabajara como índice, o bien se pueden elegir varios campos para trabajar como índice pero recuerde que el primer campo que se elija será el campo que tendrá el índice primario. En la pantalla que se presenta a continuación se mostrara la estructura de datos que se ha generado para nuestro sistema desde FoxPro conteniendo los índices que se utilizaran.

En esta pantalla se puede ver que de un lado se muestra la estructura y del otro lado se muestran los índices, note que todos los campos de la estructura están del lado de los índices, esto no quiere que se ha creado un índice para cada campo, en esta pantalla se eligen los índices seleccionando el campo que se quiere agregar como índice y se dará un clic en la opción de agregar, después se nos mostrara la pantalla de guardar en la cual escribiremos el nombre que tendrá nuestro archivo de índice. Con la opción de modificar podremos hacer algunos cambios como podría ser ordenar en forma ascendente o descendente, etc.

Para abrir un índice se hace por medio de la línea de comando tecleando:

Use <nom_tabla> index <nom_indice_

O bien

Set index to <nom_indice>

Al igual que en Clipper las tablas se pueden abrir con la palabra Exclusive al final que indicara que la tabla no se podrá utilizar en red, en caso de que FoxPro se encuentre instalado en el servidor, nunca se debe de abrir una tabla con la palabra Exclusive.

Otro punto muy importante en FoxPro con relaciona las tablas es la gran potencia relacional que tiene este. Las relaciones como ya se explico en el capitulo I son utilizadas para poder trabajar con dos o mas bases de datos al mismo tiempo. En Clipper se puede hacer pero no tiene la potencia que tendrán en FoxPro es por ello que en este caso nos detendremos a analizar un poco mas esto.

El punto mas importante para poder relacionar tablas es que las dos o mas tablas que se vayan a relacionar deben de estar indexadas por medio de un campo común a las dos, es decir que en las bases de datos debe de existir un campo que sea igual en todas.

Una vez que contamos con las tablas indexadas utilizaremos las siguientes expresiones para poder crear una relación. Se tienen que establecer áreas de trabajo para cada una de las bases de datos, es decir que cada una de las tablas que se van a relacionar deberán trabajar en un área especifica. El área se identifica por un numero y se asigna a las tablas por medio de los comandos siguientes:

```
Select 1
Use tabla1
Select 2
Use tabla2
```

De esta forma estamos asignando a la tabla 1 el área 1 y a la tabla 2 el área 2. Una vez que ya se asignado el área se puede volver a habilitar la tabla únicamente seleccionando el área de esta.

Ya que se asigno el área de trabajo se puede crear la relación utilizando el comando:

Set relation to <campo_indice> Into <tabla_a_relacionar>

En el ejemplo siguiente se vera mejor como se crea la relación entre 3 tablas:

```
CLOSE ALL  ** Cierra todas las tablas **
CLEAR      **Limpia la pantalla **
SELECT 1  **Selecciona el área 1**
USE TABLA1 **Abrir tabla1 para el área 1**
SELECT 2  **Selecciona el área 2**
USE tabla2 INDEX ind_1 **Abrir tabla2 con índice ind_1 para el área 2**
SELECT 3  **Selecciona el área 3**
USE tabla3 INDEX ind_2  **Abrir tabla3 con índice ind_2 para el área 3**
SELECT 1  **Utiliza el área 1 es decir la tabla1 **
SET RELATION TO ind_1 INTO tabla2  ** Crea relación de la tabla 1 con la
tabla 2 **
SET RELATION TO ind_2 INTO tabla3  ** Crea relación de la tabla 1 con la
tabla 3 **
```

En el ejemplo anterior se esta creando una relación de la tabla2 y la tabla3 con la tabla 1, cada relación se esta dando por un índice diferente, cada una de estas relaciones se pueden utilizar para diferentes tareas sobre todo de informes y consultas. En FoxPro se pueden crear tantas relaciones como se considere necesario.

III.2.1.2 Etiquetas

Como ya se menciono en el capitulo anterior, las etiquetas son una forma de presentar la información que se tiene en la base de datos, ya sea en papel, en pantalla o directamente a un archivo de texto.

En el lenguaje que estamos revisando las etiquetas también se pueden realizar de dos formas, programando o utilizando el asistente de FoxPro. Para programar las etiquetas se hará de la misma forma en que se realizan en Clipper (capitulo II), es decir que utilizaremos los ciclos y una serie de comandos para formar un programa que pueda organizar la información para

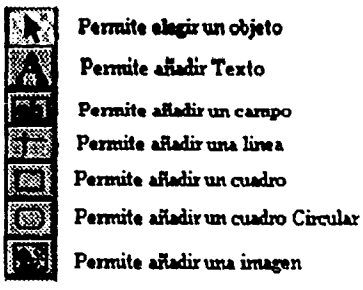
poder ser mostrada en forma de etiqueta. Por ejemplo si compiláramos y ejecutáramos en FoxPro el mismo programa que se presento en el capitulo anterior referente a las etiquetas, el programa no provocaría ningún error y se ejecutaría de igual manera que en Clipper, pero en FoxPro podemos agregar a los comandos algunas otras opciones para poder determinar los tipos y tamaños de letras con la finalidad de dar a la etiqueta una mejor presentación.

Pero en el lenguaje que en esta ocasión estamos revisando programar una etiqueta resulta algo innecesario ya que el asistente con el que se cuenta ofrece muchas ventajas para poder generar estas etiquetas. Revisaremos a detalle la forma de crear etiquetas ya que los informes y formularios se crean de manera muy similar.

El asistente con el que se cuenta contiene una serie de botones que son los diversos objetos que se pueden añadir a la etiqueta. Estos objetos se refieren a los campos, texto o líneas que se pueden agregar.

La forma de trabajar en este asistente será muy fácil, imaginemos la pantalla de paint de windows, en la cual para poder dibujar una línea se tiene que dar un clic en el botón de línea y posteriormente dibujarla en la pantalla, en FoxPro también se hace de la misma manera, es decir, que si queremos agregar a nuestra etiqueta un campo únicamente tenemos que dar clic en el botón de campo y dar un clic en la pantalla.

En la figura siguiente tenemos los botones con los que se cuenta en el asistente de etiquetas, a continuación se explica como se utilizan estos:



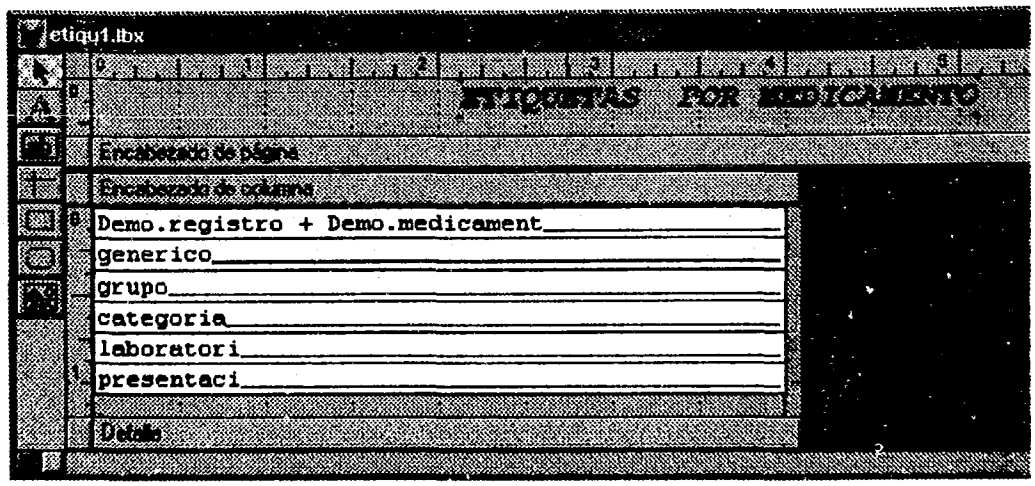
Ya se explicó como se debe hacer para poder agregar un objeto a la etiqueta pero cuando se quiere agregar un campo se debe de indicar cual es el campo que se va a añadir, en el párrafo anterior se explico que se selecciona el botón y posteriormente se da un clic en la pantalla, a continuación aparece una pantalla en la cual se tecleará el nombre de campo en la parte que pide la expresión, además también se puede teclear un formato especial para los campos, este formato indicara el tipo de campo que se va a mostrar así como también algunas otras opciones que se pueden añadir a los campos.

Por otro lado al seleccionar un objeto que se encuentre en nuestra etiqueta contaremos con un menú que se llama objeto con el cual podremos dar la presentación al objeto, es decir el tipo de fuente que se utilizara, el ancho que tendrán las líneas, los colores tanto de líneas como de cuadros, etc.

En los párrafos anteriores vimos como es que se pueden agregar los objetos a la etiqueta, pero también es importante mencionar como se hace para crear una nueva etiqueta. Así como en la base de datos se elegía la opción nuevo del menú archivo, para crear una etiqueta se hace de la misma forma solo que obviamente en este caso se selecciona la opción de etiqueta, a continuación se muestra en pantalla todos los tamaños que se tienen para las etiquetas por lo que se seleccionara uno de estos, cabe mencionar que este tamaño es independiente al tamaño de la hoja con el que se cuente para imprimir.

Al seleccionar un tamaño de etiqueta se mostrara en pantalla ahora las opciones mencionadas anteriormente.

En la pantalla siguiente se muestra el formato de etiqueta que se utiliza para nuestro proyecto.



Nótese en la figura que además el asistente cuenta con un área designada para el encabezado y el pie de pagina en dado caso que el usuario tenga que agregar alguna de estas dos herramientas.

También es importante mencionar que se puede modificar el tamaño de cualquier objeto de la misma forma que se hace en cualquier aplicación de windows. En el caso de las etiquetas de nuestro sistema estamos utilizando un tamaño para poder acomodar dos columnas de etiquetas en cada hoja por lo que se muestran dos encabezados, uno para la pagina y uno para la columna, lo mismo para con el pie de pagina. En este caso únicamente hemos añadido encabezado para la pagina.

Para poder utilizar estas etiquetas en el proyecto o en algún sistema se hará por medio del comando Label Form <archivo>, es decir utilizando la misma sintaxis que en el capitulo anterior, a diferencia que en FoxPro podemos agregar la sintaxis To File <archivo> y las etiquetas se guardaran en un archivo de texto.

III.2.1.3 Informes

La manera de trabajar los informes será de manera muy similar a la de las etiquetas, es decir tendremos dos formas de generarlos, por medio de programación y por medio del asistente.

Por medio de programación se realiza de la misma manera que en Clipper, es decir acomodando los campos de la forma que el programador lo considere y utilizando las bases de datos que se quieran. En el ejemplo final se utilizo esta técnica para poder imprimir las recetas.

El asistente será el mismo que se utilizo para crear etiquetas, la única diferencia radica en que en este no se solicita el tamaño de la etiqueta sino que el tamaño estará establecido por el tamaño que tenga como predeterminado la impresora actual. Entonces los informes y las etiquetas se crean de la misma manera, se contara con un área para el encabezado, otra para el pie de página y el área donde se agregaran los objetos que se denomina detalle.

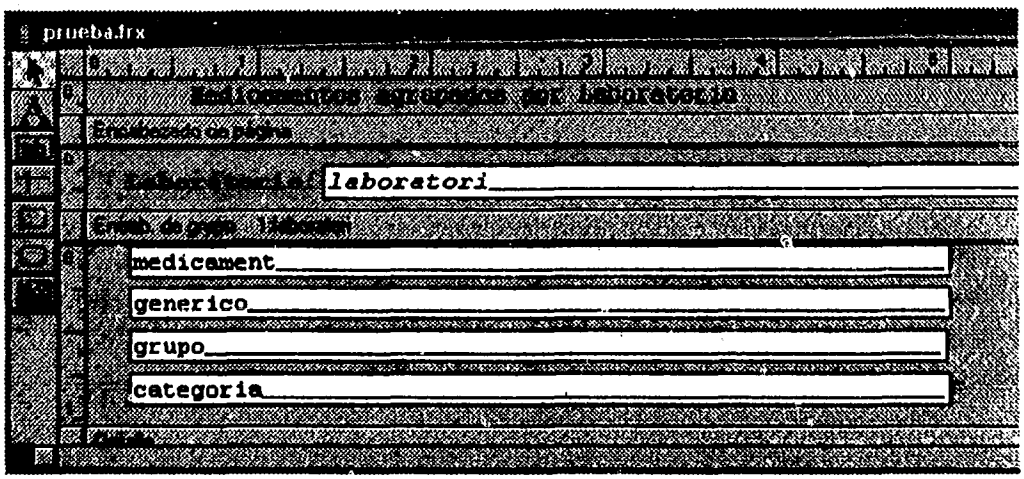
Una ventaja que presentan los informes en FoxPro es que se pueden realizar de varias tablas, es decir que se puede utilizar una relación dada para poder generar un informe, o también se puede utilizar una consulta (mas adelante revisaremos como se crean estas) para poder crear el informe.

Al crear un informe con tablas relacionadas primero debe de crearse la relación y posteriormente el informe, en este caso cuando añadimos un campo al informe no solo se indicara el nombre del campo sino que también se indicara el nombre de la tabla a la cual pertenece este campo. Esto se puede hacer tecleando <nom_tabla>.<nom_campo>, con esta sintaxis estaremos indicando el nombre de la tabla y del campo.

En nuestro proyecto no utilizamos el asistente para crear nuestro informe ya que las bases de datos que utilizamos en el sistema no se pueden relacionar ya que son completamente

independientes unas de otras. Al tener las tablas de esta manera no se pueden crear informes por medio del asistente con varias tablas.

En el asistente también se pueden crear los grupos que ya vimos en el capitulo anterior. Cabe mencionar que para poder crear un grupo se debe de indexar en base al campo que se va a utilizar como clave de grupo. Par poder crear el grupo se debe seleccionar la opción agrupar del menú informe, una vez que selecciona este se debe de teclear el nombre del campo que se va a utilizar como clave de grupo. En el ejemplo que se muestra a continuación crearemos un informe por grupo utilizando el campo *laboratori* de nuestra tabla.



Note que cuando agrupamos, también aparece en la pantalla un área especial para el encabezado y el pie de pagina de cada grupo.

Otro detalle que también podemos agregar a nuestros informes es la posibilidad de totalizar algún campo de tipo numérico, es decir que se pueden realizar cálculos aritméticos dentro de la misma creación del informe con la finalidad de que el informe se muestre de acuerdo a las necesidades que el usuario tiene. Para poder agregar una operación aritmética, cuando se agrega un campo no solo se indicara el nombre del campo sino que además podemos crear una formula utilizando los comandos que nos ofrece FoxPro. Esto le indicara al informe que al momento de imprimir mostrara el resultado de esta formula.

Por otro lado, al igual que en Clipper, una vez que ya se ha creado el informe se tendrá que mandar llamar desde nuestro sistema, esto se hace con el comando.

Report Form <nom_informe> To <nom_archivo>

Este comando se puede enfocar también hacia la impresora añadiendo To Print al final del mismo. Al utilizar este se podrán manejar las condiciones que se vieron en el capitulo anterior, es decir realizar los informes creando filtros que únicamente impriman algunas partes y no toda la tabla

III.2.2. Manejo de Presentaciones

Las presentaciones en FoxPro se pueden considerar como buenas ya que permiten gran diversidad de herramientas para estas, desde los colores que se manejan hasta los diversos gráficos que se pueden añadir.

En este lenguaje comenzaremos a manejar una herramienta para presentaciones que será esencial en este y en los siguientes lenguajes que se verán. Esta herramienta se denomina formulario o pantalla. Antes de entrar de lleno a los formularios hablaremos un poco del manejo de los colores que se utilizaran en este lenguaje.

En FoxPro los colores se pueden utilizar de tres formas, la primera muy sencilla es configurando los colores directamente desde windows, es decir que si en el sistema no se establecen colores se tomaran los colores con los cuales se encuentre configurado windows, en caso de que se quisiera cambiar los colores se tendrá que hacer desde el panel de control.

Por otro lado, el programador puede establecer los colores que se utilizaran desde el menú, como se utilizan gráficos (líneas, botones, etc.), los colores se pueden configurar desde el

menú Objeto, ya que este permite configurar los colores y tamaños de los objetos que se hayan añadido.

Una tercer forma es por medio de programación, es decir utilizando el comando Set Color, este comando ya se estudio un poco en el capitulo anterior, solo que en FoxPro ofrece mas herramientas para este comando.

En este lenguaje se manejan parejas de colores, esta será formada por el color del primer plano y el color de fondo. Básicamente FoxPro maneja los siguientes ocho colores básicos: rojo, verde, azul, cyan, magenta, amarillo, blanco y negro. De igual forma que en Clipper estos colores se manejan por medio de letras, por lo que para crear una pareja de colores se utilizara:

<color de primer plano> / <color de fondo>

Al igual que en Clipper se cuenta con dos comodines que son: (+) que indica un primer plano con color mas claro y (*) que indica el color de fondo mas claro.

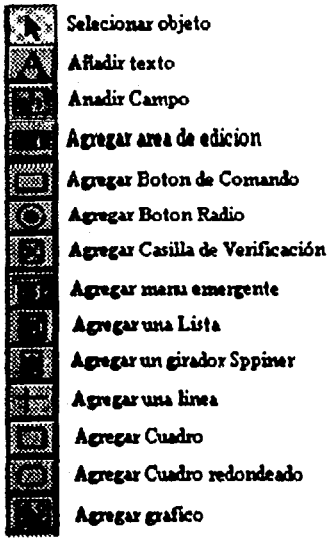
Otra forma de trabajar los colores, es creando combinaciones entre los tres colores primarios por medio de la sentencia RGB( ), los colores se manejaran por números utilizando el 0 para menor intensidad y el 255 para mayor intensidad, se tienen que declarar tres colores para la combinación, el rojo, el verde y el azul. Como se realizan parejas de colores se tienen que declarar 6 números, por ejemplo para declarar el color rojo sobre gris oscuro se declara:

RGB(255,0,0,128,128,128)

Una vez que ya explicamos como se manejaran los colores analizaremos ahora los formularios, estos son la herramienta mas básica para realizar sistemas en lenguajes que trabajan en el ambiente Windows.

Los formularios serán las pantallas de presentación, por ejemplo en Word, el formulario será la pantalla en la que trabajamos. Contando los botones, las barras de desplazamiento, etc. En FoxPro no existen grandes herramientas para formularios pero con las que cuenta podemos crear un sistema con una presentación amigable.

Los formularios en FoxPro cuentan con muchas herramientas, estas herramientas se muestran en la figura que se presenta a continuación, en cada una de las figuras se contiene una pequeña explicación de para que sirve cada botón:



En estos formularios, al seleccionar el objeto, se podrá indicar el código que se aplicara cuando se ejecuta el sistema. Para el código tendremos des eventos, el evento When y el evento Valid (estos eventos puede cambiara según el objeto). El evento valid se ejecutara cuando dentro del sistema se seleccione el objeto. Por ejemplo se añade un botón de comando y se teclea un subprograma en el mismo, cuando se ejecuta el sistema y se da clic en el botón se ejecutara el subprograma que se añadió. El evento When se utiliza para poner condiciones a la selección del objeto. En la parte de programación revisaremos mas a detalle esto.

Cabe mencionar que cuando se guarda un formulario también se genera un archivo de código que tendrá todo el programa que se utiliza para poder generar el formulario. Este archivo no se debe de modificar ya que el formulario no aceptara los cambios que se realicen en el archivo de código.

Formularios se pueden crear tantos como el programador considere necesario pero para poder mandar llamar un formulario desde otro se llamara el archivo de código que en este caso son los archivos con extensión spr.

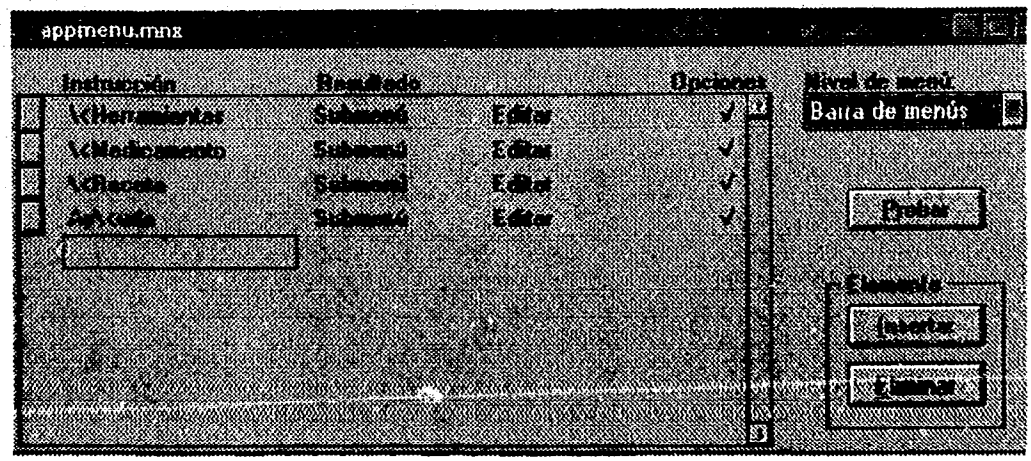
III.2.2.1. Menús

En los sistema basados en windows, el menú es indispensable ya que los usuarios de Windows se encuentran muy acostumbrados a todas las ventajas que este ambiente nos ofrece. Los menús que se crearan en FoxPro serán los menús clásicos utilizados en Windows. Estos menús permiten trabajar con el Mouse y con el teclado.

Cuando se trabaja con un sistema, el cual contara con un menú, se utilizara el archivo appmenu.mnx que contiene el menú clásico que FoxPro genera para cualquier sistema, este menú será el que se mandara llamar desde nuestro programa principal. El menú general se podrá modificar con las opciones que se necesiten para nuestro sistema pero siempre utilizando el archivo que ya se menciono. Pero también podemos crear otros menús y utilizarlos dentro de nuestro sistema pero se recomienda no hacer esto sino que se debe de basar en un solo menú y este será el mostrado por el archivo appmenu.mnx. Cabe mencionar que además de tener el archivo appmenu.mnx también se genera el archivo appmenu.mpr que contiene el código utilizado para poder trabajar con el menú.

Lo primero que haremos para crear un menú o utilizar uno ya creado será elegir la opción abrir del menú archivo y seleccionar el tipo de archivo de menú. En caso de que se quiera crear un nuevo menú se dará un clic en el botón Nuevo, en caso de querer abrir uno ya existente, se seleccionara y se dará clic en el botón Aceptar.

En la pantalla que se presenta a continuación crearemos nuestro menú y la utilizaremos para explicar como es que se crea o modifica un menú.

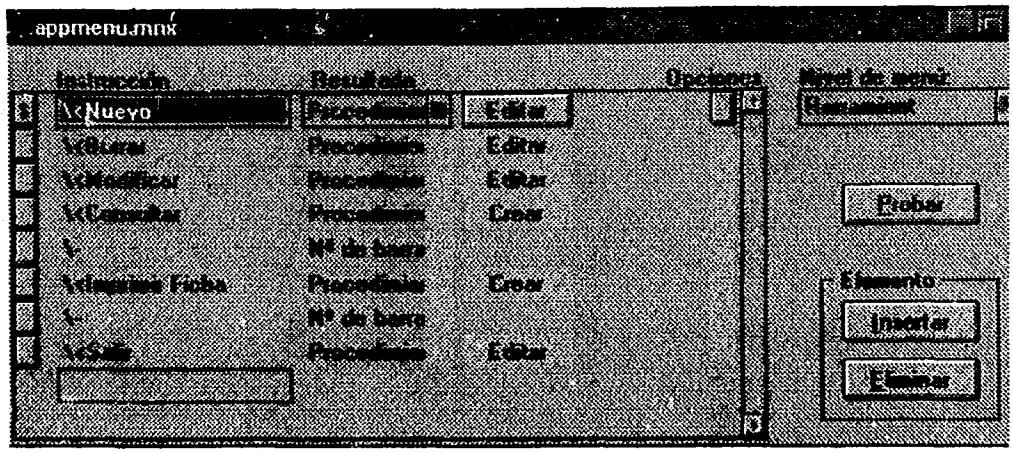


En este caso este será nuestro menú principal, este contendrá 4 Opciones, se teclea el nombre de la opción y se elige el resultado que se obtendrá de esa opción, en este caso las cuatro opciones son submenus, es decir que cada una de estas opciones contiene un menú dentro de cada una de ellas.

La sintaxis \< sirve para indicar que letra aparecerá subrayada con la finalidad de utilizar la combinación de teclas Alt-Letra para elegir esa opción.

Por ejemplo en la opción de herramientas aparecerá subrayada la letra H y podremos elegir esta opción tecleando Alt-H.

Revisemos ahora una de las opciones para ver como queda el submenu, en la siguiente pantalla vemos las opciones que contiene el menú Herramientas.



Para poder accesar a este submenu basta con seleccionar la opción herramientas y dar un clic en la opción Editar de la pantalla que ya se presento.

En esta pantalla aparecerán al igual que en la anterior las opciones que contendrá en este caso el menú herramientas, al igual que en la pantalla anterior en esta únicamente se escribe en el espacio asignado para ello el nombre de la opción y el resultado. En este caso el resultado será un procedimiento. Al dar un clic ya sea en la opción Crear o editar nos llevara a un pantalla en la cual se tecleara el código correspondiente al procedimiento de la opción

Al escribir nosotros el código para los procedimientos este se guarda también en el archivo de código que ya mencionamos anteriormente.

Como se puede observar en los párrafos anteriores la forma de crear un menú es muy sencillo únicamente lo que presenta la dificultad es el código que se debe de generar para los distintos procedimientos que se añaden.

Para mandar llamar el menú desde el programa principal se llamara el archivo appmenu.mpr ya que este contiene el código, este se encargara de hacer los enlaces con el archivo mnx para poder trabajar con le menú.

III.2.2.2. Gráficos y botones

Los gráficos y botones ya se han explicado en las diferentes aplicaciones de FoxPro. Por ejemplo ya sabemos como es que podemos agregar un campo gráfico, ya sea de imagen o sonido, también ya vimos las diferentes opciones que nos presenta un formulario para poder trabajar en el.

Estos objetos se llevan a la pantalla de trabajo como si estuviéramos trabajando en Paint, es decir se da un clic en el objeto y a continuación damos un clic en el área de trabajo, el objeto se puede mover por la pantalla hasta que lo coloquemos en la posición que consideremos adecuada, además también podemos cambiar el tamaño de la manera tradicional de windows, es decir seleccionando el objeto y ajustando el tamaño por medio del Mouse.

La organización de la pantalla depende del usuario ya que el usuario puede decidir que los botones contengan texto o que los botones contengan un gráfico, cuando añadimos un botón en pantalla se solicita el nombre del botón, en este caso contamos con la opción de que en lugar de que el botón sea normal, el botón sea una imagen, aunque en este caso cabe mencionar que tamaño del botón estará determinado por la imagen por lo que si se quiere hacer un botón de tamaño pequeño se recomienda que primero se arregle la imagen del tamaño que se quiere quede el botón.

Además de que también se puede configurar el fondo del formulario para que contenga un gráfico o un color especifico como si se tratara del papel tapiz de windows. Para hacer esto basta con seleccionar la opción distribución del menú pantalla, cuando hacemos estos nos aparece una pantalla en la cual podemos determinar el tamaño de la pantalla, el color o gráfico de fondo y el tipo de letra que utilizara el formulario.

III.2.3. Programación

La programación de FoxPro para Windows es parecida a Clipper en cuestión de la sintaxis que se maneja, obviamente como ya se ha mencionado en varias ocasiones los lenguajes que FoxPro maneja cuenta con otras aplicaciones y ventajas que en Clipper no se tenían.

FoxPro maneja de igual forma los comandos utilizados ya sea para crear ciclos o para la toma de decisiones y se trabajan de igual manera que en Clipper por lo que no los mostraremos en este capitulo.

La programación se hace también por medio de procedimientos y funciones, la diferencia radica en que como en FoxPro utilizamos ya sea formularios o menús, los procedimientos se programan dentro de estos y de esta forma se evitara que el usuario programe todo.

En este lenguaje también se manejan las variables de igual forma que en Clipper, es decir tenemos variables publicas y privadas, además también podemos utilizar los campos de las bases de datos como variables.

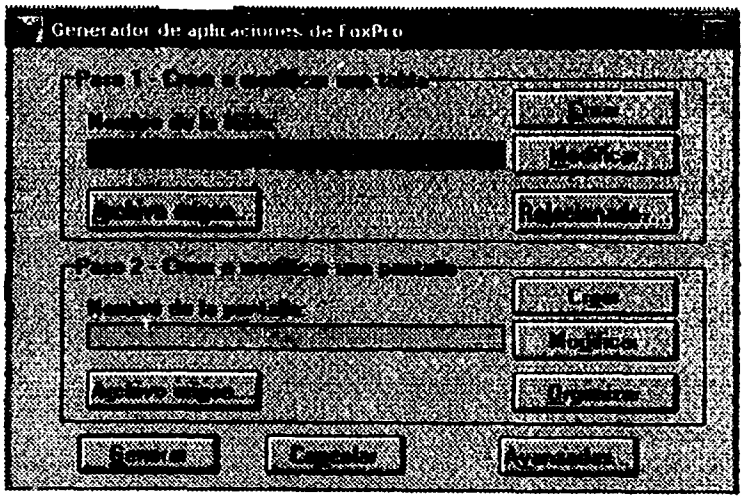
FoxPro contiene varias variables ya definidas que se pueden utilizar en cualquier parte del programa ya que estas son publicas, pero el usuario puede crear sus propias variables según las necesidades que se tengan.

Con lo anterior se podría pensar que programar en este lenguaje es muy fácil pero no es así, debido a que los comandos contienen nuevas aplicaciones, la sintaxis de estos se ha vuelto mas dificil, aparte de esto FoxPro cuenta con un conjunto de reglas para la programación, por ejemplo una regla que provoca muchos problemas es que en este lenguaje no se permite anidar mas de 5 niveles de Read, dentro de la programación del capitulo anterior y este se vio que el comando Read es muy utilizado sobre todo para poder trabajar con el comando @...get (comando indispensable en este lenguaje), el comando Read es muy importante ya

que como ya se explico FoxPro genera el código de la mayoría de las aplicaciones con las que se cuenta, por lo que utiliza estos comando

Para entender mejor lo anterior veamos lo siguiente: por ejemplo en un formulario que contiene varios botones, utiliza el comando Read para poder esperar que el usuario de un clic a cualquiera de estos botones, para leer cualquier valor desde el teclado o el Mouse se utiliza el comando Read, pero cuando utilizamos este comando varias veces (como se hace en otros lenguajes) el sistema marcara errores y no se podrá ejecutar de una manera optima. Esto solo es un ejemplo de varias reglas que se deben de seguir para programar en este lenguaje, si bien es cierto que mucha de la programación la realiza automáticamente FoxPro también es cierto que programador debe de generar el código para todos los botones o procedimientos de menú que se requieran.

Ahora, la forma de juntar todas las cosas que hemos hecho se realizan creando aplicaciones, esto se menciona aquí ya que al crear las aplicaciones automáticamente FoxPro creara el código de lenguaje para el archivo principal que será el encargado de manipular todas las cosas que hemos agregado a nuestro sistema. Para poder crear la aplicación, es necesario elegir la opción aplicación del menú ejecutar, al estar en esa pantalla se selecciona la opción nuevo, en pantalla nos muestra lo siguiente.



En esta pantalla se escribe el nombre de la tabla principal que se manejara, además también se escribe el nombre del formulario general, ya que a pesar de que se pueden hacer varios formularios es necesario definir uno que sea el principal y que mande a llamar a los otros.

Una vez que ya se han tecleado estos dos datos se da un clic en el botón generar. Al hacer esto se van a generar varios archivos de código, el archivo que será el principal, es decir el que mandara a llamar a las demás aplicaciones será el archivo. Scaffold.prg, este declarara las variables, mandara a llamar el menú, abrirá la base de datos y abrirá el formulario.

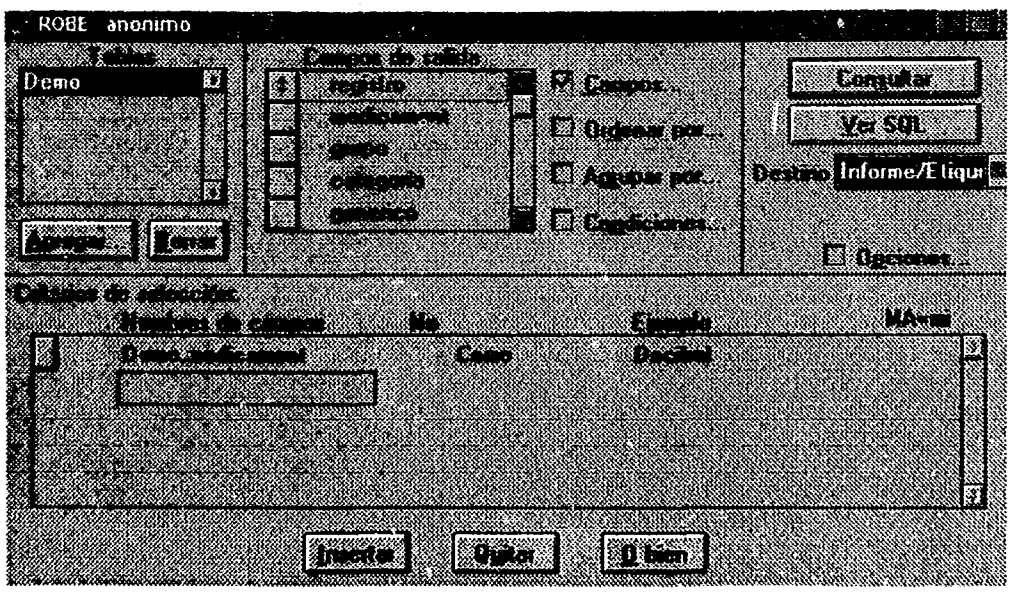
Note que no se solicita el nombre del menú, esto es por que por default se utiliza el archivo appmenu.mnx que ya se explico.

III.2.4. Otras herramientas

FoxPro contiene otras herramientas que son muy útiles para el manejo de las bases de datos. Hablaremos primero de las consultas.

Las consultas son una herramienta que ofrece FoxPro para crear filtros de información y que en los informes o etiquetas únicamente utilicemos cierta información que nos convenga. Este filtro se crea mediante condiciones que el programador va a indicar al lenguaje.

La forma de crear consultas es muy sencilla ya que el proceso de crear es el mismo que ya se visto, es decir se elige la opción nuevo del menú archivo y a continuación se elige la opción consulta y Aceptar. Al dar Clic en Aceptar aparecerá la pantalla en la cual indicaremos la tabla que utilizaremos para las consultas. Para crear consultas se nos muestra la siguiente pantalla.:



Esta pantalla se encuentra dividida en tres partes, la primera será para las tablas, la segunda para los campos y la tercera que será en la que el usuario definirá los criterios de filtro que se crearan. En la parte de las tablas se podrá utilizar una consulta de una sola tabla o de varias, cuando hacemos una consulta con varias tablas es necesario definir cual será el criterio que las enlazara, por ejemplo tenemos la tabla Demo (que es la que contiene los medicamentos) y añadimos la tabla de receta, al hacer estos en pantalla se solicitara que indiquemos por medio de que campo se enlazaran podremos elegir cualquier campo de las dos bases pero es importante elegir bien ya que de esto dependerá el resultado de la consulta Este enlace quedará formado por una expresión como la que sigue:

Receta.Medicament Como Demo.Medicament

En este caso además de estar creando el enlace también estamos creando una condición de filtro por lo que el resultado de esta consulta será mostrar los registros que cumplan con la condición de que el nombre del medicamento sea igual en las dos bases de datos, en el caso de que una de las dos tablas estuviera vacía el resultado de la consulta también seria vacío ya que no existe un solo registro que cumpla la condición.

Ahora en la parte del campo podemos indicar que campos queremos que se muestren en nuestra consulta y además también podemos definir por medio de que campo queremos que salgan ordenados, esto es muy independiente al índice que pueda tener creado la tabla. Además de que también podemos crear grupos como ya lo vimos en los informes. Para realizar cualquiera de las tareas que se acaban de mencionar únicamente basta con dar clic en el botón correspondiente y seleccionar los campos que utilizaremos. Cuando la consulta se este haciendo con dos o mas tablas, los campos por los que se podrá hacer la selección de campos será únicamente por medio de la primer tabla.

En la parte de las condiciones se tendrán que indicar tres puntos, el nombre del campo, la palabra de condición y el ejemplo. Podemos crear varias líneas de condición, es decir que podemos crear una consulta en la cual se tengan varios criterios de filtro, en este caso se irán realizando los filtros uno a uno hasta lograr el resultado de la consulta. Es importante mencionar las palabras de condición que tenemos, la propia palabra nos dice la condición que será: Como, Exactamente Como, Mayor Que, Menor Que, Entre y En. Se podrá elegir cualquiera de estas palabras para crear la condición.

Ahora bien, el resultado de la consulta puede ir a tres partes diferentes, la pantalla, un informe o a una nueva tabla, en el último caso se debe de indicar el nombre de la tabla.

Otra herramienta de FoxPro es el administrador de catálogos, esto es algo parecido a las aplicaciones que se mencionaron el punto de programación, en este caso se deben de ir añadiendo todas las aplicaciones que requiere el sistema para un buen funcionamiento, desde las tablas, hasta los programas de aplicación. El sistema se puede ejecutar desde aquí pero su ejecución no es muy confiable ya que si por error no añadimos una aplicación, el sistema marcara error y detendrá su ejecución.

FoxPro cuenta además con una serie de asistentes que permiten al usuario realizar informes, formularios, etiquetas, etc. Estos asistentes crean las aplicaciones mas sencillas por lo que no

son muy recomendables para sistemas mas complicados, pero es una buena forma de comenzar a trabajar con este software.

III.3 Ventajas y desventajas ante los demás paquetes

Existen ventajas y desventajas muy notorias en FoxPro. La principal desventaja que presenta FoxPro es que no permite crear archivos ejecutables, esto es un gran problema ya que para poder trabajar en un sistema realizado en este lenguaje es necesario tener instalado FoxPro.

Al compararse este lenguaje con los basados en DOS este tiene la ventaja de proporcionar bastantes herramientas de presentación como ya se pudo ver, además de que este lenguaje es el primero que proporciono la herramienta de formularios.

Cuando nace FoxPro para Windows fue un lenguaje muy utilizado pero debido a que las reglas de programación con las que cuenta son algo complejas, los programadores prefirieron trabajar con otros lenguajes, sobre todo que comenzaba a nacer Visual Basic en el mercado.

El hecho de comparar FoxPro con los lenguajes que veremos mas adelante, es decir los lenguajes visuales, resultaría un tanto absurdo debido a que los lenguajes visuales se caracterizan por su facilidad de manejo, pero la ventaja que ofrece este lenguaje es que todos los comandos con los que cuenta para programar están enfocados al manejo de las bases de datos.

Otra ventaja que ofrece FoxPro para Windows es que no es muy difícil aprender a manejarlo, ya que las herramientas de ayuda y los asistentes permiten que el usuario conozca el ambiente que se esta trabajando de forma sencilla.

III.3.1 Compatibilidad

La compatibilidad de FoxPro es otra de las ventajas que ofrece. La primer característica de compatibilidad que ofrece este lenguaje es que la programación es parecida a Clíper o Dbase.

En esta parte es importante mencionar a Dbase para Windows, este lenguaje es muy parecido a FoxPro, es decir la programación es muy parecida, Dbase también cuenta con asistentes, Dbase tampoco genera archivos ejecutables, etc. Pero la ventaja que ofrece FoxPro ante Dbase es precisamente la compatibilidad. FoxPro es un software creado por Microsoft, este hecho permite que este lenguaje pueda trabajar sin ningún problema con el software de Microsoft (Office o el mismo Windows), y es conocido que Microsoft es gran líder en cuestión de software.

La compatibilidad de FoxPro en cuestión a la programación es casi nula, ya que los comandos que se manejan son muy exclusivos y sobre todo por las ya mencionadas reglas, si queremos utilizar un modulo de programación en otro lenguaje tenemos que realizar bastantes cambios a los comandos por lo que no es recomendable hacer esto.

Como ya explico en los sistemas de bases de datos creados en FoxPro podemos insertar imágenes y sonidos, esta es otra posibilidad de compatibilidad, se pueden trabajar todas las imágenes de Clipart u otras galerías, es decir todos los archivos gráficos, además de todos los sonidos con extensión wav.

III.3.2 Migración.

Hablar de migración en este lenguaje es algo complicado. La migración siempre buscara la mejora de un sistema, pero como este lenguaje en cuestión de programación no es muy compatible con los demás lenguajes, la migración resultaría difícil. Se menciono en el punto anterior la similitud entre FoxPro y Dbase, por lo que entre estos dos lenguajes la migración

seria sencilla, pero migrar de FoxPro a otros lenguajes como podrían ser los visuales resulta hasta cierto punto imposible.

La versión con la que trabajamos en este capitulo fue una versión que esta completamente dedicada a Windows 3.X, después de esta versión la siguiente fue Visual FoxPro 3.0, pero aun así la migración no resulta ser muy fácil, trabajar con Visual FoxPro es trabajar en un ambiente de programación orientada a objetos, cierto es que los comandos de FoxPro siguen existiendo en la versión visual pero al trabajar con la versión mas actual de este lenguaje, este mismo sugiere cambiar la programación y trabajar mas con las nuevas herramientas que se ofrecen.

Actualmente la versión mas reciente de FoxPro es Visual FoxPro 6.0. Estas versiones ya proporcionan enlaces con HTML, Java, etc.

Concluiremos este capitulo diciendo que: FoxPro para Windows es un lenguaje que ofrece bastantes herramientas para la creación y manejo de sistemas de bases de datos, cierto que la forma de trabajar con este lenguaje es sencilla, pero la programación resulta compleja. Además no genera archivos ejecutables que en la actualidad es una herramienta muy importante. Por todo lo revisado en este capitulo sugerimos que para realizar un sistema en este lenguaje se conozcan al 100 % las reglas de programación. Pero si queremos realizar un sistema bajo el entorno de Windows elijamos otro software.

IV. Visual Basic en las bases de datos

IV.1 Conceptos Generales.

Visual Basic es uno de los lenguajes de programación mas utilizados actualmente, esto se debe a que cuenta con muchas herramientas gráficas, este lenguaje esta enfocado 100 % a la realización de aplicaciones para Windows, se basa en un proverbio que dice "vale mas una imagen que mil palabras".

La primer versión de Visual Basic fue creada para trabajar bajo el ambiente de Windows 3.X, hasta la versión 3.0 se manejo una versión de 16 bits, la versión 4.0 podía trabajar en los dos ambientes, es decir ya fuera para Windows 3.X o para Windows 95. Actualmente se cuenta con la versión 6.0 enfocada a Windows 98.

En el capitulo anterior hablamos acerca de los formularios y vimos las herramientas con las que contaba FoxPro para realizar estos, en Visual Basic los formularios se vuelven la herramienta mas directa y completa que se nos ofrece ya que contienen muchos objetos que nosotros podemos añadir a estos formularios. De hecho el primer paso para la realización de una aplicación estará basado en la planeación de las pantallas y para ello nos haremos varias preguntas: ¿Qué menús se desean? ¿Cómo va la ventana en la que se ejecuta la aplicación? ¿Cuántas ventanas mas debe haber? ¿Dónde se colocaran los botones y controles?.

Ahora bien es importante mencionar que Visual Basic es el lenguaje que permite realizar las pantallas de una manera muy sencilla y simple, de ahí que muchos programadores lo prefieran.

En este lenguaje se introduce el concepto de programación orientada a objetos, en el punto de programación hablaremos de los comandos y formas que se utilizan para trabajar en este lenguaje.

Lo anterior se menciona ya que los lenguajes que se revisaron anteriormente estaban enfocados a la creación y manejo de las bases de datos, Visual Basic es un lenguaje que no esta enfocado 100% al manejo de estos temas, aunque contiene muchas aplicaciones para las bases de datos. De hecho para crear una base de datos debemos de contar con herramientas que veremos mas adelante.

Visual Basic es un lenguaje que no solo permite crear archivos ejecutables, sino que además permite crear archivos de librería DLL que en Windows son muy utilizados. La realidad es que al trabajar con Visual Basic se puede realizar lo que el usuario se imagine ya que este lenguaje fue creado para crear, es decir se puede considerar como una aplicación que genera mas aplicaciones, y el usuario puede estar seguro de que estas aplicaciones funcionaran de la mejor manera.

Ahora regresando a la creación de sistemas de bases de datos, si bien es cierto Visual Basic no es lenguaje que este orientado a las bases de datos, pero también es cierto que con las aplicaciones con las que cuenta se puede hacer un enorme sistema de base de datos y el usuario puede estar seguro de que este sistema funcionara de la manera mas optima que se puede ver, dependiendo obviamente del programador que realice este sistema.

Para crear estos sistemas, Visual Basic cuenta con una edición profesional que será con la que nosotros trabajaremos, esta versión contiene una herramienta que se conoce como control de datos, esta herramienta trabaja casi todo el potencial que contiene Access, de hecho se recomienda que cuando se quiera hacer un sistema de bases de datos se utilice Access y Visual Basic de manera conjunta para que el sistema funcione de manera mas optima.

En este capitulo utilizaremos unos conceptos que aunque ya los conocemos en la moderna terminología de las bases de datos cambian, por ejemplo una base de datos manejada por Clipper o FoxPro era únicamente una tabla de datos, en la actualidad la base de datos no es únicamente la tabla, sino que además contiene las consultas, los índices, los informes, las

etiquetas, etc., es decir la base de datos es todo lo que nosotros hemos visto, obviamente lo fundamental en una base de datos es la tabla, puede hacer falta la consulta o el informe pero nunca puede hacer falta la tabla ya que esta contiene los datos, en Visual Basic la tabla se le denomina malla, aunque en este trabajo la seguiremos llamando tabla para evitar confusiones.

Otra parte fundamental en la creación de sistemas de bases de datos en Visual Basic es el manejo de SQL, "este lenguaje consiste en sentencias muy próximas al ingles diseñadas para seleccionar registros de una tabla de acuerdo con un criterio dado." De hecho nosotros ya trabajamos un poco con SQL en la parte de consultas del capitulo anterior.

IV.1.1 Características Técnicas.

Para poder trabajar con Visual Basic se recomiendan dos cosas: la primera de ellas es conocer el manejo de Windows y la segunda será tener nociones de programación en alguna de las versiones anteriores de Basic.

Lo anterior es muy importante ya que aunque la programación en este lenguaje cambia, se debe de tomar en cuenta que los fundamentos de programación serán los mismos, en el punto de programación se vera mas a fondo esto.

En nuestro caso, es decir para las bases de datos, se recomienda que antes de instalar Visual Basic, se verifique que en el equipo se encuentre instalado Access en alguna de las versiones actuales, de preferencia en la versión de Office 97, ya se había mencionado que para crear y manejar sistemas de las bases de datos es recomendable que Visual Basic y Access trabajen de manera conjunta.

Cuando se ocupa la versión estándar de Visual Basic es muy necesario que se tenga instalado Access, ya que esta versión no contiene la herramienta de control de datos, esta versión no es muy recomendable para crear sistemas de este tipo. En caso de utilizar la

versión profesional, no es necesario tener instalado Access pero si recomendable, esta edición cuenta con herramientas para crear y manejar las bases de datos pero todas estas herramientas se basan en Access.

Se recomienda que cuando se instala Visual Basic, se realice la instalación completa para evitar que hagan falta algunas herramientas, la instalación de este software es muy simple y amigable para el usuario por lo que cualquier usuario que conozca Windows puede realizar la instalación.

Al inicio de este punto se menciono que es importante se conozca el manejo de Windows, pero esto no solo debe de ser como un usuario habitual de Windows, es decir no suficiente con que se conozca el manejo de ventanas y del Mouse. En algunas aplicaciones de visual Basic es necesario trabajar con librerías DLL, por lo que es necesario conocer como trabajan estas en Windows, aunque no es difícil entender el funcionamiento de estas.

IV.1.1.1 Requerimientos en hardware.

Las versiones mas actuales de Visual Basic desafortunadamente requieren de un buen equipo para el desarrollo de sistemas, aunque para poder trabajar con la versión 3.0 se pueden utilizar equipos no muy sofisticados. Las características que se mencionan a continuación se refieren a un equipo para trabajar con la versión 3.0

Procesador 386 o superior.
4 Mb en memoria RAM
Monitor Color Vga
Disco duro de 200 Mb o mayor.
Unidad de disquete 3 ½
Tarjeta de vídeo Vga
Mouse y Teclado

Estas características son las básicas para trabajar con la versión 3.0, la instalación de Visual Basic requiere al menos 13 Mb de memoria libre, pero hay que tomar en cuenta que esta versión es para Windows 3.X y se debe de instalar DOS, Windows, Access y Visual Basic, parte de los archivos que se van generando cuando se crean aplicaciones.

Ahora revisemos las características básicas que se requieren para trabajar con una versión superior a la 3.0.

Procesador 486 o superior.
8 Mb en memoria RAM
Monitor Color Vga
Disco duro de 500 Mb o mayor.
Unidad de disquete 3 ½
Tarjeta de vídeo Vga
Mouse y Teclado

En este caso se toma un procesador 486, pero es mas recomendable contar con un procesador superior ya que en estas versiones se trabajara con Windows 95 y este software requiere de un buen procesador para trabajar adecuadamente. La capacidad de disco duro aumenta bastante ya que también se tiene que tener instalado Windows, Access y Visual Basic, este tamaño se toma en cuenta ya que debido a la compatibilidad que tiene Visual Basic algunas veces es necesario tener instalado otro software que se requiera según las necesidades del programador.

IV.2 Principales Herramientas.

Hablar de todas las herramientas con los que cuenta Visual Basic requeriría de muchas paginas y necesitaríamos hacer un libro dedicado a Visual Basic para mostrar todas. En este trabajo hablaremos de las herramientas pero aquellas que estén orientadas al manejo de las bases de datos.

Estas características son las básicas para trabajar con la versión 3.0, la instalación de Visual Basic requiere al menos 13 Mb de memoria libre, pero hay que tomar en cuenta que esta versión es para Windows 3.X y se debe de instalar DOS, Windows, Access y Visual Basic, parte de los archivos que se van generando cuando se crean aplicaciones.

Ahora revisemos las características básicas que se requieren para trabajar con una versión superior a la 3.0.

Procesador 486 o superior.
8 Mb en memoria RAM
Monitor Color Vga
Disco duro de 500 Mb o mayor.
Unidad de disquete 3 ½
Tarjeta de vídeo Vga
Mouse y Teclado

En este caso se toma un procesador 486, pero es mas recomendable contar con un procesador superior ya que en estas versiones se trabajara con Windows 95 y este software requiere de un buen procesador para trabajar adecuadamente. La capacidad de disco duro aumenta bastante ya que también se tiene que tener instalado Windows, Access y Visual Basic, este tamaño se toma en cuenta ya que debido a la compatibilidad que tiene Visual Basic algunas veces es necesario tener instalado otro software que se requiera según las necesidades del programador.

IV.2 Principales Herramientas.

Hablar de todas las herramientas con los que cuenta Visual Basic requeriría de muchas paginas y necesitaríamos hacer un libro dedicado a Visual Basic para mostrar todas. En este trabajo hablaremos de las herramientas pero aquellas que estén orientadas al manejo de las bases de datos.

En la edición profesional de este lenguaje básicamente existen dos grandes herramientas para los sistemas de bases de datos: el administrador de datos y el generador de informes. El administrador de datos que será como un manejador de bases de datos muy parecido a Access, y el generador de informes que permite crear no solo informes sino que también permite crear las etiquetas. La edición estándar de Visual Basic no cuenta el administrador de datos.

La ventaja que ofrecen estas dos herramientas es que son muy sencillas de manejar ya que trabajan de forma parecida a otras aplicaciones existentes para Windows.

Otra herramienta es la ayuda que proporciona Visual Basic, esta ayuda no implica solamente los comandos de programación o los posibles métodos y propiedades que ofrecen los objetos de este lenguaje, además de lo anterior contiene un tutorial que explica de una forma sencilla como se crean las principales aplicaciones en este lenguaje.

IV.2.i Manejo de Archivos

Visual Basic es un lenguaje que maneja muchos tipos de archivos, sobre todo para las aplicaciones que vamos a crear, pero por otro lado es bueno comentar que estos archivos no solo son utilizados pro Visual Basic sino que pueden ser utilizados por otras aplicaciones de Windows. Por ejemplo los archivos de base de datos son 100% compatibles con Access o Excel, las librerías que se generan pueden ser utilizadas por Windows en otras aplicaciones. Aunque es importante mencionar que estos archivos son generados por Visual Basic para utilizarlos dentro de sus propias aplicaciones.

Pero no solo genera archivos que pueden ser utilizados por otras aplicaciones de Windows sino que además genera archivos que son exclusivos de Visual Basic. Como ejemplo de lo anterior podemos mencionar los archivos de proyecto, también al crear los informes o etiquetas se genera un archivo exclusivo de Visual Basic.

Detengámonos un poco a revisar los archivos de proyecto ya que estos son los que marcan la base de toda aplicación en este lenguaje, ya que este contiene el control de todos los formularios que se vayan generando, el orden en que se utilizaran estos, etc.

Cuando se comienza a generar una aplicación en Visual Basic se van realizando los formularios que se van a utilizar para la presentación de nuestro sistema. Cada uno de estos formularios (que ya sé vera mas adelante como se realizan) es guardado en un archivo con extensión FRM. Cabe mencionar que estos formularios además de la presentación contienen también el código para el programa. En un sistema se pueden tener tantos formularios como se necesiten, pero es necesario llevar un control sobre estos. Dicho control es llevado por el proyecto, este es guardado con la extensión VBP.

Cuando se genera un archivo ejecutable o una biblioteca DLL el archivo al cual se le enlaza es el archivo de proyecto ya que este guarda toda la información y el control de todas las herramientas en el sistema.

A pesar de que los formularios y los proyectos son archivos que guardan mucha información, los tamaños de estos no son muy grandes por lo que las aplicaciones que se generan no utilizan mucho espacio en disco. Pero cuando se genera un archivo ejecutable este si requiere de mas espacio en disco.

Por otro lado en las bases de datos a pesar de que se pueden tener varias tablas dentro de una sola base, el tamaño de esta no será muy grande como se podría pensar aunque esta no solo guarde las tablas sino que también guarda los índices y las consultas.

IV.2.1.1 Tablas

A partir de este capítulo el concepto de tabla se podrá manejar también como malla. Estos por que en la moderna terminología de bases de datos se le ha llamado así.

En este lenguaje existe un detalle muy particular, ya que las tablas no se manejan de forma independiente pero si de forma individual, es decir existe una base de datos que contiene todas las tablas que vayamos a utilizar para nuestro proyecto. Estas tablas se pueden utilizar de forma individual o realizar relaciones entre estas.

Actualmente esta es la forma en la que se manejan las bases de datos, en los dos capitulos anteriores, los lenguajes manejaban las tablas de forma directa, es decir para abrir una tabla solo bastaba con utilizar el comando y abrirla, en este lenguaje no es así pero haciendo una analogía, necesitaríamos abrir primero la base de datos y posteriormente abrir la tabla.

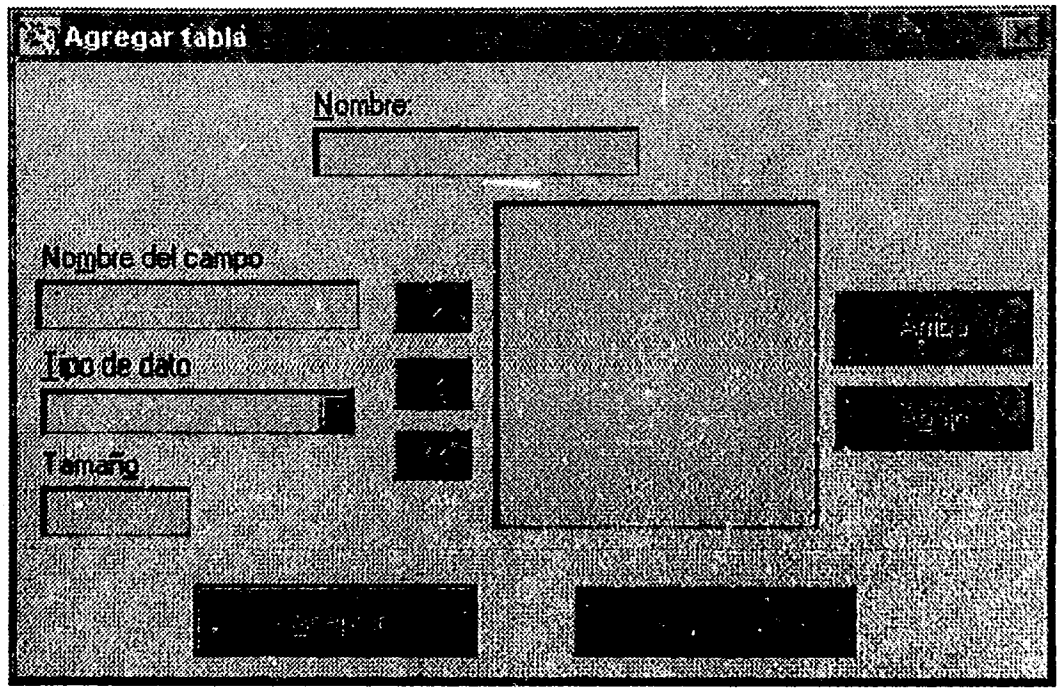
Se podría pensar que al utilizar las tablas de esta forma el sistema es mas tardado o el programador debe de utilizar mas comandos, eso hasta cierto punto puede ser verdadero pero ya veremos mas adelante que esta forma de trabajar las bases de datos simplifica muchas tareas además de que evita tener muchos archivos para un solo sistema.

En este capitulo mencionaremos como ya lo hemos hecho, la forma de manejar las tablas desde el administrador de datos y desde el punto de vista de la programación.

Como ya se había mencionado anteriormente Visual Basic provee una herramienta denominada "administrador de datos". En este se pueden realizar las tareas mas básicas que se podrían realizar en Access, es decir crear una tabla, modificarla, trabajar con los índices, etc.

La forma de trabajar en esta herramienta es muy sencilla, ya que cuenta con menús y botones que van guiando al usuario para crear las bases de datos. Lo primero que tenemos que hacer es elegir la opción Nueva Base de Datos del menú Archivo, a continuación aparecerá la pantalla en la cual teclearemos el nombre y la ubicación del archivo MDB. Cuando creamos la base de datos aparece una pantalla en la cual iniciaremos a crear o migrar tablas.

Una vez en la pantalla de la base de datos creada se dará a un clic en la opción nuevo para crear una nueva tabla. De esta forma comenzaremos a crear una tabla para la base de datos. Cuando creamos una nueva tabla nos aparece la siguiente pantalla:



En esta pantalla la opción nombre indicara el nombre de la tabla que vamos a crear, una vez que hemos tecleado el nombre teclearemos los campos que la tabla contendrá. Como en los lenguajes anteriores tenemos que indicar el nombre del campo, el tipo del campo y el tamaño. Cuando ya estamos seguros del campo que vamos a añadir, se añadirá mediante el botón con el símbolo >, esto añadirá el campo al recuadro de la derecha, lo que indica que ese campo ya pertenece a nuestra tabla.

Estos son los procesos para crear una base de datos y una tabla, como ya se menciono anteriormente se pueden crear varias tablas en la base de datos según se necesite.

Para crear las tablas en Visual Basic existen los tipos de datos tradicionales aunque en este lenguaje se conocen con otros nombres pero que son mas entendibles para el usuario, por ejemplo el tipo de campo que ahora conocemos como Carácter en Visual Basic se conoce como de tipo Texto, obviamente existen los campos numéricos, de fecha, memos, lógicos, etc.

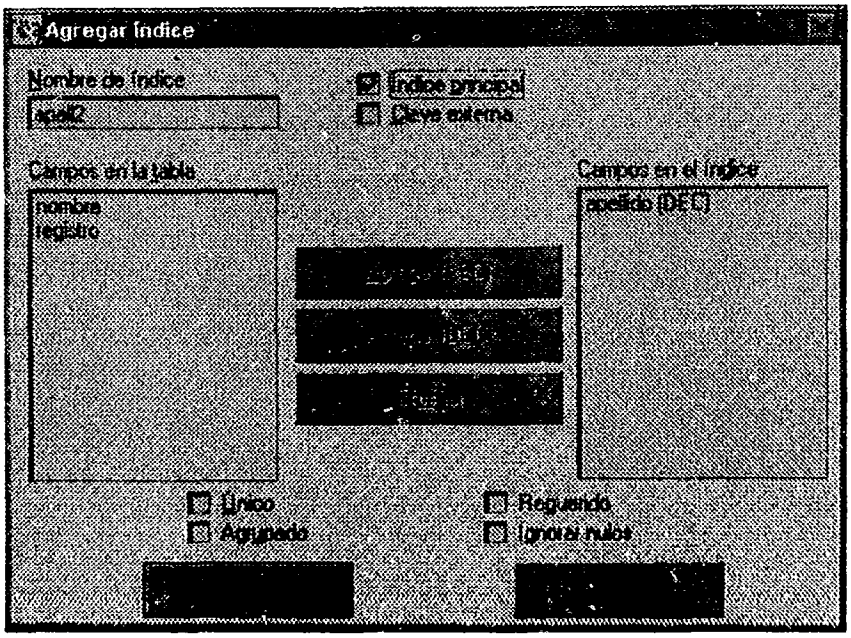
Desde este administrador, cuando se abre una base de datos y se selecciona una tabla (esto se hace seleccionando la opción Abrir en el menú archivo) aparecen iluminadas las opciones de Diseño y Abrir, la opción de Abrir será un pequeño editor con el cual podremos añadir, borrar o modificar registros.

Por otro lado la opción de Diseño permite realizar cambios a la estructura de la tabla. Además esta opción permite crear y modificar los índices que nuestra tabla requiere. Veamos esto mas afondo.

Ya sabemos que los índices son una herramienta indispensable en las tablas, en Visual Basic estos índices tienen las mismas funciones que ya hemos visto en los otros lenguajes. Pero revisemos como se trabajan.

Al elegir la opción Diseño aparece una pantalla en la que aparece la opción Indices, basta dar un Clic en esa opción para entrar a trabajar con los índices. Al igual que con las tablas podemos crear un nuevo índice o editar uno que ya se tenga, en este caso crearemos uno nuevo.

La pantalla que muestra Visual Basic después de dar clic en la opción mencionada es la siguiente:



En este caso se ha elegido el campo apellido como el campo que se utilizara como índice, el nombre de índice será Apell2, y además este será el índice primario, como ya se había revisado si vamos a tener varios índices, es necesario tener uno primario. Para crear el índice únicamente basta con escribir el nombre, seleccionar el campo y dar un clic en el botón con el símbolo Agregar ASC o Agregar DEC. Después de ello basta seleccionar las opciones que el índice llevara y dar Clic en la opción Aceptar.

Además de seleccionar un índice primario, se puede indicar que este sea único. Al utilizar un índice único el motor de Access se utiliza para determinar si alguien esta intentando introducir datos duplicados.

Cabe mencionar en esta parte que los nombres de las tablas no se ajustan a las convenciones normales de nombres de archivos del DOS, esto es puesto a que el motor de Access almacena toda la información dentro del archivo de bases de datos. Esto es una gran diferencia respecto a los demás manejadores de bases de datos.

Los índices van a ser independientes para cada una de las tablas pero podemos utilizarlos para relacionar tablas y recordemos que para ello el campo de los índices debe de ser el mismo.

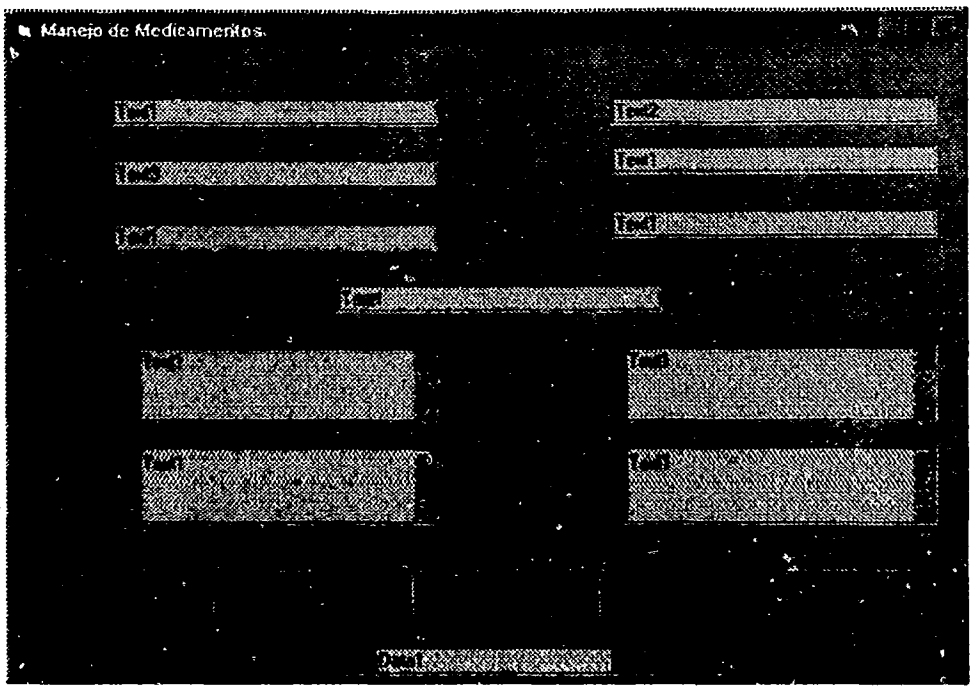
Otro concepto dentro de las bases de datos son las claves primarias. Un tipo especial de índice primario se deriva de este concepto. Las claves primarias son campos especiales cuyos contenidos deben ser exclusivos de una tabla. Las claves primarias son necesarias cuando se necesita concatenar (unir) la información de dos tablas diferentes. Se utiliza la clave primaria en un campo cuando se quiere unir la tabla de una base de datos con otra tabla diferente.

Ahora veamos como se trabaja con las bases de datos por medio de la programación. Como Visual Basic no es un lenguaje 100% dedicado al manejo de bases de datos no existen muchos comandos para el manejo de las mismas pero es importante mencionar que con pocas herramientas se pueden hacer muchas cosas sobre todo para la generación de sistemas de bases de datos.

Cuando queremos trabajar con una base de datos, es necesario agregar a nuestro formulario el objeto DataControl, mas adelante mostraremos una pantalla que lo contenga para verlo en forma gráfica.

El DataControl designa la base de datos y la tabla que se utilizara, los demás objetos que se añadan como listas y cuadros de texto podrán utilizar esta base de datos con solo hacer referencia al DataControl. El data control proporciona herramientas para navegar en una base de datos, por ejemplo ir al final, ir a un registro anterior, etc. Pero para poder crear otras aplicaciones con la base de datos, enlazaremos el DataControl con una propiedad conocida como Recordset. Este permitirá realizar bastantes tareas relacionadas con una tabla, por ejemplo: localizar un registro, añadir o borrar registros, etc.

En el ejemplo que se presenta a continuación veremos una pantalla utilizada para nuestro sistema y mostraremos el código asociado a algunos botones para demostrar lo que se menciono en los párrafos anteriores.



El objeto que se encuentra en la parte inferior de la pantalla es el DataControl, en este caso se llama Data1, observe que cuenta con flechas para poder visualizar los registros, los cuadros de texto que se muestran en esta pantalla están relacionados al DataControl por lo que es en estos en los que se muestra la información, en el sistema final puede verificar las propiedades de estos para comprender mejor lo anterior.

Ahora se mostrara parte del código asociado al botón Nuevo y al botón Eliminar:

```
Data1.Recordset.AddNew
Data1.Caption = "Nuevo registro"
Data1.Enabled = False
cNew.Enabled = False
```

Las líneas anteriores pertenecen al botón Nuevo para el suceso Clic, en este caso se puede observar que el DataControl es asociados al Recordset y se indica que se va a añadir un nuevo registro. En ese momento el mensaje del DataControl será Nuevo registro, y el DataControl será deshabilitado en lo que el usuario teclea el contenido del nuevo registro en los cuadros de texto.

```
Data1.Recordset.Delete
Data1.Recordset.MoveNext
```

Estas dos líneas son parte del suceso Clic del botón eliminar, sucede lo mismo que para el ejemplo anterior, es decir se unen el DataControl y el Recordset para borrar el registro que se encuentre visualizado en ese momento. En este caso previamente se pide una confirmación de que realmente se quiere borrar el registro. Consulte el código del sistema final para aclarar dudas.

Independiente al DataControl se pueden utilizar otros comandos para poder abrir tablas de bases de datos, tal como puede ser OpenDatabase u OpenTable, la sintaxis de estos comandos no la tomaremos en este trabajo ya que se considera mas sencillo trabajar con el DataControl, aunque para algunas aplicaciones, sobre todo cuando no se cuenta con la versión profesional de Visual Basic, es mas conveniente utilizar estos comandos. Puede verificar la sintaxis de estos comandos en la misma ayuda de Visual Basic.

IV.2.1.2 Etiquetas

Las etiquetas y los informes en Visual Basic no tienen mucha importancia, y un ejemplo de esto es que la versión Estándar de Visual Basic no contiene ninguna herramienta para la creación de informes y etiquetas solo por medio de la programación.

Por otro lado la versión profesional de visual Basic contiene una herramienta que se conoce como Crystal Reports. Esta herramienta se podría considerar como un software externo a

Visual Basic, pero no es así, como ya se menciono la versión profesional de Visual Basic proporciona herramientas para el manejo de las bases de datos.

Se puede accesar a Crystal Reports de dos formas, la primera de ellas es eligiendo esta herramienta desde la barra de tareas de Windows, esta opción se encuentra dentro del grupo de Visual Basic 4.0, por otro lado una vez que se esta dentro de Visual Basic, dentro del menú Complementos se encuentra la opción Generación de Informes que nos lleva directamente a Crystal Reports.

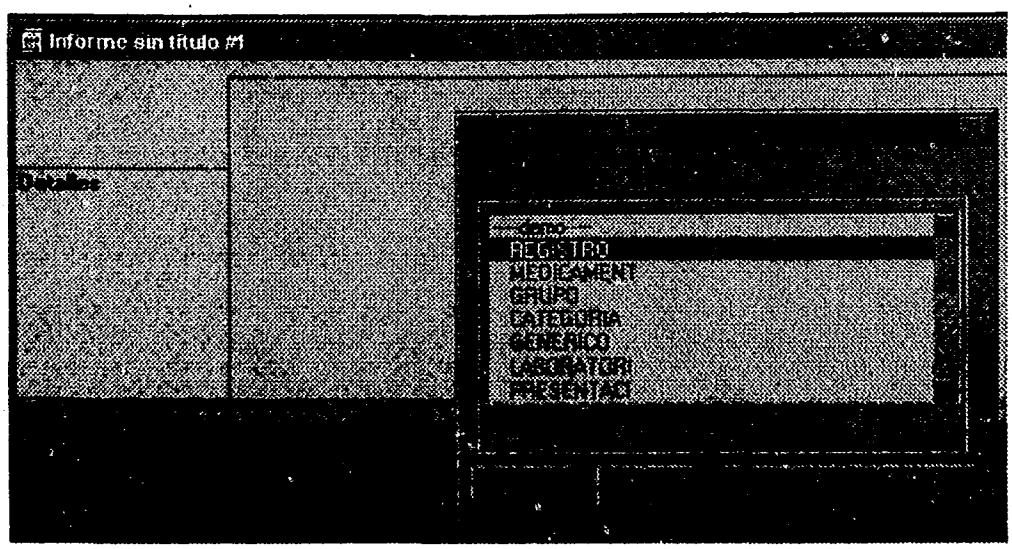
Al trabajar con Crystal Reports podremos generar informes y etiquetas de forma sencilla. En esta sección nos dedicaremos a explicar como crear etiquetas, aunque los informes se crean de manera muy similar.

Una vez que accesamos a Crystal Reports elegimos la opción Nuevo del menú Archivo y seleccionar la opción informe. Aparecerá una pantalla en la cual tenemos que dar Clic en la opción etiqueta y elegir la opción Aceptar. Una vez que hemos hecho esto se nos mostrara la pantalla de Abrir en la que se seleccionara la base de datos que vamos a utilizar para la etiqueta. A continuación aparecerá una pantalla en la que seleccionaremos el tamaño de la etiqueta y el tamaño de la pagina así como los márgenes de la misma.

En esta pantalla se recomienda que no se cambie el tipo de etiqueta que por default muestra Visual Basic, esto es debido a que, aunque se cuenta con otros tipos estos ya se encuentran establecidos y no es posible modificarlos y si tomamos el tipo estándar podremos acoplar este a las necesidades del usuario.

El tamaño también se puede dejar en 0 centímetros, ya que este se podrá modificar conforme se vayan añadiendo campos a la etiqueta. Posiblemente lo que podemos modificar es el margen de la pagina. En resumen es recomendable que se Acepten os tamaños y tipos que aparecen establecidos, las modificaciones serán hechas al momento de insertar los campos.

Después de seleccionar tamaño y tipo de etiqueta se nos mostrara la siguiente pantalla:



En el recuadro se puede ver los campos de la base de datos, cabe mencionar que se muestran los campos de todas las tablas que pudiera contener la base de datos.

Cuando se quiere insertar un campo a la etiqueta, se dará Doble Clic sobre el campo del recuadro y un clic en el área de la etiqueta que será la que esta rodeada por las lineas. El área del campo aparecerá por medio de X, es decir si el campo tiene una tamaño de 10 al insertarlo aparece como XXXXXXXXXX.

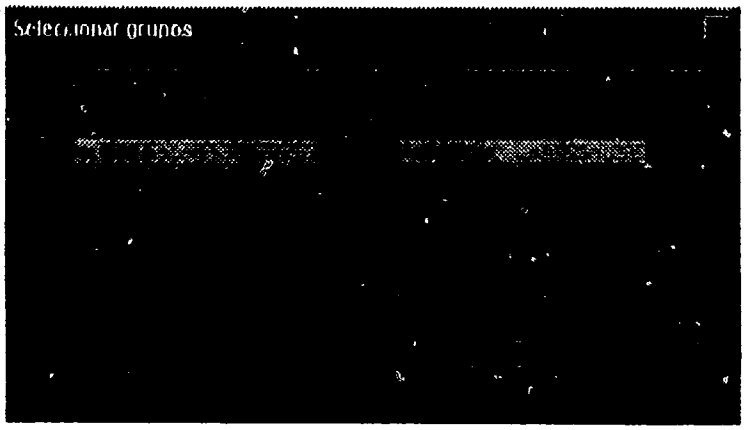
Ya que se inserto el campo solo basta acomodarlo y escribir el titulo que quiere que aparezca en el campo. Este proceso se tiene que hacer para todos los campos que se quiera introducir a la etiqueta. Una vez que se terminado de insertar todos los campos se elige la opción Terminar del recuadro.

En caso de que se quiera ampliar el área para la etiqueta basta con dar Enter como si fuera un archivo de texto y así se ampliara esta área.

Otra herramienta que nos ofrece Crystal Reports es la crear grupos, es decir informes basados en grupos, aunque en Visual Basic no funcionan igual que en los lenguajes anteriores, ya que en este caso al determinar un grupo únicamente se imprimirán los registros que cumplan con una condición determinada.

Por ejemplo si creamos un grupo en el cual se tomen los registros que cumplan con que el Grupo = Antibióticos, únicamente imprimirá los registros que cumplan con esta condición y ningún otro.

Para crear los grupos basta con seleccionar un campo de los añadidos a la etiqueta y seleccionar la opción Seleccionar Grupos del merú Informe. En este aparecerá la siguiente pantalla:



En esta pantalla se puede ver dos recuadros, en el primero pondremos la palabra de condición que eligiremos de entre una lista que proporciona Visual Basic, en el otro recuadro podemos teclear el dato para que cumpla la condición o dar un Clic en la opción Examinar Datos de Campo. Con la cual se nos mostrara una lista con los datos que contiene la base con relación a este campo.

Depende del programador las etiquetas que se van a generar en el caso del trabajo que estamos haciendo puede checar el sistema final para ver un ejemplo detallado de las etiquetas que se generaron.

IV.2.1.3 Informes

Los informes se generan de igual forma con Cristal Reports, la forma de generarlos es la misma que para las etiquetas, es decir podemos crear un informe seleccionando la opción Nuevo del Menú archivo, en este caso únicamente hay que seleccionar la base de datos que vamos a utilizar para la creación del informe. El tamaño se establecerá según el tipo de pagina que se encuentre determinado en la impresora.

Se mostrara una pantalla muy parecida a la que se muestra para crear etiqueta, es decir se mostrara un recuadro con el nombre de los campos de todas las tablas que pertenezcan a la base de datos abierta.

En el caso de las etiquetas únicamente aparece un área denominada Detalles, en el caso de las etiquetas aparecen tres áreas. En la parte superior aparece el área de encabezado, después el área de detalles y a continuación el área de pie de pagina. El área de encabezado y el área de pie de pagina pueden ser utilizados por el usuario según sus necesidades. Las tres áreas permiten la inserción de campos y de texto en general.

Se añaden los campos de la forma en que ya vimos en la sección de etiquetas, en este caso al añadir el campo aparece en la pantalla un nombre para este campo, es decir a diferencia de que en las etiquetas únicamente aparecía el área del campo compuesta por X, en los informes además de esta área aparece el nombre que tiene el campo en la tabla, este nombre se puede modificar según las necesidades del usuario, la modificación se realiza como se fuera un editor de texto. De igual forma para agrandar el área de detalles del informe se hace por medio de la tecla Enter.

Se pueden realizar también los procesos mencionados en las etiquetas como son crear grupos, y además se puede añadir otra base de datos para crear un informe mas completo, en este caso elegimos la opción Agregar Base de Datos del menú Base de Datos y nos muestra una pantalla en la cual podemos abrir una nueva base de datos.

La presentación tanto de informes como de etiquetas es responsabilidad del programador pero de acuerdo a las necesidades del sistema que se esta realizando.

IV.2.2. Manejo de Presentaciones

Hablar de presentaciones en Visual Basic es un tema muy sencillo, pero la responsabilidad del programador en este parte es muy importante. Debido a que las aplicaciones que se generan son para Windows es muy importante que la presentación de nuestra aplicación sea buena y acorde al ambiente en el que estamos trabajando y Visual Basic ofrece muy buenas herramientas para la presentación.

En este punto hablaremos mas que nada de las propiedades y presentación de la aplicación en general, en el punto de Gráficos y Botones hablaremos mas a detalle de las herramientas de presentación que se ofrecen para estos.

La generación de un sistema siempre inicia con la creación de un formulario principal y de los formularios auxiliares que se ocuparan. Por ello la importancia que tiene que desde su inicio el sistema se comienza a mostrar de buena manera.

Comenzaremos por explicar como se utilizan las propiedades. Al crear un formulario o añadir un objeto, seleccionaremos el objeto y utilizaremos la tecla F4 para poder ver las propiedades del objeto o en el caso de que no tengamos ningún objeto seleccionado se mostraran las propiedades para el fondo del formulario. La pantalla de propiedades se divide en dos partes, del lado derecho se encuentra el nombre de la propiedad y del lado izquierdo

el valor. Si queremos cambiar el valor de la propiedad solo basta con posicionarnos en ella y cambiar este valor ya sea con el Mouse o con el teclado.

Para la presentación del formulario utilizaremos 4 propiedades, aunque existen mas mostraremos las que mas se utilizan:

- Appearance: Establece el estilo de dibujo de los controles que se añadirán a este formulario, puede tener dos valores:
  0 Liso. Dibuja los controles sin efectos visuales.
  1 3D. Dibuja los controles con efectos tridimensionales.
- BackColor. Establece el color de fondo del formulario En la parte que tenemos designada para cambiar el valor, el color es representado por un numero Hexadecimal. Al dar Clic en este numero se muestra en pantalla una paleta de colores en la cual podemos escoger algún color dando clic sobre el.
- Font. Permite cambiar el tipo de letra que se utilizara en todo el formulario. Al dar clic en esta opción se muestra una pantalla en la cual podemos seleccionar el tipo y tamaño de letra.
- ForeColor. Permite designar un color para el primer plano, es decir el texto y los gráficos que se muestren en el formulario. Trabaja de igual manera que BackColor.

Es importante mencionar que cada uno de los formularios que se puedan contener en una aplicación son independientes unos de otros por lo que es necesario determinar la presentación para cada uno de estos según las necesidades del usuario. Se recomienda que los formularios se vayan realizando en el orden en el cual se manejaran durante la ejecución, ya que por default el formulario principal será el que se realizo primero. Aunque es posible modificar el orden de los formularios, es decir que el mismo programador pueda definir cual será el formulario principal, es una buena costumbre ir generando el sistema en un orden adecuado.

Como propiedades también se debe de asignar el nombre del formulario y el mensaje que aparecerá en parte superior de la ventana. Para el nombre se ocupara la propiedad Name y para el titulo de la ventana se ocupara la propiedad Caption.

Otra parte importante de los formularios es el tamaño y la posición en la que se muestra en la pantalla. Cuando estamos generando una aplicación se muestra en pantalla una posición y un tamaño para el formulario, cuando se ejecuta, la posición y el tamaño es la misma que cuando se genero. Por ello es importante que se establezca desde un principio estos dos parámetros. Como el formulario es una ventana mas el tamaño y la posición se asignan de la misma manera que una ventana de Windows, es decir con el Mouse.

Cuando se ejecuta una aplicación, se muestra en pantalla todos los formularios que se generaron, obviamente existe uno que es el principal y será el que este activo al principio, y que será encargado de llamar a los otros. Para poder llamar a un formulario dentro de otro y poder tenerlo como activo se utiliza el comando:

<Nombre_del_formulario>.Show 1

Este formulario permanecerá activo hasta que se mande llamar a otro desde un modulo.

Los cambios a las propiedades de presentación se pueden ejecutar de forma manual al crear la aplicación o también se pueden ir modificando según la secuencia del programa en el código correspondiente a cada modulo.

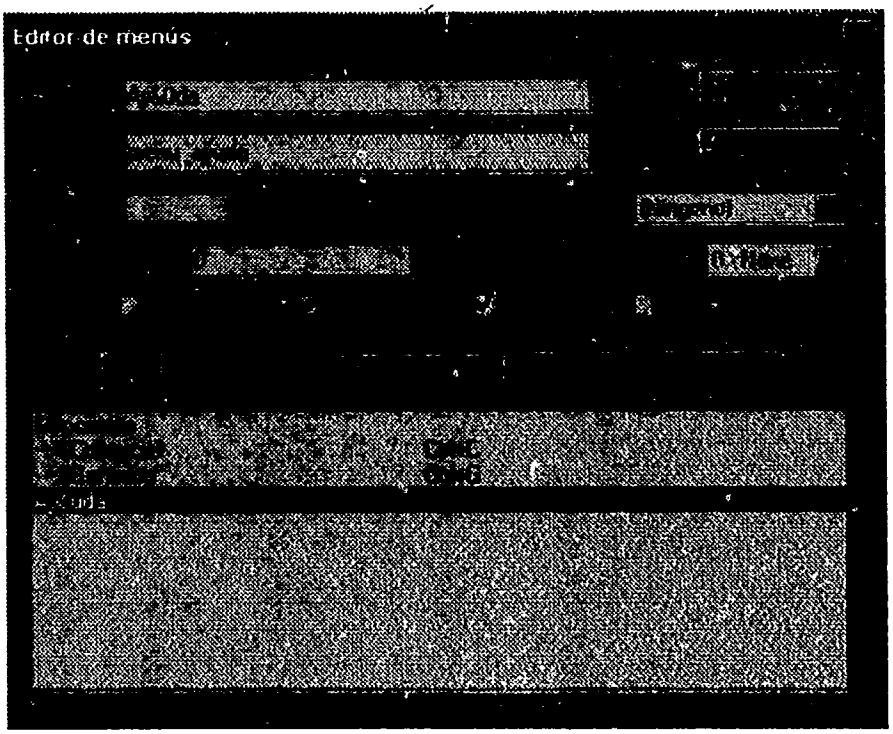
IV.2.2.1. Menús

Debido a que el ambiente que se maneja en Visual Basic es totalmente gráfico, muchos de los usuarios se olvidan casi por completo del menú. Aunque como sabemos algunos de los usuario están tan acostumbrado a este ambiente que se recomienda que cualquier sistema

que se genere cuente con un menú. Sobre todo por que los menús son parte esencial en una ventana de Windows y al ejecutar nuestro sistema será por medio de ventanas.

Lo primero que tenemos que tomar en cuenta es que cada formulario que se crea debe de contener su propio menú, es decir que se debe de crear un menú para cada uno de los formularios que tienen el proyecto.

Para poder crear un menú, tenemos que estar posicionados en el formulario. Seleccionamos la opción generación de menús del menú formulario. Aparecerá la pantalla que se muestra a continuación.



En esta pantalla tenemos que teclear los datos que llevara el menú. Para cada una de las opciones debemos que teclear dos datos obligatoriamente, estos datos serán Caption y Name. Cabe mencionar que cada una de las opciones del menú también se considera un objeto, por lo que Caption y Name son dos propiedades de este objeto. Note en la figura

que en el recuadro inferior se muestran las opciones que tendrá el menú. El nivel principal, es decir el menú horizontal, serán las opciones que se tienen hasta la izquierda del recuadro, note que cada una de esas opciones contiene debajo varias opciones mas pero en un nivel mas abajo, estas las podemos identificar porque inician con 4 puntos, esto indica que se encuentran en el segundo nivel, el segundo nivel serán las opciones del menú principal, es decir los menús verticales.

Entre mas puntos tenga una opción al principio es el nivel en el que se encuentra. En visual Basic se pueden generar opciones de menú hasta 6 niveles.

Para poder hacer que una opción cambie de un nivel a otro basta con dar Clic en las flechas de izquierda y derecha que se muestran en la pantalla que ya se mostró. Además de teclear la opción del menú note que se encuentra un recuadro con la opción ShotCut, esta le permitirá añadir una opciones de tecla rápida para cada una de las opciones del menú. Se muestran varias combinaciones de tecla rápida de las cuales puede escoger la que mejor se adecue a su menú. En el recuadro d inferior se muestra la opción de menú con su correspondiente combinación de tecla rápida. Además note también que a algunas opciones se han tecleado con el símbolo &, este indica que la letra que sigue al símbolo aparecerá subrayada para poder accesar a esta opción por medio dela tecla ALT y la letra correspondiente.

Como ya se menciono cada una de las opciones del menú serán un objeto y por lo tanto contienen eventos, dentro de estos eventos se teclearan los módulos de programa que se ejecutaran cuando se seleccione alguna opción. Para entrar a los eventos solo basta crear el menú, una vez estando en el formulario, este aparecerá con el menú correspondiente. Selección la opción a la que le va a agregar código, en la pantalla aparecerá el Evento Clic de esa opción, teclee el código correspondiente en esa pantalla.

IV.2.2.2. Gráficos y botones

En esta sección seguiremos hablando un poco acerca de las propiedades de los botones y como podemos añadir gráficos a los formularios.

En la figura siguiente se muestra todos los objetos que podemos añadir a nuestro formulario.:

Para poder insertar uno de estos objetos en el formulario basta con dar doble clic sobre el objeto y aparecerá dentro del formulario, es decisión del programador la posición y el tamaño que este objeto tendrá, cabe mencionar que el objeto se puede manipular como un dibujo dentro de cualquier aplicación de Windows, es decir se puede modificar su posición arrastrando el objeto hacia otro lugar y el tamaño se puede cambiar con el Mouse. Aunque existen propiedades para determinar la posición y el tamaño del objeto es preferible hacerlo con el Mouse.

Como ya se menciono cada uno de los objetos aquí mostrados contiene varias propiedades que sirven para poder determinar los valores con los cuales se iniciara un objeto en la ejecución de una aplicación. Cada una de las propiedades cuenta con valor por Default. Pero casi todos los valores se pueden cambiar.

A continuación se mostraran las propiedades que se recomienda cambiar en todos los objetos.

Name Esta propiedad permite asignar un nombre al objeto para poder hacer referencia a el en algún modulo de programa.
Caption. Esta propiedad asigna el mensaje con el que será mostrado el objeto. Por ejemplo un botón de comando necesita un mensaje como Aceptar, Cancelar, etc.
Visible: Permite definir si el objeto será visible o no. Contiene dos valores True que inicia que el objeto será visible y False, que indica que el objeto no estará visible en la ejecución.

Aunque existen mas propiedades estas dependerán de la aplicación que se quiera hacer por lo que no se explicaran aquí, pero es importante mencionar que a excepción de BackColor, las demás propiedades mostradas en la sección de Presentaciones se pueden utilizar para cada uno de los objetos.

Para insertar dibujos a los formularios basta con añadir a este el objeto de inserción de gráficos, al insertar este podemos seleccionar la propiedad Picture, una vez hecho esto aparece una pantalla en la cual le indicaremos el nombre del archivo gráfico que se va a insertar. Además Visual Basic proporciona una galería de archivos gráficos que pueden ser utilizados en las aplicaciones que se generen. La galería ofrece una serie de archivos metafile, este tipo de archivos es muy recomendado ya que a pesar del tamaño muy dificilmente se distorsionan, es decir que el dibujo puede estar de cualquier tamaño en el formulario y siempre se vera bien. Por lo anterior es recomendable utilizar estos archivos para anexarlos a la presentación de nuestro sistema.

IV.2.3. Programación

El tema de programación en este lenguaje es muy importante ya que de este partiremos como una introducción para los demás lenguajes que analizaremos en este trabajo. Por lo que revisaremos primero como es la programación en general y posteriormente revisaremos los comandos mas simples que se utilizan en Visual Basic.

Visual Basic hace mas sencilla la creación de grandes programas mediante las modernas técnicas de programación modular. Esto significa que se puede dividir un programa en módulos mas sencillos de manejar y por tanto menos sensibles a los errores. Idealmente los módulos realizan una sola tarea y tienen una interfaz bien definida con el resto del programa por lo que puede ser codificado y verificado independientemente.

En los capítulos anteriores revisamos dos lenguajes que en cierta forma también trabajaban con programación modular, ya que cada una de las tareas que realizaban se referían a un modulo en particular. Pero en Visual Basic esto es mas sencillo ya que cada uno de los módulos en realidad pertenecera a cada uno de los objetos que contiene el formulario. Es decir, en el formulario se anexan varios objetos que al ser referenciados realizan una tarea, esta tarea obviamente es una pequeña parte de código de programa.

Al realizar esto los módulos son cortos y se dedican exclusivamente a una tarea especifica, esto es porque cada uno de los objetos contiene a su vez sucesos, es decir las diferentes tareas que se pueden realizar con un objeto. Por ejemplo con un botón, el usuario puede dar un clic, un doble clic, etc. cada una de estas actividades es un suceso y el programador será el encargado de escribir el código para cada uno de estos sucesos, por lo que un solo objeto puede contener varios módulos de programa.

Al tener los programas de esta forma la ejecución del mismo resulta muy simple para Visual Basic. Revisemos estos tres pasos para ver como se hace esto:

1. Visual Basic supervisa las ventanas y los controles de cada ventana para todos los sucesos que cada control puede reconocer (movimientos de ratón, clics, etc.).
2. Cuando Visual Basic reconoce un suceso, examina la aplicación para comprobar si se ha escrito un procedimiento para ese suceso.
3. Si se ha escrito un procedimiento para ese suceso. Visual Basic ejecuta el código del procedimiento y vuelve al paso 1.
4. Si no hay escrito un procedimiento de suceso. Visual Basic espera al siguiente suceso y vuelve al paso 1.

Al realizar la ejecución de los programas de esta forma es mas fácil identificar donde podemos tener errores y examinar la forma en que se esta ejecutando nuestro sistema.

Los programas en los lenguajes de programación convencionales se ejecutan de arriba abajo. En los antiguos lenguajes la ejecución comenzaba en la primera fila y se desplaza según el flujo del programa a las distintas partes según se necesite (ejemplo de este caso seria Clipper). Un programa en Visual Basic funciona de otro modo. El núcleo de un programa en visual Basic es un conjunto de diferentes partes de código que son activadas por los sucesos que se les ha indicado que reconozcan. Esto tiene gran importancia porque ahora en lugar de diseñar un programa que haga lo que el programador piense que debe hacer, el usuario tiene el control.

Ya hemos analizado como funciona y como se trabaja con la programación en los lenguajes visuales pero ahora veremos como se codifica en este lenguaje.

La programación que se realiza en los lenguajes visuales se conoce como programación orientada a objetos. En sus inicios esta programación llamo la atención de los programadores aunque resultaba un tanto difícil. Con el tiempo y a través de esta técnica de programación aparecieron los lenguajes visuales.

Entre los lenguajes que nacieron fue visual Basic el primero que demostró gran poder para realizar aplicaciones sobre todo para Windows. Pero sobre todo resalto mucho que la programación en este tipo de lenguajes resultaba muy sencilla.

En el punto de Presentaciones, al hablar de formulario se menciono que cada uno de los objetos contiene una serie de propiedades y sucesos. En los sucesos como ya se menciono se escribirá el código correspondiente. Lo importante de esto son las propiedades. Cada uno de los objetos contendrá un nombre diferente para cada formulario, es decir si yo inserto una caja de texto a esta se le asignara un nombre mediante la propiedad Name, supongamos que le asignamos el nombre de CajaText, para las cajas de texto existen varias propiedades, entre ellas esta una que indica el contenido de la caja de texto, esta propiedad se conoce como Text, si en alguna parte del código yo necesito hacer referencia al contenido de la caja de texto, ya sea para consulta o modificación, el código que se ocupara será CajaText.Text=<contenido>, al hacer esto hago referencia al contenido de la caja de texto, si yo quisiera cambiar el contenido únicamente tendrá que teclear lo mismo pero con un contenido diferente.

Note en este ejemplo que no se esta utilizando un comando en particular sino que únicamente se esta haciendo referencia al objeto (que se llama CajaText) y a la propiedad que queremos utilizar (en el ejemplo es la propiedad Text). Lo mismo sucede con cada uno de los objetos que se pueden insertar.
Para el caso de los sistemas de bases de datos, es de forma muy parecida ya que algunos de los objetos contienen propiedades que referencian a bases de datos. Por ejemplo en el punto de las tablas vimos como se puede buscar un registro en particular abriendo una base de datos.

Como en cualquier lenguaje existen comandos que permiten realizar otras actividades, por ejemplo el comando MsgBox "mensaje" que muestra en pantalla un recuadro con el mensaje que nosotros queremos y la opción Aceptar. En este capitulo no nos detendremos a revisar

muchos comandos ya que como se ha menciona la finalidad del trabajo no es mostrar un manual de visual Basic sino de varios lenguajes.

Como la programación cambia un poco revisaremos en el siguiente cuadro la forma de generar bucles o ciclos y la forma de trabajar con las condiciones.

Bucles	Bloques de Condición

Do While i <= 100
Print i
I = i + 1
Loop
	If Var1 > 0 And Var2 > 0
	Var3 = Var2 / Var1
	Else
	Var3 = 0
	End If

Como se puede ver en el cuadro la forma de realizar estas dos tareas es muy parecida a la revisada en los dos capítulos anteriores.

Además Visual Basic cuenta con ayuda para revisar el código rápidamente, este lenguaje proporciona:

- Verificación automática de tareas
- Utilidades para depurar código.

Estas herramientas trabajan de manera automática, ya que al estar tecleando las instrucciones automáticamente se va revisando la sintaxis y mostrar un mensaje de error cuando encuentre alguna falla dentro del programa que estamos codificando.

Por ultimo mencionaremos la forma en que se declaran las variables que se puede utilizar en Visual Basic. El comando para declarar variables es el comando DIM (una de las palabras Global o Static) y se utiliza de la siguiente manera:

DIM <variable> AS <Tipo_de_variable>

El tipo de variable puede ser: Integer (entera), Long (Entero Largo), Single (Real), Double (Real Doble), String (Cadena).

Además de este tipo también existen otros para las bases de datos como son: Table, Database, con los cuales indicaremos que la variable se referira a un nombre de tabla o de base de datos.

Las palabras Global o Static cumplen la misma función que la palabra Public en Clipper , es decir que serán utilizadas en varios módulos del programa, ya que al declararlas con DIM únicamente podrán ser utilizadas por el modulo en el que estén declaradas.

IV.2.4. Otras herramientas

Visual Basic cuenta con muchas mas herramientas de las que se han mostrado en este trabajo pero no son enfocadas al manejo de bases de datos. En esta parte mencionaremos algunas herramientas que pueden ser utilizadas para trabajar con los sistemas de bases de datos aunque depende del programador el uso de estas.

Para poder facilitar la generación de formularios, Visual Basic ofrece una herramienta conocida como "Diseñador de formularios de datos", para poder accesar a esta herramienta basta con seleccionar del menú Complementos. Para poder generar un formulario por medio de esta herramienta basta con indicarle el nombre de la base de datos y los campos que se añadirán al formulario. También tenemos que indicar el nombre del formulario. Estos datos son ingresados por el programador en una pantalla que muestra este diseñador. Al dar un Clic en Aceptar se genera en pantalla un formulario muy sencillo pero muy practico sobre todo para la visualización y manejo de los registros.

La herramienta anterior es recomendada solo para aplicaciones sencillas, pero se puede utilizar para generar un formulario y después modificarlo de acuerdo a las necesidades del usuario. Para ello podemos utilizar las herramientas presentadas en este capitulo.

En la sección de tablas de este capitulo hablamos del administrador de datos, se menciono como trabajar con las tablas desde esta herramienta, pero además también nos permite crear relaciones entre las tablas, cabe recordar que para poder hacer relaciones entre tablas están deben de contar con un campo índice en común para ambas. Dentro del administrador de datos y una vez que abrimos una base de datos, basta con seleccionar la opción Relaciones y nos aparecerá una pantalla en la cual indicaremos las tablas que vamos a relacionar y el campo en común a las dos. Las relaciones como ya se menciono en el capitulo 1 son muy importantes para evitar tener tablas con una gran cantidad de campos. En nuestro sistema no utilizamos relaciones ya que aunque las tablas trabajan de manera conjunta son muy independientes unas de otras.

En el administrador de datos también se pueden ejecutar y guardar algunas instrucciones de SQL. Estas instrucciones son utilizadas sobre todo para consultas. Para poder utilizar estas instrucciones se requiere que el programador tenga conocimientos sobre SQL. Para teclear y ejecutar estas funciones es necesario tener abierta una base de datos, en la parte inferior de la pantalla del administrador se cuenta con un área exclusiva para instrucciones SQL.

Además de las herramientas presentadas anteriormente se cuenta también con los objetos OLE. Estos objetos son muy utilizados para aplicaciones de Windows. La vinculación e incrustación de objetos (OLE) es una tecnología que permite que los programadores de aplicaciones basadas en Windows creen aplicaciones que puedan presentar datos de muchas diferentes aplicaciones y permite que los usuarios editen dichos datos desde la aplicación con la que fueron creados. En algunos casos, el usuario puede incluso editar los datos desde dentro de la aplicación Visual Basic. Estos objetos son muy prácticos para crear aplicaciones que requieran de utilizar todas las herramientas que el software de Windows nos proporciona, por ejemplo podemos añadir una tabla de Excel y un gráfico del mismo.

IV.3 Ventajas y desventajas ante los demás paquetes

Si comparamos a Visual Basic con los lenguajes mencionados en los capítulos II y IV, resultaría obvio que Visual Basic es mejor a ellos. Visual Basic no solo es un lenguaje muy completo sino que además es muy sencillo trabajar con el.

El ambiente que Visual Basic nos ofrece es muy amigable para el programador, no solo en cuestión de presentación sino que también en la programación. Ya se menciono en una sección que la programación se hace por pequeños módulos en cada una de las secciones de un objeto, esto facilita mucho la detección de errores sobre todo de sintaxis. La ventaja que ofrece Visual Basic en este caso es que va revisando la sintaxis desde el momento en el que se va tecleando el código y no espera a ser compilado o ejecutado para poder presentar los errores.

En la versión 5.0 cuando se va codificando alguna instrucción el sistema va mostrando las diferentes opciones que tiene esa instrucción por ejemplo, cuando estamos declarando una variable tecleamos DIM <variable> AS <tipo>, después de teclear la palabra AS, visual Basic muestra en ese renglón una lista de los diferentes tipos de variables que se manejan. Esto es una gran ayuda para el programador no solo por el tiempo que evita estar tecleando sino que los errores de sintaxis disminuyen de gran forma.

Tal vez se podría pensar que Visual Basic no ofrece muchos comandos para el manejo de las bases de datos como la hace FoxPro para Windows, pero como ya vimos en este capitulo no es así, si bien es cierto que este lenguaje no es enfocado directamente a las bases de datos también es cierto que cuenta con muchas herramientas para hacer un gran sistema de base de datos sin muchos problemas. La gran capacidad que tiene Visual Basic para manejar bases de datos se puede comparar con los mejores manejadores de bases de datos y este lenguaje saldría bien librado.

Tal vez la única desventaja que pudiera tener Visual Basic son los requerimientos en hardware que tiene, ya que para poder generar y trabajar con un buen sistema de bases de datos requiere de un buen equipo de computo.

En los siguientes capítulos revisaremos dos lenguajes parecidos a Visual Basic, en estos capítulos se ira haciendo mención de las diferencias que se tienen con Visual Basic y cual de los tres es mejor para trabajar con aplicaciones para Windows.

Si bien es cierto que ya se reviso FoxPro para Windows, al ambiente que este lenguaje manejaba era de 16 bits, a partir de Visual Basic hablamos de aplicaciones de 32 bits y por lo tanto de mas herramientas en los lenguajes. Al comparar FoxPro para Windows con Visual Basic observaremos que el segundo es muy superior, desde la programación que en FoxPro es programación estructurada y en Visual Basic es orientada a objetos. Cierto es que FoxPro cuenta con mas de 300 comandos para el manejo de sistemas de bases de datos, pero Visual Basic cuenta con grandes herramientas para sustituir a todos esos comandos. Por otro lado la presentación que ofrece una aplicación de 32 bits siempre será superior a una aplicación realizada bajo Windows 3.X. FoxPro es un gran lenguaje pero para aplicaciones de 16 bits.

Por el otro lado, cuando hablamos de Clipper se menciono que era un lenguaje que trabajaba bajo ambiente DOS, por lo que seria algo difícil compararlo con Visual Basic.

IV.3.1 Compatibilidad

Al ser un lenguaje de 32 bits y dedicados a crear aplicaciones para Windows 95 o 98, Visual Basic cuenta con una gran compatibilidad con casi todo el software soportado por Windows.

Se ha mencionado ya la conexión que existe entre Access y Visual Basic para el manejo de bases de datos. Pero la compatibilidad no es solo con Access, en la propiedad Connect del DataControl podemos definir de que tipo será la base de datos, entre otras tenemos a Excel

en varias versiones, a Dbase, FoxPro, Lotus, Paradox. Al utilizar una base de datos de cualquiera de estas aplicaciones trabajaría sin ningún problema en Visual Basic, pero obviamente es mas recomendable trabajar con bases de datos de Access.

Por otro lado ya se explico el tipo de objeto OLE, como se pudo ver por medio de este objeto podemos agregar a nuestros formularios datos de diferentes aplicaciones de Windows. Además por medio del Objeto OLE podemos agregar también sonidos y videos a nuestras aplicaciones.

Todas las características de compatibilidad se dan debido a que Visual Basic es un producto mas de la familia Microsoft, por lo tanto interactua con el software de Windows sin ningún problema. Y los archivos de bases de datos pueden ser utilizados por cualquier aplicación de Windows.

Además Visual Basic también genera archivos DLL, es decir librerías de Windows que pueden ser utilizadas por otros lenguajes o aplicaciones del mismo Windows.

En la actualidad se puede encontrar la versión 6.0 de Visual Basic. Esta versión cuenta con nuevas herramientas de compatibilidad sobre todo para aplicaciones de Internet, Java y las nuevas plataformas de red que nacen día con día.

Visual Basic es un lenguaje que va evolucionando conforme evoluciona la computadora, por lo mismo ofrece una gran compatibilidad con las nuevas aplicaciones para Windows.

IV.3.2 Migración.

La migración de aplicaciones generadas en Visual Basic no es muy recomendable sobre todo cuando se intenta migrar a un lenguaje no visual, es decir que no maneje programación orientada a objetos, de hecho no se puede migrar de un lenguaje de programación estructurada a un lenguaje de programación orientada a objetos.

Hablemos primero del caso en que se quiere cambiar una aplicación a una versión superior dentro del mismo Visual Basic, este es un paso muy simple, solo basta abrir el proyecto en la nueva versión y automáticamente Visual detecta que la aplicación fue generada en una versión anterior, se muestra una pantalla en la que se pregunta al usuario si se quiere guardar la aplicación con la nueva versión, en caso de que el usuario no cambiara la versión, el proyecto será abierto únicamente de lectura. Si el usuario decide guardar el proyecto en la nueva versión no podrá ser abierto por versiones anteriores.

No es muy recomendable migrar un sistema de Visual Basic a otro lenguaje a menos que sea por medio de librerías. A pesar de que los lenguajes Visuales manejan formularios y programación modular orientada a objetos, cada uno cuenta con una extensión diferente de los archivos que genera por lo tanto no se pueden abrir en otros lenguajes.

Pero las librerías son archivos públicos de Windows que pueden ser utilizados por cualquier aplicación que se ejecute en este ambiente. Por ello se pueden utilizar estas librerías para poder migrar algún sistema entre lenguajes visuales.

Si bien es cierto que se puede hacer lo anterior cabe mencionar que no es recomendable ya que cada lenguaje maneja sus objetos de forma diferente y las propiedades no suelen ser las mismas por lo que se tendría que hacer muchas modificaciones a los sistemas cuando estos fueran migrados.

En este capitulo podemos concluir mencionando que en este momento Visual Basic es uno de los lenguajes mas comerciales y cuenta con gran aceptación en el mercado mundial, no solo por las aplicaciones para bases de datos sino por todas las herramientas que ofrece. Es muy recomendable utilizarlo para generar sistemas de bases de datos, aunque es muy recomendable que para hacerlo se trabaje en conjunto con Access para tener mas potencial.

Para gente que no conoce de lenguajes visuales es muy recomendable comenzar a generar aplicaciones con la versión 4.0 de Visual Basic que ofrece un ambiente sencillo y amigable. Para programadores mas experimentados se propone la versión 6.0 que cuenta con nuevas herramientas sobre todo en lo relacionado al internet.

V. Delphi en las bases de datos

V.1 Conceptos Generales.

Analizaremos en este capitulo un lenguaje que para muchos programadores es el mejor en este momento, Delphi vino a ser la competencia principal de Visual Basic sobre todo para sus primeras versiones. Delphi es un lenguaje de la familia Borland. Borland es para muchos el mejor fabricante de lenguajes de programación, desde Lenguaje C hasta Delphi.

Hasta la versión 3 de Visual Basic se tuvieron muchas deficiencias sobre todo que eran aplicaciones de 16 bits. Cuando nace Delphi algunos programadores lo prefirieron a Visual Basic no solo por las aplicaciones de 32 bits sino por el ambiente que presentaba en ese momento. Actualmente Delphi y Visual Basic son los mejores lenguajes para todo tipo de aplicación sobre el ambiente Windows.

Según lo creadores de Delphi este fue el primer lenguaje que creo aplicaciones realmente ejecutables, es decir que fue el primer lenguaje que creo archivos EXE para aplicaciones de Windows, al crear aplicaciones para Windows también permite crear archivos de librería, es decir archivos con extensión DLL:

Visual Basic y Delphi son lenguajes muy similares, sobre todo en el ambiente de trabajo. En este capitulo iremos revisando todas las herramientas que nos ofrece Delphi y en algunos casos se harán las comparaciones necesarias con Visual Basic, sobre todo para facilitar al programador la elección de algunos de estos dos lenguajes.

Este lenguaje cuenta con varias versiones la que utilizaremos para trabajar en este capitulo será la Versión 3. A su vez esta versión cuenta con diferentes versiones, la versión standard, la profesional y la versión Cliente/Servidor. Esta ultima será la utilizada para la realización del sistema final.

Al igual que en Visual Basic dependiendo de la versión que se utilice serán las herramientas que se nos proporcionen, por ello es que utilizaremos la versión 3.0 Cliente servidor para mostrar ejemplo y crear el sistema final.

Delphi cuenta con un ambiente muy sencillo para trabajar, la base de este lenguaje son también los formularios. Cuando se realiza un proyecto se pueden realizar tantos formularios como el sistema lo requiera.

En la ventana principal de Delphi se muestran tres áreas, El área para formulario, el área de la barra de herramientas y el área del inspector de componentes.

Como la base de la creación de aplicaciones serán los formularios trabajaremos conceptos tales como propiedades, sucesos y componentes. Cabe mencionar que para cada uno de los componentes se tienen varias propiedades que se pueden cambiar durante la creación del sistema, para esto únicamente basta con seleccionar la propiedad y escribir con el teclado o seleccionar con el mouse el nuevo dato que tendrá la propiedad, en caso de que el cuadro de propiedades no estuviera visible basta con pulsar la tecla F11 para poder visualizar este recuadro.

Para crear una aplicación se debe de planificar lo que se va a presentar al usuario, es decir diseñar las pantallas. ¿Qué menús se desean?, ¿Cuántas ventanas habrá?, ¿Dónde deben situarse los botones de ordenes que pulsara el usuario para activar aplicaciones?, estas son las preguntas básicas que se deberán contestar para crear una aplicación en Delphi.

Lo anterior se recomienda ya que cuando se esta elaborando la interfaz, los botones de ordenes y otros componentes que se han colocado en una ventana vacía reconocerán automáticamente las acciones del usuario como los movimientos del ratón y pulsaciones de botones.

Delphi es un lenguaje que al igual que Visual Basic ofrece grandes herramientas para el manejo y creación de sistemas de bases de datos. Cuenta con objetos que se enfocan al manejo de las bases de datos. De hecho la barra de herramientas se divide en varias subcarpetas, cada una de estas subcarpetas cuenta con componentes para las diferentes aplicaciones y existe una especial para las bases de datos.

En general Delphi cuenta con mas de 70 componentes para todas las aplicaciones de este lenguaje.

De hecho si en Visual Basic se recomendaba tener un manejador de bases de datos como lo es Access, para el manejo de sistemas de base de datos con Delphi se recomienda tener instalado Paradox o Dbase para Windows. Estas dos aplicaciones no son muy utilizadas pero es recomendable tenerlas sobre todo cuando se esta trabajando con una base de datos externa, aunque Delphi en sus versiones profesional y Cliente/Servidor cuenta con un manejador de bases de datos en particular, es decir propio de Delphi.

Al igual que en Visual Basic, en Delphi contamos con componentes para el control de datos, además en este lenguaje y sobre todo por la versión que se utiliza, puede ser muy utilizado SQL sobre todo para crear consultas y otras aplicaciones de bases de datos.

En este lenguaje regresaremos al concepto de tabla tradicional que ya habíamos manejado y no al de base de datos que manejamos en el capitulo anterior aunque como ya se vera mas adelante se puede también tener una base de datos con varias tablas, pero para ello se tienen que utilizar otras herramientas.

En resumen para crear una aplicación en Delphi se deben seguir los siguientes pasos:

1. Hacer a medida las ventanas que el usuario ve.
2. Decidir los sucesos que reconocerán los componentes de la ventana.

3. Escribir los procedimientos de sucesos para esos sucesos y los procedimientos que sirvan para complementar los sucesos.

V.1.1 Características Técnicas.

Delphi es un lenguaje que se ha creado para poder realizar aplicaciones para Windows sobre todo para 32 bits. Para poder trabajar en este lenguaje se recomienda que el programador sea un gran conocedor de programación en Windows, aunque la programación no es muy complicada hay algunas aplicaciones que requieren de conocimientos avanzados de Windows, sobre todo cuando se va a trabajar con librerías, es decir con archivos con extensión DLL.

Es importante mencionar que Delphi es un lenguaje basado en Pascal, por lo que es recomendable tener conocimientos básicos sobre programación en Pascal, sobre todo para la utilización de algunas sintaxis propias de este lenguaje y que provocan problemas en Delphi.

Dentro de los sistemas de bases de datos, aunque Delphi cuenta con herramientas para la utilización de tablas, es recomendable tener instalado un manejador especial para las bases de datos tal como lo puede ser Dbase o Paradox, aunque se puede utilizar otro manejador de bases de datos, se recomienda estos dos debido a que son del mismo fabricante de Delphi y utilizando cualquiera de estos dos manejadores evitara problemas sobre todo en cuestión de compatibilidad.

Se recomienda utilizar la versión profesional y la versión Cliente/servidor para poder aprovechar al máximo las herramientas que se ofrecen en este lenguaje. Utilizando alguna de estas versiones no será necesario contar con un manejador externo de bases de datos, pero como se menciona en el párrafo anterior es muy recomendable tener instalado Dbase o Paradox, esto es porque aunque se tiene aplicaciones dentro de Delphi para las bases de datos es mas recomendable dedicar a Delphi 100 % a crear la aplicación y no estarse preocupando por la estructura de la base de datos, de hecho la herramienta que proporciona

Delphi es algo parecida a la que se mostró en Visual Basic, es decir es una herramienta externa al lenguaje que ofrece aplicaciones pero no tantas como las que puede ofrecer un manejador mas completo.

En resumen para poder programar sistemas de bases de datos con Delphi debemos de tener instalado en nuestro equipo: Windows 95 ó 98, Delphi (ya sea la versión profesional o Cliente/servidor) y un manejador como Dbase o Paradox, haciendo lo anterior tendremos las herramientas suficientes para generar un sistema muy eficiente.

V.1.1.1 Requerimientos en Hardware.

Al ser Delphi un lenguaje que crea aplicaciones de 32 bits, obvio es que se requiere de un muy buen equipo para poder aprovechar al máximo las herramientas que nos ofrece este lenguaje.

La primera versión de Delphi podía trabajar incluso en un ambiente de Windows 3.X, por lo que no requería de un gran equipo, pero por eso mismo se desperdiciaba las herramientas que este lenguaje proporcionaba, aunque comparado con Visual Basic ofrecía mas herramientas. Aunque la versión 1.0 de Delphi se ejecuta en ambiente de Windows NT o Windows 95 únicamente crea aplicaciones de 16 bits.

La versión 2.0 y la versión actual requieren de un equipo que cumpla con las siguientes características básicas:

Procesador 486 o superior.
8 Mb en memoria RAM
Monitor Color Vga
Disco duro de 500 Mb o mayor.
Unidad de disquete 3 ½
Tarjeta de vídeo Vga

Mouse y Teclado

Las características presentadas son similares a las que se presentaron en el capitulo anterior, esto se debe a que los lenguajes son muy parecidos y es por ello que requieren del mismo tipo de equipo. Para la versión Cliente/Servidor se recomienda un equipo mas potente con la finalidad de poder aprovechar sobre todo las ventajas de red que esta nos ofrece.

Para la Versión 3.0 se requiere de un equipo mejor al que se acaba de presentar sobre todo porque esta versión cuenta con muchas aplicaciones para trabajar con paginas WEB y otras herramientas de la red. Pero para el caso que nos corresponde, es decir para las bases de datos con el equipo presentado se puede diseñar un buen sistema, aunque es importante tomar en cuenta que estas son las características básicas, que de poder mejorarlas obviamente mejorara el rendimiento de nuestro sistema. De hecho recomendamos esta versión para la creación de los sistemas.

Como ya se menciono en Delphi regresamos al concepto de las tablas, estas contaran con un tamaño según la cantidad de información que se maneje pero es recomendable asignar bastante espacio en disco duro sobre todo para evitar que si se añaden mas registros provoque que el disco duro se sature de información y por lo tanto nuestro sistema no funcione adecuadamente.

V.2 Principales Herramientas.

Al igual que Visual Basic, Delphi es un lenguaje que cuenta con una infinidad de herramientas para todo tipo de aplicaciones bajo el ambiente de Windows y hablar de todas requeriría de un libro completo de este lenguaje, por lo que en este caso únicamente mencionaremos las herramientas que nos convengan para los sistemas de bases de datos.

Como ya se ha venido mencionando, para este trabajo utilizaremos la versión 3.0 edición Cliente/Servidor, por lo que las herramientas que se mencionen serán las que nos proporciona esta versión.

Delphi no solo es un lenguaje de programación sino que es todo un software que proporciona toda una serie de diferentes aplicaciones para crear bueno sistemas. Comenzaremos por decir que cuenta con un editor de imágenes, que aunque es muy simple puede ser utilizado para crear iconos o imágenes que podamos utilizar para nuestras aplicaciones. La forma de trabajar en este editor es muy simple ya que es muy parecido a Paint de Windows.

Para las bases de datos contamos con tres aplicaciones que serán las que nos servirán para trabajar con los datos. Por principio de cuentas contamos con una herramienta conocida como Database Desktop. Esta herramienta nos permite crear y modificar estructuras de tablas para bases de datos.

Por otro lado se cuenta con un administrador BDE que es muy utilizado para crear alias (mas adelante veremos que es esto y para que se ocupa), este administrador sirve para llevar el control de las tablas. También se cuenta con una herramienta para poder crear los informes y etiquetas conocido como ReportSmith, en el punto de informes y etiquetas revisaremos esta herramienta a fondo.

En adelante nos dedicaremos a ampliar como es que se ocupan estas herramientas y sobre todo como es que las podemos adecuar a nuestros sistemas.

Además se cuenta con muchos componentes que podemos utilizar no solo para bases de datos sino para otro tipo de aplicaciones. En la versión cliente/servidor además ya se había mencionado que se cuenta con muchas herramientas para el manejo con redes y sobre todo para aplicaciones con la red Internet.

V.2.1 Manejo de Archivos

Al igual que Visual Basic, algunos de los archivos que se generan en Delphi son utilizados por Windows. Por principio de cuentas recordemos que Delphi crea tanto archivos ejecutables (EXE), y de librerías (DLL). Estos archivos pueden ser utilizados pro otros lenguajes.

Dentro de los archivos que se crean exclusivamente para el uso de Delphi encontramos de varios tipos. Hablaremos primero de los archivos de un proyecto. Recordemos que un proyecto es aquel archivo que contiene todos los formularios, los componentes y los módulos de lenguaje. Este archivo tiene siempre la extensión DPR en el nombre del archivo. Aunque este archivo es de proyecto cada uno de los formularios se guarda en un archivo diferente con una extensión DFM. Por lo que cada uno de las aplicaciones generadas cuenta con un solo archivo de proyecto pero con varios archivos de formulario.

Además de los archivos mencionados en el párrafo anterior, Delphi guarda archivos con extensión PAS. Esta extensión indica que el archivo contiene código de programa, es decir estos archivos se almacenan en un formato ASCII. En este archivo se guardan todos los códigos de suceso (en Delphi a cada uno de estos códigos de suceso se le llama Unidad), es decir cuando se va tecleando el código para que los componentes respondan a los sucesos, este código se va guardando en un archivo con extensión PAS.

Resumiendo, un proyecto esta entonces formado por uno o mas archivos formulario y uno o mas archivos unidad. Para poder saber que archivos constituyen un proyecto se puede utilizar la ventana Project Manager, esta ventana se activa desde el menú View. El administrador de proyectos contiene una lista de formularios y las unidades que contiene cada proyecto.

Hablando de los sistemas de bases de datos, los archivos que se utilizan será únicamente los relacionados a las tablas que se requieren para la ejecución de nuestro sistema. Las tablas

que se utilizan en el manejador de bases de datos de Delphi y por lo tanto en los sistemas serán las tablas con la extensión DBF.

En Visual Basic trabajamos las bases de datos de una manera diferente, es decir contábamos con un solo archivo para las bases de datos que contenía varias tablas y varios índices. En Delphi se pueden trabajar las tablas de forma independiente o podemos crear un Alias para varias tablas. El Alias será un subdirectorio en el cual se encuentran las tablas que utiliza un sistema de bases de datos (mas adelante veremos como se crean los Alias). De esta forma podremos accesar a las tablas de una manera parecida a Visual Basic, pero siempre se tendrá en cuenta que este es únicamente un alias y las tablas son completamente independientes.

También para crear los informes y las etiquetas utilizaremos otro tipo de archivos, estos serán generados por el ReportSmith y por el editor de reportes, las etiquetas y los informes serán del mismo tipo aunque en presentación cambien un poco. La extensión para este tipo de archivos será QRP.

Note que desde el manejo de los archivos se comienzan a observar las diferencias entre Delphi y Visual Basic, y aunque en Delphi tenemos que utilizar otras técnicas para poder manejar las tablas en Visual Basic no se crean los archivos que contienen el código, es decir los archivos de unidad.

V.2.1.1 Tablas

El manejo de tablas en Delphi difiere un poco al de Visual Basic, ya que como ya se ha mencionado regresaremos al concepto tradicional de tablas, es decir tendremos cada una de nuestras tablas de forma independiente a menos que se trabaje con relaciones. Pero aunque fisicamente las tendremos así, de forma lógica las tendremos en una especie de base de datos que en Delphi se conoce como Alias.

Un Alias indica a los componentes de Delphi cual será la ruta en la que buscara las tablas, por lo que se recomienda que todas las tablas que se utilicen en un sistema se guarden en un solo subdirectorio y a este directorio se le cree un alias para trabajar directamente con este y no tener problemas de andar localizando las tablas en directorios independientes. Por ahora estudiaremos como trabajar con Database Desktop que será el manejador de tablas que proporciona Delphi.

Para poder accesar a esta aplicación se puede hacer por medio del menú Tools y seleccionando la opción Database Desktop. Una vez que se ha abierto esta aplicación comenzaremos por explicar como se crea una tabla desde aquí.

Comenzaremos por seleccionar la opción New del menú File, aparecerá un submenu en el que le indicaremos que queremos crear una tabla. A continuación le tenemos que indicar que tipo de tabla crearemos (en este manejador se pueden crear tablas para Dbase, Paradox, FoxPro, etc.). Para nuestro caso la crearemos de tipo Paradox.

A continuación comenzaremos a indicar el nombre, tamaño y tipo de cada uno de los campos que nuestra tabla llevara.

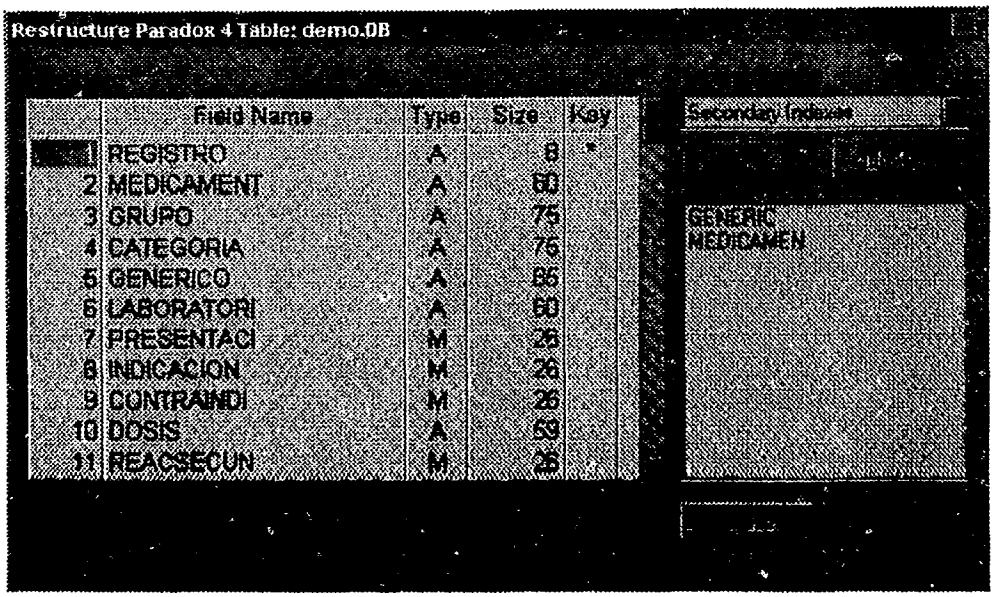
Los tipos de campos que se podrán insertan a las tablas dependerán del tipo de tabla que elegimos, pero es recomendable trabajar con los tipos de campos clásicos, es decir:

- Numéricos
- Carácter
- Fecha
- Memo
- Lógico

Para facilitar el trabajo al creador de la tabla se proporciona la siguiente herramienta: cuando queremos llenar el tipo de campo podemos teclear la barra espaciadora en la columna

correspondiente a Type, se mostrara en pantalla los tipos de campo que se permiten para el tipo de tabla que se va a crear.

En la pantalla siguiente se muestra la tabla que se utilizara en nuestro caso para poder crear el sistema final.



Como puede observar la relación de tablas es muy sencilla. Ahora no tomaremos un momento para revisar los índices, cabe mencionar la importancia que tendrán los índices en las tablas de bases de datos, recuerde que estos índices los utilizaremos sobre todo para búsquedas y ordenaciones que se tengan que hacer durante la creación de nuestro sistema.

En la pantalla anterior puede ver que existe una columna con la palabra Key, note que el primer campo de la tabla cuenta con un asterisco en esa columna. Esto se debe a que ese campo será el índice primario de nuestra tabla.

Recordemos que en Visual Basic el archivo de base de datos contenía todas las tablas, índices y consultas relacionados con la base, pero en FoxPro y Clipper la tabla era un archivo por si solo y aparte existían los archivos para los índices.

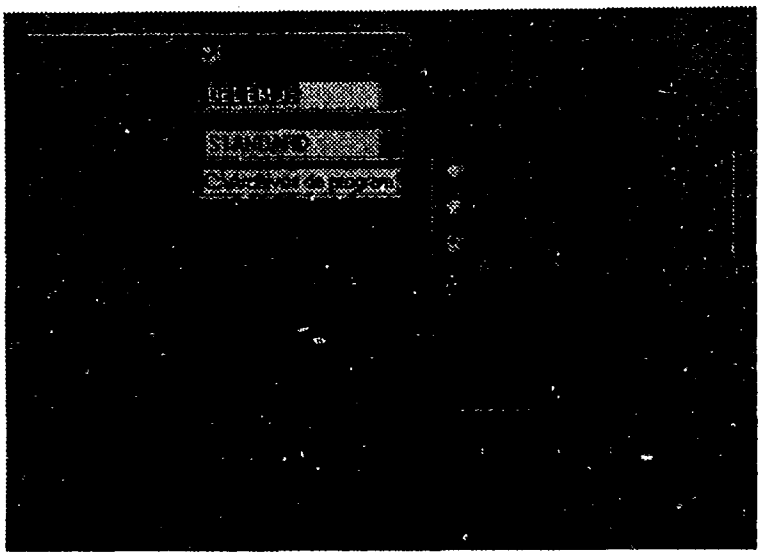
En Delphi (o sea en Paradox) como ya hemos dicho una tabla será un archivo independiente pero los índices formaran parte de este archivo, por lo que en el caso de Paradox no es necesario crear otro archivo para índices (en caso de que la tabla fuera de Dbase los archivos de índice son independientes).

Como ya se menciono, el asterisco indica que ese campo será el índice primario, pero en el caso de que se quisiera crear uno o mas índices secundarios, si se tuviese que crear los archivos necesarios para ello, de hecho únicamente un campo dentro de la tabla puede aparecer con asterisco dentro de la columna Key. Para crear índices secundarios se debe hacer lo siguiente:

- En el recuadro Table Properties elija la opción Secundary Indexes
- Se dará clic en la opción de Define ...
- En la pantalla de índices se ira indicando cual o cuales serán los campos que formaran parte de ese índice, seleccionando el campo y dando Clic en el botón con este símbolo >>.
- Una vez que selecciono el campo basta con dar Clic en la opción Aceptar.
- Puede crear tantos índices secundarios como sea necesario.

Una vez que ya creo la tabla puede insertar elementos ahí mismo eligiendo la opción Edit Data del menú Table. Así mismo puede reestructurar la tabla en caso se que así lo necesitara eligiendo la opción Reestructure del menú Table.

Por ultimo, para crear un alias se puede hacer también desde esta aplicación. Seleccione la opción Alias Manager del menú Tools, Delphi presentará la pantalla que se muestra a continuación:



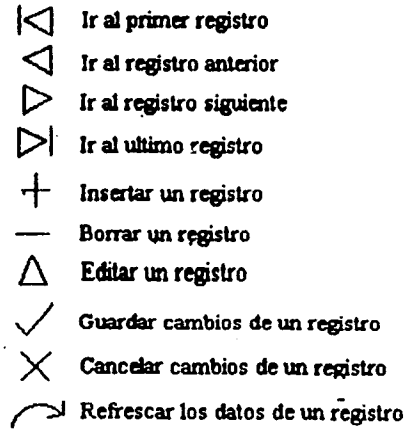
Elija la opción New y a continuación teclee el nombre del alias en el recuadro de Database Alias y el recuadro de Path tecleara la ruta donde se encuentran las tablas que contendrá este alias. Se da un clic en Aceptar y el alias quedara creado en los archivos de Delphi.

Una vez que ya hemos visto como se crean las tablas en este manejador hablemos de cómo se manejan desde el lenguaje. En Visual Basic contábamos con varios objetos para el manejo de bases de datos, en Delphi sucede lo mismo pero en el caso de la versión que estamos manejando contamos con mas componentes para el manejo de bases de datos, estos componentes los revisaremos todos en el punto Gráficos y Botones pero en este caso es importante mencionar tres que serán los encargados de indicar la tabla y en que condiciones se trabajara.

Para trabajar con una tabla en un formulario es necesario insertar 4 tipos de componentes como mínimo:

- Un componente Ttable: En este se modificaran varias propiedades; la propiedad DatabaseName en la que se indicara el Alias o el subdirectorio donde se encuentra la tabla que se va a manejar. La propiedad TableName en la que se escribirá el nombre de la tabla. Y la propiedad Active que se cambiara a True para visualizar los registros.
- Un componente DataSource: es este se escribirá el nombre del componente Ttable en la propiedad Dataset para crear el vinculo.
- Un componente DBNavigator: este será muy parecido al DataControl de Visual Basic con la diferencia de que este cuenta con muchos mas bonotes que el DataControl. En este tendremos que escribir el nombre del componente DataSource en la propiedad del mismo nombre.
- Componentes auxiliares: estos podrán ser componentes como botones de comando, cuadros de texto, etiquetas, DBGrid, etc.

Los dos primeros componentes que se acaban de mostrar son componentes no visibles, es decir que solo se utilizan durante la creación del sistema y el usuario final no puede verlos y por supuesto no puede modificarlos. El DBNavigator es el componente que se utiliza principalmente para mover el puntero dentro de la tabla, es decir poder visualizar los diferentes registros de la tabla, además de permitir lo anterior cuenta con botones para añadir, editar y borrar registros, en la tabla siguiente se muestra los botones que aparecen y para que sirven al usuario final.



La facilidad de tener estas herramientas permite al programador codificar menos líneas de código para poder crear un sistema que manipule bases de datos. Dependerá del usuario si utilizar el DBNavigator o utiliza botones de comando para crear aplicaciones con otra presentación, aunque esto implica generar mas líneas de código.

Es importante mencionar que existen componentes dedicados al manejo de tablas, por ejemplo aunque existe un cuadro de texto también existe un cuadro de texto pero dedicado a mostrar los valores de un campo de una tabla, es decir estos componentes tendrán propiedades pero enlazadas con los componentes Ttable y DataSource, esto es una facilidad para los programadores porque estos componentes son dedicados a las tablas y no se tendrá que preocupar por otras aplicaciones.

Además de tener las herramientas ya presentadas se contienen comandos que permiten manipular los datos de las tablas, en este punto no nos detendremos a analizar estos comandos ya que los analizaremos en el punto dedicado a programación. En esta parte solo mencionaremos que los comandos no se basan en una sola herramienta como en Visual Basic era el Recordset sino que en este lenguaje existen muchos comandos independientes, cada uno enfocados ya sea al manejo de registros, campos o a la misma tabla.

V.2.1.2 Etiquetas

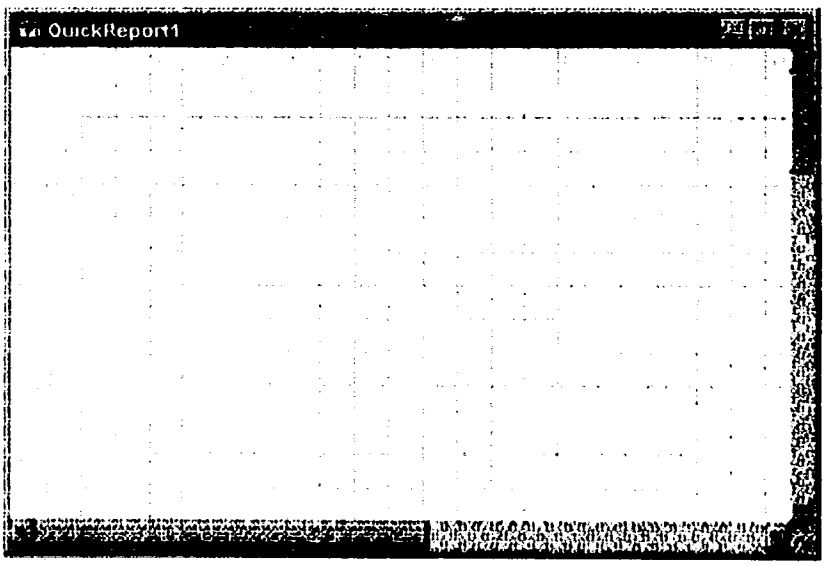
Las etiquetas y los informes son muy importantes para los sistemas de bases de datos. En este punto mencionaremos como se realizan los informes y etiquetas ya que en Delphi se realizan de la misma manera, la principal diferencia será que en las etiquetas únicamente trabajaremos con una tabla de datos y con los informes podremos trabajar con varias tablas al mismo tiempo.

En las primeras versiones de Delphi las etiquetas y los informes se realizaban por medio del ReportSmith. El ReportSmith es una herramienta muy parecida a Crystal Reports de Visual Basic, es decir que es una herramienta independiente al lenguaje que permite crear informes

y etiquetas de una manera muy sencilla. La versión 3 Cliente/Servidor ya no cuenta con el ReportSmith pero la herramienta que se ofrece para crear un reporte es mucho mas sencilla que el ReportSmith.

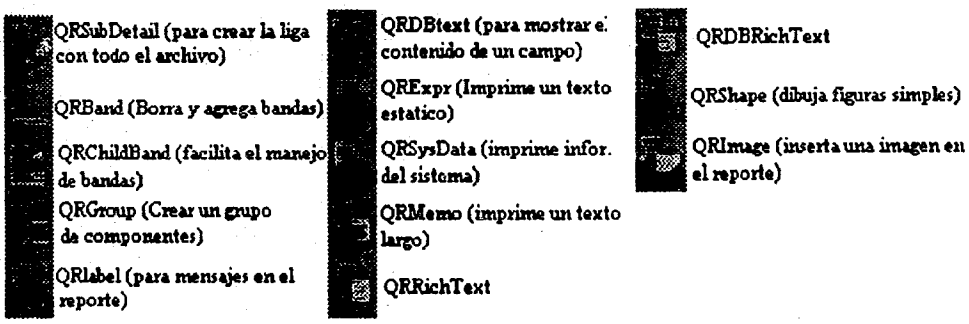
La facilidad de manejo en esta versión de Delphi se nota también en la forma de generar reportes, ya que se trabaja como si fuera una pagina en la que estoy escribiendo lo que quiero ver y estoy estableciendo la forma en la que lo quiero ver.

Los reportes (etiquetas e informes) en la versión que estamos manejando se realizan de forma muy parecida a un formulario, es decir cuando queremos crear un reporte seleccionaremos la opción New del menú File, en pantalla aparecerán diferentes opciones, el programador deberá seleccionar la opción Report, al hacer esto aparecerá una pantalla parecida a la de un formulario pero de color Blanco y con márgenes, observe la siguiente figura.



En esta pantalla se deben de insertar ciertos componentes que se mencionaran después de presentar los folders que sirven para generar reportes. Como ya se menciono los componentes se han dividido en folders, existe un folder llamado Qreport, este contiene una

serie de componentes que sirven para poder generar un reporte. Es importante mencionar que muchos de los componentes de otros folders no se pueden insertar en este tipo de formulario. En la figura siguiente se muestran los componentes que conforman este folder.



Para poder crear una etiqueta por medio de esta herramienta primero debemos de asignar el tamaño que vamos a utilizar para la etiqueta, en el formulario de reporte se deberá de dar Doble Clic, a continuación aparecerá una pantalla en la que podrá configurar no solo el tamaño de la etiqueta sino también el tamaño de la letra, los márgenes, la orientación de la hoja, etc. Esta configuración se deberá de hacer no solo para las etiquetas sino que también para los informes.

Una vez que ya conocemos los componentes del Folder Qreport mencionaremos los componentes que obligatoriamente deben de estar para poder generar una etiqueta.

- Componente Ttable: este indicara el alias y la tabla para la cual se formaran las etiquetas, se modificaran las propiedades Active, TableName y DatabaseName.
- Componente DataSource: este creara el enlace entre el componente Ttable y los demás componentes, en este se modificara la propiedad DataSet.
- Componente QRSubDetail: este creara la liga y hará que se impriman en las etiquetas todos los registros de la tabla, en caso de no tener este componente únicamente se imprimirá el primer registro de la tabla. En este componente se modificaran las propiedades DataSet, FooterBand, Master y Aling to Bottom.

- Componentes QRLabel : Estos componentes indicaran los mensajes que aparecerán en la etiqueta, se modificara la propiedad Caption. Se tendrán tantos como sean necesarios, incluso para títulos dentro de una pagina.
- Componentes QRDBText: este será el que mostrara el valor del campo al que se le asocie, es decir será como un cuadro de texto de solo lectura pero asociado a un campo de la tabla. Se tendrán tantos componentes de este tipo como campos queremos mostrar en nuestra etiqueta. Se tiene que modificar las propiedades DataSet y DataField.

En la pantalla se acomodaran estos componentes según la presentación que se quiera dar a la etiqueta, si se elige un tamaño grande de pagina para las etiquetas se deberán poner tantas etiquetas como se quiera aparezcan en la pagina, es decir si hacemos una sola etiqueta en este formulario se imprimirá una sola etiqueta en esa pagina, esta es una desventaja pero por la facilidad que se ofrece para crear estas etiquetas no resulta muy difícil para el programador realizar esta acción.

V.2.1.3 Informes

Los informes se realizan de la misma manera que en las etiquetas, es decir utilizaremos el formulario de reportes, en este caso lo primero que haremos sea configurar el tamaño de la pagina y el tipo de letra que se utilizara de la misma manera que para las etiquetas.

Ya conocemos los componentes que tenemos para crear los reportes, para los informes podemos utilizar varias tablas, es decir que un solo informe podrá contener varias tablas, para hacer esto se deberán tener un DataSource y un Ttable para cada una de las tablas que se utilizaran en el informe.

En las etiquetas se menciono que el componente QRSuBDetail permitia imprimir todos los registros de una tabla, cuando generamos un informe con varias tablas por lo regular es únicamente una la que imprime todos sus registros y las demás son auxiliares e imprimen únicamente un solo registro por lo que la propiedad Dataset de este componente deberá ser

definida respecto a la tabla que imprimirá todos. Por lo que respecta a los demás componentes se configuraran las propiedades mencionadas en las etiquetas pero con la diferencia que para las etiquetas la propiedad Dataset será la misma para todos los componentes y para los informes variara según la tabla que contenga el campo que queremos imprimir.

También es importante mencionar que solo se imprimirá lo que se muestra en el formulario de reportes, es decir en Delphi no se puede asumir que se irán reproduciendo los valores en una sola pagina, por ejemplo si en un informe únicamente colocamos un componente QRDBText y además contamos con un componente QRSubDetail se imprimirán tantas paginas como registros tiene la tabla, no se asume por ejemplo que se imprimirán varios registros en una sola pagina ya que en el formulario únicamente se inserto un solo componente.

Para poder dar presentación tanto a etiquetas como informes se puede utilizar componentes como QRImage, QRDBImage, etc. estas herramientas permitirán que se impriman algunas imágenes en un informe como puede ser el logotipo de alguna empresa, un dibujo en particular, etc.

V.2.2. Manejo de Presentaciones

La presentación como ya se ha visto en los demás lenguajes es una parte esencial en la creación de sistemas. Al ser Delphi un lenguaje enfocado a crear aplicaciones de 32 bits, resulta necesario contar con herramientas para presentación bastante poderosas y la versión que estamos utilizando las ofrece.

Al igual que en el capitulo anterior comenzaremos explicando como se realizan los formularios y cuales serán las principales propiedades que tendremos que configurar para poder tener una buena presentación. Los diferentes componentes que proporciona Delphi los estudiaremos un poco mas adelante (revisar punto de Gráficos y Bonotes).

Como ya se menciono, la base de una aplicaciones en un lenguaje visual son los formularios, mucho dependerá los colores y la presentación física de este formulario para ser agradable al usuario final, recordemos que muchos usuarios se van por lo bonito que se ve una aplicación mas que por lo que hace. Al final de cuentas será en el ambiente en el que el usuario estará trabajando y es recomendable que este sea un ambiente grato para dicho usuario.

Cuando se crear un formulario en la pantalla aparece un recuadro acerca de las propiedades que corresponden ya sea a un componente o al mismo formulario en particular. En Delphi al igual que en Visual Basic podremos configurar estas propiedades únicamente cambiando el valor correspondiente en la parte derecha del recuadro de propiedades. En Delphi a la pantalla de propiedades se el conoce como Object Inspector. Este contiene dos folders, el primero que contiene las propiedades y el segundo que contiene los eventos para el objeto seleccionado. En caso de que no aparezca el Object Inspector lo puede habilitar pulsando la tecla F11.

En Delphi el tamaño de formulario que se utilice durante el diseño será el mismo que se vera durante la presentación, por lo que es recomendable desde un principio adecuar el tamaño del o los formularios que se vayan a utilizar. Recodemos que una aplicación puede tener tantos formularios como sea necesario.

Para cambiar el tamaño del formulario basta con utilizar el mouse como si se tratara de cambiar el tamaño a una ventana tradicional, es decir se posiciona en alguna esquina y se arrastra el mouse hasta el nuevo tamaño.

Una vez que definimos el tamaño de nuestro formulario modificaremos algunas de las propiedades para dar una mejor presentación: Entre las propiedades que se pueden configurar las principales son:

- Caption : esta propiedad modifica el nombre que llevara nuestra ventana en la parte superior. Para escribir un nombre únicamente basta con teclearlo en el recuadro.
- Name : esta propiedad asigna un nombre al formulario que será utilizado por el mismo lenguaje para hacer referencia a el.
- Color : esta propiedad definirá el color de fondo que tendrá el formulario, Delphi cuenta con toda una gama de colores establecidos, para cambiar el color basta con dar clic en la flecha de menú y nos mostrara un cuadro de lista con todos los posibles colores. La combinación de colores de cada aplicación es responsabilidad del programador.
- Font : determinara el tipo de letra que se utilizara en el formulario. Esta propiedad es importante pero es mas recomendable utilizarla en los diferentes componentes ya que cada uno puede requerir un diferentes tipo de letra.
- FormStyle:
- Menú

Aunque existen mas propiedades depende del tipo de aplicación que se este generando las que se puede utilizar. Para cada uno de los formularios que se van a generar se recomienda tomar los mismos valores de las propiedades, recuerde que cada formulario es independiente por lo que cada uno se puede configurar de distinta forma pero esto no es recomendable para una aplicación seria.

Es importante recomendar aquí que los formularios se vayan generando según el orden en el que se pueda presentar la ejecución, es decir se recomienda que primero se genere el formulario principal que mandara a llamar a los demás. Para poder generar un formulario ya que se ha realizado otro, basta con seleccionar la opción New Form del menú File. Este nuevo formulario contendrá los valores que por default proporciona Delphi.

Para mandar a llamar a otro formulario generados en la misma aplicación se utiliza el comando:

<Nombre_del_formulario>.ShowModal;

Este comandos es utilizado principalmente cuando se cuenta con varios formularios y es necesario estar llamando a unos dentro de otros. Esto es muy recomendable ya que se aprovecha al 100% el ambiente de Windows y una sola aplicación no se desempeña en una sola pantalla, el hecho de que existen varios formularios no solo ayuda a una buena presentación sino que además facilita en mucho al programador la realización del sistema.

Con lo mostrado en este punto ya conocimos como configurar el ambiente de un formulario, los componentes que contenga dicho formulario dependerán de la aplicaciones que se este ejecute, hasta este momento hemos visto ya varios componentes sobre todo para las bases de datos, en el punto de Gráficos y botones revisaremos otros componentes que se pueden utilizar en sistemas de bases de datos.

V.2.2.1. Menús

La creación de menús en Delphi es muy sencilla ya que se basa en las propiedades que ya hemos visto, es decir cada una de las opciones del menú será como un componente mas que cuenta con propiedades y sucesos (aunque el único suceso con el que se cuenta es el de OnClic).

Para crear un menú tenemos que agregar el componente MainMenu del folder Standard a nuestro formulario, recuerde que como cada formulario es independiente se debe de crear un menú para cada uno de los formularios.

Para agregar este componente basta con dar doble Clic sobre el componente ya mencionado.

Una vez que el componente se encuentra sobre el formulario se deberá dar doble clic sobre este componente, aparecerá la siguiente pantalla en la cual se generara el menú:

Note que en la parte superior aparece un pequeño cuadro, si escribimos algún nombre en la propiedad Caption este será el nombre que aparecerá en el menú principal, es decir el menú horizontal, cuando se teclea el nombre y se pulsa la tecla Enter automáticamente aparece otro cuadro del lado derecho del primero, esto será para añadir otra opción a nuestro menú. En resumen cada cuadro que aparece en la parte superior será una opción del menú horizontal y la propiedad caption proporcionara el nombre a estas opciones.

Una vez que ya tenemos las opciones de nuestro menú horizontal comenzaremos a teclear los submenus, es decir los menús verticales que aparecerán en cada opción. Para hacer esto basta con posicionarnos en alguna de las opciones que ya se teclearon. Debajo de estas aparecerán unos cuadros en los cuales por medio de la propiedad Caption pondremos los nombres que llevaran las opciones. Note que en este caso también ira apareciendo un cuadro nuevo debajo del ultimo al cual se le asigno nombre.

Al igual que en Visual Basic podemos hacer que nuestro menú se vea mas presentable. Al teclear el nombre de la opción en la propiedad Caption se puede teclear el símbolo (&) para indicar que letra del nombre de la opción aparecerá subrayada para poder accesar a esta

opción por medio de la combinación de la tecla ALT y una letra opcional, por ejemplo si tecleamos &Archivo en la propiedad Caption, la opción del menú aparecerá Archivo y podremos accesar a esta opción mediante la tecla Alt-A.

Si queremos que aparezca una línea en el menú vertical para dividir varias opciones basta con teclear un guión en la propiedad caption del cuadro en el cual queremos que aparezca la línea.

Además también contamos con la propiedad ShortCut que cuenta con varias alternativas para poder hacer que alguna de las opciones de nuestro menú pueda ser accesado por alguna tecla o alguna combinación de teclas, cuando tecleamos el nombre en la propiedad Caption podemos elegir alguna de las opciones que se presentan en la propiedad ShortCut. Por ejemplo si en nuestro menú contamos con la opción de ayuda (recuerde que tradicionalmente en Windows se accesa a la ayuda por medio de la tecla F1) una vez que elegimos en que parte del menú aparecerá esta opción teclearemos la palabra Ay&uda en la propiedad Caption y en la propiedad ShortCut elegimos la opción F1, note que en el cuadro aparece aparte del nombre, la opción que usted eligió.

Una vez que ha terminado de crear el menú cierre la ventana de edición de menú y en su formulario aparecerá el menú que usted acaba de crear. Para hacer alguna modificación una vez que ya se tiene creado el menú basta con volver a dar doble clic sobre el componente MainMenu que se encuentra en nuestro formulario.

Para poder agregar líneas de código a cada una de las opciones de nuestro menú y que de esta manera respondan al suceso, basta con teclear doble clic sobre cada una de las opciones en la pantalla de edición del menú y aparecerá la pantalla en la que se teclea el código, es decir la que lleva el archivo con extensión PAS. Como en un principio se dijo cada una de las opciones del menú es considerada como un objeto mas por lo que cada opción debe contener código para poder responder al suceso OnClic, además de que cada una de estas opciones cuenta con mas propiedades que puede ayudar a que la presentación sea mejor.

Puede consultar el menú de nuestro sistema final para poder observar las diferentes opciones en la propiedad Caption y en la propiedad ShortCut y los diferentes códigos contenidos en cada una de las opciones.

V.2.2.2. Gráficos y botones

Ya vimos anteriormente como se crea un formulario y que propiedades podemos cambiar para tener una buena presentación. También ya revisamos que los tipos de componentes que tenemos que añadir para poder crear un sistema con bases de datos, mencionamos el componente Ttable, DataSource, DBNavigator y componentes auxiliares, estos últimos serán los que revisaremos en este punto.

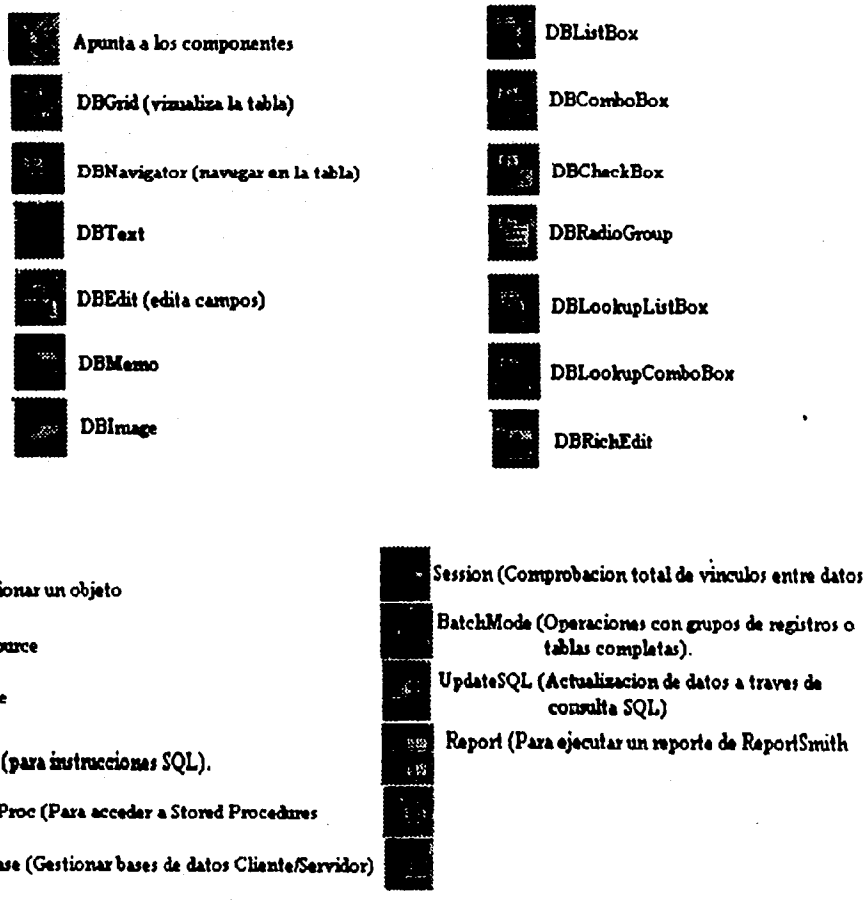
La versión que estamos trabajando cuenta con muchos componentes para todo tipo de aplicaciones, detenernos a estudiar todos requeriría de mucho espacio y mucha información que para el caso de las bases de datos no resultaría muy necesaria, por lo que solo revisaremos los componentes que se proporcionan para las bases de datos.

En Delphi existirán dos tipos de componentes, los que sirven únicamente para diseño y los que sirven para la ejecución, los de diseño serán componentes que se podrán configurar sus propiedades pero no se verán durante la ejecución, es decir el usuario final no sabrá que existen estos, ejemplo de estos componentes serán el Ttable y DataSource. Los componentes de ejecución serán todos los demás componentes, es decir aquellos en los que se utilizaran los sucesos para poder ejecutar acciones pero que serán visibles al usuario durante la ejecución.

En Visual Basic vimos que existían objetos que se podían utilizar en cualquier aplicación, es decir el cuadro de texto lo podíamos utilizar para una aplicación de bases de datos o para otro tipo de aplicación. En Delphi, aunque se puede hacer esto, existen componentes especiales para las bases de datos, por ejemplo existe el mismo cuadro de texto pero este

enfocado a un campo en una tabla. En Delphi 3.0 existen tantos componentes que se han agrupado en folders.

Los componentes para aplicaciones de bases de datos se dividen en dos folders, el primero conocido como Data Access que contiene los controladores para el acceso de bases de datos locales o de red, el segundo folder que contendrá los componentes para trabajar con las tablas y los registros es el folder de Data Controls En las figuras siguientes se muestran todos los componentes que proporciona Delphi para el manejo de bases de datos. La primera figura para Data Access y la segunda para Data Controls.



Los componentes que se presentan en las dos figuras anteriores serán los que presenta la versión 3.0 Cliente/Servidor que es la que estamos utilizando, estos componentes pueden variar según la versión de Delphi que se utilice aunque desde la versión 1.0 se contemplan la mayoría de estos componentes.

Para poder utilizar alguno de estos componentes bastara con dar doble clic en alguno de ellos y en la pantalla aparecerá el componente que se ha seleccionado, a continuación deberá acomodar el componente en la posición y tamaño que convenga al programador (los componentes de diseño se podrán colocar en cualquier lugar ya que no serán visibles durante la ejecución) y modificar las propiedades de acuerdo a las necesidades del sistema. Se mencionara a continuación algunas de las propiedades que se recomienda cambiar para los componentes principalmente del tipo Data Controls.

- Caption : será el mensaje que mostraran en pantalla los componentes, es decir para un botón de comando será el mensaje que se muestre en el.
- Name : será el nombre por medio del cual se podrá hacer referencia al componente en algunas partes del lenguaje, aunque Delphi asigna por Default un nombre a los componentes se recomienda que el programador asigne sus propios nombres.
- Datasource: esta propiedad determina el nombre del DataSource que se esta utilizando, esta propiedad será la que ligara al componente con la tabla de datos.
- ReadOnly : en caso de estar con un valor True hará que los componentes únicamente muestren los datos pero que no se puedan modificar. Esta propiedad es muy recomendada para cuestiones de seguridad.
- Visible : permite que el componente este visible o no durante la ejecución o durante algunos momentos dentro de la ejecución.

Las otras propiedades se cambiaran de acuerdo a las necesidades del usuario. El DataSource será la propiedad mas importante ya que esta será la que diferencie entre componentes para bases de datos y componentes para otras aplicaciones. Estos últimos también pueden ser utilizados, al igual que los Data Controls, contendrán propiedades como Name y Caption, el

programador puede utilizar los componentes que considere apropiados para el sistema pero se recomienda que se trabaje con ambos grupos de componentes para crear un buen sistema.

Como ya se menciono los componentes se encuentran acomodados en folders según para el tipo de aplicación que serán utilizados. Existen los siguientes folders: standard, aditional, Win32, System, Internet, Data Access, Data Control, y otros mas. Básicamente para un sistema de bases de datos como el que estamos creando en este trabajo utilizaremos componentes de Standard, Aditional, Data Access y Data Control. Mostrar todos los componentes de cada uno de los folders seria una tarea tardada y no muy interesante para el manejo de bases de datos.

Por otro lado, un punto muy importante es el de inserción de gráficos a nuestros formularios, recordemos que algunas aplicaciones requieren de insertar por ejemplo algún símbolo de empresa, algún dibujo en particular, etc. Por ello Delphi proporciona de forma independiente un software para la realización y modificación de imágenes ya sea de tipo Icono, mapa de bits, etc. Esta herramienta se conoce como Image Editor y se maneja de manera muy parecida a Paint de Windows.

Para insertar una imagen a un formulario se insertara el componente Image del folder Standard, una vez que se inserto basta con modificar la propiedad Picture, esta propiedad nos mostrara un cuadro para abrir una imagen que ya tengamos guardada en nuestro disco. En la mayoría de tipos de imágenes el tamaño estará determinado por la misma imagen por lo que si tenemos una imagen grande en tamaño del componente debe ser de igual tamaño. Es importante mencionar que la imagen se utiliza estilo papel tapiz de nuestro formulario, es decir que podemos tener otros componentes encima de la imagen y esta únicamente funcionara como fondo del formulario.

Con todas las herramientas que acabamos de mostrar nos podemos dar cuenta de la potencia en presentación que tenemos con Delphi y además de la facilidad que proporciona para trabajar con estas herramientas, aunque no se ha ampliado mucho en varios temas, se puede

consultar cualquier libro de Delphi o la misma ayuda del lenguaje para ampliar los conceptos aquí presentados ya que en este trabajo nos interesa mas las herramientas para bases de datos.

V.2.3. Programación

Esta sección la dividiremos en dos partes, la primera en la que hablaremos acerca de los fundamentos de programación empleada en Delphi y en la segunda hablaremos de cómo se pueden manejar algunos comandos para trabajar con las bases de datos.

En el capitulo pasado hablamos un poco acerca de la programación orientada a objetos. En Delphi volvemos a utilizar este concepto, recordemos que en esta programación siempre haremos referencia a objetos (que en este lenguaje hemos manejado como componentes).

Al igual que en Visual Basic el código se divide en pequeños fragmentos, recordemos que cada uno de estos fragmentos tiene una tarea especifica y solo responderá a algún suceso dentro de un componente, es decir cada uno de los componentes cuenta con sucesos que serán en los que se encontrara el código de programa. Durante la ejecución cada uno de estos sucesos solo responderá al ser llamado por el usuario, por ejemplo contaremos con el suceso OnClic en algún componente, en este suceso se teclea código, durante la ejecución cuando el usuario da un clic en ese componente se ejecuta lo que contenga este suceso. Algunos autores no consideran esta programación como la tradicional programación orientada a objetos, si bien es cierto que se considera mas programación modular por que en si todo el programa se divide en sucesos la mayoría de los comandos hace referencia a componentes, es decir a objetos. Entonces resumiendo se puede considerar una programación orientada a objetos pero no de forma tradicional sino modular.

Este tipo de programación tiene muchas ventajas, desde la velocidad hasta la detección de errores, es considerada mas rápida ya que en los programas tradicionales la ejecución se tenia que realizar desde la primer línea y hasta la ultima, en este caso solo se ejecutará lo que

responda a un suceso cuando este sea llamado. Por otro lado los errores de programación no suceden tan a menudo y si se tienen son mucho mas fáciles de detectar y de corregir. Sencillamente con Delphi programar para Windows no solo se ha convertido en una actividad mas eficiente sino también mas divertida, al estar dividida la programación en módulos se encuentra mas fáciles los errores, además los mensajes de error en este lenguaje son muy claros, aunque presenta ciertas desventajas a la detección de errores de Visual Basic.

Recordemos también que todos los sucesos quedan guardados en un solo archivo con extensión PAS. En este archivo se guardan todos los sucesos que se han tecleado y a pesar de estar en un solo archivo únicamente será ejecutado el modulo y no todo el programa.

En resumen a lo anterior, cuando la aplicación se ejecuta ocurre lo siguiente:

1. Delphi supervisa las ventanas y los componentes de cada ventana para todos los sucesos que puede reconocer cada componente (movimientos del ratón, pulsaciones, teclas y cosas similares).
2. Cuando Delphi detecta un suceso, si no existe una respuesta interna incorporada para el mismo, Delphi examina la aplicación para ver si se ha escrito un procedimiento de sucesos para ese suceso.
3. Si se ha escrito, Delphi ejecuta el código del procedimiento de sucesos y vuelve al paso 1.
4. Si no se ha escrito, Delphi espera hasta el próximo suceso y vuelve al paso 1.

Estos pasos se realizan continuamente hasta que la aplicación finaliza: normalmente para que Delphi actúe debe tener lugar un suceso. Así los programas conducidos por sucesos son reactivos mas que activos razón por la cual son mas fáciles de usar.

Una vez que ya se explico como funciona la programación explicaremos de manera sencilla como se añade el código a los sucesos y cuales serán las reglas y los comandos mas simples

para crear ciclos y condiciones. Para añadir un suceso basta con seleccionar el componente en el cual se añadirá el código. El Object Inspector se divide en dos partes, el folder de propiedades y el folder de sucesos, en este ultimo se encuentran todos los sucesos que pueden ser ejecutados con ese componente, al dar clic sobre algún suceso aparece en la pantalla el archivo con extensión PAS ubicado en la parte correspondiente a ese suceso. Cada suceso se encontrara encerrado entre las palabras clave Begin ... End; , si se cuenta con conocimientos de programación se sabrá que estas palabras son utilizadas en todos los programas de Pascal, ya que son parte de la sintaxis de mismo lenguaje (recuerde que Delphi es un lenguaje basado en Pascal).

Otra regla importante será la de poner punto y coma para dar por terminada cada sentencia utilizada en el código.

Hablemos ahora de las variables, recordemos que aunque los campos de la tabla pueden ser utilizados como variables, dentro de un sistema siempre requeriremos de otro tipo de variables auxiliares para tareas especificas. La declaración de variables en este lenguaje es muy simple ya que se realiza de la misma forma que en tradicional Pascal, es decir se utiliza la siguiente sintaxis:

<variables>: <tipo_de_variable>;

Como se trabaja con Pascal debemos seguir las mismas reglas de programación que sigue este lenguaje, es decir para la declaración de variables se hace antes del Begin por medio de un comando llamado VAR, a continuación se declaran las variables que se necesiten.
Para tener mas información acerca de las variables puede consultar cualquier libro de Delphi o incluso algún libro de Pascal.

También consulte el sistema final para ver como se hizo la declaración de variables en el sistema que estamos realizando.

Para crear ciclos y condiciones utilizaremos los comandos que ya hemos visto en los capítulos pasados cabe recordar que aunque son los mismos comandos y tiene las mismas funciones la sintaxis de estos será distinta. En el cuadro que se presenta a continuación se puede ver la sintaxis para el comando IF y los comandos WHILE y FOR.

Bloques de condición	Bucles

if <condición> then
begin
if <condición> then
<comandos>
else
<comandos>
end;
	While <condición> do
	Begin
	<comandos>
	end;
	for <inicio> to <fin> do
	begin
	<comandos>
	end;

Para los bucles tendremos dos comandos en comando While y el comando For, en el comando For marcaremos el inicio y fin del bucle, en el comando While únicamente indicaremos que ese bucle se cumplirá mientras se cumpla la condición.

Una vez que ya revisamos los fundamentos de programación hablaremos un poco acerca de los comandos que se proporcionan para el manejo de bases de datos.

Existe una serie de comandos para el manejo de bases de datos desde código de lenguaje, la mayoría de estos comandos están asociados al TdataSet, es decir a los componentes DataSource y Ttable. Cada uno de los comandos asociados a estos componentes están a su vez asociados a una serie de propiedades. La propiedad mas utilizada para estos comandos es Value, por ejemplo para poder ver el contenido de un campo una vez que nos posicionamos en el, se hará mediante el comando Tfield.Value, este comando puede proporcionar a una variable el valor de un campo para realizar operaciones con el.

Para crear ciclos y condiciones utilizaremos los comandos que ya hemos visto en los capítulos pasados cabe recordar que aunque son los mismos comandos y tiene las mismas funciones la sintaxis de estos será distinta. En el cuadro que se presenta a continuación se puede ver la sintaxis para el comando IF y los comandos WHILE y FOR.

Bloques de condición	Bucles

if <condición> then
begin
if <condición> then
<comandos>
else
<comandos>
end;
	While <condición> do
	Begin
	<comandos>
	end;
	for <inicio> to <fin> do
	begin
	<comandos>
	end;

Para los bucles tendremos dos comandos en comando While y el comando For, en el comando For marcaremos el inicio y fin del bucle, en el comando While únicamente indicaremos que ese bucle se cumplirá mientras se cumpla la condición.

Una vez que ya revisamos los fundamentos de programación hablaremos un poco acerca de los comandos que se proporcionan para el manejo de bases de datos.

Existe una serie de comandos para el manejo de bases de datos desde código de lenguaje, la mayoría de estos comandos están asociados al TdataSet, es decir a los componentes DataSource y Ttable. Cada uno de los comandos asociados a estos componentes están a su vez asociados a una serie de propiedades. La propiedad mas utilizada para estos comandos es Value, por ejemplo para poder ver el contenido de un campo una vez que nos posicionamos en el, se hará mediante el comando Tfield.Value, este comando puede proporcionar a una variable el valor de un campo para realizar operaciones con el.

El componente Ttable es muy utilizado durante la generación de código para sistemas de bases de datos, además de las propiedades que ya se han presentado cuenta con una serie de propiedades que pueden ser utilizadas para que respondan a ciertos sucesos durante la ejecución del sistema. Por ejemplo se puede utilizar la propiedad IndexName para habilitar un índice, es decir si la tabla cuenta con varios índices se puede elegir alguno de estos mediante esta propiedad. Esta propiedad es utilizada principalmente para buscar u ordenar en una tabla.

Para buscar un registro se utiliza otra propiedad asociada a la tabla, esta propiedad es FindKey, esta propiedad buscara un dato dentro de la tabla pero en el campo que este de índice en ese momento. Por ejemplo para buscar un medicamento llamado Acanol utilizamos la siguiente instrucción:

```
Table1.FindKey("Acanol")
```

Table1 es el nombre del componente Ttable y en ese momento se debe tener la tabla ordenada por medio del índice Medicamento.

Existen también comandos como BookMark que permitirán guardar el contenido de un campo o todo el contenido de un registro para poder ser utilizado en otras partes del sistema

V.2.4. Otras herramientas

Al igual que en Visual Basic, en Delphi existen muchas herramientas para todo tipo de aplicaciones, en la versión que estamos utilizando existen muchas herramientas para el manejo y creación de aplicaciones con Internet. En este punto hablaremos de algunas otras herramientas pero que podemos utilizar para agregar a nuestros sistemas de bases de datos.

En Delphi también encontramos una herramienta que genera formularios muy simples, estos formularios se generan sobre todo para manejo de bases de datos. Para poder generar un

informe de este tipo seleccionaremos la opción Form Wizard del menú Database, en la pantalla se irán solicitando ciertos datos que Delphi necesita para crear el formulario. Se pueden crear formularios utilizando tablas o Query's (SQL). Después de indicar a Delphi lo que queremos crear solicitara el nombre de la tabla a utilizar, y los campos que se añadirán a este formulario. A continuación solicitara la posición que tendrán los campos dentro del formulario (horizontal o vertical). El formulario que se genera es muy sencillo por lo que no se recomienda si el sistema que se esta creando es complejo. Esta herramienta se recomienda para aprender a manipular los datos y los componentes pero se aconseja realizar posteriormente los cambios que el programador considere convenientes para crear un formulario mas complejo.

Otra aplicación para bases de datos será el SQL Explorer, este es utilizado principalmente para crear consultas y algunas modificaciones a los registros de la tabla. Es recomendable para conocedores de lenguaje SQL. Para poder accesar a esta herramienta seleccionaremos la opción Explorer del menú Database. En esta herramienta podemos crear también Alias para algunas tablas. La pantalla de Explorer se divide en dos partes, en la parte de la izquierda se encuentran los diferentes alias que se han generado. Al dar doble clic sobre alguno de los alias en la parte de la izquierda se muestra información sobre estos alias y las tablas que contienen. En la parte izquierda podemos crear Query's, es decir consultas de tablas. Las Query's son herramientas utilizadas en algunos lenguajes para evitar trabajar con toda la tabla. Serán consultas de las mismas tablas pero que crearan filtros de datos.

Al igual que en Visual Basic, al ser Delphi un lenguaje enfocado a crear aplicaciones para Windows, una buena herramienta para presentación será la de los objetos Ole, la forma de agregar estos objetos será por medio del componente Ole del folder System. Basta con dar clic en el objeto para poder insertar un objeto en este recuadro, recuerde que los objetos que se pueden añadir dependerán de las aplicaciones que se tengan instalados en su PC.

Delphi proporciona componentes para poder accesar a bases de datos remotas, es decir trabajar con sistemas de bases de datos en un ambiente de red. Para poder hacer lo anterior

se cuenta con componentes como RemoteServer y ClientDatSet. Estos componentes servirán para crear la conexión con un sistema remoto en el cual podemos accesar a una tabla lejana.

Para poder organizar nuestro sistema contamos con una herramienta conocida como Project Manager, esta herramienta será utilizada para organizar todos los formularios que contiene nuestro sistema. Además podemos configurar el ambiente en el cual se hará la compilación del sistema. En el menú View se selecciona la opción Project Manager, se muestra en la pantalla un recuadro en el cual aparecen los formularios que contiene la aplicación. El botón Options es el encargado de configurar el ambiente, al dar clic en este botón aparecen varios folders en los cuales se puede configurar las diferentes opciones que se presentan.

La versión que estamos manejando de Delphi esta dedicada a aplicaciones con Windows 95, es decir aplicaciones de 32 bits, pero no por eso se olvida de las aplicaciones de Windows 3.X, por lo que proporciona un folder de componentes para dichas aplicaciones. Estas aplicaciones se proporcionan debido a que existen todavía muchos usuarios acostumbrados al ambiente de Windows 3.X, por lo que es bueno tener considerados estos componentes aunque trabajarlos en una aplicación de 32 bits resultaría un desperdicio.

V.3 Ventajas y desventajas ante los demás paquetes

En un principio de este capitulo se menciono que para muchos programadores Delphi es el mejor lenguaje de programación que existe en este momento. Tal vez sea asi, pero no hay que dejar afuera lenguajes como Visual Basic o Visual C. El comparar a Delphi con lenguajes como Clipper o FoxPro no resulta muy sencillo ya que en aquellos lenguajes se vio que eran para aplicaciones en DOS o Windows 3.X y se estableció las ventajas de estos lenguajes, tal vez la única desventaja que podría presentar Delphi, es que este ultimo requiere de un equipo muy potente para poder trabajar sin problemas, además de que recuerde que FoxPro y Clipper son lenguajes enfocados a la creación y manejo de sistemas de bases de datos 100%.

Se mostró en este capitulo que Delphi cuenta con herramientas para el manejo de bases de datos como lo es el DataBase Desktop y el DataBase Explorer, pero estas herramientas no cuentan con todas las características necesarias para considerarlas un manejador de bases de datos como lo seria Access o Paradox, por ello tanto en Visual Basic como en Delphi es recomendado tener instalado un manejador de bases de datos aparte de los que ofrecen los lenguajes en cuestión. Las aplicaciones que ofrecen estos lenguajes se deben de considerar únicamente herramientas auxiliares para el manejo de sistemas de bases de datos.

En los dos últimos capítulos hemos revisado los lenguajes que a mi juicio son los mejores y no solo por que permiten crear aplicaciones muy potentes sino sobre todo por la facilidad que ofrecen para crear estas. En las siguientes líneas se tratara de mencionar las principales diferencias que existen entre Delphi y Visual Basic para poder establecer así las ventajas y desventajas que ofrece Delphi, aunque es muy importante mencionar algo, las ventajas y desventajas serán a juicio del autor y pueden diferir de otras opiniones.

Cuando nace Delphi lo hace como la principal competencia de Visual Basic, de entrada Delphi presenta ciertas ventajas sobre todo en las primeras versiones (estamos hablando de la versión 3 de Visual Basic y la versión 2 de Delphi):

- Las aplicaciones desarrolladas con Delphi son básicamente tan rápidas como las desarrolladas en C o C++.
- Con Delphi pueden construirse programas ejecutables reales que incluyen DLLs.
- Pueden reconstruirse objetos reutilizables siguiendo los paradigmas de la programación orientada a objetos.

Con el paso del tiempo nacen nuevas versiones de ambos productos, y estos son mejorados al grado que es muy dificil definir cual de los dos es el mejor. Ambos lenguajes trabajan con objetos o componentes, cada uno de estos componentes cuenta con sucesos que responderán a algunos eventos sobre todo del ratón y del teclado. Ambos lenguajes cuentan con aplicaciones especiales para el manejo de bases de datos y generación de reportes. El

ambiente de ambos es muy parecido, es decir para los dos lenguajes la base de un sistema serán los formularios que estarán contenidos en un proyecto. La primera diferencia que encontramos entre estos lenguajes es que Delphi cuenta con componentes dedicados al manejo de tablas y campos de una base de datos aunque esta no es una ventaja ya que en Visual Basic los objetos pueden ser utilizados para bases de datos o para otra aplicación sin problema alguno.

La principal desventaja que ofrece Delphi con respecto a Visual Basic será que las aplicaciones para bases de datos ocupan mas espacio de almacenamiento en disco por varias razones. En Visual Basic existe una única base de datos que contiene todas las tablas e índices que pueden ser utilizados por el sistema, aunque en Delphi existe el concepto del Alias, las tablas son almacenadas fisicamente en un disco y por lo tanto ocupan mas espacio. Por otro lado Delphi genera un archivo con extensión PAS que contiene el código de los sucesos que intervienen en un sistema, recordemos que en Visual Basic este código forma parte del mismo archivo de formulario, esta es una ventaja y desventaja a la vez, la desventaja es que el archivo con extensión PAS ocupa un espacio en disco y por eso Delphi requiere de mas espacio, pero la ventaja es que para los programadores es bueno tener el código en un archivo en particular sobre todo por cuestiones de seguridad.

La programación en ambos lenguajes es modular y orientada a objetos, pero las reglas de sintaxis de Pascal son un poco mas complicadas que las reglas que sigue Basic, por lo que para una persona inexperta es mas sencillo aprender Basic que Pascal por lo que la programación en Visual Basic es un poco mas fácil que en Delphi.

Algunos autores consideran mas factible trabajar en Delphi ya que estos consideran que Delphi es mas rápido que Visual Basic, esto es considerado por que recordemos que Delphi es de la familia Borland y esta solo se dedica a crear lenguajes de programación por lo que por algunas personas son considerados mejores los lenguajes de Borland que los lenguajes de Microsoft.

Recordemos que este trabajo muestra las herramientas de ambos lenguajes y queda a consideración del lector su propia elección.

En el siguiente capitulo revisaremos otro lenguaje enfocado a aplicaciones para Windows por lo que en ese capitulo iremos haciendo algunas comparaciones entre Visual C con Delphi y Visual Basic.

V.3.1 Compatibilidad

Delphi es un software generado para Windows 95, por eso mismo es un lenguaje que puede ser compatible con varias aplicaciones de Windows. Delphi es un lenguaje de la familia Borland (para muchos programadores el mejor fabricante de lenguaje de programación), por ello mismo no es muy compatible con las aplicaciones Microsoft, es decir Access y Excel aunque se pueden introducir estas por medio del OLE.

Delphi es un lenguaje compatible pero no al 100%, por ejemplo podemos utilizar tablas de cualquier tipo para aplicaciones de bases de datos pero algunas de estas son serán 100 % compatibles con el Database Desktop, de hecho algunas veces no se podrán abrir tablas generadas en otros manejadores de bases de datos que no sean Paradox o Dbase por eso mismo es importante recordar que las tablas que serán compatibles y por lo tanto fáciles de trabajar en esta herramienta serán las tablas de Paradox y de Dbase, estos dos últimos también de la familia Borland. Al utilizar tablas de otro tipo principalmente se encuentran problemas con los índices, ya que como se vio en el punto de tablas, en este tipo de lenguajes los índices se consideran parte del mismo archivo de tabla.

Con lo mencionado en el párrafo anterior se recomienda que es preferible utilizar Paradox o Dbase para manejar las bases de datos que se utilizaran durante el sistema. En caso de utilizar otro tipo de tabla es recomendable convertirla a una tabla Paradox o Dbase con la ayuda de otro software para Windows como puede ser Access o el mismo Access.

Para las tablas de bases de datos también ofrece gran compatibilidad con SQL, de hecho ya se ha visto que cuenta con una aplicación especial para crear Query's para SQL. Hablando de bases de datos, los lenguajes deben ser compatibles con SQL ya que este es manejado para crear consultas que evitan que el programador trabaje con toda una tabla completa, ya que al crear una consulta se creara un filtro que hará que aparezcan únicamente ciertos campos que cumplan una condición especifica.

Al presentar Delphi la herramienta OLE podemos introducir en nuestro sistema cualquier objeto de todas las aplicaciones de Windows, es decir podemos agregar a nuestro sistema sonidos e imágenes o cualquier otro tipo de objeto como puede ser una tabla de Excel o una diapositiva de PowerPoint, etc. Esta es una herramienta de compatibilidad que presentan todas las aplicaciones para Windows y Delphi no se podía quedar sin ella.

Delphi genera archivos DLL, es decir librerías para Windows, estas librerías pueden ser utilizadas por otras aplicaciones y Delphi también puede utilizar librerías de Windows o generadas por otros lenguajes como puede ser Visual Basic.

La versión que estamos utilizando al ser una versión Cliente/Servidor ofrece muchas herramientas para red y aplicaciones Java e Internet, estas herramientas muestran la gran compatibilidad que ofrece Delphi en estas áreas de la computación, es decir en cuestión de red Delphi se puede conectar a cualquier servidor y utilizar aplicaciones remotas, en el caso que nos interesa se puede conectar a una base de datos remota y accesarla siempre y cuando esta cumpla con las condiciones de compatibilidad presentadas en este punto. Pero lo importante de lo anterior es tomar en cuenta que Delphi Cliente/Servidor es muy compatible para trabajar en red.

Tal vez para algunos autores Delphi no sea un lenguaje muy compatible pero es bueno recordar que Delphi proporciona todas las herramientas posibles para crear cualquier tipo de aplicación por lo que no requiere de muchos componentes externos.

V.3.2 Migración.

Al igual que en Visual Basic la migración de aplicaciones generadas en Delphi será muy complicada sobre todo por las herramientas que manejan este tipo de lenguajes. Recordemos que no es posible migrar un lenguaje en programación estructura a programación orientada a objetos y mucho menos es posible hacer lo contrario.

Dentro de las mismas versiones de Delphi algunas cosas han ido cambiando por lo que se pueden tener algunos problemas al tratar de migrar un sistema sobre todo de bases de datos a otra versión dentro del mismo Delphi, por ejemplo si contamos con un sistema en Delphi y este sistema cuenta con reportes generados en ReportSmith y es migrado a la versión 3.0 Cliente/Servidor encontramos que esta versión cuenta con otras herramientas para generar reportes y no con el ReportSmith, esto provocara ciertos problemas porque habrá que diseñar nuevamente los reportes. Pero al ser el mismo lenguaje obviamente rescataremos la mayor parte del sistema por lo que solo habrá que hacer una pequeñas modificaciones el nuevo sistema para que funcione adecuadamente con la nueva versión. Es importante recordar que no se puede migrar de una versión nueva a una versión anterior.

Para poder utilizar algunas aplicaciones realizadas en Delphi en otro lenguaje para Windows como Visual Basic únicamente se puede hacer a través de las librerías, es decir los archivos DLL, aunque estas algunas veces no funcionan muy bien sobre todo cuando son utilizadas por otro lenguaje distinto al lenguaje que las creo.

Aunque Delphi y Visual Basic son muy parecidos internamente cambian mucho y la forma de generar un proyecto o un informe es muy diferente desde la misma extensión con la que son guardados en disco, por lo que no es recomendable migrar una aplicación hecha en Delphi a Visual Basic o viceversa. Lo mismo sucederá con Visual C.

Dentro de lo que nos ocupa, las tablas que se generen tanto en Database Desktop de Delphi como en Wizard de Visual Basic pueden ser utilizadas en ambos lenguajes obviamente

convirtiendo estos datos al tipo de tabla que se necesite, esto se puede hacer con ayuda de un manejador de bases de datos como lo es Access.

En este capitulo podemos concluir con lo siguiente: Delphi es un lenguaje muy completo para todo tipo de aplicación y aunque no esta muy dedicado al manejo de bases de datos es muy sencillo generar grandes aplicaciones con tablas de datos. En este lenguaje al igual que en Visual Basic es mas recomendable utilizar todas las herramientas posibles y evitar programar mucho, evitar tener muchas líneas de código para hacer que la aplicación sea mas rápida y que sea mas fácil de realizar.

En este lenguaje se tiene que aprovechar al máximo las herramientas de presentación que se tiene recuerde que de la vista nace el amor y un sistema con una buena presentación tendrá gran aceptación por todos los usuarios. No se puede definir que Delphi sea mejor que Visual Basic o viceversa lo que si podemos decir es que Delphi es un gran lenguaje que nos permite hacer todo lo que nuestra imaginación como programadores nos proporcione y sobre todo que nos permitirá hacerlo de una manera muy sencilla.

VI. Visual C en las bases de datos

VI.1 Conceptos Generales.

Este será él ultimo lenguaje que revisaremos en este trabajo. Visual C es un lenguaje que tampoco es dedicado al manejo de las bases de datos pero eso no impide que tenga una gran potencia en la creación y manejo de sistemas de este tipo.

Visual C es de la familia Microsoft, es decir que su compatibilidad con Windows y las aplicaciones para Windows es muy extensa y eso permite que los sistemas que se generan en este lenguaje sean muy completos.

Visual C permite crear sistemas bajo el ambiente DOS y Windows, en este capitulo tomaremos las aplicaciones para Windows para poder comparar a este lenguaje con los dos anteriores, es decir Visual Basic y Delphi. Esta comparación la haremos debido a que en la actualidad los tres lenguajes mas utilizados y comerciales son Visual Basic, Delphi y Visual C. Básicamente los tres lenguajes son parecidos, aunque con diferencias muy claras principalmente en cuestión de programación, aunque en los tres la base serán los formularios, los objetos y los sucesos que responden a estos sucesos, mas adelante veremos como se manejan todos estos conceptos en Visual C.

Visual C es un entorno de programación en el que se combinan la programación orientada a objetos (C++) y el sistema de desarrollo diseñado especialmente para crear aplicaciones gráficas para Windows.

Este lenguaje incluye un conjunto completo de clases que permiten crear en forma intuitiva las aplicaciones para Windows y que permiten manejar los componentes de Windows según a su naturaleza de objetos, es decir que esta aplicación es una implementacion que utiliza las funciones del API de Windows encapsulando todas las estructuras y llamadas a dichas funciones en objetos fáciles de utilizar.

Como ya se conoce, en los lenguajes visuales la base son los formularios, Visual C para Windows esta centrado en dos tipos de objetos, ventanas y controles, las ventanas serán lo que conocemos como formularios y los controles serán los que conocemos como objetos y componentes. Al igual que en los demás lenguajes visuales cada objeto (ventanas y controles) esta ligado a un código inactivo hasta que se dé el suceso que lo activa. Por ejemplo podemos programar un botón (objeto que se puede pulsar) que responda a un clic del ratón.

Hasta este momento podríamos pensar que Visual C es un lenguaje que al igual que Visual Basic o Delphi crea las aplicaciones de forma muy sencilla, pero la realidad es otra, tal vez la principal desventaja que tiene Visual C es que este es un lenguaje no muy amigable para el programador, por ejemplo cuando en Visual Basic o Delphi iniciábamos la aplicación automáticamente en pantalla aparecía la ventana de formulario en la que podríamos trabajar, en Visual C no sucede esto sino que hay que realizar otras cosas para poder crear la ventana, esto lo veremos de manera mas amplia en la sección de presentaciones. Por otro lado los controles no cuentan con tantos sucesos y propiedades como en los lenguajes anteriores por lo que si requerimos sucesos mas especializados será responsabilidad del programador realizarlos.

A pesar de lo anterior Visual C es un lenguaje muy competitivo, y esto se debe principalmente a los antecedentes que este lenguaje tiene. Lenguaje C fue uno de los mejores lenguajes en su tiempo (aunque algunas empresas aun utilizan este), de hecho C fue el primer lenguaje que se dedico de forma directa a la programación orientada a objetos. Si hablamos de los lenguajes base de los tres visuales que estamos analizando veremos que C es un lenguaje mas completo que Pascal o Basic (estos últimos lenguajes base de Visual Basic y Delphi), de ahí que muchos programadores prefieran trabajar con Visual C que con los otros. En este capitulo iremos viendo las ventajas y desventajas que nos ofrece este lenguaje para poder generar una conclusión al final de este capitulo de que tan bueno resulta ser Visual C.

Visual C proporciona muchas herramientas para la creación de las aplicaciones, en este trabajo iremos mencionando las que se utilizaran para crear los sistemas de bases de datos. Por principio de cuentas mencionaremos que Visual C proporciona una librería de clases conocida como MFC.

Estas librerías serán las que se encargaran de que los programas fuentes sean mas cortos ya que tienen implementadas muchas de las labores que antes tenia que hacer el programador. En lo que nos interesa el MFC ofrece métodos para el acceso a una base de datos, estas clases son: CdaoWorkspace, CdaoDatabase, CdaoTableDef, CdaoRecordset, CdaoQueryDef. Estas se irán explicando mas adelante, sobre todo en el punto de tablas que será donde se manejen.

A pesar de ofrecer herramientas para al manejo de las bases de datos, desafortunadamente Visual C no proporciona un manejador de bases de datos, es decir que en Visual C se puede generar un sistema que maneje las bases de datos pero esta se tiene que crear en otra aplicación. Esto es una gran desventaja para Visual C, pero ya veremos mas adelante como nos adecuamos a este lenguaje.

Para poder trabajar con todas las clases, menús, ventanas, etc. se utiliza una ventana conocida como WorkSpace. Esta ventana se encuentra siempre en la parte Izquierda de la ventana y esta dividida en cuatro folders: ClassView, ResourceView, FileView, InfoView. Cada uno de estos folders contiene las distintas herramientas que hemos insertado a nuestra aplicación.

En resumen Visual C es un lenguaje no tan sencillo pero que ofrece gran potencia, en este capitulo se mostraran las herramientas para crear un sistema de bases de datos exclusivamente, por lo que pueden quedar afuera varias herramientas para otro tipo de aplicación, pero no podemos ampliar mucho estos conceptos por que este trabajo no esta dedicado a eso.

VI.1.1 Características Técnicas.

Las aplicaciones que crearemos en Visual C serán bajo el ambiente de Windows por lo que al igual que en los lenguajes anteriores se recomienda tener conocimientos en Windows, además de los conocimientos básicos se recomienda al programador tener conocimientos en el manejo de librerías.

Por otro lado se recomienda conocer bien lenguaje C, en este caso no solo se debe conocer superficialmente el lenguaje ya que en Visual C se programa mas que en los lenguajes anteriores. Se recomienda tener conocimientos de programación orientada a objetos en laguna de las versiones de C++ ya que el código que se utiliza en este lenguaje es precisamente orientada a objetos y no de programación estructurada. Lo anterior es muy importante ya que C++ requiere de una sintaxis muy particular, esto lo veremos mas detallado en el punto de programación peor es importante tomar en cuenta estos puntos.

Visual C también contiene dos versiones, la versión estándar y la versión profesional, ambas contienen el mismo compilador del lenguaje aunque la versión profesional contiene mas herramientas sobre todo para aplicaciones diferentes de las bases de datos, a pesar de ello es mas recomendable tener instalada la versión profesional completa para aprovechar las herramientas extras que esta versión pueda proporcionar.

Para el tipo de sistemas que nos interesa tomaremos en cuenta que como ya se menciono Visual C no contiene ningún manejador de bases de datos dentro de las aplicaciones que proporciona. Por lo anterior es recomendable contar con algún manejador instalado. Al ser Visual C un lenguaje de la familia Microsoft cuenta con gran compatibilidad sobre todo con manejadores de la misma familia, la mayoría de los autores que han escrito sobre visual C recomiendan contar con Access, FoxPro o Dbase, este ultimo de Microsoft por que la versión de Borland puede traer varios problemas de compatibilidad. En lo particular se recomienda tener instalada la ultima versión de Access.

Aunque se podría trabajar con manejadores como FoxPro o Dbase se recomienda Access por la forma en que maneja las bases de datos, recordemos que maneja un solo archivo que contiene todas las tablas, índices y consultas, a diferencia que los otros manejadores requieren de un archivo por cada tabla o índice que se requiera dentro del sistema.

En la creación de sistemas de bases de datos se pueden utilizar dos métodos de acceso a una base de datos. A parte de las clases que se mencionaron el punto anterior (estas clases son dedicadas para DAO) también se puede accesar a una base de datos mediante ODBC. Nosotros trabajaremos mediante el primer método que es el mismo que se utiliza en Visual Basic. Pero en caso de querer utilizar ODBC tenemos que tomar en cuenta que se requiere del gestor de controladores (ODBC.DLL) y el controlador ODBC de la base de datos a la que se quiere acceder. Lo anterior puede resultar transparente para el programador ya que el ODBC es instalado directamente por Windows dentro del panel de control pero es bueno tenerlo en cuenta en caso de tener algunos problemas.

VI.1.1.1 Requerimientos en hardware.

Los requerimientos que tendrá Visual C para trabajar de forma adecuada pueden resultar en una desventaja de este lenguaje, pero debido a la gran potencialidad que proporciona requiere de un muy buen equipo para poder generar el tipo de aplicaciones que el usuario desee. Por otro lado debemos tomar en cuenta que Visual C es un lenguaje directamente para Windows 95 y no para Windows 3.X, al mencionar esto debemos de tomar en cuenta los requerimientos mínimos que necesita Windows 95.

Para realizar aplicaciones sencillas con Visual C los requerimientos básicos son:

Procesador 486 o superior.
16 Mb en memoria RAM
Monitor Color Vga

Disco duro de 500 Mb o mayor.
Unidad de disquete 3 ½
Tarjeta de vídeo Vga
Mouse y Teclado

Tome en cuenta que estos requerimientos son los básicos, para poder crear un buen sistema es recomendable tener un equipo mejor al que se acaba de presentar. Debemos de tener en cuenta varias cosas, en Visual Basic o Delphi aunque era recomendable tener un manejador de bases de datos instalados no era necesario, en Visual C este manejador es obligatorio, también es importante mencionar que las aplicaciones en Visual C generan mas archivos que en los dos lenguajes anteriores y estos archivos requieren de espacio suficiente en disco duro.

Para crear un buen sistema de bases de datos el equipo que se acaba de presentar podría provocar algunos problemas sobre todo en la velocidad a la que se hiciera y ejecutara el sistema. En este caso se recomienda un equipo mejor al presentado pero basados en los requerimientos de instalación del lenguaje se presenta la información antes mencionada.

VI.2 Principales Herramientas.

En este trabajo utilizaremos la versión 5.0 Profesional de Visual C++, por lo que a partir de aquí nos dedicaremos a hablar de versión.

Visual C++ al igual que los dos lenguajes anteriores cuenta con muchas herramientas para la creación de aplicaciones en Windows. Desdichadamente Visual C++ no proporciona muchas herramientas para la creación de sistemas de bases de datos. Pero veremos que con las herramientas que se nos presentan podemos crear buenos sistemas.

La principal herramienta que proporciona Visual C es utilizada para crear cualquier tipo de sistemas pero para las bases de datos proporciona ciertas partes para la configuración. Estas

configuraciones las veremos mas adelante, en este punto nos dedicaremos a explicar de forma teórica La herramienta de la que estamos hablando se conoce como MFC AppWizard.

El AppWizard es un poderoso asistente que escribe por usted el código estructural de los programas. Es recomendable utilizar el AppWizard siempre que inicie la creación de un sistema, aunque sea un programador experto se pueden aprovechar muy bien las diferentes configuraciones que nos proporciona este. Cuando se elige la opción New del menú File siempre se activa la pantalla del AppWizard, por lo que siempre se iniciara la creación de un sistema con esta herramienta.

El AppWizard creara todos los archivos que utiliza Visual C para el manejo de sus aplicaciones, a estos archivos serán a los que se les harán modificaciones posteriormente para adecuarlos a nuestro sistema. En la primer pantalla del AppWizard se le indicara el nombre de la aplicación, el tipo y el subdirectorio en donde se guardaran los archivos, cabe mencionar que Visual C crea una carpeta especial en la ruta que se le indique en la que guardaran todos los archivos que se utilicen. Mas adelante aprenderemos a usar el AppWizard.

Cuando se crea una aplicación en Visual C++ se crean una serie de clases que contendrán todo el código que el programa necesite, para las bases de datos se crean una serie de clases especiales que ya se habían mencionado pero que analizaremos mas adelante. Lo importante aquí es que Visual C++ proporciona otro asistente conocido como ClassWizard, este permite agregar código a los controles de su programa. Recordemos que en los lenguajes anteriores cada componente o objeto contenía sucesos que contenían cierto código de lenguaje y dicho código solo respondía cuando el suceso era ejecutado, en Visual C++ sucede lo mismo (aunque no se contienen tantos sucesos como en los lenguajes anteriores). Por medio del ClassWizard podemos ver los objetos con los que cuenta nuestra ventana y los sucesos que corresponden a cada objeto. Por medio de esta herramienta podemos

agregar código a los sucesos. Es importante mencionar que este código será guardado en las clases que previamente se crearon.

Además de contar con las herramientas ya mencionadas, Visual C++ proporciona una serie de herramientas para proporcionar una mejor presentación a nuestras aplicaciones, como por ejemplo cambiar los iconos que se presentan en alguna de nuestras ventanas, agregar opciones al menú que se va a presentar, etc. Estas herramientas la iremos revisando según lo necesitemos.

VI.2.1 Manejo de Archivos

Como ya se menciono al utilizar el AppWizard se crean todos los archivos necesarios para el manejo de nuestro sistema, recuerde que todos los archivos se guardan en una carpeta en particular. El nombre de esta carpeta será el mismo que nosotros definimos como el nombre de nuestra aplicación. Supóngase el caso de que nuestra aplicación se llama recetario. En la ruta que se le indica en el AppWizard crea una carpeta llamada recetario y en esta deposita todos los archivos que necesita la aplicación.

A continuación se presenta una lista con los archivos que se crean y el contenido de cada uno de ellos. Para estos archivos nos estamos basando en que la aplicación se llama recetario:

Recetario.h. Es el fichero de cabecera principal de la aplicación, Incluye otros ficheros de cabecera necesarios para la aplicación. Incluido Resource.h (aunque esta ultima también aparece como archivo independiente en la lista) y declara la clase aplicación CrecetarioApp.

Recetario.cpp. Es el fichero fuente principal de la aplicación contiene la definición de clase CrecetarioApp.

Recetario.rc. Contiene una lista de todos los recursos que la aplicación utiliza, como iconos, bitmaps, menús, etc. Este fichero puede editarse directamente desde AppStudio. Este archivo utiliza otra serie de archivos que se han guardado en otra carpeta con el nombre RES, los siguientes dos archivos pertenecen a esta carpeta.

Recetario.ico. Contiene el icono que va a utilizar la aplicación.

Recetario.rc2. Contiene los recursos que no son editados con AppStudio.

Recetario.clw. Este fichero contiene información utilizada por el ClassWizard para editar las clases existentes, añadir nuevas clases, generar los mapas de mensajes y los mapas de datos de las ventanas de dialogo.

Mainfrm.h, Mainfrm.cpp. Estos dos archivos son para la ventana marco principal. Contienen la declaración y definición de la clase CmainFrame, que esta derivada de CframeWnd y controla las características de una SDI (single Document Interfase).

Recetariodoc.h, recetariodoc.cpp. Contienen la declaración y definición de la clase CrecetarioDoc (el documento).

Recetarioview.h, recetarioview.cpp. Contienen la declaración y definición de la clase CrecetarioView (la vista del documento). Los objetos vista son utilizados para ver los objetos documento.

Recetarioset.h, recetarioset.cpp. Contiene la declaración y definición de la clase CrecetarioSet. Esta clase deriva de CdaoRecordset.

Stdafx.h, Stdafx.cpp. Son utilizados para construir el fichero de cabecera precompilado denominado stdafx.pch y un fichero precompilado de tipos (PCT) denominado atdafx.obj.

Como puede ver se crean muchos archivos y no es recomendable por ningún motivo borrar alguno de estos. La mayoría de estos archivos no ocupan mucho espacio en disco pero cuando se va agregando código o clases estos archivos van creciendo, además se crean otros archivos durante la compilación y ejecución del programa que pueden ocupar mas espacio.

También es importante mencionar que como nuestro sistema es de bases de datos no podemos dejar fuera los archivos que contienen los datos. Es importante recordar que se recomienda trabajar con Access por lo que ya sabemos que se deberá de contar con un archivo con extensión MDB y los archivos auxiliares y de seguridad que la base de datos necesita. Es recomendable que el archivo de bases de datos se encuentre almacenado en la misma carpeta que los otros archivos para evitar tener desorganizado nuestro directorio.

Cuando se hace la compilación del programa se crea una carpeta con el nombre DEBUG, esta contiene los archivos de carga o mejor conocidos con extensión OBJ además del archivo ejecutable, recuerde que la mayoría de los archivos mantienen el nombre de la aplicación por lo que en nuestro caso el archivo ejecutable se llamaría Recetario.exe. Además se crea un archivo OBJ para cada uno de los archivos con extensión CPP que se presentaron anteriormente.

Durante este capitulo se ira haciendo mención a estos archivos cuando los utilicemos para que se entienda mejor su contenido. También puede consultar los archivos en el sistema final para verificar su contenido.

VI.2.1.1 Tablas

En este punto analizaremos como se trabajan los campos y registros dentro de una tabla con Visual C++. Recordemos que para nuestro sistema utilizaremos Access por lo que es importante conocer el funcionamiento de los registros es esta aplicación. Esto es importante ya que en este lenguaje no se explicara como crear una tabla sino como trabajar con ella una

vez ya creada, esto ultimo porque recordemos que Visual C++ no proporciona una herramienta para crear tablas.

En Visual C++ generalmente se utilizan los siguientes tipo de datos para los campos:

- Texto: Cualquier tipo de carácter por lo general con una longitud menor a 255 caracteres.
- Memo: Aceptan también cualquier tipo de carácter pero para proporciones de texto mayores.
- Numero: Solo aceptara datos numéricos que se aplican luego a operaciones de calculo.
- Fecha/hora. Son de un tamaño establecido, en la mayoría de los casos se transforman internamente en formato de números y solo al mostrarlos se convierten en datos horarios.
- Moneda: Acepta valores monetarios y numéricos que se pueden utilizar en operaciones matemáticas.
- Si/No: tipo de datos para campo con valores de tipo booleano.

Al igual que en los demás lenguajes al crear una tabla se debe de tomar en cuenta que campo utilizaremos como índice. Habitualmente el primer campo siempre es tomado como llave principal de la tabla. Recuerde que el índice es muy importante sobre todo para cuando queremos realizar relaciones entre tablas, pero además es muy utilizado para realizar búsquedas y ordenaciones.

Como ya se ha mencionado cuando se crea una aplicación con las bases de datos, se crean ciertas clases que son exclusivas para este tipo de aplicación. A continuación explicaremos cada una de estas clases:

CdaoWorkspace: esta clase alberga la funcionalidad de dirigir una sesión con una base de datos desde el alta en una base de datos. En los casos normales solo se utilizara un Workspace. Esta clase permite contener varias bases de datos abiertas.

CdaoDatabase: Esta clase establece una conexión con una base de datos. Si desea abrir varias bases de datos tendrá que crear varias instancias de CdaoDatabase incrustadas en un Workspace.

CdaoTableDef: Contiene la definición de una tabla de base de datos. Por tanto con ella podrá examinar la estructura de los campos de una tabla, determinar si se pueden editar campos determinados, crear tablas, etc.

CdaoRecordset: es una colección de registros en una tabla, que puede constar de los campos de una o varias tablas de la base de datos. La tabla de un recordset no es mas que una forma temporal de representación de los datos. Al crear esta clase se crean automáticamente las anteriores.

Existen tres tipos de instancias de CdaoRecordset:

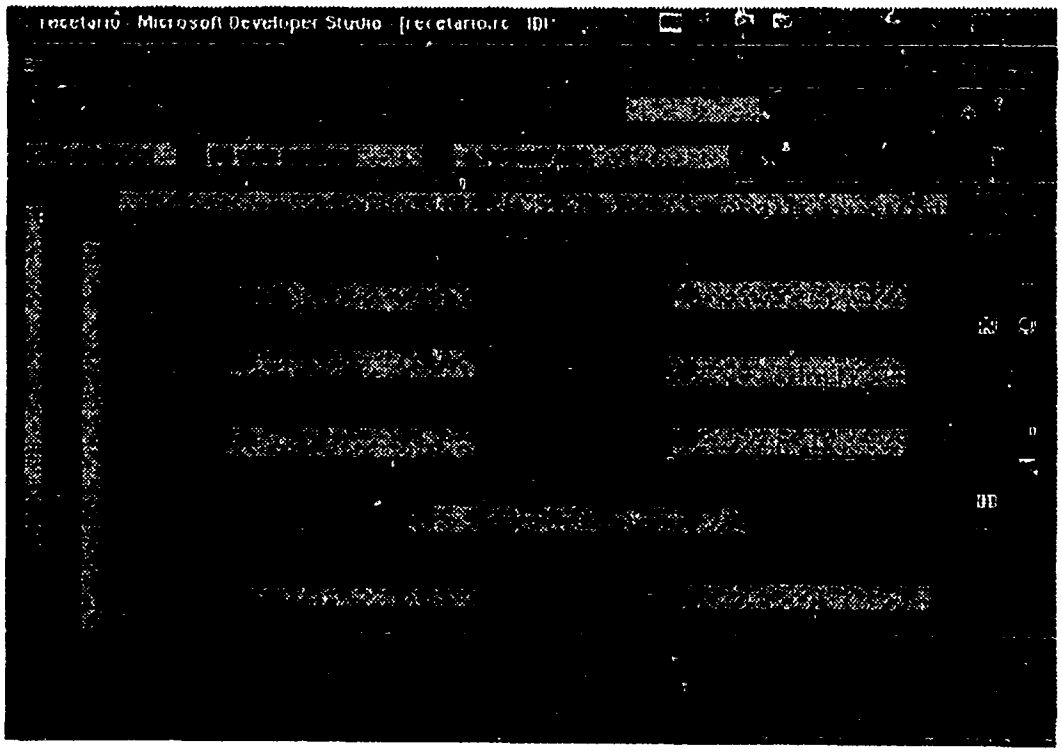
1. Con un table-type-recorset, puede examinar los datos de una tabla, modificar, eliminar o añadir nuevos registros.
2. Un dynaset-type-recordset es capaz de modificar datos, cuyos campos no se encuentran en una sola tabla, sino en varias tablas de base de datos.
3. Los snapshot-type-recordset representan copias estáticas de datos que se pueden utilizar para búsquedas o para crear informes puesto que son de solo lectura. También puede contener campos de varias tablas de base de datos.

CdaoQueryDef: podrá crear nuevas tablas y con ellas nuevas representaciones de los datos de su base de datos. Sin embargo este tipo de instancias tienen mas el carácter de una consulta a una base de datos, es decir en la mayoría de los casos están vinculadas a una condición para los registros y campos que debe contener una tabla.

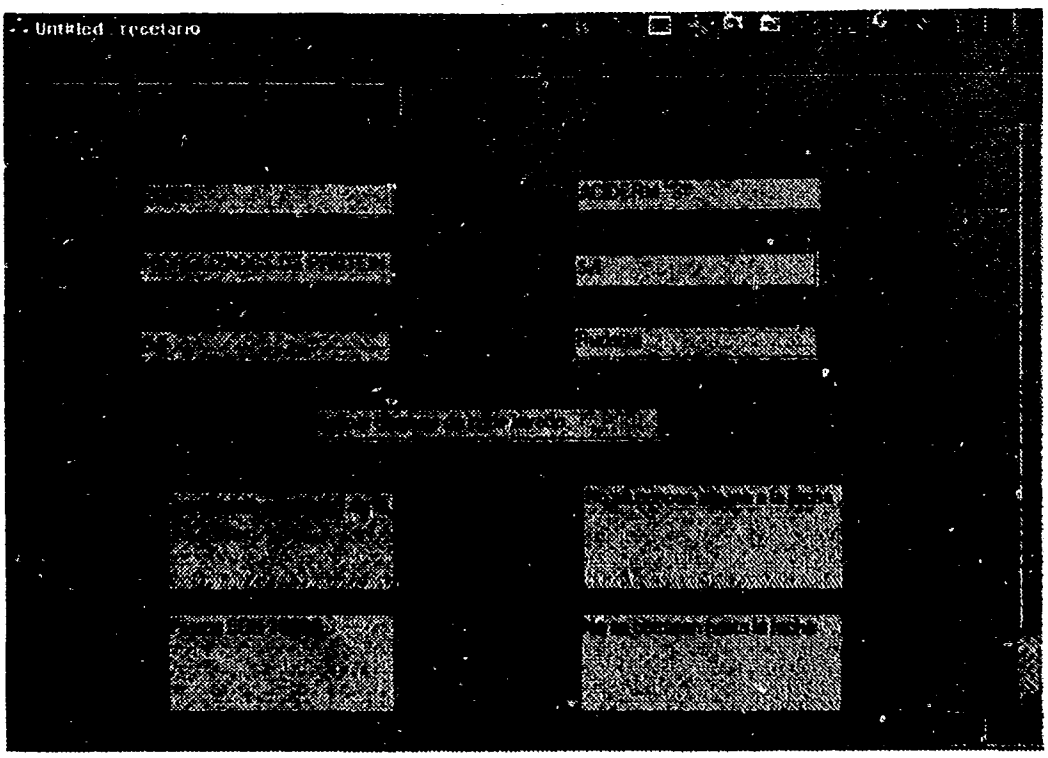
Para el manejo de las bases de datos es muy importante tener en cuenta la clase Crecordset. En el ejemplo que utilizamos en el punto de manejo de archivos, la clase seria Crecetarioset

que esta definida en el archivo recetarioset.h, puede consultar este archivo y se dará cuenta de que contiene un campo de datos para cada campo de la tabla. Durante la ejecución del programa estos campos contienen siempre los valores del registro actual.

Una vez que hemos revisado los conceptos básicos que debemos tomar en cuenta para la creación de un sistema de bases de datos hablaremos de cómo manejarlos desde Visual C++. Para hablar de cómo se manejan los datos tomaremos en cuenta que ya se cuenta con una base de datos creada y que utilicemos la tabla de medicamentos como ejemplo. También pensaremos que ya se utilizo el AppWizard para crear la aplicación con la base de datos y que ya contamos con la ventana principal que se presenta en la siguiente figura (para saber como llegamos a esta figura consulte el punto de manejo de presentaciones de este capitulo):



Si en este momento ejecutáramos el sistema, la pantalla se mostraría como en la siguiente figura:



Note en esta ultima figura que se añaden en la parte superior una serie de botones ya conocidos por que los manejamos en otros lenguajes. Recordemos que estos botones permiten desplazarse por medio de la tabla, mostrando los registros en el área que hemos destinado para ello. Hasta el momento solo hemos creado una ventana para la visualización de registros pero a continuación mostraremos como se insertan, borran y ordenan los registros.

Primero mencionaremos de forma teórica que procesos se siguen para poder insertar, modificar o eliminar registros.

Al editar un registro primero se modifica el texto en el cuadro de texto. Tan pronto el usuario modifica el registro actual se deben desarrollar los siguientes procesos:

- El registro se prepara para la edición activando el método Edit de la clase recordset.

- Se activa el método de la clase View UpdateData con el que se guardan las modificaciones de los elementos de control de los campos de la clase recordset.
- Se activa el método Update de la clase recordset con el que se actualiza el registro de la tabla con los valores de los campos.

Acciones similares se realizan para insertar un nuevo registro a una tabla:

- Primero se activa el método AddNew de la clase recordset que crear un registro nuevo y vacío. Se activa además el modo de edición para este nuevo registro.
- Posteriormente se tiene que asignar nuevos valores a los campos de la clase recordset. Este proceso se realiza de nuevo con el método UpdateData de la clase View.
- Con el método Update de la clase recordset, se guardan en la tabla los cambios en los campos.

Para eliminar un registro es necesario activar el método Delete de la clase Recordset y cuidar de que se determine un registro actual nuevo. Note que en los tres casos tiene una gran trascendencia la clase recordset. Ya que revisamos de forma teórica este proceso revisaremos ahora los procesos reales que debemos seguir para agregarle la opción de insertar y borrar a nuestro sistema. Conociendo estos paso podemos darnos cuenta de cómo se manejan las tablas en Visual C++.

Añada en el menú las dos opciones que requerimos, es decir insertar y borrar (para verificar como se insertan las opciones al menú consulte el punto menús de este capitulo). Los identificadores que se deberán utilizar son ID_RECORD_ADD e ID_RECORD_DELETE. A continuación realice los siguientes pasos:

1. En la ventana de Workspace, pulse con el botón derecho del Mouse en el nombre de clases CrecetarioView y seleccione Add Variable del menú.
2. Asigne a la variable el tipo BOOL y como identificador m_AddMode. Esta variable será auxiliar, esta será la forma de agregar variable s a nuestras clases.

3. En la barra de herramientas se encuentra una barra en la cual por medio de menús verticales podemos elegir la clase que queremos visualizar. En este caso selección le clase CrecetarioView y modifique el constructor conforme a lo siguiente:

```
CRecetarioView::CRecetarioView()
     : CDaoRecordView(CRecetarioView::IDD)
{
    //{{AFX_DATA_INIT(CRecetarioView)
    m_pSet = NULL;
    //}}AFX_DATA_INIT
    m_AddMode = FALSE;
}
```

Realmente si visualiza esta clase vera que solo debe de insertar una línea. Es importante mencionar que las mayúsculas y minúsculas son diferentes en Visual C++ por lo que deberá tener cuidado al teclear código.

Ahora programaremos las rutinas para las opciones que acabamos de añadir. En primer lugar para insertar registros.

1. Abra el archivo RecetarioView.cpp.
2. En la barra de herramientas bajo Object ID, seleccione ID_RECORD_ADD y bajo messages seleccione COMMAND.
3. Confirme el cuadro de dialogo Add Member Function.
4. Escriba las instrucciones siguientes para el método OnRecordAdd:

```
void CRecetarioView::OnRecordAdd()
{
    if ( m_AddMode )
         OnMove( ID_RECORD_FIRST );
    OnGetRecordset()->AddNew();
    m_AddMode = TRUE;
    UpdateData( FALSE );
}
```

3. En la barra de herramientas se encuentra una barra en la cual por medio de menús verticales podemos elegir la clase que queremos visualizar. En este caso selección le clase CrecetarioView y modifique el constructor conforme a lo siguiente:

```
CRecetarioView::CRecetarioView()
     : CDaoRecordView(CRecetarioView::IDD)
{
    //{{AFX_DATA_INIT(CRecetarioView)
    m_pSet = NULL;
    //}}AFX_DATA_INIT
    m_AddMode = FALSE;
}
```

Realmente si visualiza esta clase vera que solo debe de insertar una línea. Es importante mencionar que las mayúsculas y minúsculas son diferentes en Visual C++ por lo que deberá tener cuidado al teclear código.

Ahora programaremos las rutinas para las opciones que acabamos de añadir. En primer lugar para insertar registros.

1. Abra el archivo RecetarioView.cpp.
2. En la barra de herramientas bajo Object ID, seleccione ID_RECORD_ADD y bajo messages seleccione COMMAND.
3. Confirme el cuadro de dialogo Add Member Function.
4. Escriba las instrucciones siguientes para el método OnRecordAdd:

```
void CRecetarioView::OnRecordAdd()
{
    if ( m_AddMode )
         OnMove( ID_RECORD_FIRST );
    OnGetRecordset()->AddNew();
    m_AddMode = TRUE;
    UpdateData( FALSE );
}
```

Ahora se tiene que programar una intervención en el desarrollo normal del método OnMove para desactivar el método Add en caso de que se produzca un cambio de registro.

1. En la barra de herramientas seleccione el identificador CrecetarioView y en el cuadro combinado Message, OnMove. Transmita la demanda pulsando el botón Si.
2. Escriba el texto siguiente para este método:

```
BOOL CRecetarioView::OnMove(UNIT nIDMoveCommand)
{
  if ( m_AddMode )
    {
      if ( !UpdateData() )
          return FALSE;
      m_pSet->Update();
      m_pSet->Requery();
      UpdateData( FALSE );
      m_AddMode = FALSE;
      return TRUE;
    }
    else
       return CDaoRecordView::OnMove(nIDMoveCommand);
}
```

Veamos ahora los pasos que tenemos que hacer para poder borrar registros de la tabla. El proceso que tenemos que realizar es como sigue:

1. Seleccione de la barra de herramientas el evento Command con él ID ID_RECORD_DELETE.
2. Vamos a agregar las siguientes líneas al método OnRecordDelete:

```
m_pSet->Delete();
m_pSet->MoveNext;
if( m_pSet->IsEOF() )
  m_pSet->MoveLast();
```

```
if (m_pSet->IsBOF() )
  m_pSet->SetFieldNull( NULL );
UpdateData( FALSE );
```

En este caso se realizan dos consultas, la primera de ellas para determinar si se alcanzo el final de la tabla, en este caso se define como registro actual el ultimo registro de dicha tabla. Por otro lado la segunda consulta es para determinar el caso en que después de borra haya quedado vacía la tabla, en este caso todos los campos tomaran un valor de 0. Note que a pesar de que hemos dicho que Visual C++ nos es un lenguaje enfocado a las bases de datos, existen comandos que permiten manipular las tablas. Esto ultimo es gracias al Recordset.

Ahora revisaremos como se ordenan los registros en las tablas por medio de Visual C++. Al contener una tabla con índices es obvio pensar que ya se encuentra ordenada y de hecho es así, pero recordemos que algunas veces necesitamos ordenar los registros de alguna manera en particular y no necesariamente por medio del índice primario.

En la clase recordset existe un campo para clasificar los registros, pero medio de este campo se le indica al sistema el campo que se hará cargo de la secuencia de orden. Este campo se conoce como m_strSort.

Utilizar lo anterior es muy sencillo, basta con agregar una línea en algún suceso en el cual requerimos una clasificación. La sintaxis que llevara la línea puede ser como sigue:

m_pSet->m_strSort = 'MEDICAMENT'

En este caso le indicaremos que la tabla se ordenara por medio del campo medicament, este campo puede ser o no índice de la tabla, en Visual C++ tenemos esta ventaja.
Cabe mencionar que al principio de esta explicación se indico que se añadieran dos opciones en el menú, en el caso de que quisiéramos agregar los botones correspondientes a los procesos anteriores es necesario realizar los pasos siguientes:

1. En el folder ResouceView de la ventana Workspace pulse doble clic en la opción Toolbar.
2. Pulse doble clic en la opción IDR.MainFrame.
3. En la ventana del lado derecho aparece la barra de herramientas que se muestra en nuestro sistema. La forma de agregar un botón es muy simple. En esta barra existe un botón vacío, sobre este pulse la tecla Supr. Esto duplicara el botón, acomódelo en la forma deseada y dibuje el símbolo que llevara este botón (el dibujo se realiza de la misma forma que en Paint).
4. Una vez que el botón esta dibujado pulse doble Clic sobre este. En la ventana que aparece se le indicara el ID al cual se hará referencia con este botón.

Para el ejemplo que estamos haciendo se añadirían dos botones, uno que indique añadir y otro que indique borrar. Par el primero se le indicada el ID ID_RECORD_ADD y para el de borrar se le asignara ID_RECORD_DELETE.

Note que en Visual C++ el manejo de las tablas no es tan fácil como en los otros lenguajes pero si es muy efectivo, Visual C++ permite realizar todos los procesos clásicos con las bases de datos y eso facilita la creación de un sistema de este tipo, solamente que el programador deberá tener conocimientos en programación ya que como se ha visto en este lenguaje se programa mas que en los dos anteriores.

VI.2.1.2 Etiquetas

Las etiquetas las revisaremos en este capitulo junto con los informes ya que Visual C++ no contiene un editor de etiquetas pero cuenta con dos herramientas que permiten de alguna forma agregar etiquetas e informes a nuestro sistema, las etiquetas en Visual C++ son básicamente igual que los informes por lo que preferiremos revisarlas de forma mas amplia en el siguiente punto. Revise la sección de informes para analizar esto.

VI.2.1.3 Informes

La impresión de informes y etiquetas desde Visual C++ resulta algo complicado sobre todo para programadores que no tengan mucho conocimiento de programación en C++ orientada a objetos y no conozcan mucho del manejo de clases de Visual C++.

Desafortunadamente como Visual C++ no contiene un manejador de bases de datos, tampoco contiene una herramienta para crear informes y etiquetas. Esto dificulta un poco la creación del sistema pero no por ello se hace imposible realizarlo.

Para poder imprimir las etiquetas y los informes desde nuestro sistema contamos con dos formas de hacerlo: utilizando herramientas ActiveX y utilizando las clases de impresión que ofrece Visual C++.

Revisemos la primera opción. Esta opción resulta más sencilla debido a que por medio de esta evitamos un poco la programación pero existe la desventaja de que debemos utilizar un generador de informes externo a Visual C++. En este trabajo ya revisamos un generador de informes que fue Crystal Reports, este generador es muy completo y es el que recomendamos utilizar.

Las herramientas ActiveX son una forma de mostrar la compatibilidad de Visual C++, esta herramienta permite introducir a nuestro sistema componentes realizados en otras aplicaciones de Windows, por ejemplo por medio de ActiveX podemos agregar a nuestro sistema una tabla de Excel, una diapositiva de PowerPoint, un reporte de Cyistal, etc.

Al introducir algún componente de otra aplicación será un objeto mas dentro de nuestro formulario o ventana, al igual que los demás componentes, este consta de una serie de propiedades que permite configurar cual será la presentación y la utilidad que este tenga para nuestro sistema.

El manejo de los objetos ActiveX es muy sencillo en Visual C++, al registrar uno de estos objetos el objeto se agregara a nuestra barra de herramientas junto con los objetos ya establecidos por el lenguaje. Por principio de cuentas debemos mencionar que los controles ActiveX serán archivos con extensión OCX y para que puedan ser ocupados en Visual C++ es necesario que se encuentren en la carpeta \\Windows\System\ en el caso de Windows 95 o 98, en caso de que algún control no se encuentre en esta carpeta puede ser copiado a ella porque de lo contrario no puede utilizarse. Visual C++ copia algunos controles ActiveX a esta carpeta durante su instalación.

Para agregar un objeto ActiveX a alguna aplicación deberá hacer lo siguiente:

- Utilice el AppWizard para crear la aplicación como comúnmente se hace.
- En el paso 2 del AppWizard aparece una casilla de verificación con la leyenda ActiveX Controls, esta casilla deberá estar seleccionada.
- Continúe la creación de su aplicación según las necesidades del proyecto.
- Una vez que ya tiene creada la plataforma de la aplicación registremos el control en Visual C++.
- Seleccione ActiveX Control Test Container en el menú Tools de Visual C++
- Seleccione Register Controls en el menú File del Test Container.
- Haga clic en el botón Register.
- Aparecerá una pantalla como si fuera a Abrir un archivo, seleccione de la Carpeta \\Windows\System el archivo OCX que corresponda al control que se agregara.
- En ese momento aparece una lista en la que se muestra el control ya registrado.
- Salga del Test Container.
- Entre a la ventana de diseño de la ventana o formulario.
- En el menú Project seleccione la opción Add to Project, en el menú que aparece seleccione la opción Components and controls.
- En el cuadro de dialogo que aparece elija la opción Registered ActiveX Controls.
- Seleccione el elemento que corresponda al control deseado.

- Confirme el cuadro de mensaje que aparece.
- En la barra de herramientas aparecerá el control ActiveX que se eligió.
- Puede añadir este control como cualquier otro, por ejemplo como un botón de comando.

La propiedades que se puedan configurar dependerán del control que se ha insertado.
En el caso de Crystal Reports que es lo que nos importa las propiedades y su explicación se presentan en la siguiente tabla.

Propiedad	Función
CopiesTo Printer	Esta propiedad establece el numero de copias a imprimir en el caso de que la salida del informe sea la impresora. Basta con indicar por medio de un numero a la propiedad las copias que se imprimirán.
PrintFileName	Como en cualquier herramienta de impresión existe la posibilidad de mandar a imprimir a un archivo, en la pantalla o en impresora, esta propiedad se utilizar en el caso de que se quiera mandar a imprimir a un archivo, en esta propiedad únicamente se le indica el nombre del archivo de salida.
PrintFileType	Esta propiedad definirá el tipo de salida, es decir el formato de la salida, esta propiedad podrá tomar 6 valores: 0 Formato de registro 1 Separado por tabulaciones 2 Formato de texto 3 Formato DIF 4 Valor separado por comas 5 Texto separado por TAB.
Destination	Esta propiedad determina cual será el medio de salida, puede tomar tres valores: 0 Ventana 1 Impresora 2 Archivo
SelectionFormula	Por medio de esta propiedad se puede imprimir solamente un grupo de registros. La formula que se teclea en esta propiedad deberá ser una formula de SQL.

Las propiedades que se acaban de presentar son las propiedades que se pueden configurar desde el código del programa, pero existen otras propiedades que se establecen desde que se esta realizando el informe, estas propiedades no pueden ser cambiadas durante la ejecución, entre estas propiedades están: el tamaño de la hoja , tamaño de la ventana, estilo, tipo de letra, etc. Estas propiedades se deben de tomar muy en cuenta sobre todo para cuestiones de presentación preliminar.

Hablaremos a continuación como se maneja la impresión desde el lenguaje para que se pueda comparar como es mas fácil crear los informes.

Revisemos ahora las clases que proporciona Visual C++ para la impresión. Estas clases son sobre todo utilizadas para aplicaciones que no contengan bases de datos, por ejemplo si creamos un editor de textos que permite la impresión de documentos.

Revisemos estas clases de forma general y posteriormente las revisaremos un poco mas a detalle.

Función o Clase	Comportamiento
OnDraw	Esta se manda a llamar cada que se imprime o se realiza una presentación preliminar. Dibuja la vista.
OnPreparePrinting	Configura el numero de la primera y ultima pagina.
OnBeginPrinting	Crea objetos GDI
OnPrepareDC (en cada pagina)	Configura el modo de asignación y, opcionalmente, detecta el fin del trabajo de impresión.
OnPrint (en cada pagina)	Genera salida especifica para impresora o llama a OnDraw

OnEndPrinting	Elimina los objetos GDI

Los objetos GDI son objetos de tipo interfaz de dispositivo gráfico, tales como fuentes, que sean necesarios para el trabajo de impresión.

La impresión básicamente es coordinada por la clase Cview, esta clase será en la que programaremos las condiciones de la impresión, es decir esta clase coordina tanto la impresión como la presentación preliminar, para hacerlo utiliza varias funciones que ya se mencionaron y que ahora revisaremos mas a detalle, al revisar estas funciones entenderemos mejor la función de la clase CView.

OnDraw: esta función trabaja en dos formas, cuando se hace una presentación preliminar y cuando se imprime. En el caso de la presentación preliminar la función OnPaint llama a OnDraw porque en este caso simplemente se esta dibujando en pantalla. Para la impresión es la función OnPrint la que manda a llamar a la función OnDraw, en este caso se manda a llamar una función OnPrint por cada pagina a imprimir. Tanto OnPaint como OnPrint son funciones virtuales de Cview. Resumiendo, OnDraw será la función encargada de dibujar en pantalla o en papel los datos a imprimir.

OnPreparePrinting: Esta función también parte de Cview es mandada a llamar al iniciar un trabajo de impresión, de hecho se llama antes de mostrar el dialogo de visualización. Si sabemos los números de la primera y ultima pagina a imprimir llamaremos a CPrintInfo::SetMinPage y a CPritnInfor::SetMaxPage en OnPrepare Printing.

OnBeginPrinting: a esta se le llama al salir del dialogo imprimir Esta función será la encargada de crear los objetos GDI que darán forma a la impresión. En este caso es recomendable utilizar OnBeginPrinting para toda la impresión y no hoja por hoja, ya que de esta manera la impresión es mas rápida.

OnPrepareDC: Esta función será llamada antes de OnPrint u OnPaint, OnPrepareDC se encarga de preparar toda la impresión, ya que esta determina el formato que tendrá nuestro trabajo, por ejemplo si no conocemos el número de paginas a imprimir esta función puede detectar el final del documento y desactivar una bandera para que la impresión se detenga. De hecho un parámetro de esta función es un puntero a la estructura CPrintInfo, esta ultima contiene las dimensiones de la página, el número de la página actual y el número de página máximo.

OnPrint: ya se ha mencionado que la función OnPrint manda a llamar a la función OnDraw y que esta ultima puede utilizarse para visualizar o imprimir, por esto antes de mandar llamar a OnPrint debe configurarse el modo de asignación (es decir, visualización o impresión). El armazón de la aplicación llama a la funciones OnPrint una vez por cada pagina a imprimir, almacenando el número de página en curso en la estructura CPrintInfo ya mencionada, de hecho también uno de los parámetros de OnPrint es un puntero a esta estructura.

OnEndPrinting: esta función de Cview se llama al final del trabajo de impresión, después de imprimir la ultima página. Esta función es utilizada para deshacernos de los objetos GDI creados con OnBeginPrinting, esto se tiene que realizar por que cada impresión es diferentes y por lo tanto los objetos GDI también.

A continuación se mostrará un ejemplo de impresión de un documento de una página para ver como se utilizan estas funciones, recuerde que algunas de esta funciones son virtuales por lo que no se encuentran fisicamente en el texto pero si los datos que muestran su funcionalidad. En este ejemplo solo se mostrará la función OnDraw y la función OnPreparePrinting.

Función OnDraw: la función OnDraw de la clase CStringView dibuja tanto en pantalla como en impresora. Además de visualizar 10 líneas de texto (ya antes insertadas en la función OnNewDocument creada por el AppWizard al crear la aplicación), dibuja un borde en torno al área imprimible y una regla básica en la parte superior y en el margen izquierdo.

```
Void CStringView::OnDraw(CDC* pDC)
 {
      int i, j. nHeight;
      Cstring str;
      CFont font;
      TEXTMETRIC tm;
      // Dibuja un borde, ligeramente mas pequeño para evitar
      // que se trunque.
      pDC->Rectangle(m_rectPrint + CRect(0,0,-20,20));
      // Dibuja reglas vertical y horizontal
      j = m_rectPrint.Width() / 1440;
      for (i=0;i<=j;i++) {
            str.Format("%02d",i);
            pDC->TextOut(0,-1*1440,str);
      }
      j = m_rectPrint.Heigth() / 1440;
      for (i=0;i<=j;i++) {
            str.Format("%02d",i);
            pDC->TextOut(0,-1*1440,str);
      }
      //Imprime el documento o 0.5 pulgadas por debajo y encima
      // con fuente roman de 10 puntos
      font.CreateFont(-200,0,0,0,400,FALSE,FALSE,0,
                  ANSI_CHARSET, OUT_DEFAULT_PRECIS,
                  CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
                  DEFAULT_PITCH | FF_ROMAN,
                  "Times New Roman");
      CFont* pOldFont = (CFont*) pDC->SelectObject(&font);
      pDC->GetTextMetrics(&tm);
      nHeight = tm.tmHeight + tm.tmExternalLeading;
      TRACE("altura fuente = %d, espacio interno = %d \n,
            nHeight, tm.tmInternalLeading);
      j= pDoc->m_stringArray.GetSize();
      for (i=0;i<=j;i++) {
            pDC->TextOut(720,-1 * nHeight - 720, pDoc->
            m_stringArray[i]);
      }
      pDC->SelectObject(pOldFont);
      TRACE("LOGPIXELSX = %d, LOGPIXELSY = %d \n",
            pDC->GetDeviceCaps(LOGPIXELSX),
            pDC->GetDeviceCaps(LOGPIXELSY));
      TRACE("HORZSIZE = %d, VERTSIZE = %d \n",
            pDC->GetDeviceCaps(HORZSIZE),
            pDC->GetDeviceCaps(VERTSIZE));
      }
```

Función OnPreparePrinting: esta función configura el número máximo de paginas del trabajo de impresión.

```
BOOL CStringView::OnPreparePrinting(CPrintInfo* pInfo)
{
      pInfo->SetMaxPage(1);
      return DoPreparePrinting(pInfo);
}
```

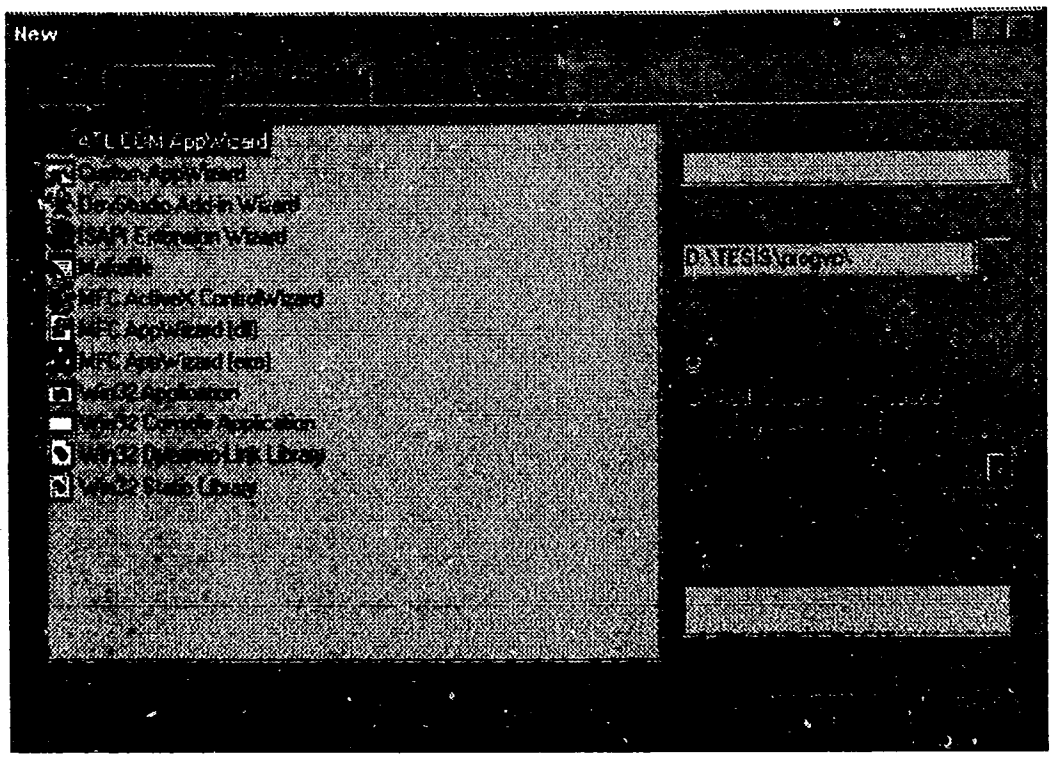
Note en este pequeño ejemplo lo problemático que resulta la impresión desde el código, en este ejemplo solo se imprimía un solo documento en una página, para poder imprimir en nuestro caso toda una base de datos o algunos registros de la base de datos, es necesario hacer un array de los registros que se van a imprimir para poder imprimir desde el arreglo, esta es también dificil ya que para ello se debe conocer el manejo de arreglos y aparte de todo estamos haciendo tarea de mas, ya que primero se deben mover lógicamente los registros a el arreglo y posteriormente se imprimen desde este tomando en cuenta todas las cuestiones ya explicadas.

Para los programadores expertos esta es una buena herramienta pero si la finalidad es obtener un sistema aprovechando todas las herramientas que nos ofrece la computación es preferible utilizar ActiveX ya que después de todo esta herramienta nos muestra la compatibilidad que ofrece Visual C++.

VI.2.2. Manejo de Presentaciones

En este punto del capitulo nos dedicaremos a revisar como se crea un formulario. El manejo de los gráficos los revisaremos en otro punto. Es muy importante tomar en cuenta las recomendaciones que se dan en este capítulo ya que aunque se realiza automáticamente la creación de las clases y las ventanas es necesario que el usuario proporcione algunos datos que van personalizados para el sistema que se este creando.

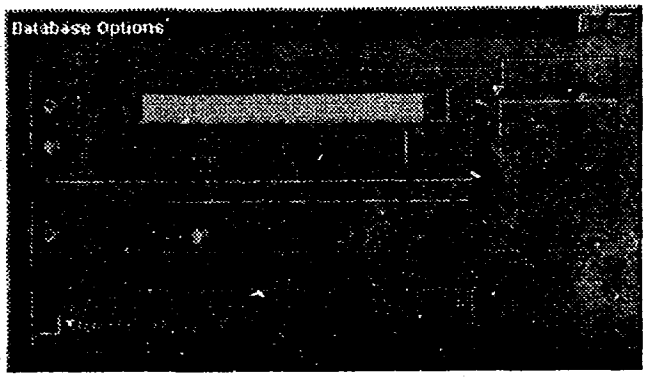
Lo primero que tenemos que hacer es seleccionar la opción New del menú File. Aparecerá la siguiente pantalla (esta es la principal de la herramienta AppWizard):



En esta pantalla se deberá elegir la opción MFC AppWizard(EXE) y deberá de teclear el nombre del proyecto además de la ubicación en la que se guardaran los archivos que ya se mencionaron anteriormente. Por último deberá de pulsar el botón OK.

En la siguiente pantalla selección la opción Single Document. Si el sistema no fuera de bases de datos pulse el botón Finish, pero como el sistema que estamos creando es para bases de datos pulse el botón Next.

En la siguiente pantalla seleccione la opción Database View Without File Support y pulse el botón DataSource Aparecerá la siguiente pantalla.



En este caso hay que tomar una decisión si el sistema se hará bajo ambiente ODBC o DAO, en caso de seleccionar la opción ODBC debe configurar previamente este ambiente mediante el Panel de Control de Windows. En nuestro caso seleccionaremos la opción DAO y pulsaremos el botón a la derecha de esta opción, en este caso se mostrar la pantalla en la que se indicara la base de datos que utilizaremos, recuerde que esta base es de Access y ya debe de estar creada. Al pulsar el botón OK se preguntara por la tabla que queremos utilizar, seleccione alguna tabla y pulse OK. Una vez que hace esto regresará a una pantalla anterior, en esta pantalla puede pulsar el botón Finish, esto le indicara al sistema que las demás opciones se tomaran por default.

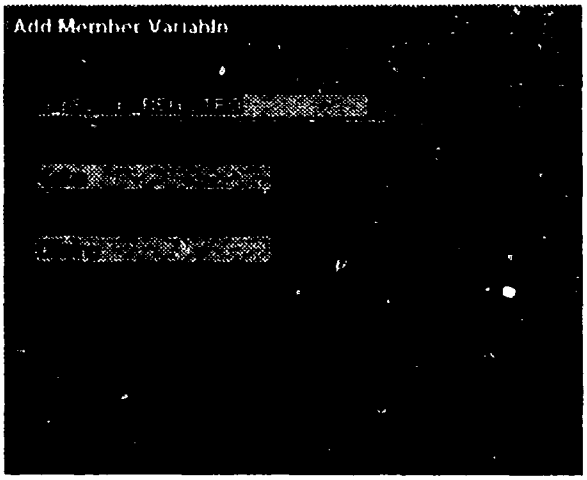
En este momento ya se han creado los archivos, ventanas y menús que necesita nuestra aplicación. Ahora lo que haremos será concentrarnos en la ventana principal. Para abrir esta seleccione el folder ResourceView en la ventana WorkSpace, en esta abra el folder de Dialog. Pulse Doble Clic en la opción IDD_RECETARIO_FORM (este será el formulario principal), en la parte derecha se abrirá el formulario hasta este momento vacío y la barra de herramientas para este menú (esta barra se muestra en la parte de botones y gráficos).

La forma de ingresar los objetos a nuestro formulario se hace seleccionando el objeto y dando un clic dentro del formulario, acomode el tamaño y la posición del objeto en la ventana. Una vez que ha insertado el objeto hay que configurarlo de acuerdo a las necesidades del sistema, para ello pulse el botón derecho del Mouse en el objeto que quiera

configurar, aparece un menú en el cual seleccionara la opción Propiedades. Escriba los nombres para el ID y el caption.

El ID será el nombre de la variable que se utilizara para hacer referencia dentro del mismo sistema, es muy parecida como la propiedad Name de Visual Basic y Delphi. Por otro lado la opción Caption contendrá el nombre que se mostrara en el objeto insertado, es decir en un botón la propiedad caption llevara el nombre de este. Esta propiedad es exactamente igual que en Visual Basic y Delphi.

Como en este caso queremos crear un formulario para desplazarnos en la tabla necesitamos insertar etiquetas y cuadros de texto que servirán para crear la ventana que se mostró en el punto de tablas. En los cuadros de texto hay que configurar el campo que mostrara dicho cuadro, para hacer esto seleccione el cuadro y manteniendo oprimida la tecla Ctrl pulse doble clic sobre el cuadro de texto. En este caso aparecerá una pantalla como la siguiente:



Esta pantalla es muy importante para los sistemas y es necesario que la tome en cuenta ya que sirve para añadir una variable a nuestro sistema, pero también es utilizada para asignar un campo a un cuadro de texto. En el recuadro Member Variable Name seleccione el campo que quiere se muestre en el cuadro de texto. En la lista se contienen todos campos de la

tabla que se selecciono durante el AppWizard. Tiene que repetir lo anterior para cada uno de los cuadros de texto que se ingresen en nuestro formulario.

En este momento ya tenemos un formulario creado. Para compilar el sistema que hemos creado hasta el momento seleccione la opción Build del menú con el mismo nombre. Si el sistema se compila sin errores puede ejecutar el sistema con la opción Execute del mismo menú Build.

En este caso el formulario creado únicamente permitirá desplazarse entre los registros de la tabla. Si quiere agregar las opciones para insertar y borrar consulte el manejo de tablas de este capitulo.

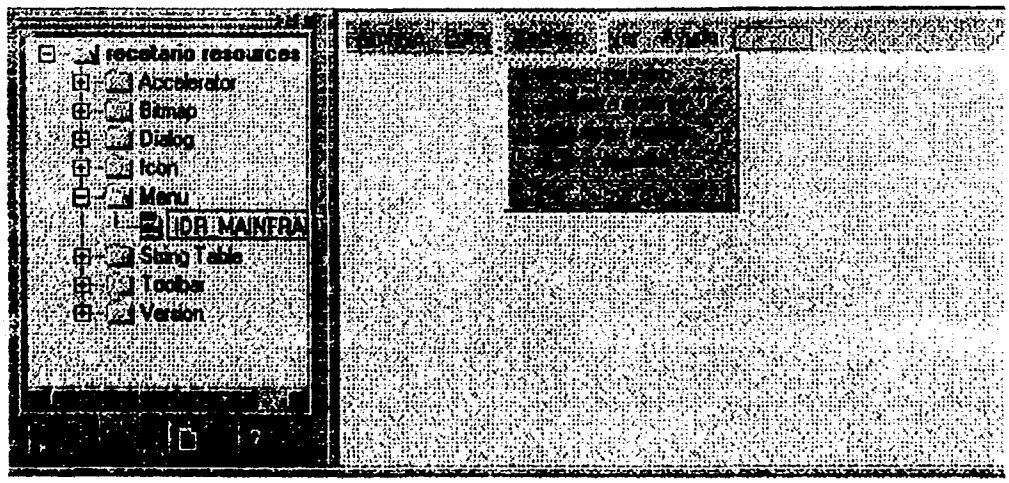
En la parte de Gráficos y botones explicaremos como darle mas presentación a los formularios y cuales son los diferentes objetos que podemos añadir a nuestro formulario. En este caso únicamente se quería explicar la forma de crear un formulario con ayuda del AppWizard.

VI.2.2.1. Menús

Loe menús en Visual C++ se realizan de una forma muy sencilla, esto es principalmente gracias al App Wizard, recuerde que este crea los archivos y objetos necesarios para la aplicación. En el caso de la aplicación que estamos haciendo, y al seleccionar las opciones que ya se mencionaron en el AppWizard este crea un menú muy simple que será cuestión del programador añadir o quitar las opciones que considere necesarias. Algunas veces este menú aparece vacío según la aplicación que se esta creando pero de todas formas crea la estructura necesaria para agregar las opciones necesarias.

Es importante mencionar que en Visual C++ cada opción del menú también se reconoce como un objeto independiente por lo que se le debe asignar un ID o variable para poder asignarle código a dicho objeto. Para poder trabajar con el menú debemos seleccionar en la

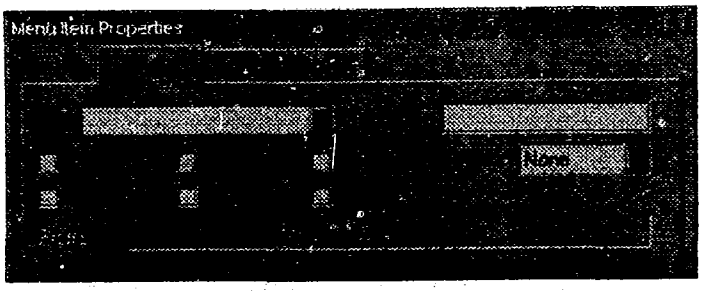
pantalla del WorkSpace el folder ResourceView en este abrir la carpeta menú, se mostrará un nombre para el menú, en el caso de nuestra aplicación este nombre será IDR_MAINFRAME. Para poder abrirlo es necesario teclear doble clic sobre el nombre del menú, en la pantalla de la derecha aparecerá el menú que ya esta creado. En nuestro caso el menú que se crea automáticamente será el que se muestra en la pantalla siguiente:



Posiblemente para el menú que estamos haciendo, la única opción que nos interesa es la de Registro, aunque como ya se comento es responsabilidad del programador el menú que utilizara el sistema final. Para borrar alguna de las opciones del menú basta con seleccionarla y pulsar la tecla Supr.

Note que en la parte derecha de las opciones aparece un cuadro vacío, este le permite crear una nueva opción para el menú horizontal, lo mismo ocurrirá si usted abre o selecciona cualquiera de las opciones horizontales, es decir aparecerá el menú vertical correspondiente a esa opción y mostrara al final un cuadro vacío para agregar una nueva opción a ese menú.

En caso de querer añadir una nueva opción basta con dar doble clic sobre el cuadro vacío ya sea el vertical o el horizontal según la opción que desee añadir. Al dar doble clic aparecerá la siguiente pantalla:



En esta pantalla se va a configurar las propiedades del objeto que se acaba de añadir al menú, en el caso de la pantalla anterior se ha seleccionado una opción ya existente para que pueda visualizar la sintáxis que deben de llevar los datos a introducir. Básicamente se deben introducir dos datos, el ID y el Caption, El ID será el nombre de la variable o función que identificara a este objeto, en caso de que el ID ya exista y únicamente quiera hacer referencia a este objeto basta con seleccionarlo de la lista, ahora bien si el ID no existe basta con teclear un nuevo nombre para que automáticamente lo cree Visual C++.

Por otro lado la propiedad Caption indicara el nombre que aparece en la pantalla, en la sintaxis del ejemplo anterior puede observar que se ocupa el símbolo &, esto sirve para indicar la letra que aparecerá subrayada en el menú y que se pueda accesar a esta opción mediante la tecla ALT combinada con la letra subrayada, además note que esta línea cuenta con \t, para los conocedores de lenguaje C esto no será raro pero para los que no conocen C, esta sintáxis indica que el siguiente texto ira después de un tabulador para que no se vea junto, es decir exista una separación entre los mensajes, a continuación lleva la combinación de teclas que nos interesa para el objeto.

Puede además añadir un comentario al objeto mediante el cuadro de texto Prompt que se encuentra al final del ejemplo mostrado, note que en este caso ya cuenta con un comentario. Puede también añadir otras opciones a este menú, por ejemplo puede indicar que quiere una línea de separador en lugar de un objeto, esto se hace por medio de la opción Separator de

la pantalla, puede también agregar un submenu a alguna opción por medio de la opción Pop-up.

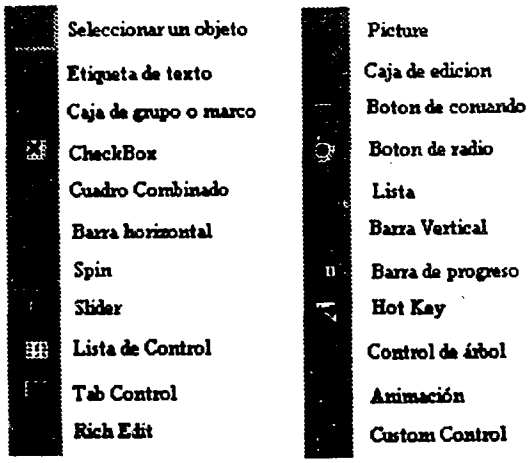
Es importante mencionar que para las opciones que se agreguen no debe de faltar la asignación del ID ya que por medio de este se crear a el enlace entre el menú y el código de las clases, de hecho si no se asigna el ID o es incorrecto Visual C++ automáticamente mandara un mensaje de error y no le permitirá seguir trabajando con el sistema hasta corregir este error.

VI.2.2.2. Gráficos y botones

Este punto lo dividiremos en dos partes, en la primera analizaremos los diferentes objetos que podemos añadir a un formulario así como las principales propiedades de estos y en la segunda revisaremos un poco acerca de los colores y de cómo añadir elementos gráficos a nuestro sistema.

En la pantalla siguiente se muestran los diferentes objetos que se pueden añadir a un formulario de Visual C++, la mayoría de estos objetos ya los conocemos por que los hemos mencionado en otros capítulos.

Lo importante en este punto es mencionar que para añadir uno de estos objetos a los formularios basta con seleccionarlo y dar un clic dentro del formulario, con esto se añadirá el objeto al formulario, ahora lo siguiente es acomodar el tamaño y la posición del objeto y esto es muy sencillo ya que recuerde que este responderá de las misma manera que cualquier objeto en todo Windows, es decir se puede mover y cambiar de tamaño por medio del Mouse.



Al añadir alguno de estos objetos a nuestro formulario hay que configurar ciertas propiedades para cada uno de ellos. Básicamente se deben de configurar las siguientes propiedades (recuerde que para accesar a las propiedades basta con dar clic con el botón derecho del Mouse y del menú que aparece seleccionar la opción Propiedades):

- ID: este indicará el nombre de la variable o función a la que se hace referencia con el objeto. Esta propiedad es muy importante ya que de esta dependen los sucesos a los que responderá el objeto. El nombre del ID será la función que se programara según el suceso que nos convenga según la aplicación. Este nombre de ID puede o no existir, en caso de que no exista se crea automáticamente pero se tiene que asignar código de programación para que responda al suceso.
- Caption: será el nombre que mostrará el objeto en pantalla.
- Visible: esta propiedad hará que el objeto sea visible o no al inicio de la ejecución del sistema. Esta propiedad puede cambiar durante la ejecución según las necesidades del sistema. Por default aparecerá seleccionada.
- Disabled: esta propiedad permite que el objeto se encuentre deshabilitado al iniciar la ejecución del sistema, al igual que la anterior puede cambiar durante la ejecución, esta propiedad no aparece seleccionada al inicio por lo que se toma un valor de Enabled.

Del formulario en si también se puede configurar ciertas propiedades, por ejemplo el tipo de letra que se mostrará en el formulario, esto se hace de la misma manera que para los objetos ya que recuerde que también el formulario en si es considerado como un objeto mas dentro de la aplicación.

Una vez que hemos revisado los objetos que se pueden añadir a los formulario revisaremos las herramientas gráficas con las que cuenta este lenguaje. Por principio de cuentas hablaremos de los iconos, estos serán los que aparecerán en la parte superior izquierda de la ventana, en el folder ResourceView de la ventana WorkSpace puede seleccionar el folder Icon, en este aparecerán los iconos con los que cuenta la aplicación.

Al dar Doble Clic sobre alguno de estos, aparecerá en la parte derecha un área de trabajo parecida a Paint de Windows, es decir aparecerá la imagen y una barra de herramientas para dibujo, esto es con la finalidad de crear o modificar sus iconos. Para crear un nuevo icono basta con dar un clic con el botón derecho del Mouse en la ventana WorkSpace, en el menú aparece la opción Insert, elija esta, a continuación se preguntara por el objeto a añadir, elija Icon y aparecerá un nuevo nombre en el folder Icon, puede comenzar a dibujar el icono que desee y configurar las propiedades para este.

De la misma manera se puede hacer para añadir cualquier otro gráfico a nuestra aplicación, es decir, una vez que hemos seleccionado la opción Insert, en la pantalla que aparece en lugar de elegir la opción icono seleccione la opción Bitmap, aparecerá un nuevo folder con el nombre BitMap y el nombre del mapa de bits que se va a insertar, puede dibujar el nuevo mapa de bits en la parte derecha como en el icono, o bien puede importar algún dibujo que ya tenga realizado, para hacer esto vuelva a dar un clic derecho en el folder Bitmap, del menú que aparece elija la opción Import, de la pantalla que aparece seleccione el dibujo que quiera importar y se creara un nuevo nombre de mapa de bits en el folder Bitmap.

Lo anterior es muy importante ya que de esta manera insertaremos nuestros dibujos o gráficos al formulario. Para poder hacerlo, inserte el objeto Picture dentro del formulario, en la propiedades del este objeto debemos realizar los siguientes pasos:

1. En la opción Type elija la opción Bitmap.
2. En la opción Image aparecerá una lista de los dibujos que se han insertado en el folder Bitmap, elija el nombre del bitmap deseado.
3. Acomode el tamaño y posición del objeto.

En resumen para insertar una imagen al formulario primero se debe insertar dentro del folder bitmap y a continuación podemos añadirlo al formulario o utilizarlo para otras actividades dentro de la misma aplicación. También se puede cargar un gráfico desde el código de un programa, esto se hace por medio de la siguiente sintaxis:

```
M_bmp.SetBmpFilename(nombre_del_archivo);
```

Esto permitirá que en alguna parte del sistema pueda mandar llamar un dibujo en caso de que así lo considere necesario, se podría pensar que esto es mas fácil que añadirlo de la forma que se explico anteriormente pero lo difícil de hacer esto es que se tiene que configurar el tamaño y la posición también por medio de código y eso requiere de mas conocimientos de programación.

Ahora para hablar de los colores es necesario que tomemos en cuenta como es que se manejan en Visual C++. La forma de manejarlos es por medio de combinaciones de colores primarios, por lo que el comando para los colores es:

```
RGB(num1,num1,num1);
```

Esta forma ya se había visto anteriormente en este trabajo, recuerde que hay que combinar los colores proporcionando un porcentaje para los colores rojo, verde y azul, los niveles de

color irán de 0 a 255, por ejemplo para utilizar un color rojo basta con declarar RGB(255,0,0), el blanco se genera con RGB(255,255,255) y el negro con RGB(0,0,0).

De hecho esta función será la que por medio de programación se debe de añadir a alguno de los componentes u objetos para proporcionarle el color. Recuerde que a pesar de que algunas tareas las realizamos gráficamente siempre se esta realizando el código de programa por medio de las clases.

VI.2.3. Programación

La programación en Visual C++ es el tema un poco mas complicado acerca de esta aplicación. Recuerde que Visual C++ trabaja por medio de programación orientada a objetos y eso conlleva muchas cosas, de hecho aunque hemos visto en otros lenguajes programación orientada a objetos no es de la misma forma que en este lenguaje. Cabe mencionar que en este capitulo no se piensa enseñar al programador todo lo referente a la programación orientada a objetos ni programar en Visual C++ ya que eso requeriría de un libro completo, pero si vamos a explicar un poco los detalles que se necesitan conocer para comenzar a programar. Por principio de cuentas se explicara muy brevemente los mecanismos básicos de la programación orientada a objetos:

Objetos: son una entidad que tiene atributos particulares, los datos y unas formas de operar sobre ellos, los métodos y procedimientos. Un objeto contiene operaciones que definen su comportamiento y variables manipuladas por esas operaciones que definen su estado.

Mensajes: Cuando se ejecuta un programa orientado a objetos, los objetos están recibiendo, interpretando y respondiendo a mensajes de otros objetos. Por ejemplo cuando hacemos un clic para maximizar una ventana, esta recibe un mensaje de que tiene que maximizarse.

Métodos: Determina como tiene que actuar el objeto cuando recibe el mensaje asociado, este se encuentra definido dentro de una clase. También lo podríamos definir como una función que contiene el código que responde a algún suceso de los objetos.

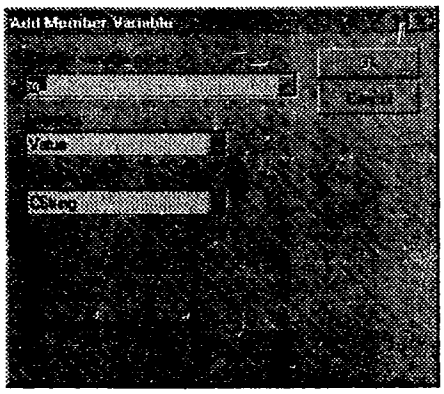
Clases: es un tipo de objetos definido por el usuario. Equivale a una generalización de un tipo especifico de objetos. Las variables de la clase tienen almacenados valores que son compartidos por todos los objetos de esa clase y cada objeto de una clase tiene sus propios valores, almacenados en las variables asociadas a cada objeto de la clase.

Como en Visual C++ se crean las clases automáticamente, únicamente el usuario se encargara de añadir objetos y métodos a cada uno de estos objetos. Al tener la programación de esta forma también es fácil detectar los errores ya que como en las clases se encuentran claramente definidos los objetos y dentro de cada objeto los sucesos es fácil encontrar cual es el suceso u objeto que contiene el error.

Hasta este momento hemos tocado varias veces el nombre ID, y en este momento hablaremos un poco mas a detalle de este para ver que cual es su función en los programas de Visual C++. Por principio de cuentas recuerde que se asigna un nombre de ID a cada uno de los objetos cuando se insertan a un formulario o un menú.

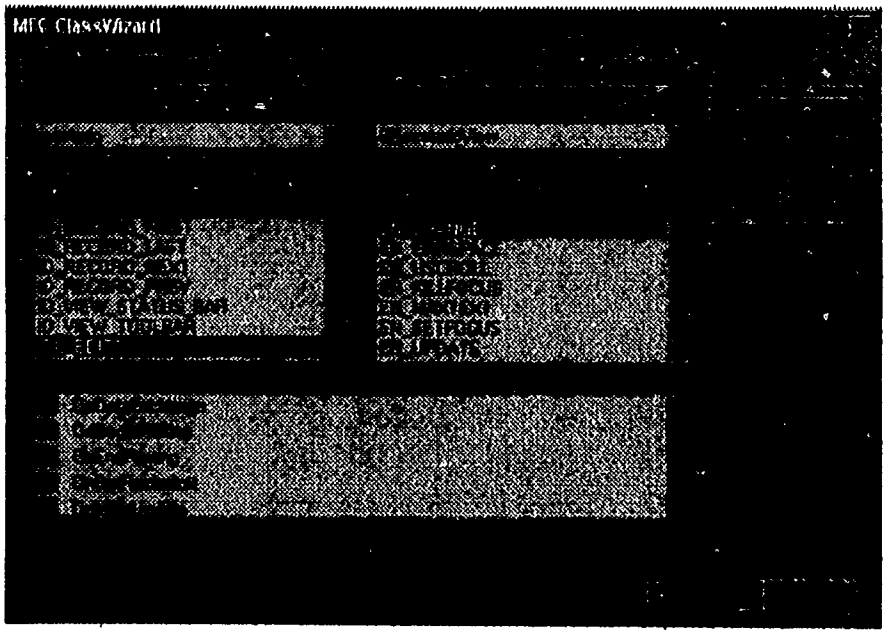
Al estar en el formulario puede seleccionar la opción ClassWizard del menú View, al hacer esto se mostrara en pantalla los nombres de ID que se han proporcionado hasta ese momento, cabe mencionar que estos serán para el formulario, lo mismo puede hacer en el caso de los menús.

A cada uno de estos ID se le debe asignar una variable para poder realizar algunos cambias mediante el código, para hacer esto seleccione el nombre de ID y elija la opción Add Variable, aparecerá la siguiente pantalla (esta será la forma de declarar variables):



En esta pantalla se le indicará el nombre, categoría y tipo de la variable. Tome muy en cuenta el nombre de la variable ya que será el utilizado para el código. Los ID serán los nombres de los métodos que responderán a los objetos.

Para añadir código a los cada uno de los ID utilizaremos la siguiente pantalla:



En esta pantalla seleccione el ID al que quiera agregarle código y elija el Message (o suceso) correspondiente, por ejemplo supóngase el caso de un objeto Check Box, el ID asociado a este objeto será el de IDC_CHECK1, y la variable que se le agrego fue m_check2. Queremos agregar código al suceso On_Click por lo que seleccione en la pantalla anterior el ID IDC_CHECK1, el message BN_CLICKED, pulse el botón Add Function y confirme en la pantalla con Enter, en este momento se acaba de crear una nueva función para este objeto, a continuación pulse el botón Edit Code, en la pantalla aparece el área donde se debe agregar el código de programación correspondiente.

De esta forma se agrega el código al sistema, note que en este caso el programador debe de indicar que se quiere crear una función y no es como en Visual Basic que ya se encontraban creados automáticamente los módulos donde se podía programar.

Por otro lado es muy importante que tome en cuenta la sintaxis que requiere Visual C++ en los comandos, recuerde que todos los comandos deben terminar con un punto y coma, aunque sean comandos de programación estructurada o de programación orientada a objetos.

Esto ultimo es porque Visual C++ además de ofrecer los comandos clásicos de programación estructurada ofrece una serie de comandos para programación en Windows como por ejemplo el MessageBox, este comandos se comento en Visual Basic y es exclusivo de programación en Windows, pero en este lenguaje llevara otra sintaxis diferente a la que se vio en Visual Basic. Puede consultar los archivos del sistema final para verificar sintaxis acerca de los comandos utilizados.

Por otro lado los ciclos y funciones de condición son las mismas que en lenguaje C y cumplen los mismo requerimientos que en ese lenguaje en la siguiente tabla puede observar esto:

Condición	Bucles o ciclos

if <condición>
{
<comandos>
}
else
{
<comandos>
}
	do
	{
	<comandos>
	}while <condición>;
	o bien;
	for(<inicio>;<tope>;<avance>)
	{
	<comandos>
	}

Como se puede observar en lenguaje C eran clásicas las llaves y en la programación de Visual C++ siguen siendo requeridas para marcar el inicio y final de una función, aunque la mayoría de las veces estas las maneja directamente Visual C++.

VI.2.4. Otras herramientas

Al igual que los otros lenguajes visuales analizados en este trabajo, existen mas herramientas aparte de las mostradas en este trabajo. En este punto analizaremos algunas de estas sobre todo que nos sirvan para el tipo de sistema que estamos generando.

Por principio de cuentas se cuenta con la clase CdaoQueryDef, recuerde que cuando creamos una nueva aplicación por medio de App Wizard y esta es para bases de datos se crean varias clases, una de ellas es precisamente CdaoQueryDef. Esta clase es utilizada para crear consultas de datos y de esta forma crear nuevas tablas pero creadas por medio de un filtro. En esta clase se pueden crear las consultas por medio de comandos SQL, este tipo de comandos ya los analizamos en el capitulo de Delphi, en este caso al hacer los filtros por medio de SQL y crear las nuevas tablas facilitaran las búsquedas y sobre todo que harán mas rápidos los sistemas. Este tipo de herramienta se recomienda usarlas cuando se utilizan tablas con una gran cantidad de datos. Una vez que ya se tienen las consultas hay que crear

un enlace entre CdaoQueryDef y CdoaRecordet para que este ultimo utilice la nueva tabla generada por la consulta.

Otra herramienta muy completa proporcionada por Visual C++ son los controles ActiveX. Los Controles ActiveX son partes de software que se pueden fácilmente agregar en los programas Visual C++. Un control ActiveX es un archivo con la extensión OCX, por lo general este archivo reside en Windows\system, este tipo de controles es muy similar a las bibliotecas DLL ya que es vinculado directamente al programa. Este tipo de herramienta es muy potente sobre todo porque permite agregar a nuestros sistemas varias aplicaciones que a lo mejor con el mismo Visual C++ no se podrían a ser o requerirían de mucho tiempo y esfuerzo para hacerlo. Esto se puede hacer una similitud con las aplicaciones OLE de Visual Basic y Delphi, ya que funcionaran de la misma manera a diferencia de que los controles ActiveX serán como bibliotecas y las aplicaciones OLE serán directamente una aplicación de Windows.

Por otro lado también se pueden utilizar aplicaciones para Internet y que por supuesto ocupen bases de datos, esto es muy recomendable sobre todo para cuando los datos con los que estamos trabajando son utilizados por varias maquinas, es decir en red, Visual C++ proporciona herramientas para trabajar en red y por supuesto en Internet.

VI.3 Ventajas y desventajas ante los demás paquetes

Primeramente mencionaremos que al ser Visual C++ una aplicación para Windows, ofrece todas las ventajas que una aplicación de este tipo nos puede ofrecer, es decir nos permite trabajar en un ambiente GUI (interfaz gráfica del usuario). Además de que también permite crear aplicaciones para DOS, esto es una gran ventaja ya que recordemos que muchos sistemas se requieren realizar en este tipo de ambiente debido a las necesidades propias de la empresa.

Developer Studio es el ambiente de Visual C++, de hecho resulta algo sorprendente para algunos usuarios que el ambiente no se llame Visual C++ (la propia barra de titulo lleva el nombre Developer Studio). Esto es debido a que Microsoft ha conseguido un entorno de desarrollo para los paquetes de programas Visual C++, Fortran PowerStation, SourceSafe, Visual Test y Development Library. Esto nos da una visión del alcance en cuestión de lenguaje que nos ofrece este ambiente por ello es que en cuestión de programación Visual C++ es uno de los mejores lenguajes y si a ello agregamos el ambiente visual apreciaremos muy bien las ventajas de esta aplicación.

Al revisar este lenguaje nos damos cuenta que es un lenguaje que requiere de mas programación que los lenguajes visuales anteriormente vistos. Pero no por ello deja de ser un lenguaje muy completo. Si comparamos a Visual C++ con Clipper (obviamente en aplicaciones para DOS) diremos que Visual C++ requiere de una programación mas avanzada, simplemente por el hecho de que Clipper maneja programación estructurada y Visual C++ orientada a objetos. A pesar de que Visual C++ ofrece muchas herramientas no es un lenguaje enfocado a las bases de datos y a pesar de ofrecer algún tipo de clases para este tipo de aplicación no llega a ofrecer todos los comandos que Clipper nos ofrece, aunque la ventaja es la facilidad con la que se trabaja en Visual C++ y sobre todo las herramientas de presentación que nos da este ultimo.

Por otro lado al comparar Visual C++ con FoxPro para Windows resulta lo mismo que con Clipper ya que mientras FoxPro nos ofrece muchos comandos para las bases de datos, Visual C++ no ofrece estos, aunque sucede los mismo, la facilidad con la que se trabaja Visual C++ y las herramientas de presentación que este ultimo ofrece. Además como se pudo ver en el sistema realizado en este trabajo las clases para bases de datos que ofrece Visual C++ permite crear buenos sistemas de bases de datos y el ejemplo es el que se realizo en este trabajo.

Ahora bien, comparar a Visual C++ con los otros lenguajes visuales mencionados en este trabajo tendremos que considerar varios puntos. Muchos programadores siempre se dejan

llevar por los antecedentes que los lenguajes visuales tienen, si pensamos en hace varios años lenguaje C era el que dominaba el mercado y el primero que introdujo el concepto de programación orientada a objetos, de ahí que muchos programadores prefieran Visual C++ a otros lenguajes, aunque este requiera de mas programación que los otros.

Visual C++ ofrece ciertas ventajas que saltan a la vista, por principio de cuentas es el único que proporciona la herramienta necesaria para iniciar cualquier tipo de sistemas y crear todos los archivos que necesitara nuestra aplicación, durante la realización del sistema únicamente se irán agregando comandos a estos archivos o clases. Además también recuerde que es único lenguaje que automáticamente crea una carpeta que contiene todos los archivos así como las subcarpetas necesarias. Otra ventaja muy notoria de Visual C++ es la capacidad que tiene para manejar las variables o el mencionado ID, recuerde que este ID será la variable que asociara el código de la clase con el objeto en cuestión.

Otra ventaja de este lenguaje es la gran potencia que tiene en cualquier tipo de aplicación ya que al ser un lenguaje orientado a objetos aprovecha al máximo las posibilidades que este tipo de programación proporciona. Además para las bases de datos cuenta con clases particulares que permiten realizar las acciones mas básicas que refieran a las tablas.

Las desventajas que este lenguaje proporciona también son muy notorias, Visual C++ no es un lenguaje enfocado a las bases de datos y a pesar de que ofrece herramientas y clases para su manejo siempre se deben de manejar estas clases por medio de programación y esto para muchos programadores no resulta muy grato. Observemos que de hecho Visual C++ es el único de los lenguajes visuales que no proporciona un manejador interno de bases de datos, ni un generador de informes, etc. Aunque estos manejadores los sustituye con clases y comandos para poder realizar todas las tareas que un sistema requiera, las acciones para esta tarea se tiene que programar manualmente y con una programación no tan sencilla y simple como en Visual Basic.

Esto ultimo es lo que no ha permitido que Visual C++ se convierta en el mejor lenguaje del mercado ya que aunque ofrece gran potencia el programador tiene que ser experto y conocer muy bien la programación estructurada de lenguaje C además de la programación orientada a objetos del mismo lenguaje.

Otro detalle que sucede con Visual C++ comparado con Visual Basic y Delphi, es que los dos últimos ofrecen mas objetos y sucesos de los que ofrece Visual C++, esto puede resultar una ventaja o desventaja a la vez, por un lado Visual C++ se centra específicamente a los objetos clásicos y mas necesarios para aplicaciones de Windows y lo mismo pasa con los sucesos, es decir existen solo los sucesos a los que responde mas cotidianamente una aplicación bajo Windows, como lo puede ser el clic, el doble clic, etc. pero por el otro lado algunas aplicaciones requieren de objetos y sucesos mas específicos y en este caso Visual C++ no los proporciona de una manera directa como los otros lenguajes, para que el sistema responda a objetos y sucesos específicos se tienen que programar y esto requiere de buenos conocimientos en programación.

Otra cosa que no ha permitido que este lenguaje tenga mucho auge en las empresas es que para muchos programadores Lenguaje C de Microsoft no proporciona toda la potencia que este lenguaje debe proporcionar, por ejemplo hace varios años la gente prefería Lenguaje C de Borland al lenguaje C de Microsoft, y desafortunadamente para Microsoft, Borland cuenta con un lenguaje parecido a Visual C++ conocido en el mercado como Power Builder, este ultimo se podría considerar como la versión Borland de Visual C++. Pero por otro lado tomemos en cuenta que el mercado de las computadoras personales ha sido dominado por Microsoft hace varios años, de hecho Windows es de esta familia, por lo que los lenguajes de Microsoft son mas compatibles con todas las aplicaciones que trabajen bajo Windows.

En herramientas de presentación no se puede decir mucho de ventajas y desventajas ya que este lenguaje ofrece las herramientas necesarias para crear una aplicación amigable para los usuarios.

VI.3.1 Compatibilidad

Ya se ha mencionado que Visual C++ es un lenguaje de la familia Microsoft, esto nos da la visión de la compatibilidad que proporciona esta aplicación, recordemos por ejemplo que Visual Basic es un lenguaje que es 100% compatible con todas las aplicaciones de Windows, con Visual C++ resulta lo mismo ya que además de las herramientas mencionadas en este capitulo, este lenguaje también cuenta con la herramienta para aplicaciones OLE, recordemos que este tipo de herramientas permite introducir cualquier tipo de aplicación de Windows a nuestro sistema. Además las ventas o formularios que se crean en Visual C++ tienen el mismo comportamiento que las ventanas de cualquier aplicación de Windows por lo que para el usuario final no resultara difícil aprender a manejar el sistema al menos en cuestión del manejo clásico.

En la presentación de este lenguaje se mencionaba que Visual C++ permitía trabajar con bases de datos de varios tipos (Access, Paradox, Dbase, etc. todas ellas de la familia Microsoft), esto es una gran ventaja ya que en este trabajo nos referimos a las bases de datos de Access por se considerado este el mejor manejador de los datos, pero existen algunos usuarios acostumbrados a otros tipos de manejadores y esto no resulta un obstáculo para Visual C++, recordemos que el sistema realizado para este capitulo se realizo por medio del método DAO, pero mencionamos también que se puede trabajar por medio del método ODBC (Open Database Connectivity). El primer método es mas recomendable para las bases de datos de Access y el otro método para otro tipo de manejador. De hecho la configuración del ultimo se realiza desde el panel de Control de Windows, lo que da una visión de que este lenguaje puede ser compatible 100% con Windows.

Además de ser compatible con Windows y con las aplicaciones de este, también es compatible al 100% con DOS, ya se menciono que Visual C++ permite crear aplicaciones para DOS, de hecho es el único lenguaje Visual que lo permite, y eso es una gran ayuda sobre todo para empresas que aun no se olvidan del viejo sistema operativo.

También se menciono durante este capitulo que Visual C++ cuenta con herramientas para crear sistemas que trabajen con paginas de Internet y con Java, esto también nos da una visión de la compatibilidad de este lenguaje ya que las herramientas que proporciona son muy completas, de hecho es uno de los lenguajes que mejor trabaja este tipo de aplicación. Por ello mismo podemos decir que Visual C++ es un lenguaje actual y no deja de fuera los nuevos entornos en los que se encuentra actualmente la computación.

Por otra parte Visual C++ también puede ser compatible con Visual Basic, esta compatibilidad esta principalmente dad por el hecho de que por medio del APP Studio se pueden incorporar controles personalizados Visual Basic (VBX) aunque solo para aplicaciones de 16 bits, es decir para Windows 3.X.

Regresando a la compatibilidad que proporciona este lenguaje con Windows debemos tomar en cuenta que la presentación de los sistemas realizados en este lenguaje esta dada por medio de imágenes en el formato de Windows o de aplicaciones de Windows, en Visual C++ es posible incluir cualquier imagen diseñada a medida o dada por un icono existente, un mapa de bits o un archivo con extensión wmf.

Además recuerde también que por medio de este lenguaje se pueden crear las clásicas librerías DLL ya mencionada en este trabajo, estas librerías son utilizadas por Windows para cualquier tipo de aplicación. Resumiendo, Visual C++ puede ser un lenguaje algo complicado para algunos programadores pero hay que tomar en cuenta que la compatibilidad de este lenguaje es muy amplia y eso resulta ser una gran ventaja para cuando se crean aplicaciones para Windows.

VI.3.2 Migración.

La migración de sistemas de Visual C++ hacia otras aplicación o viceversa resulta muy difícil, esto es debido principalmente a la forma en la cual se programa, recuerde por ejemplo que otros lenguajes aunque ocupan programación orientada a objetos no manejan las clases

de la misma forma en que lo hace este lenguaje, incluso recuerde que los sucesos que nos ofrece Visual C++ son distintos a los que ofrecen otros lenguajes.

En el punto anterior se menciono que se podían añadir ciertas aplicación de Visual Basic pero para 16 bits, los sistemas que estamos realizando en Visual C++ son de 32 bits y el migrar aplicaciones de 16 bits a un sistema para Windows 95 puede resultar en problemas para este ultimo.

En este lenguaje se cuenta con una herramienta conocida como Active X, esta herramienta permite insertar a nuestro sistema ciertas aplicaciones o herramientas de otro tipo de lenguaje, por ejemplo podemos insertar un informe realizado en Crystal Reports, de Visual Basic, entre otras aplicaciones, esto facilita un poco la migración de un sistema realizado en otro lenguaje a Visual C++, por ejemplo si quisiéramos migrar un sistema realizado en Visual Basic a Visual C++, por medio del Active X podemos reutilizar algunas partes generadas en Visual Basic y esto nos facilitaría el trabajo, aunque esto ultimo no es muy recomendable ya que tendremos la problemática de que algunas partes del sistema se tienen que generar en Visual C++ y tendremos que componer el sistema de acuerdo a lo que ya se tiene generado, de hecho es mas recomendable generar un nuevo sistema y a este agregarle algunas aplicaciones generadas en el otro sistema con el fin de facilitar un poco la elaboración del nuevo.

Se pueden migrar sistemas de una versión anterior a una versión mas reciente de Visual C++, aunque al hacer esto debemos de realizar algunos cambios al sistema en la nueva versión para que funcione de manera adecuada, aunque es importante mencionar que estos cambios son mínimos. También podemos utilizar código realizado en lenguaje C de programación estructurada o C++ de programación orientada a objetos, aunque con estos últimos dos lenguajes debemos de tomar en cuenta que deberán ser de la familia Microsoft porque como ya se menciono que como existe la versión de Borland se pueden presentar algunos problemas aunque la programación sea muy parecida.

Al haber revisado este lenguaje podemos concluir con lo siguiente: Visual C++ es un lenguaje que nos ofrece una gran potencia para la realización de sistemas de cualquier tipo, nos ofrece muchas herramientas de programación, muy buenas herramientas para presentación, etc. pero es un lenguaje un poco difícil para cuando no se ha tenido experiencia con lenguaje C++, de hecho resulta bastante complejo realizar un sistema en este lenguaje debido sobre todo a que muchas de las herramientas tienen que se programadas por el mismo usuario, aunque el lenguaje crea la mayoría de las clases necesarias para cualquier sistema, estas clases tienen que complementarse con programación por medio del usuario. También es importante tomar en cuenta que Visual C++ no ofrece una herramienta como manejador de bases de datos y eso es una gran desventaja, ya que aunque ofrece herramientas para ligar bases de datos con nuestros sistemas los programadores están acostumbrados a que la mayoría de los lenguajes visuales ofrecen facilidad de manejo con las bases de datos y contienen su propio manejador. A juicio del autor Visual C++ es buen lenguaje pero no muy recomendable para sistemas de bases de datos, sobre todo si no se es un programador experto.

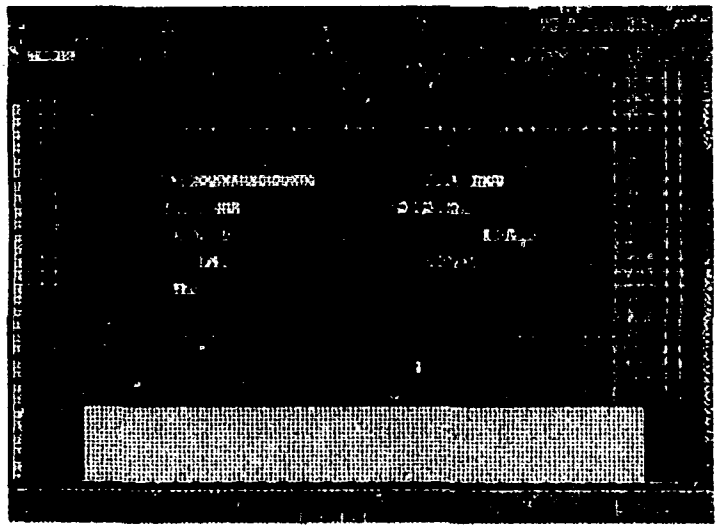
APENDICE A. PRESENTACION DE PANTALLAS

En este apéndice se presentaran las pantallas básicas de los diferentes sistemas, esto es con la finalidad de mostrar las diferentes presentación que los sistemas tienen y para poder visualizar de manera gráfica el manejo de la información. Las pantallas se presentan con información para visualizar también los datos que se consideran los adecuados para un buen manejo del sistema.

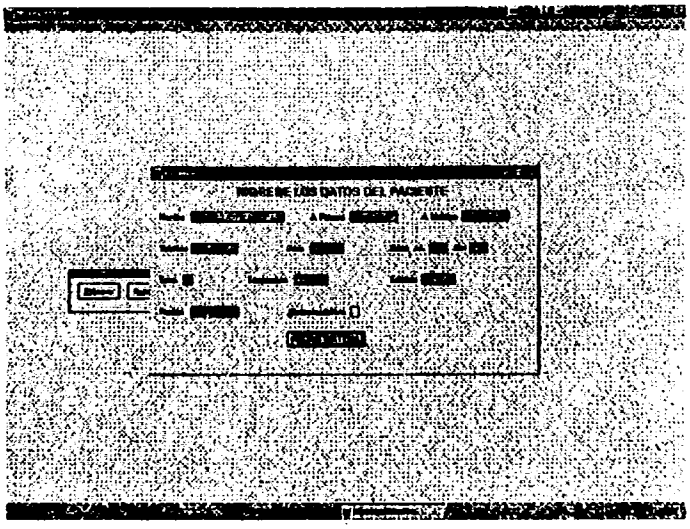
Para poder comparar mejor estas pantallas se han organizado de acuerdo a la tarea para la cual fueron diseñadas, por lo cual para cada una de las tareas encontraremos 5 pantallas pertenecientes cada una a cada uno de los lenguajes revisados en este trabajo.

Pantalla para la recepción de datos del paciente.

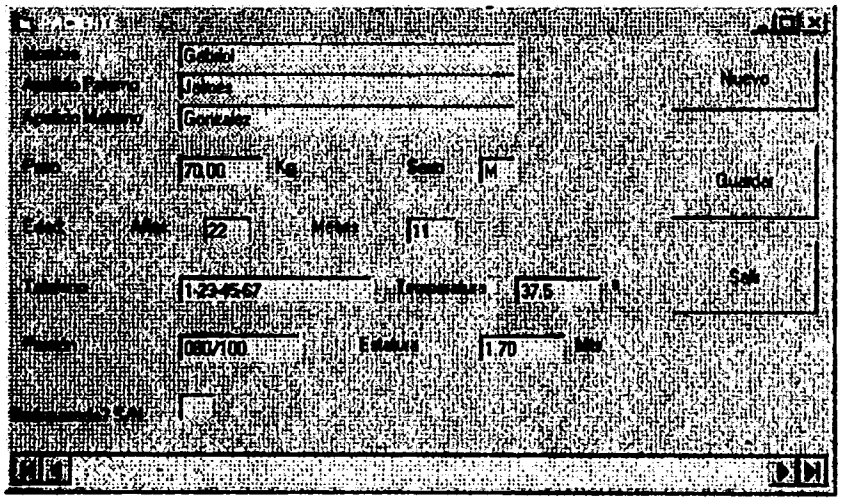
Clipper



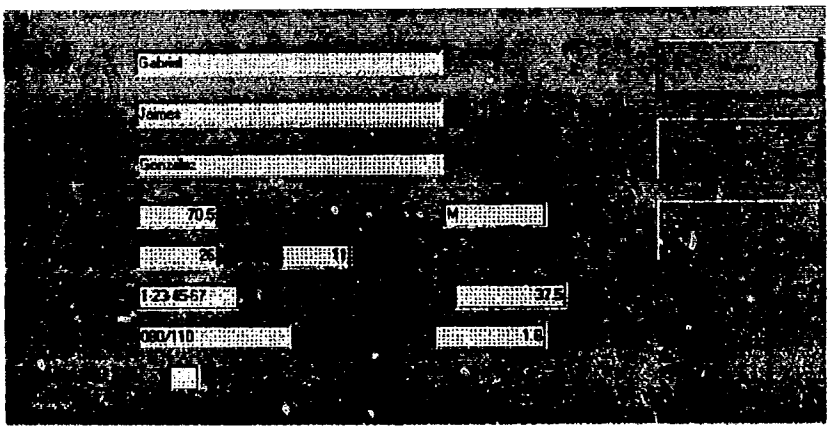
FoxPro



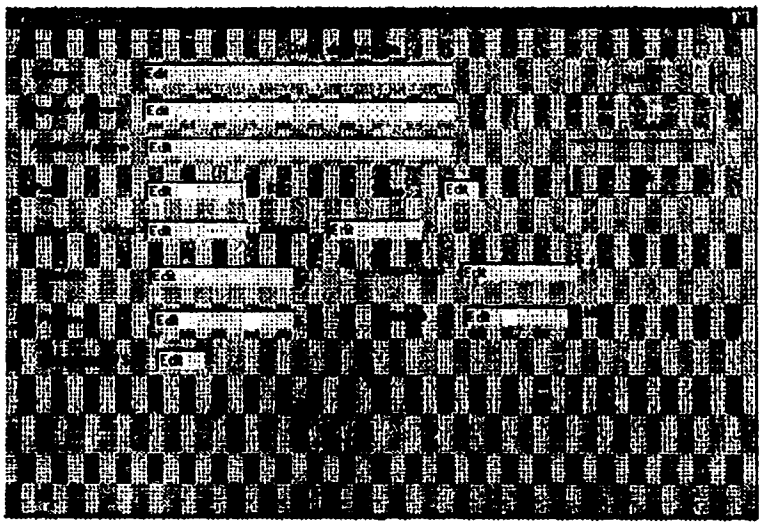
Visual Basic



Delphi

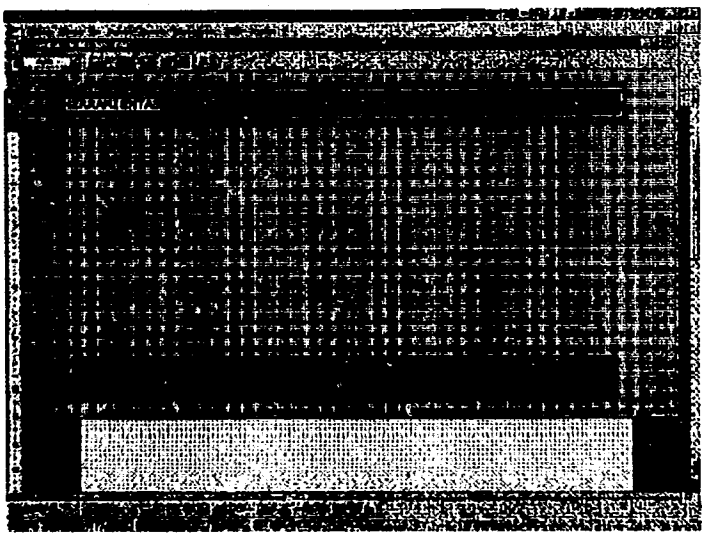


Visual C++



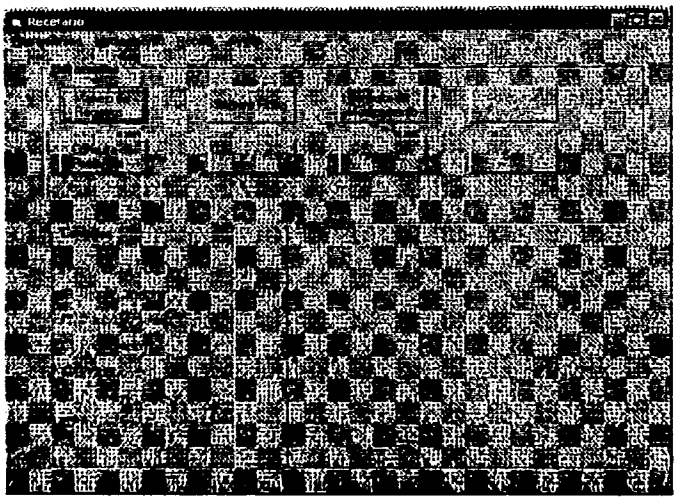
Pantalla general del sistema

Clipper

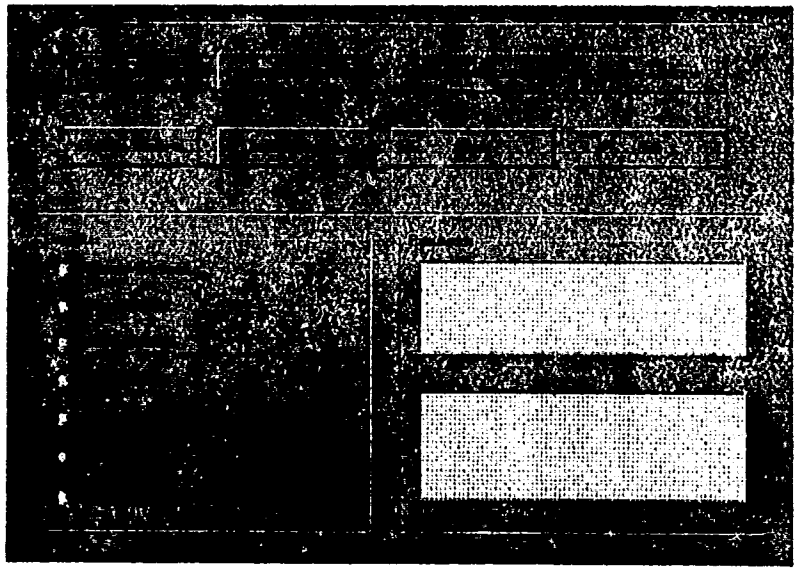


FoxPro

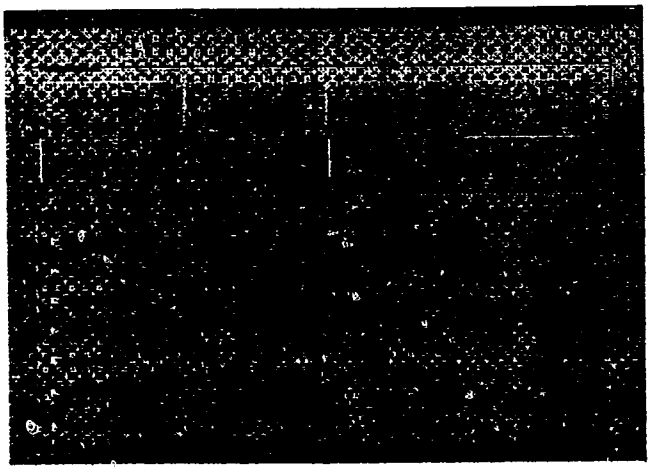
Visual Basic



Delphi

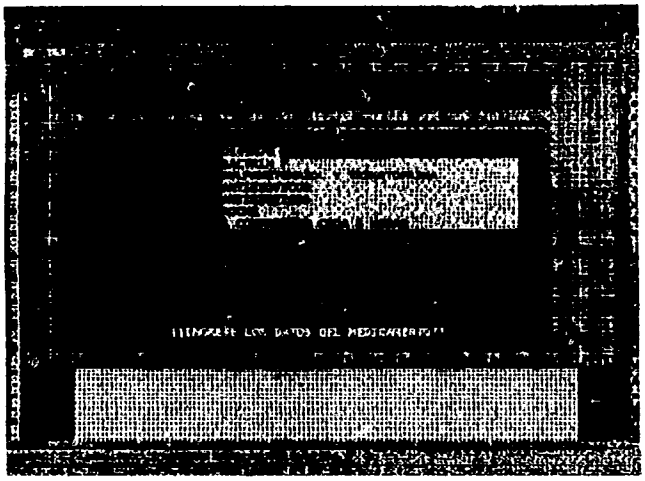


Visual C++

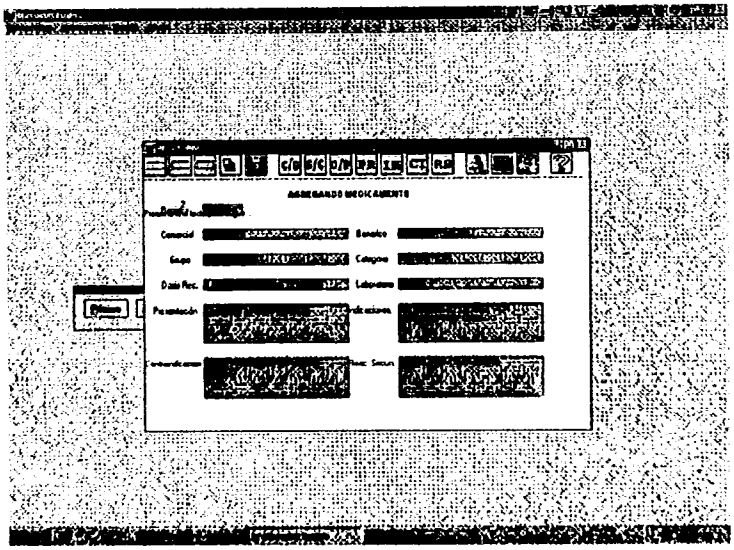


Pantalla de manejo de registros

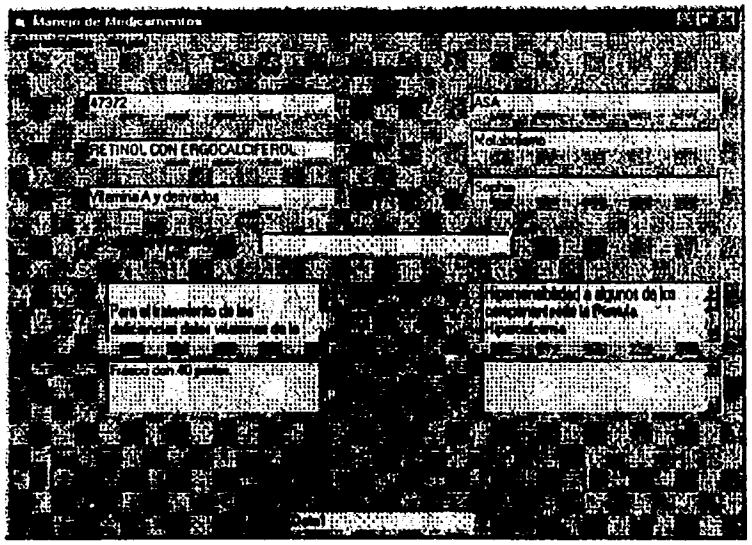
Clipper



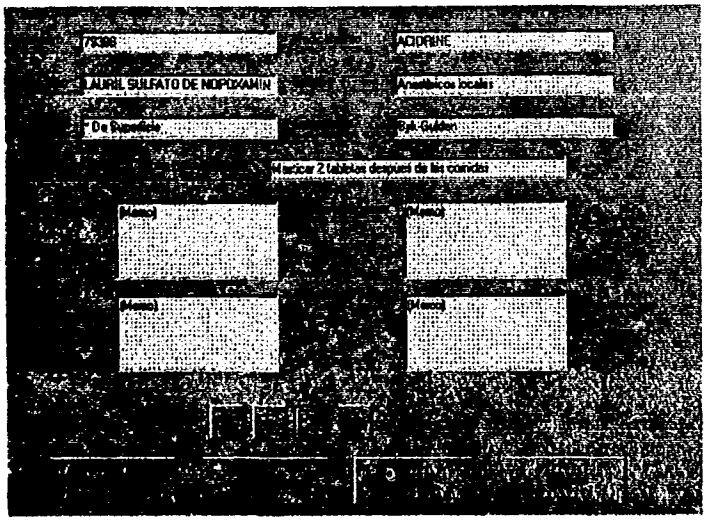
FoxPro



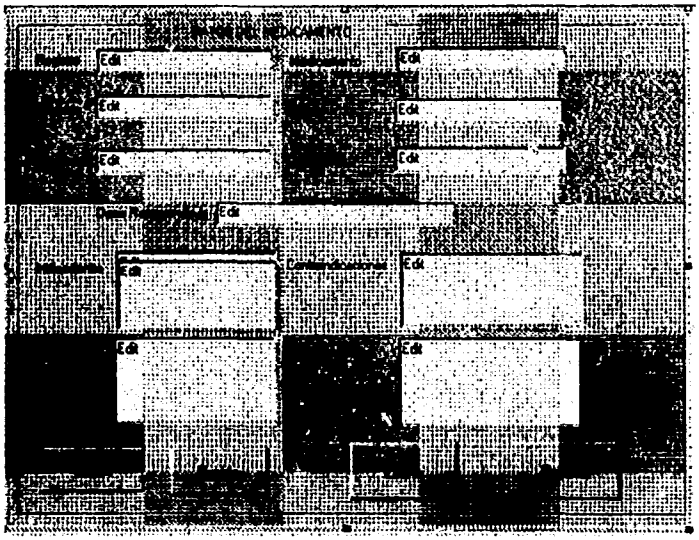
Visual Basic



Delphi

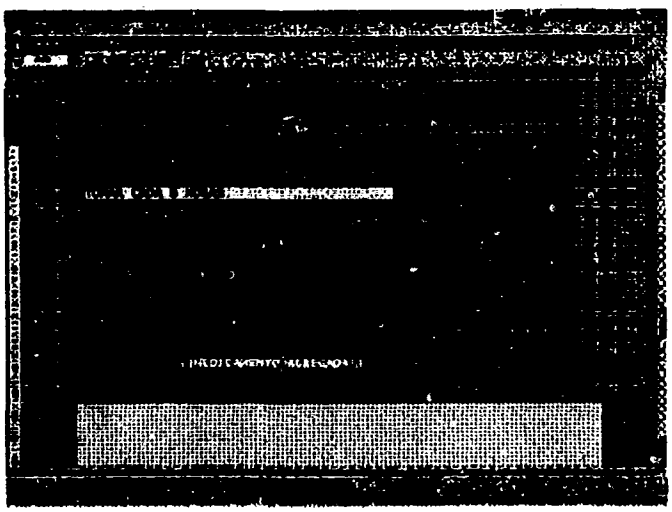


Visual C++

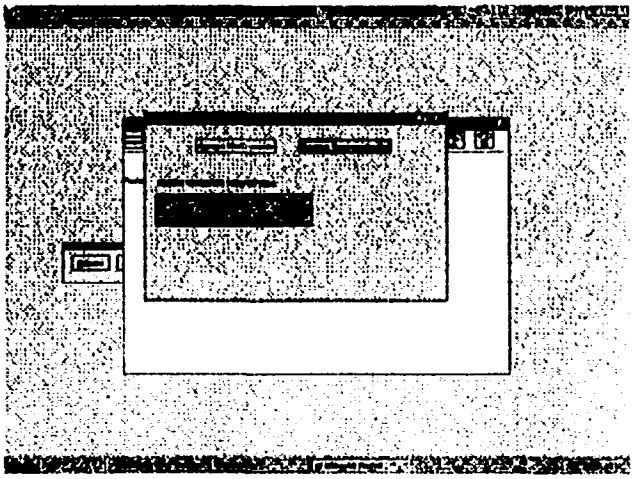


Pantalla de agregar medicamento o recomendación a receta.

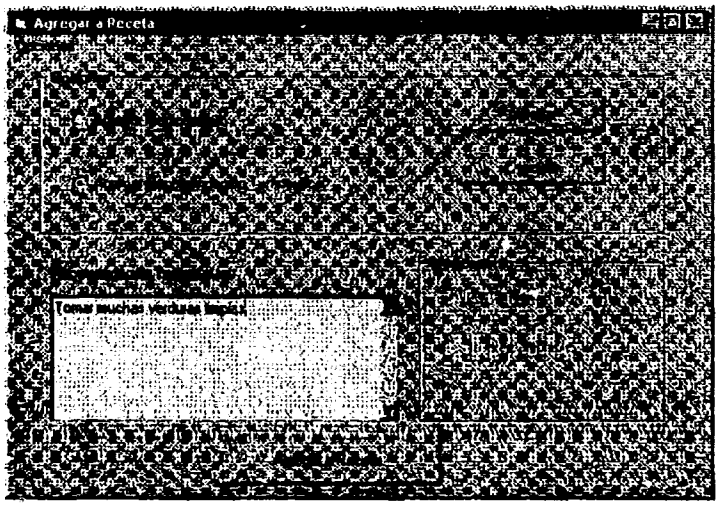
Clipper



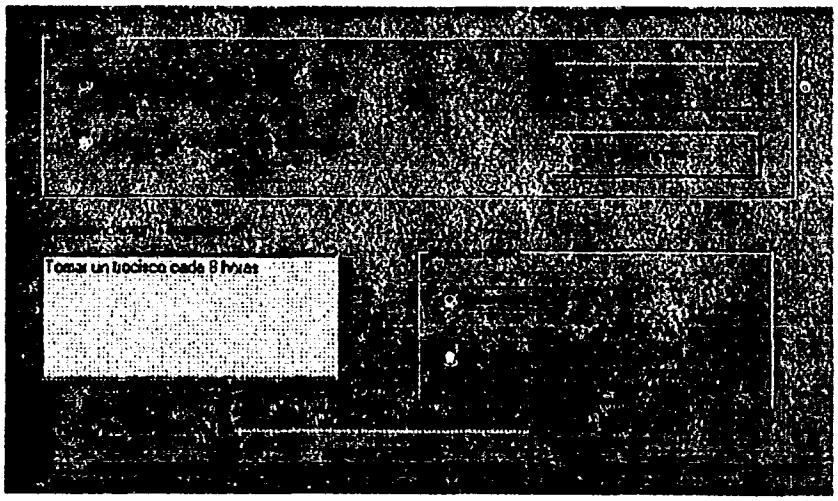
FoxPro



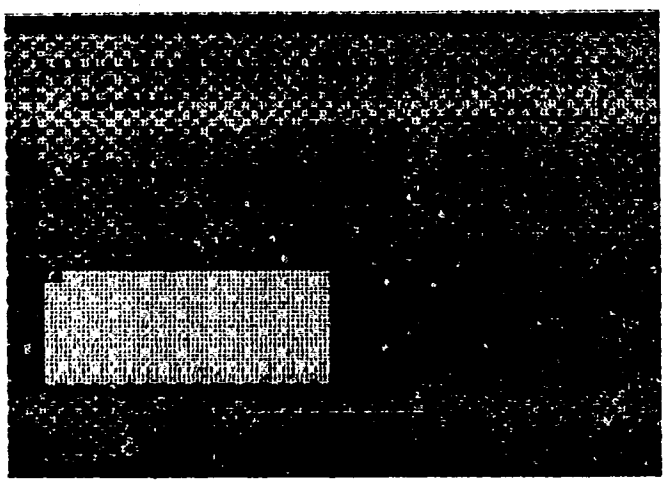
Visual Basic



Delphi

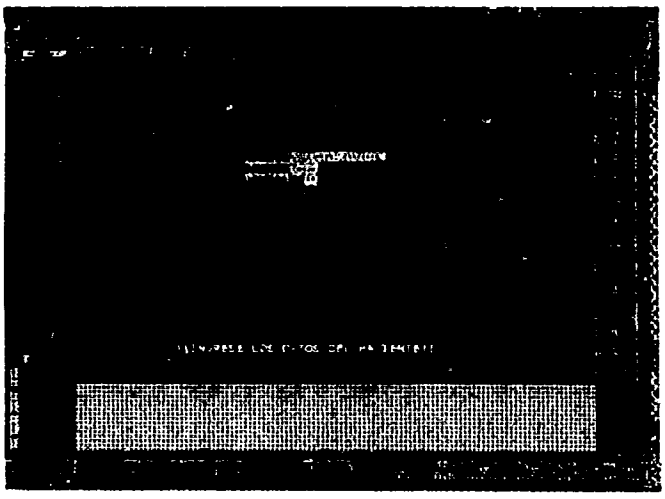


Visual C++

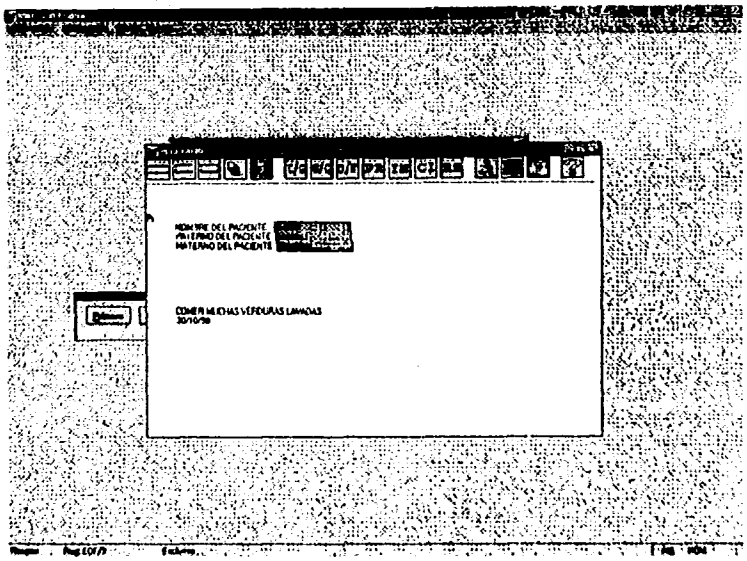


Pantalla de consulta de pacientes

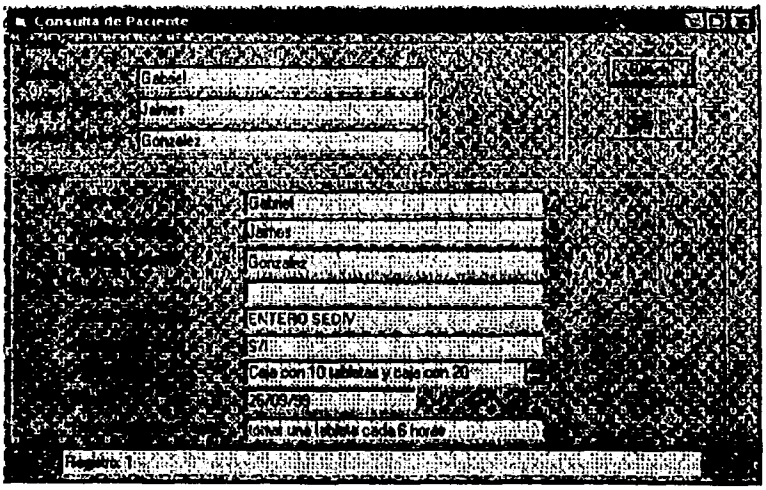
Clipper



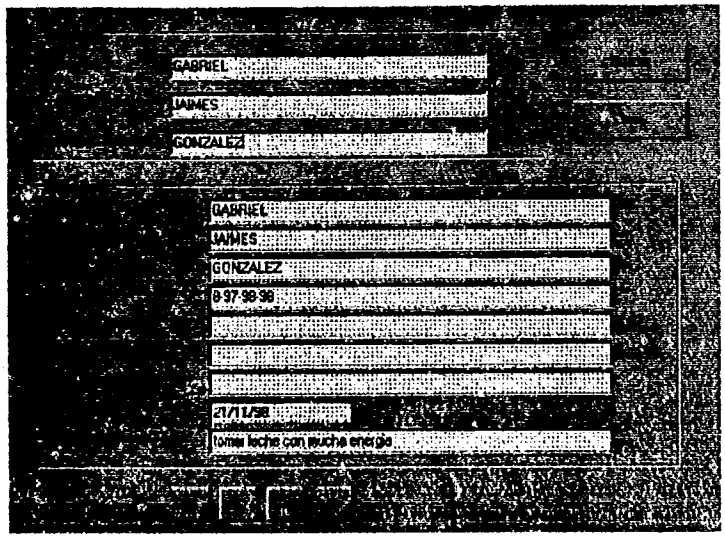
FoxPro



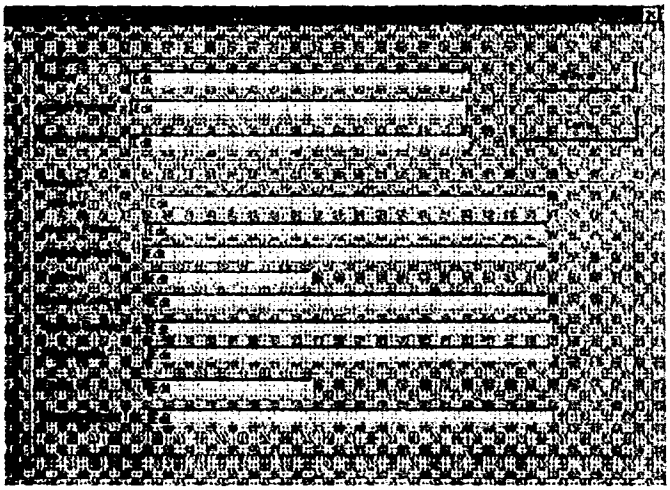
Visual Basic



Delphi

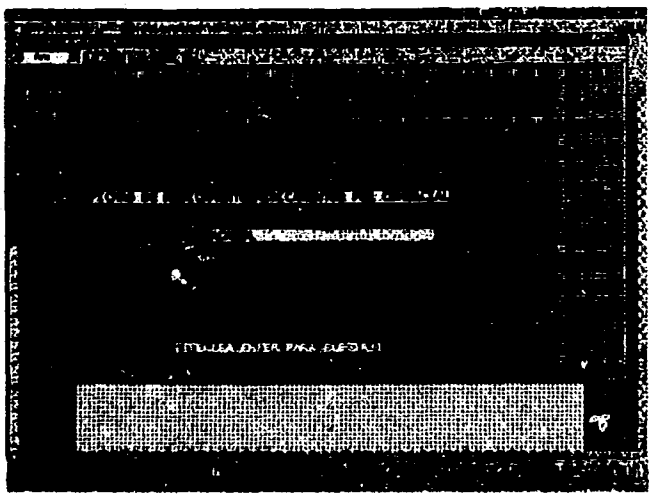


Visual C++

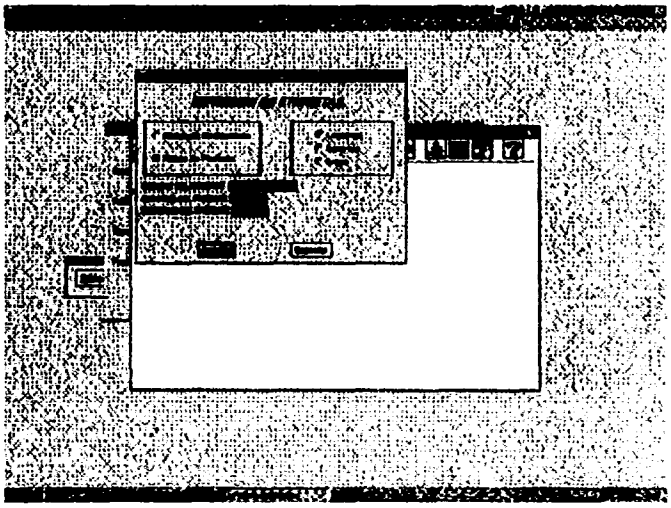


Pantalla de impresión de fichas.

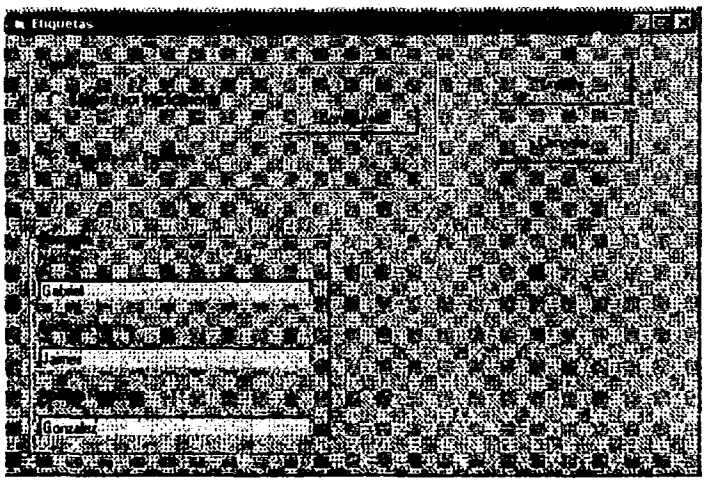
Clipper



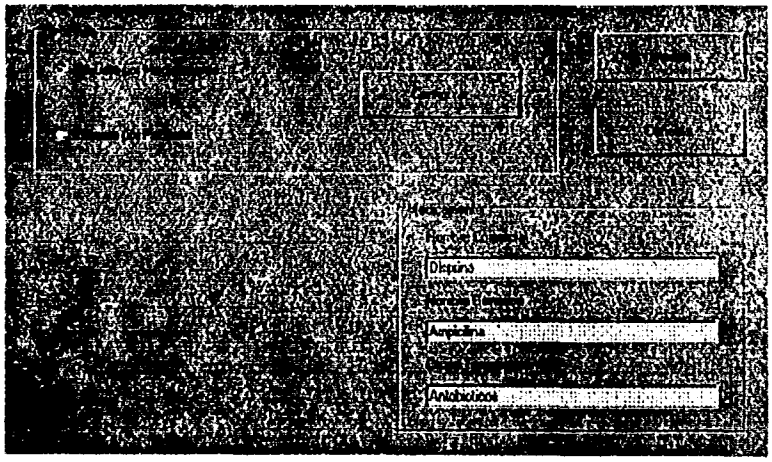
FoxPro



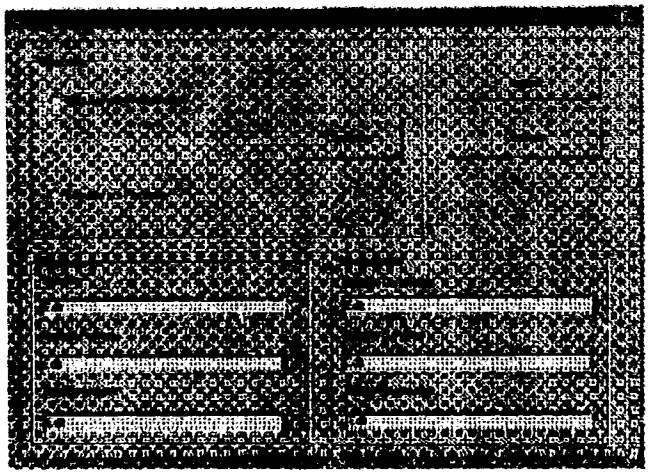
Visual Basic



Delphi

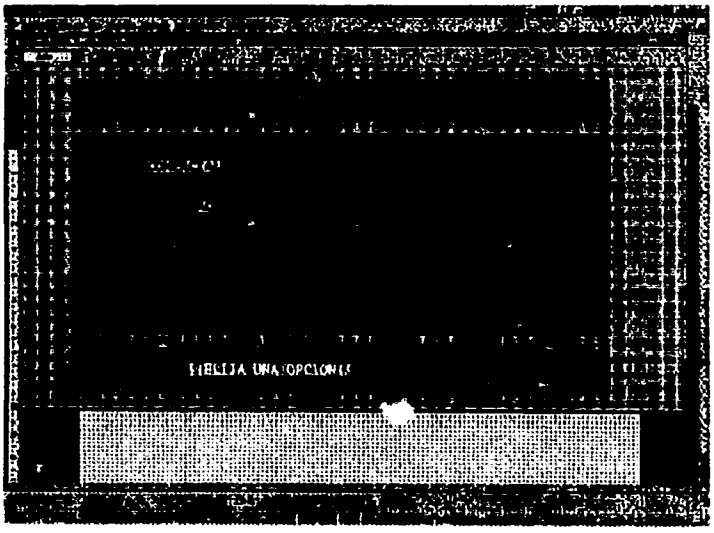


Visual C++

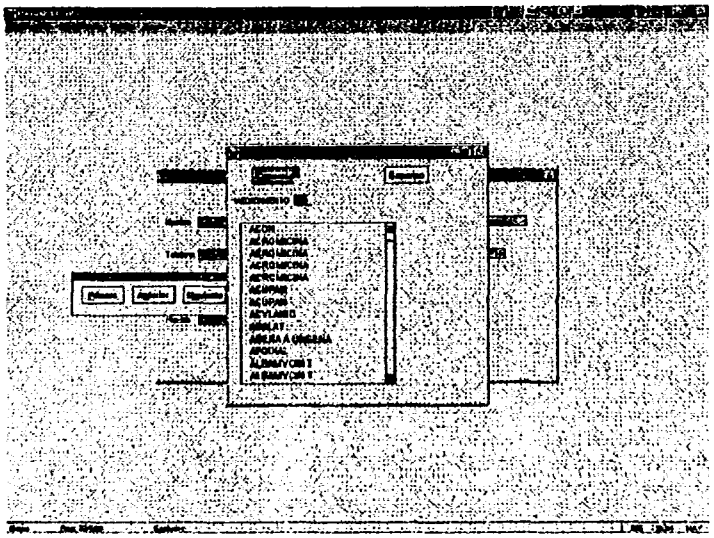


Pantalla de consulta de medicamentos.

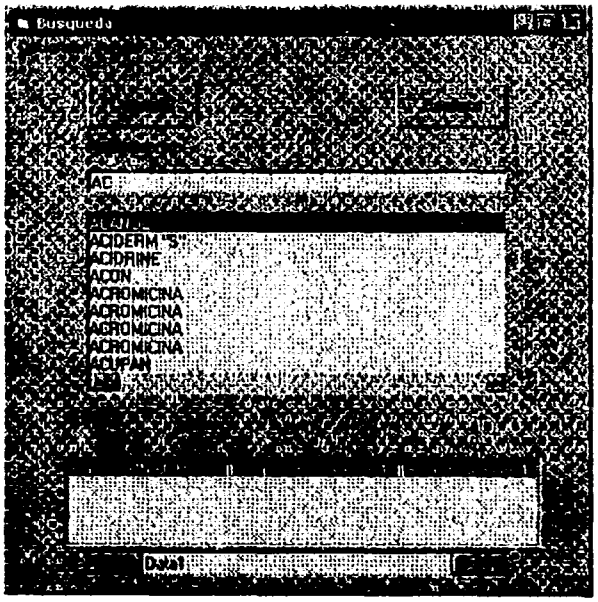
Clipper



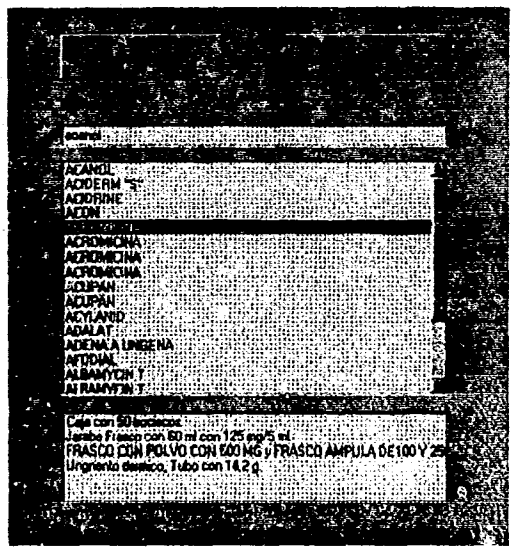
FoxPro



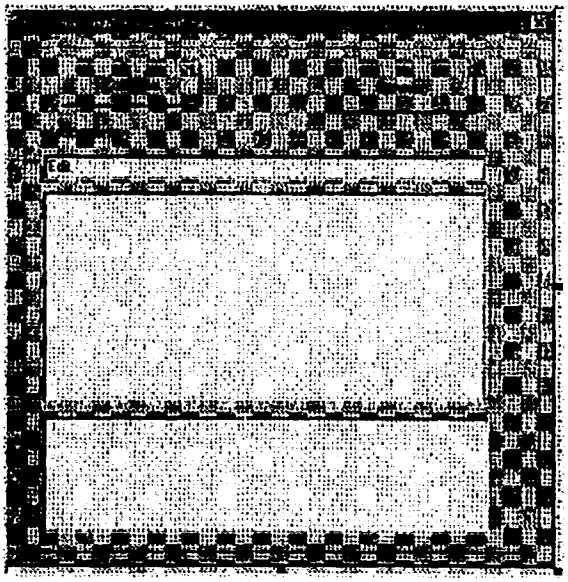
Visual Basic



Delphi



Visual C++



APENDICE B. PRESENTACION DE RECETAS.

DR Leonardo Mendoza Del

RAUL PEREZ PEREZ EDAD 22
PESO 75.500 TEL 1-23-45-67 PRESION 080/100
TEMPERATURA 37.50 ESTATURA 1.70

ACANOL (LOPERAMIDA)
Frasco CON 60 ML. NO INTERCAMBIABLE
Columbia
TOMAR CADA 6 HORAS

()

COMER MUCHA FIBRA

Av. R1 # 39 Col. Jardines de Sta. Clara
TELEFONO 5-67-87-88 D.G.P. 68768
FAX 7-87-67-68 S.S.A. 6768

DR Leonardo Mendoza Del RAUL PEREZ PEREZ EDAD 22
PESO 76 TEL 1-23-45-67 PRESION 080/100
TEMPERATURA 37.50 ESTATURA 1.70

ACON (PALMITATO DE VITAMINA A)
INTERCAMBIABLE
Rhone-Poulenc Rorer
TOMAR CADA 8 HORAS

0
INTERCAMBIABLE

COMER MUCHA FIBRA

Av. R1 # 39 Col. Jardines de Sta. Clara
TELEFONO 5676788 D.G.P. 68768
FAX 7876768 S.S.A. 6768

Doctor: Leonardo Mendoza Delgado
Nombre: RAUL PEREZ PEREZ Edad: 22 Anos
Telefono: 1-23-45-67 Peso: 75.50 Sexo: M
Temperatura: 37.50 Estatura: 1.70 Presion: 080/100

1 ACON PALMITATO DE VITAMINA A
Frasco con 15 ml y gotero. no Intercambiable
Rhone-Poulenc Rorer
TOMAR CADA 6 HORAS

2 no Intercambiable

COMER MUCHA FIBRA

S.S.A 6768
D.G.P. 68768
Av. R1 # 39 Col. Jardines de Sta. Clara
Tel: 5-67-87-88 Fax: 7-87-67-68

Doctor: Leonardo Mendoza Delgado

Nombre: RAUL PEREZ PEREZ

Telefono: 1-23-45-67 Peso: 75.5 Sexo: M

Temperatura: 37.5 Estatura: 1.7 Presion: 080/100

ACON PALMITATO DE VITAMINA A

Frasco con 15 ml y gotero. NO INTERCAMBIABLE

Rhone-Poulenc Rorer

TOMAR CADA 6 HORAS

NO INTERCAMBIABLE

COMER MUCHA FIBRA

S.S.A. 6768

D.G.P. 68768

Av. R1 # 39 Col. Jardines de Sta Clara

Tel. 5-67-87-68 Fax. 7-87-67-68

Doctor: Leonardo Mendoza Delgado

Nombre: RAUL		PEREZ	PEREZ	Edad: 22 Anos
Telefono: 1-23-45-67		Peso:	75.50	Sexo: M
Temperatura:	37.50	Estatura:	1.70	Presion: 080/100

1 ACON — PALMITATO DE VITAMINA A

Frasco con 15 ml y gotero.

Rhone-Poulenc Rorer

no Intercambiable

TOMAR CADA 6 HORAS

2

no Intercambiable

COMER MUCHA FIBRA

S.S.A 6768
D.G.P. 68768
Av. R1 # 39 Col. Jardines de Sta. Clara
Tel: 5-67-87-88 Fax; 7-87-67-68

Conclusiones.

En este trabajo hemos revisado 5 lenguajes todos ellos enfocados a las bases de datos. Por principio de cuentas como ya se menciono en este trabajo, las bases de datos forman tal vez la principal herramienta para todo tipo de aplicación, simple y sencillamente porque permiten almacenar grandes cantidades de información que puede ser manipulada posteriormente. Dentro del ambiente computacional es obvio que las bases de datos no pueden pasar desapercibidas y por ello es que en todos los lenguajes de programación existen herramientas para el manejo de estas, aun mejor, existen manejadores que permiten realizar tareas específicas con las bases de datos ya que son completamente dedicados a estos ambientes.

Este trabajo lleva a dos conclusiones básicas la primera de ellas nace de una pregunta ¿Es mejor utilizar manejadores de bases de datos que un lenguaje de programación para manipular la información?. Y la respuesta puede variar según las necesidades y prioridades de la gente. En forma muy personal considero que los lenguajes de programación son mejores por un simple hecho, estos sirven para crear sistemas que manipulen la información, es decir por medio de un lenguaje de programación podemos crear un manejador de bases de datos pero enfocado a las necesidades de una aplicación concreta. Si bien es cierto en los manejadores de bases de datos existen algunos productos muy buenos pero todos manejan las bases de datos de forma general, es decir no se enfocan a una aplicación directa como lo hace un sistema generado con un lenguaje cualquiera. También existe una ventaja de los lenguajes sobre los manejadores, estos últimos por lo general solo se enfocan a las bases de datos y el lenguaje por lo regular contiene otro tipo de herramientas que podemos utilizar para hacer que el sistema maneje no solo la información sino otros detalles que la aplicación requiera.

En el párrafo anterior no se debe confundir, estamos hablando de como manejan la información ambos ambientes, porque después de todo la base de datos siempre es almacenada en un manejador pero la forma de manipulación de la información es lo que nos

interesa. Por ejemplo hablemos de las cuatro acciones básicas de las bases de datos, altas, bajas, modificaciones y consultas, al generar un sistema que realice estas 4 acciones se puede hacer que el sistema sea muy amigable para el usuario incluso aunque este no sepa de computación y que por lo tanto resulte sencillo el manejo de la información, pero por otro lado el manejador ya se encuentra configurado y diseñado para realizar las cuatro tareas de forma determinada y se hace necesario que el usuario que manipule la información tenga conocimientos aunque sean básicos de computación.

En este trabajo revisamos básicamente lenguajes de programación aunque Clipper y FoxPro se podrían considerar manejadores de bases de datos, pero ambos cuentan con una amplia gama de comandos para ser considerados lenguajes de programación.

Durante cada capitulo se trato de recopilar las principales herramientas que ofrecen estos lenguajes para trabajar con las bases de datos. Además también se presentaron las ventajas y desventajas consideradas por el autor sobre los lenguajes presentados con la finalidad de tener una mejor visión de este software.

La segunda conclusión a la que se podría llegar (de hecho podría ser la conclusión general del trabajo) seria determinar cual de los lenguajes revisados es el mejor pero hacerlo resultaría hasta cierto punto dificil debido a las herramientas que nos ofrece cada uno de ellos pero cada lector puede decidir cual es el mejor de acuerdo a sus necesidades. Esto ultimo es muy importante ya que cada programador es diferente y tiene diferentes formas de pensar además también debemos de tomar en cuenta las necesidades que tiene el sistema que se quiere crear.

Para poder determinar que un lenguaje va de acuerdo con nuestro sistema debemos contemplar varias cuestiones:

- Bajo que ambiente correrá nuestro sistema.
- Con que equipo se cuenta actualmente en la empresa.

- Cuenta la empresa con los recursos económicos necesarios para adquirir nuevo equipo en caso de que el que exista no satisfaga los requerimientos del software.
- Que cantidad de información maneja el sistema.
- Con que licencias de software cuenta la empresa o cuanto esta dispuesto a invertir en estas licencias.
- Cual será el tipo de presentación que se quiere proporcione el sistema.

Note que poder determinar el lenguaje va enfocado mucho a la economía que la empresa o el usuario pueda ofrecer. Independientemente de los requerimientos que el usuario tenga algunas veces los sistemas no se pueden realizar en forma completa debido a las posibilidades económicas que la empresa ofrece, por ejemplo, alguna empresa requiere un sistema para Windows que maneje muchas herramientas, una buena presentación, va a manejar millones de datos, actualmente lo mas conveniente seria recomendar alguno de los lenguajes visuales (Visual Basic, Delphi o Visual C++) para realizarlo, pero que sucede si la empresa solo cuenta con equipos 386 y 486 y no dispone de recursos para adquirir equipo nuevo. Obviamente, como el sistema es necesario, el programador debe optar por realizarlo en uno de los lenguajes visuales mencionados pero en sus primeras versiones o en FoxPro para Windows. Es importante destacar que todos estos datos se obtienen durante el análisis del sistema.

Como se pudo ver en este trabajo, los cinco lenguajes ofrecen alguna forma de hacer todas las tareas relacionadas a un sistema de bases de datos, por lo que no se puede decir cual de estos lenguajes esta incompleto ya que cada uno ofrece las herramientas de acuerdo al ambiente para el que están diseñados. Algunos programadores o usuarios eligen realizar los sistemas en el lenguaje mas actual para buscar que tenga mas herramientas, algunas veces esto es bueno pero no siempre ya que debemos tener en cuenta que el programador y por lo tanto el lenguaje se debe adaptar a las necesidades del usuario o de la empresa que adquirirá el sistema y no al revés ya que al final de cuentas la empresa es el cliente que esta requiriendo un servicio dependiendo de sus recursos y necesidades. Algunas veces la empresa requiere un sistema que este trabajando en un maquina 286 o 386 y que no va a

interactuar un usuario directamente con el, simplemente el sistema estará realizando algunas labores, por ejemplo en algunos pisos de las instalaciones de Pemex, se cuenta con un sistema (hecho en Dbase III Plus) que se encarga de recibir el numero del empleado para registrar su hora de entrada y su hora de salida, es decir realiza las labores de checador. En este caso la empresa no requirió de un lenguaje actual ni nuevo para realizar este sistema por que el mismo sistema no lo requería.

Los cinco lenguajes que se revisaron en este trabajo son los que se consideran mejores en el ambiente para el cual fueron creados, y ya quedo establecido que las necesidades del sistema determinaran el lenguaje que se ha de utilizar. Lo anterior queda claro para diferenciar cuando utilizar Clipper en vez de Visual Basic o Delphi, pero que pasa cuando la empresa tiene las posibilidades económicas para generar un sistema de 32 bits, en este caso tenemos varios posibles lenguajes (Visual Basic, Delphi y Visual C++, entre otros) y una difícil elección.

Algunas veces la empresa tiene ya elegido el lenguaje en el que el sistema se realizara y esto basado en las licencias que la empresa tiene, algunas empresas (Bancomer por ejemplo) desde el principio adquirieron licencias de un lenguaje en particular y todos los nuevos sistemas se realizan en este lenguaje. Pero algunas empresas no cuentan con licencias y están dispuestas a adquirir alguna dependiendo de la decisión del programador. Aquí hay una ventaja, cuando se deja a decisión del programador el lenguaje, este se decide por el lenguaje que mas conoce o que se le hace mas sencillo y eso ayudara a que el sistema tenga mejores resultados.

A continuación presentamos un cuadro en el cual se resumen las herramientas mas comunes de los lenguajes revisados, este cuadro tiene la finalidad de presentar sobre todo al programador un esquema general de los lenguajes para facilitar su elección.

	CLIPPER 5.2	FOXPRO 2.6	VISUAL BASIC 4.0	DELPHI 2.0	VISUAL C++ 5.0
¿Para qué ambiente fue diseñado?	Dos	Windows 3.x	Windows 95	Windows 95	Windows 95
¿Contiene un manejador de base de datos integrado?	SI	SI	SI	SI	NO
¿Permite utilizar bases de datos de varios manejadores?	No	Si	Si	Si	Si
¿Cuenta con buenas herramientas para presentación	No	No	Si	Si	Si
¿Qué tipo de programación maneja?	Estruct.	Estruct.	Orient. A objetos	Orient. A objetos	Orient. A objetos
¿Permite manejar relaciones?	Si	Si	Si	Si	Si
¿Al termino del sistema requiere de mucho espacio para almacenarse?	Si	Si	No	No	No
¿Resulta fácil la programación?	Si	No	Si	Si	No
¿Los informes realizados ofrecen buena presentación?	No	Si	Si	Si	Si
¿La manipulación de la información de las tablas es sencilla?	Si	Si	Si	Si	Si
¿Contiene herramientas independientes a las bases de datos?	No	No	Si	Si	Si
¿Ofrece una buena compatibilidad?	No	Si	Si	Si	Si
¿Existe suficiente documentación y manuales de consulta?	Si	No	Si	No	Si

Al revisar este cuadro, en forma personal llego a determinar que el mejor lenguaje es Visual Basic, en las siguientes líneas tratare de explicar porque esta conclusión:

Visual Basic es el lenguaje mas comercial del momento, la facilidad que tiene la programación en este lenguaje es visible desde el primer momento en que se empieza a

trabajar sobre el, Visual Basic cuenta con herramientas muy potentes para todo tipo de aplicación (entiéndase, base de datos, Internet, redes, etc.). Comparado con Delphi que es muy parecido, encuentro la ventaja de que hay mas información sobre Visual Basic, es decir Visual Basic cuenta con mas manuales de consulta y aprendizaje que Delphi y eso es un gran ventaja, además siendo realistas el mercado de la computación en este momento esta ganado por la empresa Microsoft, en la actualidad es difícil encontrar una máquina que no tenga instalado Windows y Office de la familia Microsoft, al ser Visual Basic de esta familia hace que los enlaces con los demás paquetes sean mas sencillos. Comparado con Visual C++ (también de la familia Microsoft) presenta también varias ventajas, en el mercado es difícil encontrar Visual C++ en español a diferencia de Visual Basic, pero por otro lado para programar en Visual C++ se requiere de grandes conocimientos de programación y esta resulta un poco compleja, recuerde que muchos de los módulos en Visual C++ tienen que ser programados por el usuario y en Visual Basic no es así. Además Visual Basic contiene internamente controles completos para las bases de datos y Visual C++ no, por ejemplo en el caso de la creación de etiquetas e informes.

Pero, a final de cuentas, la elección de un lenguaje o manejador es única y exclusivamente personal y depende de lo que se quiera hacer, lo importante es que se obtengan los mejores resultados y el programador y el usuario queden convencidos de lo que se hizo y que la ventaja de poder manejar bases de datos se aproveche al máximo.

Referencias Bibliográficas.

Wiederhold, DISEÑO DE BASES DE DATOS , Edit. Mc Graw Hill, 1986

Goerge Tsu-der Chou, DBASE III PLUS GUIA DEL PROGRAMADOR, Edit Anaya,1990

S.M. Deen, FUNDAMENTOS DE SISTEMAS, Edit. Gustavo Gili S.A. , 1987

Roger S. Pressman, INGENIERIA DEL SOFTWARE, UN ENFOQUE PRACTICO, Edit. Mc Graw Hill, 1988

Cesar E. Rose, ARCHIVOS. ORGANIZACION Y PROCEDIMIENTOS. ,Edit. Computec, 1993

José Javier García-Badell, CLIPPER 5.2 A SU ALCANCE, Edit. Mc Graw Hill. Serie Mc Graw Hill de Informática, 1994

Stephen J. Straley, CLIPPER 5.0, Grupo Noriega Editores. Serie Megabyte, 1993

Francisco Marín Qrirós, CLIPPER 5 REFERENCIA RAPIDA, Edit. Macrobit, 1991

Sistemas de Imágen y Palabra, MICROSOFT FOXPRO PARA WINDOWS PASO A PASO, Edit. Mc Graw Hill, 1993

Sistemas de Imagen y Palabra, MANUAL PARA EL MANEJO DE FOXPRO 2.6 PARA WINDOWS, Edit Mc Graw Hill, 1993

Sal Ricecardi, MICROSOFT FOXPRO PARA MS-DOS, Edit. Mc Graw Hill, 1990

Sal Ricecardi, GUIA COMPLETA DE MICROSOFT FOXPRO, Edit. Mc Graw Hill, 1992

Gary Cornell, MANUAL DE VISUAL BASIC 3 PARA WINDOWS, Edit. Mc Graw Hill, Serie Mc Graw Hill de Informática, 1995

Gary Cornell, PROGRAMACION DE BASES DE DATOS CON VISUAL BASIC, Edit. Mc Graw Hill, 1995

Marco Antonio Tiznado Santana, EL CAMINO FACIL A VISUAL BASIC 4.0, Edit. Mc Graw Hill, Serie Enter, 1996

Francisco Charte Ojeda, VISUAL BASIC 5.0, Edit. Anaya Multimedia, 1997

Nathan Gurewich, APRENDIENDO DELPHI 3.0 EN 14 DIAS, Edit. Prentice Hall, 1998

Ori Gurewich, PROGRAMACION EN DELPHI, Edit Prentice Hall, 1997

Francisco Charte Ojeda, DELPHI 3.0, Edit. Anaya Multimedia, 1998

David J. Kruglinski, VISUAL C++, APLICACIONES PARA WINDOWS, Edit. Mc Graw Hill, 1997

David J. Kruglinski, PROGRAMACION AVANZADA CON VISUAL C++, Edit. Mc Graw Hill, 1998

Nathan Gurewich, APRENDIENDO VISUAL C++ 5 EN 21 DIAS, Edit Prentice Hall, 1998

Ori Gurewich, EL GRAN LIBRO DE VISUAL C++, Edit Prentice Hall, 1997

AYUDA EN LINEA CLIPPER 5.2

AYUDA EN LINEA FOXPRO 2.6 PARA WINDOWS

AYUDA EN LINEA VISUAL BASIC 4.0

AYUDA EN LINEA DELPHI 3.0

AYUDA EN LINEA VISUAL C++ 5.0