24
2 ej

UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"ARAGON"

FALLA DE ORIGEN

IMPLEMENTACION DE UN SISTEMA DE PRESTADORES DE BIENES Y SERVICIOS EN IXTAPA-ZIHUATANEJO

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N :

CRISTINA HERNANDEZ CAMPOS
VERONICA IVONNE VIRGILIO CUEVAS

ASESOR: M. en I. JUAN CARLOS ROA BEIZA

SAN JUAN DE ARAGON, EDO. DE MEX. 1995

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Primero contrui sobre la arena, después sobre la roca.
Cuando la roca se quebró
no constrúi sobre nada.
Después a menudo volví a construir
sobre arena y rocas, como saliera, pero
había aprendido.

Bertolt Brecht.

INDICE

CAPITULO II

PLANTEAMIENTO DE LA PROBLEMATICA Y PROPUESTA DE SOLUCION

CAPITULO III

DESARROLLO DEL SISTEMA

OBJETIVO:

El sistema deberá contemplar la solución de los siguientes requerimientos:

- Se utilizaran metodologías de programación Orientada a Objetos, tomando como base de desarrollo el lenguaje de programación visual C++.
- Uso de un menú gráfico para el acceso a otros menús, utilizando para ello dos dispositivos de acceso, el mouse y el teclado.
- Información sobre la Historia general del lugar
- Manejo de los mapas de las ciudades para la pronta localización de bienes, servicios y atracciones turísticas
- Localización e Información específica sobre un punto geográfico de interés deseado, utilizando tres radios de acción de diferente longitud
- Información sobre bienes, servicios y atracciones turísticas, de las ciudades mediante un ambiente gráfico
- Opción de un formato de Quejas donde el interesado pueda indicar algún Bien, Servicio o Atracción turística que no haya encontrado, o alguna característica que le gustaría que el sistema tuviera.
- Opción de un menú de ayuda al usuario en donde se explique el manejo del sistema.
- Obtener un sistema amigable y confiable

Introducción

Dado que el uso de la computadora se ha extendido hacia la mayoría de las actividades que el ser humano realiza, el siguiente trabajo tiene como propósito desarrollar e implementar un sistema para el sector tutístico, el cual permita al visitante o turista de algún lugar, localizar de manera ágil y eficiente, sitios de interés en un mapa, y que además le proporcione información turística sobre el lugar de una manera fácil.

Para llevar a cabo el propósito anterior nos basaremos en la técnica de la programación orientada a objetos, es decir, que mediente objetos gráficos se desarrolle un ambiente amigable para el usuario final. Dicho sistema sólo proporcionará información sobre los bienes, servicios y atracciones turísticas de las ciudades de Ixtapa y Zihuatanejo basandose en los mapas del lugar para la localización de los sitios de interés.

En cuanto a la elaboración del sistema se utilizó el lengueje de programación Visual C++, ya que este cuenta con poderosas herramientas que nos facilitan la programación, otorgandonos el beneficio de crear un código de programación formal.

CAPITULO I

CONCEPTOS TEORICOS

1.1 Teoria de Programación Orientada a Objetos

1.1.1 CONCEPTOS BASICOS

La programación orientada a objetos fue desarrollada, esencialmente, por las limitaciones que otras técnicas de programación tenían y tienen, como es el caso de la programación estructurada.

La idea fundamental de la programación estructurada es romper un programa en unidades más pequeñas denominadas funciones en C (procedimientos, subprogramas y suburbanas en otros lenguajes). Cada una de las funciones (al menos idealmente) tienen un propósito claramente definido así como una interfaz a las otras funciones del programa.

La programación estructurada utiliza el método descendente y de refinamiento sucesivo, y cada una de las funciones se invocan sucesivamente. Los programas estructurados son más fáciles de escribir y mantener que los programas no estructurados. Sin embargo, a medida que el tamaño de los programas aumenta y se hacen muy grandes y más complejos, el énfasis se pone en los tipos de datos que se procesan.

Las estructuras de datos en un programa se hacen tan importantes como las operaciones realizadas por ellos. Los tipos de datos se procesan en muchas funciones, dentro de un programa estructurado y cuando se producen cambios en esos tipos de datos, las modificaciones se deben hacer en cada posición que actúa sobre esos tipos de datos dentro del programa .

Esta tarea además de consumir gran tiempo puede ser frustante en programas que contengan millares de líneas de código y centenares de funciones. Esta circunstancia se produce esencialmente porque muchas funciones comparten datos (variables globales).

El último inconveniente se produce cuando un equipo de programadores trabaja en una aplicación. Ya que en un programa estructurado, a cada programador, se le asigna construir una tarea especifica de funciones y manejar los tipos de datos de esta. Dado que diferentes programadores manipulan funciones independientes que comparten datos, los cambios que un programador hace a los datos deben ser reflejados en el trabajo del resto del equipo. Los problemas y los errores de comunicación se reflejarán en el diseño final.

PROGRAMACION ORIENTADA A OBJETOS

La programación orientada a objetos enfatiza en los datos, al contrario de la programación estructurada que enfatiza en los algoritmos. La programación orientada a objetos permite organizar los datos de su programa al igual que los objetos que forman el mundo real; por ejemplo: los departamentos de una gran compañía (ventas, contabilidad, personal, técnico, etc.), o los diferentes vehículos que se construyen en una fábrica de automóviles, etc.; todos ellos son objetos.

En la programación orientada a objetos automóviles, árboles, departamentos, publicaciones, se conocen en general como objetos. Un objeto contiene en una sola entidad y con un único nombre las estructuras de datos y las acciones (procedimientos y las funciones) que actúan sobre este.

Así pues, una clase (objeto en general) es una plantilla o modelo que define los datos y las funciones que actúan sobre esos datos (llamados métodos). Por ejemplo, una clase puede describir las características fundamentales de un empleado de un hotel (nombre, título, salario, cargo, departamento al que pertenece, etc.), mientras que un objeto representará un ejecutivo específico (Andrés Montes, Lic. en Turismo, 5000.00, Gerente, etc.).

IDENTIDAD DE OBJETOS

Los objetos en una base de datos orientada a objetos, normalmente corresponden a una entidad en la empresa que está modelando la base de datos. Una entidad conserva su identidad aún cuando algunas de sus propiedades cambien con el tiempo. Igualmente, un objeto conserva su identidad aún cuando algunos o todos los valores de las variables o las definiciones de los métodos cambien con el tiempo. Este concepto de identidad no se aplica a las tuplas[1] de una base de datos relacional. En los sistemas relacionales, las tuplas de una relación se distinguen únicamente por los valores que contienen.

[1] En la administración de bases de datos relacionales, un registro o una fila.

A continuación se presentan varias formas de identidad:

- Valor. Se utiliza un valor de dato por identidad. Esta es la forma de identidad que se usa en los sistemas relacionales.
- Nombre. Se utiliza un nombre facilitado por el usuario por identidad. Esta es la forma de identidad que normalmente se usa para variables en los procedimientos. A cada variable se la da un nombre que identifica de manera única a la variable sin importar el valor que contenga.
- Incorporación. Una noción de identidad es incorporar en el modelo de datos el lenguaje de programación, y no se requiere que el usuario proporcione ningún identificador. Esta es la forma de identidad que se usa en los sistemas orientados a objetos.

En general un objeto tiene asociado:

- Un conjunto de variables que contiene los datos del objeto. El valor de cada variable es un objeto.
- Un conjunto de mensajes a los que el objeto responde.
- Un método que es un trozo de código para implementar cada mensaje. Un método devuelve un valor como respuesta al mensaje.

El término mensaje en un contexto orientado a objeto no implica el uso de un mensaje físico en una red de computadoras, sino que se refiere al paso de solicitudes entre objetos sin tener en cuenta detalles específicos de implementación.

Puesto que el único interfaz externo que presenta un objeto es el conjunto de mensajes al que responde, es posible modificar la definición de métodos y variables sin afectar a otros objetos. También es posible sustituir una variable por un método que calcule un valor. Por ejemplo, un objeto de documento puede contener una variable de tamaño que contenga el número de bytes de texto en el documento o bien un método de tamaño que calcule el tamaño del documento leyendo el documento y contando el número de bytes.

La capacidad de modificar la definición de un objeto sin afectar al resto del sistema está considerada como una de las mayores ventajas del modelo de programación orientado a objetos.

El proceso de programación en un lenguaje orientado a objetos se caracteriza por:

- Crear clases que definen la presentación y el comportamiento de los objetos.
- Crear objetos a partir de la definición de la clase.
- Realizar la comunicación entre objetos a través del envío de mensajes (invocación a las funciones miembro).

Así por ejemplo, en C++ el tipo de clase (class) es una estructura (struct) más funciones, que manipulan los campos de datos. Su sintaxis es similar a una estructura, sustituyendo struct por class y añadiendo una posibilidad de controlar el grado de acceso a los datos y métodos de la clase manifestados en declaraciones public (pública), private (privada) y protect (protegida).

PROPIEDADES FUNDAMENTALES DE LA PROGRAMACION ORIENTADA A OBJETOS.

Las propiedades fundamentales de la programación orientada a objetos son:

- Abstracción de datos
- Encapsulación (encapsulamiento y ocultación de datos).
- Herencia
- Polimorfismo
- Reutilización

ABSTRACCION DE DATOS

La abstracción fue un concepto introducido en programación estructurada. Se define como la capacidad para examinar algo sin preocuparse de sus detalles internos. En un programa estructurado, es suficiente conocer la tarea específica que hace un procedimiento dado, y no es tan importante cómo se realiza esa tarea. Esta propiedad se conoce como abstracción funcional.

La abstracción de datos es a los datos lo que la abstracción funcional es a las operaciones. Con la abstracción de datos, las estructuras y los ítems[2] de datos se pueden utilizar sin tener que preocuparse sobre los detalles exactos de su realización (implementación).

[2] Ítem, elemento. Una unidad o miembro de un grupo.

La abstracción de datos libera al programador de preocuparse sobre detalles no esenciales. La abstracción de datos ha existido formalmente en todos los lenguajes de programación para elementos complicados. Sin embargo, han sido los lenguajes modernos como Modula-2, Ada y los lenguajes orientados a objetos (puros Smalltalk o híbridos C++ y Turbo Pascal 5.5/6.0/7.0) quienes han permitido definir sus propios tipos abstractos de datos.

Tipos abstractos de datos (TAD) es un tipo de dato definido por el programador que se puede manipular de un modo similar a los tipos de datos definidos o incorporados al sistema. Al igual que con los tipos definidos en el lenguaje, un TAD pertenece a un conjunto de valores legales de datos (rango) y se pueden realizar sobre esos valores un número de operaciones primitivas. Los usuarios pueden crear variables con valores cuyo rango esta en el conjunto de valores legales y pueden operar sobre esos valores las operaciones definidas.

Un objeto es simplemente un tipo abstracto de datos y, por ello, las nociones de programación orientada a objetos se construyen sobre la idea de dicho tipo de datos, añadiéndoles otras propiedades, como pueden ser la ocultación y la reutilización de código.

ENCAPSULAMIENTO Y OCULTACION DE LA INFORMACION

La abstracción y el encapsulamiento son conceptos complementarios: la abstracción se enfoca a la vista exterior de un objeto y el encapsulamiento (también conocido como ocultación de la información) previene a otros objetos ver su interior, donde el comportamiento de la abstracción se ha realizado (implantado).

ENCAPSULACION

El encapsulado protege del cambio a los usuarios de clases. Al proporcionar una especificación explícita para el empleo de un objeto de cierto tipo, los usuarios de clase y quienes la instrumentan establecen un acuerdo que define la responsabilidad de cada uno. El usuario se compromete a manipular los objetos sólo en una forma consistente con la especificación de la interfaz. El constructor garantiza que, siempre y cuando el usuario sólo se comunique con los objetos a través de la interfaz pública, los objetos siempre producirán los resultados esperados.

Algunos escritores han llegado al punto de llamar a este acuerdo un "contrato" , haciendo referencia al usuario como el "cliente" o "comprador" y al constructor como el "vendedor" o "agente". Esta metáfora muestra poco sobre las relaciones establecidas alrededor de la construcción y el uso de clases y objetos. Sin importar cuáles sean los nombres que se empleen, lo principal que debe tenerse en cuenta es la razón para distinguir entre el papel del comprador y el del usuario.

El encapsulado propicia una separación flexible de las preocupaciones, entre el constructor y el usuario. Esta separación de preocupaciones es posible a través de una interfaz pública a un objeto que puede hacer obligatoria un lenguaje de programación. Aunque es agradable que el usuario pueda contar con cierto comportamiento con base en una especificación de clase, el constructor es quien adquiere más libertad a partir de la relación.

El concepto de acceso público a funciones como una interfaz a módulos de programa no es exclusivo de la programación orientada a objetos. Ocultar datos y encapsular a través de módulos y archivos separados de código fuente se practica mucho en C.

El usuario puede construir una clase ficticia que simule los objetos con los que está trabajando el constructor. Esto permite al usuario avanzar y en realidad compilar, enlazar y ejecutar código que crea objetos y envía mensajes a los objetos que con el tiempo instrumentará el constructor. El compilador verifica que el usuario no haya violado nada de lo acordado acerca del protocolo de transmisión de mensajes. El constructor puede avanzar y probar muchas instrumentaciones distintas de una clase dada y efectuar experimentos para determinar cuáles son más eficientes en términos de espacio, velocidad y facilidades de desarrollo. El acuerdo da al constructor la máxima flexibilidad posible para probar diseños alternativos.

HERENCIA

La idea de clases conduce a la idea de herencia. En nuestra vida diaria el concepto de clases se divide en subclases. Así, las clases animales se dividen en: mamíferos, anfibios, insectos, etc. El principio en que se basan las sucesivas subclases es que cada subclase comparte características comunes con la clase con la que se deriva o desciende, por ejemplo, todas las clases de vehículos tienen un motor, ruedas y un volante o manillar. Además de las características comunes, cada subclase tiene sus propias características particulares, por ejemplo, número de ruedas, número de asientos, diferente portaequipajes, etc.

En la programación orientada objetos, las clases se pueden subdividir en subclases dicho en otro modo, una clase se puede deducir o derivar de otra clase.

La clase original se denomina clase base y, merced a la propiedad de herencia, una clase derivada o descendiente puede heredar las estructuras de datos y los métodos de su clase base.

Además, la nueva clase puede añadir nuevos elementos datos y métodos a los que hereda de su clase base. Cualquier clase base (incluida una descendiente) puede tener cualquier número de clases descendientes.

Mediante el mecanismo de la herencia se puede construir una jerarquía de clases que adoptará la forma típica de árbol, donde la clase base se llamará clase padre, y la clase descendiente o derivada, clase hija.

Existen dos tipos de herencia: simple y múltiple. La herencia simple, es aquella en la que una clase solo puede tener un ascendiente. Herencia múltiple se refiere a la capacidad de las clases para heredar variables y métodos de múltiples superclases, o es aquella en que un objeto puede tener más de un descendiente. La mayoría de los lenguajes soportan herencia simple. Solo algunos incorporan la propiedad de herencia múltiple: C++ (a partir de 2.0) y Eiffel. Un caso típico de herencia múltiple es el popular término multimedia.

Cuando se emplea la herencia múltiple es posible que se de ambigüedad en el caso en que pueda heredarse la misma variable o método de más de una superclase. No todos los casos de herencia múltiple conducen a ambigüedad.

¿Cuándo heredar?

Siempre que se construya una nueva clase, se debe determinar primero si se puede utilizar una clase preexistente como clase de base. A menudo se encuentran clases que proporcionan casi el comportamiento indicado. Estos son buenos candidatos, ya que se pueden heredar todas las características deseadas. Para deshabilitar una función de la clase progenitora o primaria, se emplea una técnica llamada sobrecarga de funciones.

¿Qué no se puede heredar?

Como en la vidad real, en C++ no todo se puede transmitir a través de la herencia. Esto se puede considerar en un principio como una desventaja o limitación artificial, pero en realidad sólo algunos casos especiales son inconsistentes por definición con la herencia:

- Constructores.
- Destructores.
- Nuevos operadores definidos por el usuario.
- Operadores de asignación definidos por el usuario.
- Relaciones Friend.

Las clases derivadas invocan automáticamente al constructor de la clase de base cuando se instancian; pero después de ser construidas, el constructor de la clase de base queda fuera de límites. Aunque los constructores de las clases de base son heredados y sólo puden ser invocados automáticamente por el compilador cuando se construye un objeto derivado. El constructor de una clase de base no pueden ser invocados de manera explicita en una clase derivada como otras funciones heredadas.

De manera analoga, los destructores están diseñados para ser invocados automáticamente cuando un objeto del campo de acción lo llama. En un programa no se permite la invocación explicita de un destructor.

Clases diseñadas para ser heredadas

Dado el enfasis que se hace en la facilidad de reutilización y la herencia en C++, la mayor parte del tiempo se crean clases que utilizan subsiguientemente como clases de base de otras clases.

POLIMORFISMO

Polimorfismo es la propiedad que permite a una operación tener diferente comportamiento en objetos diferentes. En otras palabras, objetos diferentes reaccionan de modo diferente al mismo mensaje. El polimorfismo juega un papel importante en la simplificación de la sintaxis al realizar la misma operación sobre una colección de objetos.

El polimorfismo requiere herencia; simplifica la tarea del programador generalizando la sintaxis de comunicación que permite tratar objetos de un tipo diferente de modo similar. El polimorfismo se aplica únicamente a los métodos heredados de una clase base.

Esta propiedad significa que un método puede tener un nombre que es compartido a lo largo de una jerarquía de clases. Cada clase puede tener una implementación diferente para ese método.

El polimorfismo depende de la ligadura, que es el proceso por el cual un método se asocia con una función real. Cuando los métodos polifórmicos se utilizan, el compilador no puede determinar a qué función llamar; la función especifica llamada depende de la clase del elemento a la cual se envia el mensaje; en consecuencia, la función a llamar se determina en tiempo de ejecución. Esta operación se conoce como ***ligadura tardía o postergada*** (late

binding[3]) ya que ocurre cuando el programa se está ejecutando. La ***ligadura temprana o previa*** ocurre para métodos no polifórmicos; el compilador conoce exactamente cuál es el método que se invoca, de modo que se puede construir una llamada directa en tiempo de compilación. Aunque la ligadura previa es muy eficiente la mayoría de los lenguajes orientados a objetos utilizan ligadura postergada para todos los métodos.

REUTILIZACION.

Una vez que una clase se ha escrito, creado, depurado y comprobado, se puede distribuir a otros programadores para utilizarla en sus propios programas. Es decir, se puede utilizar como bloque básico para construir un programa. Esta operación se llama ***reutilización***.

Los programas orientados a objetos se construyen a partir de componentes reutilizables de software. Esta operación es similar al modo en que una biblioteca de funciones, en un lenguaje procedural se puede incorporar en diferentes programas.

El concepto de herencia en programación orientada a objetos proporciona una extensión importante a la idea de reutilización. Un programador puede tomar una clase existente, perteneciente a una biblioteca de clases y, sin modificarla, le puede añadir características y propiedades adicionales.

[3] ligadura tardía, ligadura dinámica. La unión (link) de rutinas en el momento de la ejecución.

En resumen la programación orientada a objetos se basa en el concepto de encapsulamiento de datos y código que opera sobre esos datos en un objeto. Los objetos estructurados se agrupan en clases. El conjunto de clases está estructurado en sub y superclases basado en una extensión del concepto del modelo entidad-relación. Puesto que el valor de un dato en un objeto también es un objeto, es posible representar el contenido del objeto dando como resultado un objeto compuesto.

1.1.2 ARQUITECTURA DE UN SISTEMA CON PROGRAMACION ORIENTADA A OBJETOS.

El estado actual de la mayor parte de la ingeniería de software está algo atrasado con respecto a las demás áreas de la ingeniería. Cuando la mayoría de los ingenieros completan su trabajo y éste se vende, esperamos que funcione de manera correcta. No esperamos que los motores de los aviones exploten o que los edificios se derrumben, pero no nos sorprende que cierto software se comporte de manera extraña. La mayoría de los productos de hardware tienen garantía, pero la mayoría de los productos de software llevan una renuncia a la garantía.

El software que se vende no esta libre de errores, pero puede decirse que cuando estos aparecen lo hacen con una frecuencia bastante baja.

Cuando las personas empiezan a programar, en general sus maestros les dicen que piensen como una computadora. Esta técnica parecía servir cuando aprendíamos a programar, puesto que escribíamos programas muy sencillos. Sin embargo, los ciclos y las ramificaciones nos proveen de una gran cantidad de combinaciones de formas, de modo que no podemos pensar en todos ellos como una computadora. La operación de la mayoría del software real depende de condiciones que no se conocerán hasta ejecutar el software. En el caso de la multiprogramación, el equipo hace varias cosas a la vez, por lo que es imposible pensar como una computadora.

Para ayudarnos a lidiar con la complejidad, se comenzó a utilizar la programación estructurada. Hubo una reducción en el código, pero la programación seguia basándose en una secuencia esperada de instrucciones de ejecución. El esfuerzo por diseñar y depurar

programas, pensando en el orden que la computadora sigue para hacer las cosas, desembocó en un software que nadie entendía del todo.

El mundo de las técnicas orientadas a objetos es muy diferente. El diseñador piensa en términos de objetos y su comportamiento, y se genera el código. El generador de código no debe tener errores, de modo que no hemos eliminado la programación manual. Sin embargo la mayoría de los sistemas se pueden construir sin tener que pensar en ciclos, ramificaciones y estructuras para el control del programa.

El contraste entre la orientación por medio de procesos y la orientación a objetos se puede resumir de la manera siguiente. El procesamiento convencional de los datos se centra en los tipos de objetos cuya estructura de datos sólo pueda controlarse mediante los métodos de la clase del objeto. Ocurren eventos que modifican el estado de un objeto. Por lo general, la propia programación de cada cambio de estado es sencilla, por lo que dividimos la programación en partes relativamente sencillas. De hecho cada objeto lleva a cabo una función específica e independiente de los demás objetos.

ANALISIS Y DISEÑO DE UN SISTEMA ORIENTADO A OBJETOS

Cuando se analizan sistemas, se crean modelos del área de aplicación que interesa. Un modelo puede incorporar un sistema, centrarse en un área de la empresa o abarcar toda la empresa. El modelado de empresas es importante para la planeación de la automatización de las mismas.

El modelo representa un aspecto de la realidad y se construye de modo que ayude a comprender ésta. El modelo es mucho más sencillo que la realidad, al igual que un avión a escala es mucho más sencillo que un avión de verdad. Podemos manejar el modelo y esto nos ayudará a idear sistemas o rediseñar áreas de la empresa.

Con el análisis orientado a objetos, la forma de modelar la realidad difiere del análisis convencional, modelamos el mundo en términos de tipos de objetos y lo que le ocurre a éstos. Esto implica también diseñar y programar sistemas de forma orientada a objetos.

La figura 1.1.2.1 ilustra nuestra forma de construir sistemas. El analista crea un modelo del área de interes. El modelo se convierte en un diseño y más adelante en un código. El modelo debe representar como perciben los usuarios finales el área o lo que los usuarios desean que realice el sistema. En la medida de lo posible, este modelo debe tener una forma comprensible para los usuarios y ayudarlos a ser creativos con repecto a sus necesidades. El técnico utiliza el modelo y crea un diseño a partir del cual se pueda escribir o generar un código.

Los modelos que se construyen en el análisis orientado a objetos reflejan la realidad de modo más natural que los del análisis tradicional de sistemas. Después de todo, la realidad consta de objetos y eventos que cambian el estado de dichos objetos. Mediante las técnicas orientadas a objetos, construimos software que modela más fielmente el mundo real. Cuando el mundo real cambia, nuestro software es más fácil de cambiar, lo que es una ventaja real.

Quisieramos capturar el punto de vista de los usuarios con respecto al mundo y traducirlo en software de la manera más automática posible. Entonces, cuando cambien las necesidades de los usuarios, el software cambiará con ellas.

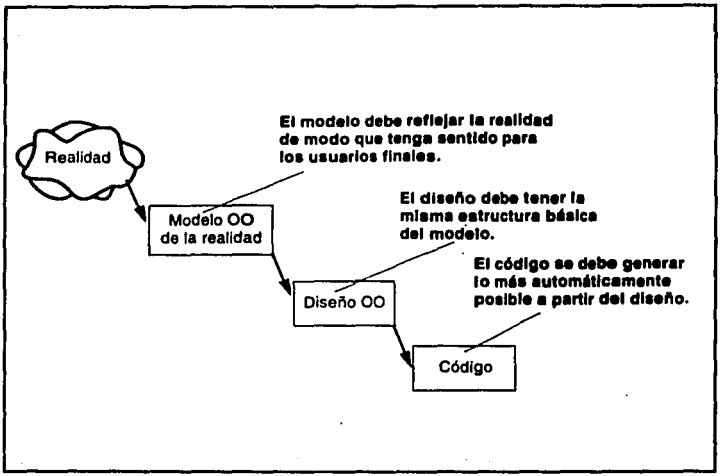


Fig. 1.1.2.1. muestra la forma de construir sistemas.

Por ejemplo, el diseñador de un microchip no pensaria en trabajar sin poderosas herramientas computarizadas. De la misma forma, se necesitan las herramientas para el diseño de software.

Con las herramientas Orientadas a Objetos, el diseño y código avanzan de manera natural a partir del modelo de alto nivel. Todo se relaciona con los tipos de objetos, los métodos que utilizan los objetos y los eventos que activan las operaciones con objetos. El depósito almacena muchas clases y nos ayuda a localizar las que sean utiles en el diseño.

DOS TIPOS DE MODELOS

En el análisis Orientado a objetos construimos dos modelos estrechamente relacionados: un modelo de los tipos de objetos y sus estructuras; y un modelo de lo que les ocurre a los objetos. Los modelos se representan mediante diagramas llamados *esquemas*. Los *esquemas* de objetos muestran las estructuras del objeto y los esquemas de eventos muestran lo que ocurre a los objetos.

El análisis y diseño Orientado a objetos tiene dos aspectos los que se ilustran en la figura 1.1.2.2. El primer aspecto se refiere a los tipos de objetos, clases, relaciones entre los objetos y herencia; se le conoce como análisis de la estructura de objetos (DEO).
El otro aspecto del comportamiento de los objetos y lo que les ocurre al paso del tiempo; se conoce como análisis del comportamiento de objetos (ACO) y diseño del comportamiento de objetos (DCO).

ANALISIS DE LA ESTRUCTURA DE OBJETOS

El análisis de la estructura de objetos (AEO) define las categorías de los objetos que percibimos y las formas en que los asociamos.

Durante el análisis de la estructura de objetos, el equipo de análisis se preocupa más por identificar los tipos de objetos que por identificar los objetos individuales en un sistema. Los tipos de objetos son importantes, puesto que crean los bloques conceptuales de construcción para el diseño de sistemas.

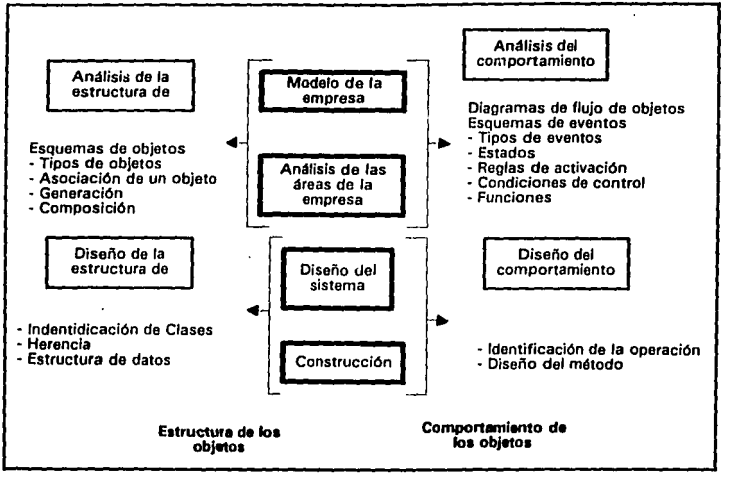


Fig. 1.1.2.2 Análisis y diseño se dividen en modelos relacionados entre sí; a la izquierda están los modelos de la estructura de un objeto y a la derecha están los modelos para el comportamiento de objetos.

En la programación orientada a objetos, estos bloques de construcción guían al diseñador en la definición de las clases y sus estructuras de datos. Además, los tipos de objetos son un índice para los procesos del sistema. En otras palabras, un objeto sólo debe ser controlado por medio de las funciones asociadas con su tipo. Así, las operaciones no se pueden definir de manera adecuada sin los tipos de objetos.

Los tipos de objetos que definimos y utilizamos son variados, puesto que los elegimos con base en la comprensión de nuestro mundo.

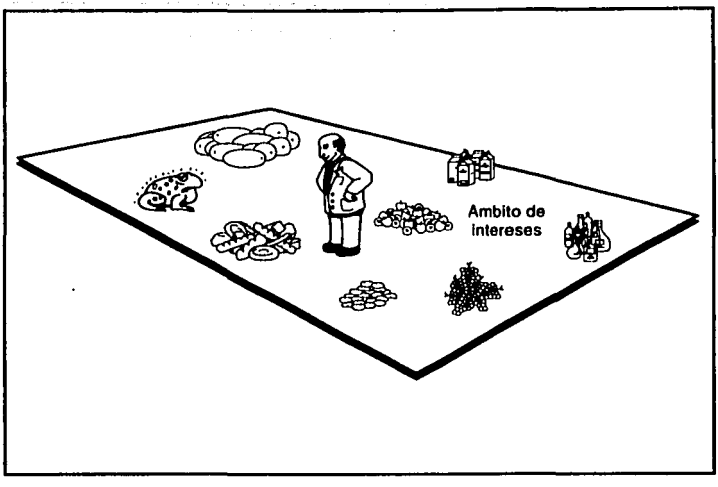


Fig. 1.1.2.3 Nosotros elegimos la forma de dar categorías a nuestro mundo.

Aunque esto pudiera parecer muy arbitrario, así funciona nuestra mente. De hecho, un objeto se puede categorizar en más de una forma.

ASOCIACIONES DE OBJETOS

Así como se ha mencionado, los tipos de objetos que clasifican los objetos de nuestro entorno son vitales en el análisis orientado a objetos: También es importante modelar la forma en como los objetos se asocian entre sí. En el análisis, es útil nombrar de alguna forma las asociaciones e indicar la cantidad de objetos de otro tipo, puesto que esto le da significado y aumenta la comprensión de la asociación.

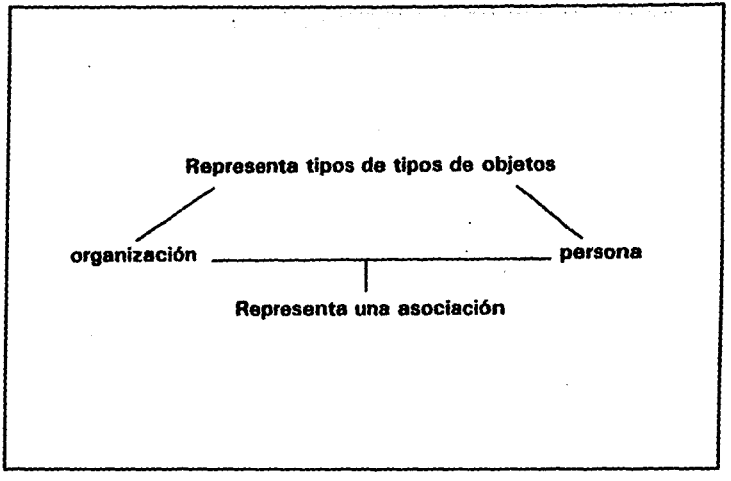


Fig. 1.1.2.4 Los objetos de un tipo se asocian con los objetos de otros tipos

Aunque la figura 1.1.2.4 ilustra las asociaciones entre dos tipos de objetos no se indica el significado de la asociación. Además en el significado, tampoco se indica la cantidad de objetos con los que un objeto dado puede y debe asociarse. En el análisis, es útil nombrar de alguna forma a las asociaciones e indicar la cantidad de objetos de un tipo dado que se deben asociar con los objetos de otro tipo, puesto que esto le da significado y aumenta la comprensión de la asociación.

GENERALIZACIÓN

La *generalización* es el resultado de distinguir un tipo de objeto como más general, o incluso, que es más que otro. Todo lo que se aplique a un tipo de objeto también se aplica a sus subtipos. Cada instancia de un tipo de objeto es también una instancia de sus supertipos.

Un tipo de objeto puede tener subtipos, sub-subtipos, etc. Por ejemplo, ácido es subtipo de líquido y ácido nítrico es un subtipo de ácido. El producto 739 es un subtipo de ácido nítrico, la figura 1.1.2.5 muestra esta jerarquía de generalización.

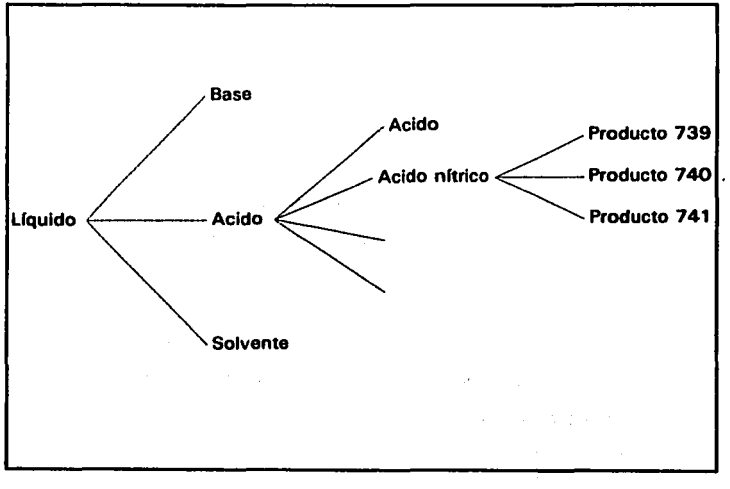


Fig. 1.1.2.5 Jerarquía de generalización.

La figura muestra una jerarquía en la que cada tipo no tiene más de un supertipo. Sin embargo es frecuente que un tipo tenga varios supertipos. Así, las jerearquías de generalización no tienen que ser necesariamente jerarquías de árbol.

Las jerarquías de generalización son importantes para el desarrollo orientado a objetos por dos razones. La primera es que el uso de supertipos y subtipos proporciona una herramienta útil para describir el mundo del sistema de aplicación. La segunda es que indica las direcciones de herencia entre las clases en los lenguajes de programción orientada a objetos.

JERARQUIAS COMPUESTAS

Algunos tipos de objetos se consideran complejos. Por complejo queremos indicar que determinados objetos estan formados por otros. Por ejemplo, un auto se puede describir como algo formado por chasis, llantas y motor. A su vez el motor esta compuesto por el monoblock, valvulas, pistones, etc. Además el objeto complejo formado por anillos, una biela y una cabeza.

En el análisis orientado a objetos, la composición de un objeto nos ayuda a describir el hecho de que los dibujos estan formados por determinada configuración de símbolos, los trabajos lo están por tareas específicas, las organizaciones por otras organizaciones, y así sucesivamente.

En los sistemas de tecnología avanzada, el análista describirá la forma en que un pedido no sólo puede constar de elementos en cada renglón, sino también contener instrucciones verbales del cliente o un diagrama hecho a mano. Los pedidos de este tipo se

llaman objetos complejos. Cada pedido se puede controlar como un objeto unico que conste de otros, los cuales a su vez, se pueden controlar de forma independiente, en caso necesario.

DIAGRAMAS DE RELACION ENTRE LOS OBJETOS

Los tipos de objetos están relacionados con otros tipos de objetos. Un empleado trabaja en una sucursal. Un cliente realiza un pedido de varios productos.

Los diagramas de relación entre entes se han venido utilizando por años en el análisis convencional de sistemas. Un diagrama de relación entre objetos es esencialmente igual a un diagrama de relación entre entes.

ESQUEMAS DE OBJETOS

La comprensión de un modelo suele ser más sencilla si los tipos de objetos y relaciones se representan mediante un diagrama de relación entre objetos; los supertipos y subtipos se representan en un diagrama de jerarquía de generalización y las estructuras compuestas en un diagrama compuesto. Sin embargo, para los usuarios más sofisticados puede ser útil presentarlo todo en el mismo diagrama. Este tipo de diagrama se llama esquema de objetos, como se muestra en la figura. 1.1.2.6.

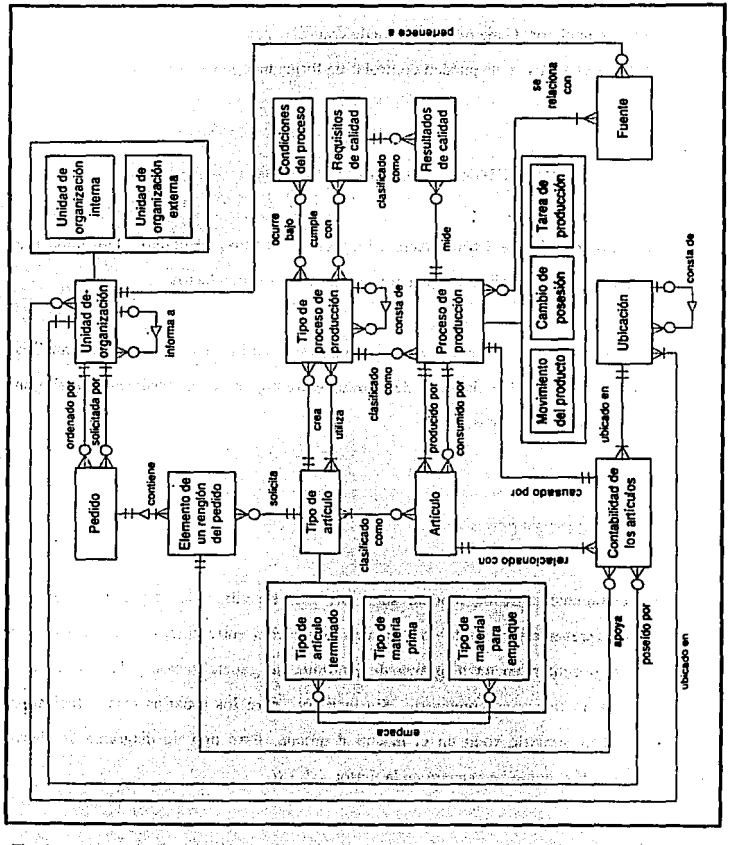


Fig. 1.1.2.6 Ejemplo de esquema de objetos para una aplicación en producción.

ANALISIS DEL COMPORTAMIENTO DE OBJETOS

En el análisis del comportamiento de objetos (ACO) realizamos esquemas de eventos que muestran eventos, la secuencia en que ocurren y cómo los eventos cambian el estado de los objetos. Así, los esquemas de eventos se deben expresar en términos de esquemas de objetos, puesto que los eventos cambian el estado de determinados tipos de objetos.

ESTADOS DE UN OBJETO

El estado de un objeto se puede definir como la colección de los tipos de objeto que se aplican a él.

Al implantarlo en un lenguaje de programación orientada a objetos, el estado se registra en los datos almacenados con relación al objeto. Se determina mediante las clases y valores de los campos de los datos asociados con el objeto. Así en general, los programadores orientados a objetos utilizan otra definición de estados.

En el lenguaje orientado a objetos, las solicitudes se envían y provocan la activación de los métodos. Los métodos cambian el estado del objeto. El estado se registra en los datos del objeto.

TIPOS DE EVENTOS

En el análisis orientado a objetos el mundo se describe en términos de los objetos y sus estados, así como de los eventos que modifican esos estados. Un *evento* es un cambio en el estado de un objeto.

Sin los eventos, el mundo no cambiaría. En un mundo sin eventos podríamos construir y generalizar bases de datos sin preocuparnos por actualizarlas. Sin embargo, en la mayoría de las aplicaciones, sí debe de cambiar el contenido de las bases de datos. Puesto que deseamos saber de esos cambios y reaccionar en forma adecuada ante ellos, debemos entender y modelar los eventos.

Los tipos de eventos indican los cambios sencillos en el estado de un objeto. Básicamente, los tipos de eventos describen las siguientes formas de cambios de estado:

- Un objeto se crea. Por ejemplo se crea una reservación aérea.
- Un objeto se termina. Por ejemplo, un producto se destruye o un contrato se termina.
- Un objeto se clasifica como una instancia de un tipo de objeto. Por ejemplo, un empleado se convierte en gerente.
- Un objeto se desclasifica como una instancia de un tipo de objeto. Por ejemplo, una firma deja de ser cliente; un producto sale del catalogo de ventas.
- Un objeto cambia de clasificación. Por ejemplo, un abogado pasa de ayudante a socio; una cuenta cambia de normal a atrasada.
- El atributo de un objeto se cambia.

Los eventos pueden asociar un objeto con otro. Por ejemplo, en la mayoría de las organizaciones, cuando un objeto se clasifica como **empleado**, debe estar asociado con un Departamento. Un objeto clasificará al objeto como **empleado**. Otro evento creará una asociación entre el objeto **empleado**, y un objeto **Departamento**. Si el empleado, cambia de departamento, se crea una nueva asignación **empleado-departamento** y se termina la anterior.

La figura 1.1.2.7 muestra los objetivos tanto para el análisis como para el diseño.

El análisis de la estructura de objetos se ocupa de definir las categorías de objetos y la forma en que los asociamos. Preguntamos: ¿Qué tipos de objetos hay? ¿Cuáles son sus relaciones y funciones? ¿Qué subtipos y supertipos son útiles? ¿Hay algún tipo de objeto compuesto por otros objetos?

Cuando el análisis pasa a la etapa del diseño de la estructura de objetos, identificamos las clases (la implantación de tipos y objetos). Se definen las superclases, subclases, rutas de herencia y los métodos a utilizar y se lleva a cabo el diseño en detalle de las estructuras de datos.

El análisis del comportamiento de objetos se ocupa de modelar lo que ocurre a los objetos al paso del tiempo. Nos preguntamos ¿En qué situación pueden estar las clases de objetos? ¿Qué tipo de eventos cambian estos estados? ¿Qué sucesión de eventos ocurre? ¿Qué funciones resultan de estos eventos y cómo se activan?

Después sigue la etapa de diseño. En ésta nos preocupamos por el diseño detallado de métodos, ya sea con técnicas por procedimientos o de no procedimientos. Se crea la entrada para los generadores de códigos. Se diseña la pantalla y se diseñan y generan los diálogos. Finalmente, se construyen y desarrollan los prototipos.

Análisis de la estructura de objetos (AEO)	Análisis de Comportamiento de objetos (ACO)
Se ocupa de los tipos de objetos y sus asociaciones	**Se ocupa de lo que le sucede a los objetos al paso del tiempo**
• Tipos de objetos y asociaciones • Diagramas de generalización • Diagramas de relación entre los objetos • Diagramas de componentes	• Diagrama de flujo de objetos • Esquemas de eventos • Diagramas de funcionamiento que muestran las funciones y la secuencia en que deben ocurrir • Estados de objetos y cambios en dichos estados

Diseño de la estructura de objetos (DEO)	Diseño del comportamiento de objetos (DCO)
Se ocupa de las clases, métodos y herencia	**Se ocupa del diseño de métodos**
• Clases • Superclases y subclases • Herencia • Estructuras de datos • Diseño de una base de datos	• Métodos y funciones • Lógica de procedimientos • Código de no procedimientos • Entrada para los generadores de códigos • Diseño de la pantalla y del diálogo • Fabricación de prototipos.

Fig. 1.1.2.7 Las dos mitades del análisis y diseño OO.

1.1.3 DESARROLLO DE SOFTWARE ORIENTADO A OBJETOS.

La velocidad a la que avanza la tecnología de hardware de computadoras es sorprendente; año con año se logran avances que conducen a la construcción de computadoras más veloces, más compactas y baratas. Sin embargo, en el aspecto de software no parece haber un desarrollo similar. Mientras que los costos de hardware han disminuido continuamente, los de software han hecho lo contrario. La construcción de software no es una tarea fácil y en muchas ocasiones los proyectos de programación sobregiran los presupuestos de tiempo y dinero.

EL PROBLEMA DEL MANTENIMIENTO DE SOFTWARE.

La construcción de software ha recibido la atención de los expertos desde hace mucho tiempo; en la década de los 70's se consiguieron avances significativos hacia el desarrollo de metodologías para construir programas en forma sistemática y a bajo costo. Como resultado de esos esfuerzos surgieron técnicas que, como el diseño estructurado y el desarrollo descendente (top-down), durante mucho tiempo han sido las herramientas utilizadas por los programadores para construir software, Aunque dichas técnicas han sido empleadas durante mucho tiempo en proyectos realmente complejos, la mayoría de los ingenieros de software coinciden en afirmar que sufren de tres grandes deficiencias:

- Los productos que resultan al emplear estas técnicas son poco flexibles.

- Los programadores que los usan tienden a concentrarse en el diseño y la implementación inicial del sistema, sin tomar en cuenta su vida posterior.

- No alientan al programador a aprovechar el trabajo de proyectos anteriores.

La desventaja de utilizar una metodología que se concentra en el diseño inicial del sistema se hace evidente si se toma en cuenta que la vida útil de un producto de software puede ser cinco ó seis veces más grande que el lapso en que se desarrolla; por ejemplo, un sistema que se desarrolla en uno ó dos años puede mantenerse trabajando durante un periodo que va de cinco a quince años.

Los gastos que se hacen durante este último período (gastos de mantenimiento) representan alrededor del 70% del costo total del sistema. La figura 1.1.3.1 ilustra la forma en que se distribuyen estos gastos.

La gráfica muestra que cerca del 60% de los costos de mantenimiento de un sistema (alrededor del 35% del costo total) se tienen que hacer por cambios en las especificaciones del usuario ó en los formatos de los datos. Es por ello que la extensibilidad (la facilidad con que se modifica un sistema para que realice nuevas funciones) debe ser uno de los objetivos primarios de la etapa de diseño.

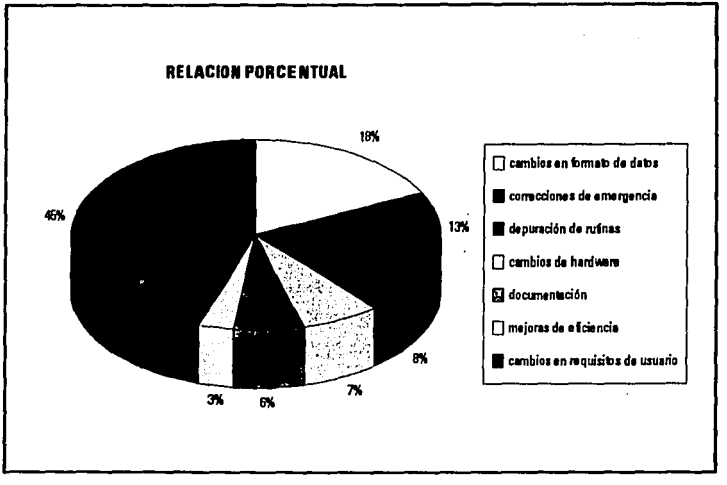


Fig. 1.1.3.1. Forma en que se distribuyen los gastos de mantenimiento de software.

La fase de mantenimiento es tan importante que qualquier método de diseño debe tener como objetivo principal producir sistemas que faciliten su propio mantenimiento. Pero, ¿cómo alcanzar este objetivo?. Los expertos recomiendan una serie de actividades y heurísticas que pueden servir como guía durante el desarrollo del sistema.

La reutilización de código es otro de los factores que no se toma en cuenta en los métodos de diseño tradicionales.

Cualquier programador que desarrolla un sistema nuevo debe escribir una buena cantidad de código que ha escrito anteriormente una y otra vez.

Las rutinas de búsqueda, ordenamiento, manejo de menús y despliegue de ventanas, entre otras, se repiten continuamente en proyectos diferentes, Los métodos de desarrollo de software deben alentar al programador a utilizar el código escrito previamente por él mismo o por otros programadores.

MODULARIDAD.

La programación orientada a objetos es un método de diseño que tiene como objetivo establecer técnicas de desarrollo de software para producir sistemas modulares.

La modularidad es una de las herramientas de diseño más poderosas para facilitar el desarrollo y mantenimiento de sistemas de software. La modularidad permite definir un sistema complejo en términos de unidades más pequeñas y manejables; cada una de esas unidades (ó modulos) se encarga de manejar un aspecto local de todo el sistema, interactuando con otros módulos para cumplir con el objetivo global.

La mayoría de los lenguajes actuales alientan el uso de la modularidad: en lenguajes estructurados como C ó Pascal, la modularidad se basa en el concepto de función (también llamada procedimiento o subrutina); en lenguajes más evolucionados, como Ada ó Modula-2, los modulos corresponden a conjuntos de funciones relacionados con las estructuras de datos que manipulan esas funciones (paquetes, en terminología de Ada).

Sin embargo, para ser realmente útil, las propiedades de un módulo deberían ser las siguientes:

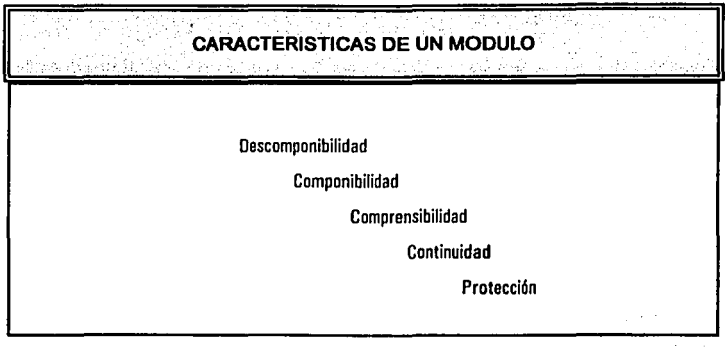


Fig. 1.1.3.2 Propiedades de la Modularidad.

a) **Descomponibilidad**: Un método de diseño modular debe permitir descomponer un problema de diseño en subproblemas más pequeños que pueden resolverse en forma independiente.

b) **Componibilidad**: Una vez que se cuenta con un conjunto de módulos que realizan una función especifica, se debe alentar al programador a usar esos módulos para construir nuevos programas.

c) **Comprensibilidad**: El lector de un programa o una librería debe ser capaz de entender el funcionamiento de cada módulo sin necesidad de consultar el texto de otros módulos.

d) **Continuidad**: Un cambio pequeño en las especificaciones de un programa debe causar cambios en un sólo módulo ó en un conjunto pequeño de ellos.

e) **Protección**: Un error de ejecución en el funcionameinto de un módulo no debe expandirse hacia los demás módulos.

Estas cinco propiedades pueden asociarse con las siguientes estrategias:

a) Un módulo debe corresponder con una unidad sintáctica del lenguaje (una subrutina, un paquete, una clase); esta unidad debe poder ser compilada por separado, tal vez para ser almacenada en una librería (con esto se mejoran la componibilidad y comprensibilidad del sistema).

b) Los módulos deben tener pocas interfaces (medios de comunicación con otros módulos), y éstas deben ser pequeñas. Un número pequeño de interfaces aumenta la independencia de los módulos, lo cual hace más fácil el proceso de componer nuevos sistemas a partir de módulos prefabricados, ayuda a evitar que los errores en un módulo se propaguen por todo el sistema y hace que cada módulo de un sistema sea más comprensible.

c) Cada módulo debe ocultar su implementación y algoritmos internos al resto del sistema (principio de ocultamiento de información). No se debe permitir que un módulo modifique los elementos internos de otros módulos; sino que la comunicación entre ellos debe realizarse mediante interfaces explícitas y bien definidas (esto favorece la comprensibilidad y la protección modular).

Un sistema orientado a objetos se puede entender fácilmente, por lo que su desarrollo, depuración y mantenimiento se facilitan en gran medida.

SOFTWARE ORIENTADO A OBJETOS.

Las técnicas en las que se basa la programación orientada a objetos (ocultamiento de información, abstracción de datos, manejo automático de memoria, polimorfismo) eran conocidas y utilizadas por los ingenieros de software desde hace muchos años; lenguajes como Ada ó Modula-2 alientan a los programadores a usar algunas de esas técnicas.

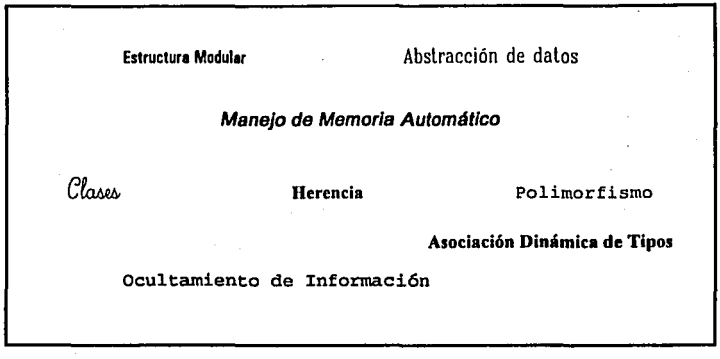


Fig. 1.1.3.3 Facilidades de la Programación Orientada a Objetos.

Entre los lenguajes orientados a objetos más populares, se encuentran Simula67, un lenguaje de simulación con facilidades para manipular eventos discretos; Smalltalk, que se ha usado principalmente para desarrollar interfaces de usuario gráficas y Eiffel, que se ha empleado en áreas diversas.

Los lenguajes orientados a objetos no habían tenido una aceptación amplia entre la comunidad de programadores.

Características como el manejo de memoria automático y la asociación dinámica de tipos imponen sobrecargas demasiado grandes a la ejecución de programas escritos en dichos lenguajes. Recientemente han surgido dos estrategias para disminuir esa sobrecarga: una de ellas consiste en utilizar lenguajes orientados a objetos para desarrollar los componentes de más alto nivel de un sistema y lenguajes funcionales para escribir las partes de bajo nivel críticas para la ejecución (una forma común de este tipo de combinaciones es utilizar Smalltalk y C).

La otra estratégia consiste en desarrollar nuevos lenguajes que nos proporcionen todas las facilidades de la programación orientada a objetos, pero que no impongan sobrecargas de ejecución demasiado altas.

Naturalmente, la programación orientada a objetos requiere el uso de lenguajes de programación idóneos. En la Programación Orientada a Objetos (POO), existen dos tipos diferentes de lenguajes: *puros e híbridos.*

Lenguajes puros.- Son aquellos diseñados en base a las propiedades específicas de la programación orientada a objetos, y tienen unas construcciones léxicas y lógicas muy diferentes a las encontradas en los lenguajes estructurados tradicionales. Su primer representante fue Simula, un lenguaje de simulación basado en Algol; hoy día, sin embargo, su representante más genuino es Smalltalk, y la versión Smalltalk/V la más difundida. Es el lenguaje que, al decir de los especialistas en POO, debería ser elegido como primer lenguaje al comenzar a estudiar POO.

Lenguajes híbridos.- Son lenguajes que añaden, a las propiedades tradicionales estructuradas, características orientadas a objetos. Hoy día son los más difundidos y, entre ellos, podemos citar: C++, Turbo Pascal (5.5, 6.0 y 7.0), Objective Pascal y Objective-C.

El debate sobre cuál lenguaje utilizar es abierto y polémico. Los puristas sugieren lenguajes puros que fuerzan al programador a utilizar técnicas orientadas a objetos; los lenguajes híbridos permiten que otros paradigmas, como la programación estructurada, puedan ser utilizados. La decisión, muchas veces, dependerá de un conjuntos de cirscunstancias impredecibles en el principio del diseño. Hoy día, los lenguajes híbridos han proliferado, debido a la gran comunidad de programadores en Pascal y C que, lógicamente tienen, a priori, asegurada una migración más fácil a Turbo Pascal 5.5, 6.0 ó 7.0 y a C++.

SMALLTALK

Es el primer lenguaje orientado a objetos. Descendiente de Simula-67, fue desarrollado en Xerox's Palo Alto Research Center (PARC). Smalltalk trata a todo como un objeto, y representa el paradigma de la programación orientada a objetos: es el lenguaje puro POO, por excelencia.

El entorno Smalltalk es orientado a objetos con ventanas, menús desplegables y emergentes, tratamiento de menús, etc. Presenta las características fundamentales de POO, y cada objeto en el sistema Smalltalk hereda del objeto raíz. Al igual que otros POO, Smalltalk diferencia entre objetos y clases.

Un objeto es una instancia o ejemplo de una clase. Todas las variables instancia de una clase Smalltalk son, por defecto, privadas a ese objeto.

Smalltalk es un lenguaje interpretado que lo hace muy flexible; pero lógicamente, limita sus posibilidades comerciales. Un programa Smalltalk debe ejecutarse desde su mismo entorno, y eso limita su transportabilidad. Esto no siempre es una desventaja. El entorno está disponible en muchas plataformas; entre ellas, DOS, OS/2 y Windows.

TURBO/QUICK/OBJECTIVE PASCAL

En 1989, Borland International y Microsoft, simultáneamente, lanzarón versiones orientadas o objetos de Pascal, Borland utilizó la filosofia C++, y Microsoft la filosofia Smalltalk, en sus versiones. Sin embargo, Borland ha progresado mucho y su última versión 7.0 incluye además, una magnífica biblioteca de clases llamada Turbo Vision, mejora de las versiones anteriores 5.5 y 6.0; por el contrario, Microsoft sigue estancada en su versión primitiva.

Estos lenguajes soportan las características mínimas orientadas a objetos, como son la creación de clases, herencia (sólo simple) y polimorfismo. La versión de Turbo Pascal es más robusta que Quick Pascal, aunque ésta es más fácil de aprender.

C++

El lenguaje de programación fué diseñado e implementado por Bjarne Stroustrup de los laboratorios Bell de la AT&T como sucesor de C. Tomando ideas de los lenguajes de programación Simula 67 y Algol 68, C++ conserva la compatibilidad con programas existentes en C y la eficiencia de C. La figura 1.1.3.4 ilustra la herencia de C++.

C++ además adiciona nuevas capacidades poderosas, haciendolo compatible para un amplio rango de aplicaciones incluyendo inteligencia artificial. C++ se tiene que tomar seriamente para el desarrollo de software porque está intimamente relacionado con C, y por el potencial que tiene para graficas e interfaces, para programación de sistemas, y para el soporte a gran escala en el desarrollo de software.

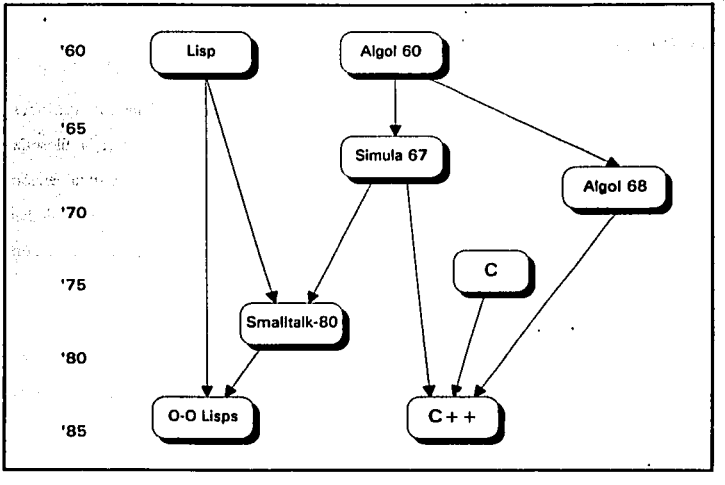


Fig. 1.1.3.4. Lenguajes que contribuyerón en la formación de C++.

El lenguaje de programación C++ permite además la abstracción de datos, en la tabla 1.1.3.1 se muestra una relación de las características de lenguajes de programación.

En ella se presenta solo Smalltalk-80 and C++ como los lenguajes en los cuales se puede realizar abstracción de datos y programación orientada a objetos.

C++ es seguramente el lenguaje híbrido más popular, existen versiones para todas las plataformas: DOS, OS/2, Windows, UNIX, XENIX, etc., y muchos distribuidores comercializan productos y compiladores C++, como es el caso de AT&T, Borland, Microsoft, Zorterch, etc.

Lenguaje POO	Programación Estructurada	Programación Modular	Abstracción de Datos
C	Si	Sí	No
Pascal	Sí	No	No
Modula-2	Sí	Sí	No
Ada	Sí	Sí	Sí
Smalltalk-80	No	No	Sí
C++	Si	No	Si

Tabla 1.1.3.1. C++ comparado sus características principales con otros lenguajes.

El lenguaje C++ se utiliza actualmente en el desarrollo de manejadores de bases de datos, sistemas operativos, compiladores, sistemas de comunicación, productos de CASE, redes y robótica.

Su gran difusión actual y la que se prevé en la década de los noventa se deberá esencialmente al proceso de estandarización a que se encuentra sometido, la versión 2.0 ya adoptada por el comité ANSI (Instituto Americano de Normas Nacionales) respectivo, es seguida, prácticamente, por todos los fabricantes. La versión 2.1 está en proceso de estandarización, y ya existen también actualizaciones de diversos fabricantes.

TURBO C++ y BORLAND C++

El compilador Turbo C++ es una implementación completa de AT&T C++ versión 2.0. Con Turbo C++, se obtienen las mismas características de C, así como de C++. Su característica fundamental es el entorno integrado de desarrollo EID (Integrated Development Environment, IDE). En la actualidad, las versiones más populares son 1.0 y 2nd edición, aunque recientemente ha aparecido la nueva versión 3.0, que corre en entornos DOS y Windows.

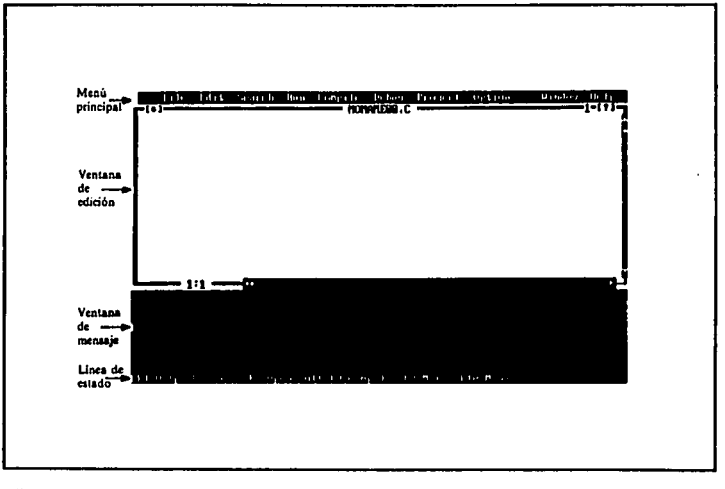


Fig.1.1.3.5. Entorno integrado Turbo C++.

Borland C++ es un paquete profesional. Añade las características básicas de los compiladores C y C++, un depurador incorporado, un profiler y un ensamblador.

Otras características sobresalientes son: soporte Windows 3.0/3.1 y módulos DLL/Dynamic Link Library (Bilblioteca dinámica de enlaces). Además Borland C++ contiene el Whitewater Group's Resource Toolkit, que permite crear, mantener, programar recursos Windows (iconos, menús, cuadros de diálogo, etc.). A finales de 1991, Borland ha presentado la versión 3.0, que soporta las versiones 2.0 y 2.1 de AT&T C++ y, en la primavera de 1992, la versión 3.1, que además soporta Windows 3.1.

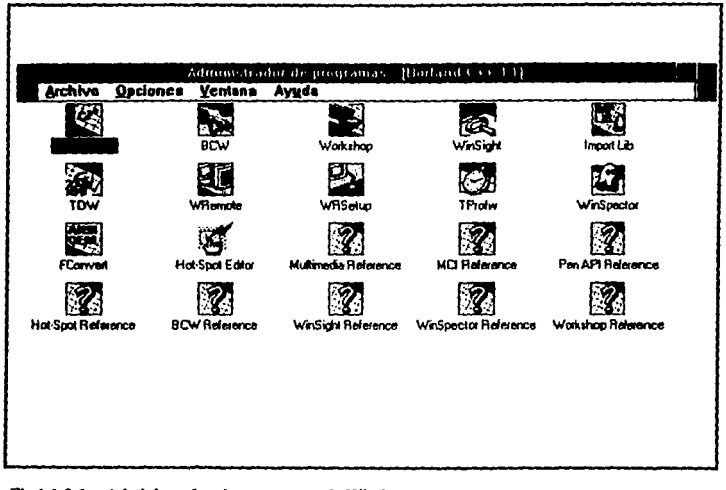


Fig.1.1.3.6. Administrador de programas de Windows.

Actualmente el compilador C++ 3.0 de Borland se nos presenta como un entorno integrado de programación para Windows que incluye todas las herramientas tradicionales para el desarrollo de una aplicación en C. Además, dispone de unas utilerías (denominadas "expertos") que facilitan de forma muy particular el desarrollo de aplicaciones bajo Windows. Por añadidura, el paquete de Borland dispone del gestor de proyectos Proyect Manager, que nos permite mantener desde una ventana el control del desarrollo de toda la aplicación.

ZORTERCH C++

Zorterch C++ es un compilador, desarrollado por la casa Zorterch, que combina las propiedades del compilador estándar C++ y un entorno integrado. Hasta la aparición de los compiladores de Borland, ha sido la estrella de la programación orientada a objetos. En la actualidad, se comercializa la versión 3.0, que soporta los entornos DOS, Windows y OS/2. Presenta dos nuevas opciones: una como entorno de desarrollo profesional y, otra, para cálculos científico-técnicos, lo que aumenta su potencia y versatilidad comparada con otros compiladores C++.

SYMANTEC C++

El compilador Symantec C++ 6.0 es la que llamaríamos, en inglés, un *outsider*, es decir, un aspirante inesperado que intenta hacerse sitio entre los dos gigantes de la programación visual en C.

Symantec C++ 6.9 es heredero directo del Zorterch. Lo primero que ha innovado es el diseño de su propio entorno integrado. De IDE ha cambiado a IDDE (*Integrated Develpment and Debugging Environment*, entorno integrado de desarrollo y depuración).

MICROSOFT C/C++ 7.0

El programa Microsoft C/C++ 7.0 (MSC) contiene un compilador C que sigue la norma ANSI. MSC necesita una PC basada en procesador 386 ó 386 con un mínimo de 3MB de RAM, un disco duro de al menos 10 MB libres, versión MS-DOS 3.3 o superior y monitores CGA, EGA, VGA o las nuevas SuperVGA (SVGA). Si se desea instalar todo el compilador C, más el kit de desarrollo SDK de Windows 3.1, se requiere entonces 55MB de disco duro.

Una de las propiedades más notables de MSC 7.0 es la biblioteca de clases MFC (Microsoft Foundation Classes) que ofrece soporte completo para objetos Windows tales como ventanas, diálogos, mensajes, fuentes, mapas de bits, etc. Esta biblioteca es una colección de más de 60 clases reutilizables C++ y que son compatibles totalmente con Windows 3.0 y 3.1. Una característica muy notable de esta biblioteca es su trasportabilidad, que no exige más que una recompilación de aplicaciones a otros entornos. Microsoft ha pensado en el futuro y la biblioteca será compatible con el próximo sistema operativo Windows de 32 bits, el NT.

MSC 7.0 incorpora un entorno integrado PWB (programmer's WorkBench) bajo DOS, como se muestra en la Figura 1.1.3.7.

Para programadores DOS ofrece, además de las herramientas citadas, una serie de propiedades notables, así como otras herramientas tales como: cabeceras precompiladas, alineación de código de máquina (posibilidad de insertar código ensamblado sin necesidad del Macro Assembler), manejador de memoria virtual, manejador de recubrimientos Overlays y una biblioteca QuickWin que permiten convertir una aplicaión MS-DOS en una

simple aplicación para Windows (utilizándolas se pueden transportar muchos programas compilados con Microsoft C/C++ para ejecutarse en una ventana sólo de texto de Windows).

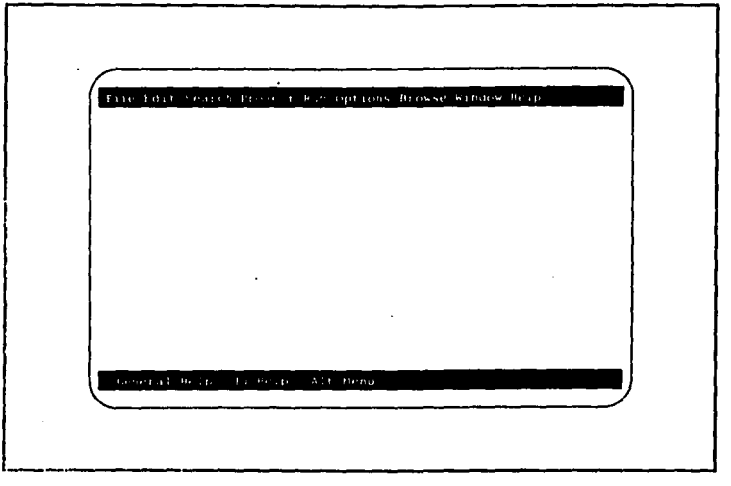


Fig. 1.1.3.7. Menú principal de PWB

MICROSOFT VISUAL C++

El Microsoft Visual C++ es uno de los dos paquetes de programación más potentes del mercado, en donde se ha conseguido un compilador capaz de realizar aplicaciones muy sofisticadas con cierta facilidad, gracias a la última versión de las librerías de clases Microsoft, las MFC (Microsoft Foundation Classes).

1.2 El lenguaje C ++ y la Programación Orientada a Objetos

1.2.1 EL LENGUAJE Y SU HISTORIA

El lenguaje C nació en los laboratorios de la Bell Telephone y ha sido estrechamente asociado con el sistema operativo UNIX, ya que su desarrollo se realizó en este sistema y debido a que tanto UNIX como el propio compilador de C y la casi totalidad de los programas y herramientas de UNIX, fueron escritos en C. Su eficiencia y claridad han hecho que el lenguaje ensamblador apenas haya sido utilizado en UNIX.

Este lenguaje esta inspirado en el lenguaje B escrito por Ken Thompson en 1970 con intención de recodificar el UNIX, que en la fase de arranque estaba escrito en ensamblador, en vistas a su transportabilidad a otras máquinas. B era un lenguaje evolucionado e independiente de la máquina, inspirado en el lenguaje BCPL concebido por Martin Richard en 1967.

En 1972, Denis Ritchie, toma el relevo y modifica el lenguaje B, creando el lenguaje C y reescribiendo el UNIX en dicho lenguaje. La novedad que proporcionó el lenguaje C sobre B fue el diseño de tipos y estructuras de datos.

Este lenguaje ha evolucionado paralelamente a UNIX, que a su vez ha pasado por diversas versiones entre las que destaca XENIX. Esta versión, fue desarrollada por Microsoft para las micro computadoras de 16 bits.

El lenguaje C ha seguido evolucionando sin perder sus características originales dando origen al lenguaje C++.

C++ esta basado en C y es un superconjunto de C. C++ retiene el poder y la flexibilidad de C para permitir la interfaz hardware-software con la programación a bajo nivel.

El poder de eficiencia y economía de las expresiones de C están contenidas en C++, sin embargo, C++ provee una plataforma que soporta la programación orientada a Objetos y los problemas de abstracción de alto nivel.

C++ es un lenguaje híbrido. El paradigma de la programación orientada a Objetos puede ser explotado totalmente para producir una solución orientada a objetos a un problema, o puede ser tratado como un lenguaje procedural que contiene algunas construcciones adicionales. Esta dualidad en C++, representa un cambio significativo para un programador principiante. Ya que no solamente esta el aprender un nuevo lenguaje sino el aprender una nueva forma de pensar y de resolver problemas.

C++ deriva de la inspiración de BCPL y Simula67. La noción de subclases y las funciones virtuales están tomadas de Simula. La habilidad para sobrecargar operadores y la flexibilidad para incluir declaraciones cerradas a su primer punto de aplicación son reminiscencias de Algol68. Es claro que C++, como otros modernos lenguajes de

programación, representa una evolución y refinamiento de algunas de las mejores características de los lenguajes previos.

C++ fue desarrollado por Bjarne Stroustrup en los laboratorios Bell a principios de los años 80's. El Dr. Stroustrup desarrollo C++ para mantener la escritura de algunas simulaciones complejas cuyas consideraciones de eficiencia preincluyen el uso de Simula67.

En julio de 1983 C++ fue primeramente instalado por el grupo de Stroustrup quien desarrollo el lenguaje. En el verano de 1987, C++ todavia seguia evolucionando. No hubo un lenguaje estándar o formal. El Dr. Stroustrup y sus colegas en los laboratorios AT & T[1] continuaron haciendo refinamientos al lenguaje de acuerdo a su experiencia con él.

Una convención a lo que han llegado los desarrolladores del lenguaje es hacer compatible a C. Esto preserva la integridad de millones de lineas de código de C, manteniendo la integridad de las bibliotecas de C y las herramientas que se han desarrollado con él, lo que trae como beneficio que haya incentivos para un programador de C, el tener que aprender C++, sin tener que sacrificar o perder lo que ha realizado hasta ese momento.

Para muchos desarrolladores de software el lenguaje es como una religión así que muchos programadores de C, no tuvieron ninguna dificultad para incorporar el C++ a sus actividades.

[1] (American Telephone & Telegraph Company) Compañía Americana de Teléfonos y Telégrafos.

El lenguaje C es sencillo, compacto, elegante y veloz por lo que resulta de gran utilidad para aquellos programadores que pretendan un software de altas miras y al mismo tiempo efectivo.

Podemos afirmar, sin lugar a dudas, que el lenguaje C, a través de su corta existencia, ha demostrado ser uno de los más versátiles, completos y estructurados, que posee como gran ventaja a destacar, la transportabilidad, es decir, la posibilidad de utilizarlo tanto en macrocomputadoras como en mini y micro computadoras.

Además, como muestra de su gran aceptación, son cada vez más numerosos los programas escritos en lenguaje C.

Se puede decir, que todo programador que se precie como tal, debe conocer a fondo el lenguaje C, porque se ha convertido en el más universal de los lenguajes y el que menos reformas sufra en un futuro próximo.

LA EVOLUCION DE LOS LENGUAJES DE PROGRAMACION ORIENTADA A OBJETOS.

Los ensambladores fueron los primeros lenguajes de programación que introdujeron representaciones simbólicas para las instrucciones de maquina. Algunos de los primeros ensambladores fueron Soap para la IBM 650 (a mediados de los años 50's) y Sap para la IBM 704 (después de los años 50's) pero el primer lenguaje de programación de alto nivel fue sin duda FORTRAN.

FORTRAN introdujo varios conceptos importantes a los lenguajes de programación, incluyendo variables, arreglos, estructuras de control (tales como las estructuras condicionales y las iterativas). FORTRAN es todavía uno de los más populares lenguajes de programación. El lenguaje tiene una activa estandarización ANSI que periódicamente realiza cambios substanciales y extensiones al lenguaje.

Todavía, después de los años 50's se encontraron con un problema cuando se desarrollaban programas grandes, que era que los nombres de las variables entraban en conflicto en diferentes partes del programa.

Los lenguajes de programación que sucedieron al FORTRAN fueron PL/1, COBOL y Algol. Los diseñadores del lenguaje Algol decidieron proveer barreras para separar nombres de variables de los segmentos del programa. Con esto nació el BEGIN y el END en el Algol 60 (Randell y Rusell, 1964).

Así sólo cuando los nombres de las variables aparecen en un bloque son conocidas unicamente en ese bloque. Este fue el primer intento de proteger o encapsular bloques en un

lenguaje de programación. Las estructuras de bloques son ahora extensamente usados en una amplia variedad de lenguajes.

A principios de los años 60's, los diseñadores del lenguaje Simula-67 (Dahl y Nygaard, 1966; dahl, Myhrhaug y Nygaard, 1970) tomaron el concepto de bloque de Algol para introducir más tarde el concepto de Objeto. Aunque las raíces de Simula fueron en Algol, Simula, fue principalmente conocido como un lenguaje de simulación. Así, los objetos de Simula tuvieron una existencia propia conteniendo datos y el poder comunicarse con otros durante una simulación.

Conceptualmente un objeto contiene ambos, los datos y las operaciones que manipulan sus datos. Las operaciones fueron llamados métodos. Simula también incorporó la noción de clases, las cuales son usadas para describir la estructura y el comportamiento de un conjunto de objetos.

La herencia de clase fue también soportada por simula. La herencia organiza las clases dentro de jerarquías, permitiendo parte de la implementación de las estructuras. Simula también distingue entre dos tipos de igualdad, idénticamente y superficial que refleja la distinción entre referente basado en valor y basados en la interpretación de objetos.

Otro importante desarrollo dentro de los lenguajes de programación fue el lenguaje LISP (McCarthy et al.,1965). Lisp fue introducido a finales de los años 50's y a principios de los años 60's, LISP fue uno de los más elegantes y refinados lenguajes utilizando pocas y simples construcciones de programación (listas y funciones) para ejecutar procesos complejos. LISP es uno de los principales lenguajes que por su refinamiento se utilizan en las aplicaciones de la Inteligencia Artificial.

lenguaje de programación. Las estructuras de bloques son ahora extensamente usados en una amplia variedad de lenguajes.

A principios de los años 60's, los diseñadores del lenguaje Simula-67 (Dahl y Nygaard, 1966; dahl, Myhrhaug y Nygaard, 1970) tomaron el concepto de bloque de Algol para introducir más tarde el concepto de Objeto. Aunque las raíces de Simula fueron en Algol, Simula, fue principalmente conocido como un lenguaje de simulación. Así, los objetos de Simula tuvieron una existencia propia conteniendo datos y el poder comunicarse con otros durante una simulación.

Conceptualmente un objeto contiene ambos, los datos y las operaciones que manipulan sus datos. Las operaciones fueron llamados métodos. Simula también incorporó la noción de clases, las cuales son usadas para describir la estructura y el comportamiento de un conjunto de objetos.

La herencia de clase fue también soportada por simula. La herencia organiza las clases dentro de jerarquías, permitiendo parte de la implementación de las estructuras. Simula también distingue entre dos tipos de igualdad, idénticamente y superficial que refleja la distinción entre referente basado en valor y basados en la interpretación de objetos.

Otro importante desarrollo dentro de los lenguajes de programación fue el lenguaje LISP (McCarthy et al.,1965). Lisp fue introducido a finales de los años 50's y a principios de los años 60's, LISP fue uno de los más elegantes y refinados lenguajes utilizando pocas y simples construcciones de programación (listas y funciones) para ejecutar procesos complejos. LISP es uno de los principales lenguajes que por su refinamiento se utilizan en las aplicaciones de la Inteligencia Artificial.

A principios de los años 70's, el concepto de abstracción de datos fue propuesto por un número de diseñadores de lenguajes con el propósito de manejar extensos programas (Parnas,1972).

Existen dos aspectos fundamentales de la abstracción de datos. Uno es agrupar las estructuras de un tipo con las operaciones definidas por su tipo. Por ejemplo, con Algol o Pascal, en donde el lenguaje no agrupa todas las operaciones en un tipo de registro del mismo modulo con la definición del tipo de registro.

El otro aspecto de la abstracción de datos es la ocultación de información, donde los detalles de implementación y representación de los objetos son escondidos y no pueden ser accesados directamente hacia los usuarios del objeto.

Los lenguajes tales como Alphard (Wulf London y Shaw, 1976) y CLU (Liskov et al,. 1977) introdujeron la abstracción de datos. En CLU, por ejemplo, los tipos de abstracción de datos fueron implementados en agrupaciones con un nombre apropiado.

Uno de los más importantes lenguajes que manejaron la abstracción de datos fue el ADA (Booch, 1986). El departamento de la defensa de los E.U. comisionaron el diseño de ADA para reducir y controlar el costo de desarrollo de software. El departamento de la defensa intento que se produjeran en ADA los nuevos sistemas que ellos manejaran.

El lenguaje contenía las usuales estructuras de control, y las definiciones de tipo, funciones y subrutinas. Las construcciones orientadas a objetos de ADA incluían diseño de paquetes.

LOS LENGUAJES ORIENTADOS A OBJETOS EN LOS 90'S

En 1982 se dijo "que la programación orientada a objetos sería para los años 80's lo, aque la programación estructurada fue para los años 70's" (Rentsch, 1982). Los 80's seran probablemente conocidos como la década que lanzó la orientación de objetos en la era de la computación.

En el año de 1986 se dio la primera conferencia, OOPSLA (Lenguajes y Sistemas de programación orientada a objetos), completamente dedicada a la orientación de objetos, y en 1988 salió la primera publicación completamente dedicada a la orientación de objetos: The Journal of Object-Oriented Programming).

Después de este evento hubo un notable incremento de las conferencias sobre Programación orientada a objetos. Simultáneamente, surgieron los lenguajes orientados a objetos tales como C++, Objective-C, Smaltalk-80, Eiffel y extensiones de LISP orientadas a objetos que llegaron a estar disponibles comercialmente. Varias de las mejores compañías de computadoras tales como AT & T, SUN, y Microsoft comenzaron a implementar estilos de programación orientados a objetos en el desarrollo de su propio software.

Durante los años 70's y los años 80's, los conceptos orientados a objetos de Simula y otros de los primeros prototipos estuvieron influenciados por los lenguajes: Smaltalk que inicialmente fué un proyecto de Xerox Palo Alto Research park (PARC).

Durante los años 70's un grupo de desarrolladores de Xe.ox PARC estuvieron revolucionando el futuro de la industria de la computación. Los desarrolladores PARC crearon tecnologías actualmente conocidas como tecnologías orientadas a objetos. El lenguaje Smaltalk fue desarrollado simultáneamente en este tiempo. En términos de la

interfaz de usuario surgieron las estaciones de trabajo y sus predecedores fueron influenciados por el diseño de los ambientes de la Apple Macintosh, Aldus Page Maker y Microsoft windows.

En la década de los años 90's, los lenguajes, las técnicas, bases de datos e interfaces de usuarios orientadas a objetos van a ser cada vez más populares. La mayoría del desarrollo de software se va a ver afectado por la orientación de objetos de una u otra forma. Los años 90's van a ser conocidos como los años de la proliferación de los lenguajes y tecnologías orientadas a objetos.

¿Por Qué Programación Orientada A Objetos?

Hace varios años, investigadores de la ciencia de la computación observaron que los programadores podían escribir y depurar más o menos la misma cantidad de código sin importar el lenguaje que utilizaran. La cantidad de trabajo es más o menos la misma, pero no así los resultados. Escribir 100 líneas de código en lenguaje ensamblador, se obtiene cierto resultado pero con el código en C se obtiene mucho más. Con esto en mente, los investigadores buscaron desarrollar lenguajes de alto nivel que multiplicaran el poder de un programador, reduciendo así el tiempo y los costos de desarrollo de los proyectos.

En la década de 1970 se volvió popular el concepto de Objeto entre los investigadores de los lenguajes de programación. Un objeto es un conjunto de códigos, datos diseñados para emular o imitar una entidad física o abstracta. Los objetos son eficientes como elementos de programación por dos razones principales: representan una abstracción directa de elementos que se utilizan comúnmente y ocultan la mayor parte de la complejidad de su implantación a los usuarios.

Los primeros objetos que se desarrollaron fueron aquellos que estaban más íntimamente ligados a las computadoras, como integer, Array y stack. Se diseñaron algunos lenguajes (como Smalltalk) como lenguajes ortodoxos en los cuales todo se define como un objeto.

La programación orientada a objetos es una metodología que da gran importancia a las relaciones entre objetos, más que a los detalles de la implantación. Las relaciones son nexos entre objetos y suelen desarrollarse a través de árboles genealógicos en los que se crean nuevos tipos de objetos a partir de otros.

Ocultar la implantación de un objeto hace que el usuario tenga que ver más con la relación que guarda un objeto con el resto del sistema y no con la forma en que se implanta el comportamiento de un objeto. Esta distinción es importante y representa una desviación fundamental de los lenguajes "imperativos" (como C), en los que las funciones y las llamadas a funciones son el centro de la actividad.

En C++, pocos objetos son parte del lenguaje mismo. El trabajo pesado y la responsabilidad de diseñar objetos corresponden al usuario. El poder de la programación Orientada a Objetos (POO) se aprovecha si se desarrollan grupos de objetos que suelen denominarse Jerarquías de clases. El desarrollo de estas jerarquías de clases es una actividad fundamental en POO.

LA EVOLUCION DE LA PROGRAMACIÓN ORIENTADA A OBJETOS EN EL USO DE INTERFACES

En los últimos años la comunicación de los usuarios con las computadoras se ha llevado a cabo a través de interfaces basadas en comandos.

En los primeros sistemas computacionales los usuarios tenían que memorizar comandos o en su defecto consultar grandes manuales para poder comunicarse con las máquinas. Las primeras microcomputadoras y mainframes, requerían de un gran conjunto de instrucciones para poder funcionar.

Los primeros usuarios de computadoras fueron ingenieros, ya que ten'ian mucho interes en conocer más acerca de su funcionamiento y de su tecnología. El uso de las interfaces basadas en comandos fue aceptada por la mayor parte de estos usuarios ya que ellos estaban acostumbrados a utilizar un lenguaje nemotecnico para poder comunicarse con las computadoras.

En los años 70's fue entonces cuando las computadoras empezaron a ser usadas por un nuevo grupo de usuarios tales como las secretarias, directores, ejecutivos y personas en general. Los comandos basados en éstas interfaces causaron fobia a los usuarios hacia las computadoras; debido a que tenían que memorizar comandos tales como "Control-Shift-D" para borrar una palabra o "Control-Escape-D" para restaurar una palabra.

Para hacer la vida más fácil al usuario, una gran cantidad de componentes han sido inventados para comunicarse con la maquina. Al principio los unicos perifericos que existian, eran el teclado y la terminal de video, pero no fue hasta finales de los años 70's que surgió al

mercado comercial un nuevo dispositivo conocido como MOUSE o "Ratón" éste cambió radicalmente el uso de las interfaces.

El mouse ya existia antes de los años años 70's, pero sólo para un selecto grupo de usuarios. En los años 60's Projects y SRI Intenational, MIT y otras universidades permitieron la invención de componentes con función de apuntadores y sistemas con ambiente gráfico (Perry y Voelcker, 1989). El mouse y el joystick son algunos de los pocos componentes apuntadores que fueron inventados en ese tiempo.

En los años 70's los diseñadores de Xerox PARC estuvieron ocupados diseñando poderosas estaciones de trabajo con ambientes gráficos. Sus experimentos se concentraron en combinar la memoria asociativa del usuario con su capacidad de manipulación directa de objetos. El objetivo básico de esas nuevas estaciones de trabajo, era que un usuario pudiera tener una poderosa computadora de escritorio totalmente dedicada al trabajo del usuario.

El concepto de utilizar la memoria asociativa del usuario es el de hacer uso de menus y cajas de dialogo que interactuen con él , en vez de tener que memorizar comandos para cada tarea, el usuario selecciona un comando de una barra de menus que a su vez este contenga un submenu con una lista de comandos disponibles. Por ejemplo la figura 1.2.1.1 despliega la barra de menus de la aplicación Write de microsoft.

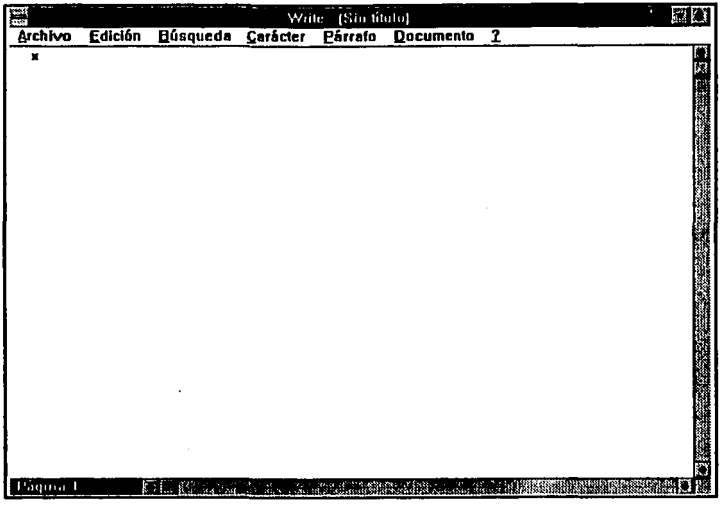


Fig 1.2.1.1 Barra de menu de Microsoft Write.

Esta barra de menus despliega una lista de comandos disponibles tales como FILE, EDIT y SEARCH. Cuando el mouse apunta a cualquiera de esos comandos una determinada acción es ejecutada y aparece un submenu con otra serie de comandos. La figura 1.2.1.2 muestra el submenu del comando **caracter** apuntado. El usuario puede entonces seleccionar diferentes estilos de los caracteres disponibles.

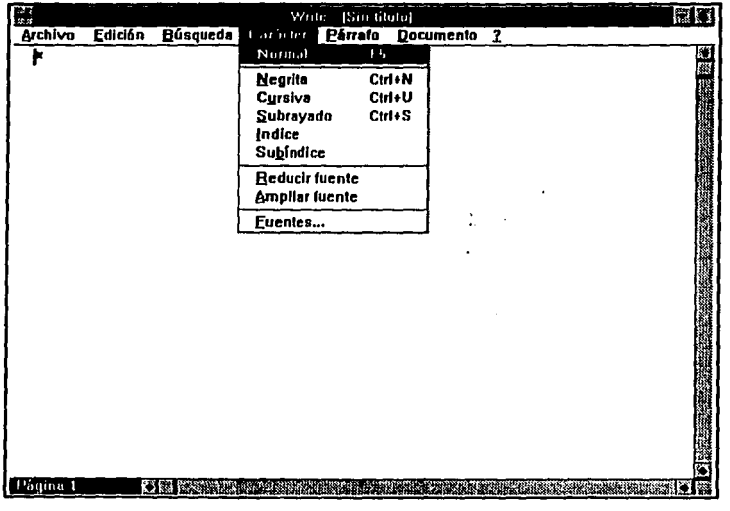


Fig. 1.2.1.2 Tipico submenu desplegable de Microsoft Write.

Las cajas de dialogo permiten la interacción entre el usuario y la computadora, las cajas de dialogo emplean una larga colección de instrucciones para el control de objetos tales como botones, barras de scroll y cajas de edición. Por ejemplo la figura 1.2.1.3 muestra una caja de dialogo que se utiliza para abrir un archivo. Esta caja de dialogo esta compuesta de dos botones llamados OPEN y CLOSE, una caja de edición que permite introducir el nombre del archivo, y una región de scroll que permite navegar a través de la lista de archivos y directorios disponibles en el disco. Apuntando sobre el botón de OPEN se permite editar el archivo.

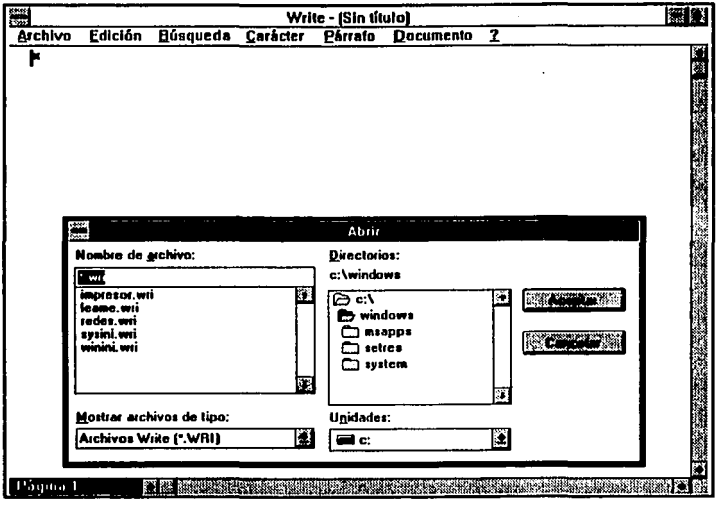


Fig. 1.2.1.3 Caja de dialogo de Microsoft.

En las interfaces gráficas, los datos textuales no son la unica forma de interacción. Los iconos representan conceptos tales como archivos de folders, cestos de basura e impresoras. Los iconos simbolizan palabras y conceptos comunmente aplicables en difererentes situaciones. La figura 1.2.1.4 muestra la utilidad Microsoft Paint con una barra compuesta de iconos. Cada uno de esos iconos representan un cierto tipo de estilo de dibujo. Apuntando sobre el icono del lápiz, por ejemplo, el cursor se comporta como la punta de un lapiz para dibujar lineas.

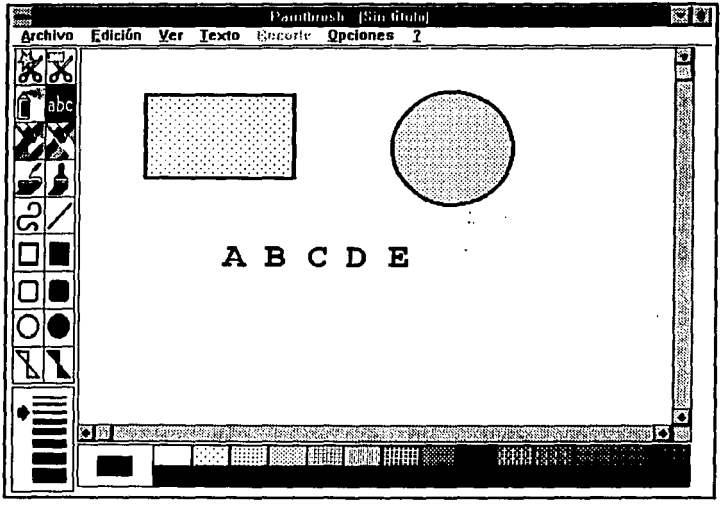


Fig. 1.2.1.4 Aplicación Paint de Microsoft.

EL USO DE LAS INTERFACES EN LOS 90´S

En los años 90´s los sistemas computacionales incluirán procesadores más poderosos, monitores de mayor resolución y dispositivos con cada vez más memoria. Gran parte de la maquinaria del futuro utilizarán computadoras más esteticas. Las interfaces de computadoras serán más intuitivas, manejaran gran diversidad de objetos y estarán provistas de ambientes visuales.

Las interfaces jugaran un papel importante en la industria. La manipulación directa de objetos será ampliamente utilizada y las computadoras serán accesibles al

público en general. En el futuro, ésta manipulación tan directa permitirá a los diseñadores utilizar interfaces seleccionando objetos directamente de la pantalla. Herramientas tales como NextStep serán comunmente utilizadas para el desarrollo de interfaces sofisticadas. Además el usuario podr'a construir aplicaciones gráficas conectando componentes parametrizados preconstruidos.

Tecnología tales como Digital Video Interactive (DVI), la de los discos opticos y procesadores de señal digital permitirán al usuario interactuar con aplicaciones de multimedia. La orientación de objetos será ampliamente explotada para integrar diversos conjuntos de objetos no importando si son éstos de texto, voz o video. Hypercard de apple y otras herramientas de hypermedia serán utilizadas con más frecuencia interconectando tales objetos.

La programación visual ayudará tanto a los programadores como a los no programadores para desarrollar diversas aplicaciones. Los lenguajes de programación visual permitirán a los programadores crear tipos de objetos y desarrollar rápidamente los flujos de control de tales aplicaciones. Visual SQL y QBE permitirán la realización de reportes o la extracción de información de bases de datos instaladas en mini computadoras y mainframes fácilmente sin tener que ser un experto en computadoras.

1.2.2 VARIACIONES ENTRE C y C ++.

Se debe pensar en C++ como un subconjunto de C. La parte más importante es que cada característica que se conozca de C estará disponible en C++. Al igual que C, los programas de C++ comienzan con una función main(.).

Todas las palabras reservadas de C y funciones de trabajo también las encontraremos en C++. Si siempre se han escrito programas en C que tomen las ventajas de los comandos de linea en los argumentos argc y argv, estaremos contentos de conocer que aún se encuentran en C++.

Con pocos cambios, en los programas de C, estos pueden correr en el ambiente de C++.

La siguiente tabla muestra algunos operadores de C y C++ los cuáles se utilizan en ambos casos y en otros tienen algunas diferencias que se describiran más adelante.

C	C++
-Funciones Prototipo.	-Funciones Prototipo.
Opcional	Requeridas
-Conversión de tipos automática.	-Conversion de tipos automática.
Igual	Cambios
-Alcances.	-Alcances.
Igual	Cambios
-Comentarios.	-Comentarios.
/* y */	//
	-Inicialización de argumentos por default.
-Variables por valor.	-Variables por referencia.
	-Sobrecargar el nombre de la función.
-Asignar y liberar la memoria.	-Asignar y liberar la memoria.
malloc( ) -> asignar	new -> asignar
free( ) -> liberar	delete -> liberar
	-El operador Scope (::).
	-Funciones en linea.

Tabla 1.2.2.1. Operadores de C y C++.

SE REQUIEREN FUNCIONES PROTOTIPO.

En C las funciones prototipo son opcionales. Si no hay conflictos entre un llamado a una función y la declaración de la misma función, el programa puede compilar.

En C++, una función prototipo es *requerida* por cada una de las funciones del programa. El programa de C++ no podra compilar a menos que siempre cada función prototipo este en su lugar. Como en C, se puede declarar una función sin teclear return. Si no se teclea return, la función asume que tiene un return interno.

En este tipo de operador se debe observar la importancia que tiene en C++, ya que se requiere siempre definir la función prototipo para cada una de las funciones de nuestros programas dentro del ambiente de C++. Las diferencias entre C y C++ son que para uno es opcional y para el otro se requiere respectivamente.

CONVERSION DE TIPOS AUTOMATICA.

C++ usa la mismas reglas de C para conversión de tipos automatica, pero con un pequeño cambio.

Aunque un apuntador void puede estar asignado al valor de otro tipo de apuntador sin teclearse en forma explicita, lo contrario no es verdad. Por ejemplo, aunque el siguiente codigo de compilador pertenezca a C, esto no va a compilar en C++.

```
void  *voidPtr;
void  *shortPtr;
voidPtr = shortPtr;     /*  <-- This line is just
                            fine... */
shortPtr = voidPtr;     /*  <-- This line is fine in C,
                            but WILL NOT compile in C++ */
shortPtr = (short *)voidPtr;  /*  <-- This works in
                                  C++ */
```

Observamos en este caso que en C++ dos apuntadores no pueden estar asignados a uno de tipo void. En cambio en C el apuntador void puede tener asignados dos apuntadores.

CAMBIOS EN LOS ALCANCES.

Existen algunas diferencias entre C y C++ envolviendo el **alcance**. El alcance de una variable define la disponibilidad de la variable durante el resto del programa. Por ejemplo, una variable global esta disponible durante todo el programa, mientras una variable local esta limitada al bloque en el cuál es declarada. La mayor parte de C++ sigue las mismas reglas de alcance de C. Sin embargo hay pocas diferencias entre ellos.

LA MARCA DE COMENTARIO //.

La marca de bloque de comentario de C, /* y */, funciona de igual forma en C++. Pero, además C++ soporta una sola marca de comentario. Cuando un compilador de C++ encuentra el caracter de //, esté ignora el resto del código de esta linea. Ejemplo:

```
void main( void )
{
     short numGuppies; // May increase suddenly !!
}
```

Como se esperaba, el carácter // es ignorado dentro de un bloque de comentario. En el siguiente ejemplo, // esta incluido como parte de un bloque de comentario.

```
void main( void )
{
     /*    Just a comment...
     //    */
}
```

Contrariamente, los caracteres de comentario /* y */ no tienen un sentido especial dentro de una linea de comentario. El inicio del bloque de comentario en el siguiente ejemplo contiene un comentario como este:

```
void main( void )
{
    // Don't start a /* comment block
       inside a single-line comment....
       This code WILL NOT compile!!! */
}
```

El compilador va a quejarse acerca de este ejemplo.

Se puede ver con estos ejemplos que en C se empieza un comentario con /* y se debe terminar con */; pero en C++ la linea de comentarios solo utiliza el operador // y este no termina con otro operador igual, es decir únicamente utiliza un sólo operador; la desventaja es que el compilador de C++ sólo permite una línea de comentario, si el comentario excede esta linea y ocupa más, este código no será compilado y tendrá un error.

MANEJANDO ENTRADAS Y SALIDAS.

En un programa estandar de C, la entrada y salida son normalmente manipuladas por las rutinas de la Libreria Estandar tales como `scanf( )` y `printf( )`. Se puede llamar a `scanf( )` y `printf( )` desde un programa de C++, esta es una alternativa elegante. El iostream facilita que el envio de una secuencia de variables y constantes a una salida normal, tal como lo hace `printf( )`. Tambien, iostream hace fácil convertir datos desde una entrada normal hacia una secuencia de variables tal como lo hace `scanf( )`.

`iostream` predefine tres entradas y salidas normales. `cin` es usada como entrada, `cout` para una salida normal, y `cerr` para salida de error. El operador << ("put to") es usado para enviar datos a un stream. El operador >> ("get.from") es usado para recibir datos desde un stream.

DEFINICION.

El operador << es también conocido como el **operador de inserción**, porque permite insertar datos dentro de un stream. El operador >> es tambien conocido como el **operador de extracción** porque permite extraer datos desde un stream.

Este es un ejemplo del operador << :

```
#include <iostream.h>

void main( void )
{
     cout << "Hello, world!";
}
```

Este programa envía el texto `"Hello, world!"` a la pantalla, tal como se usa `printf( )`. El archivo include `<iostream.h>` contiene todas las definiciones necesarias para usar iostream. Puesto que << es un operador binario, este requiere 2 operandos. En este caso, los operadores cout y la cadena `"Hello, world!"`. El destino de stream siempre aparece del lado izquierdo del operador <<.

Como con cualquier otro operador, se puede usar más de un << en una linea. Este es otro ejemplo:

```
#include <iostream.h>

void main( void )
{
     short i=20;
     cout << "The value of i is " << i;
}
```

Este programa produce la siguiente salida:

```
The value of i is 20
```

`iostream` conoce todo acerca de la construcción de este tipo de datos de C++. Esto significa que las cadenas de tipo texto se imprimen como cadenas de tipo texto, `shorts` como `shorts`, y `floats` como `floats`, completas con punto decimal. Estos no necesitan un formato especial.

En este caso se tienen para C las entradas con el operador `scanf( )` y las salidas con `printf( )`; En cambio para C++ se tienen 3 tipos de operadores: `cout` para salidas, `cin` para entradas y `cerr` para salidas de error; además se debe utilizar la librería iostream.h para que el compilador entienda el manejo de dichas entradas y salidas. Además C++ utiliza los operadores << de inserción y >> de extracción, para sus entradas y salidas. Estos operadores serían la diferencia entre C y C++; mientras que `printf` y `scanf` utilizan los parentesis ( ); `cout`, `cin` y `cerr` utilizan sólo << ó >>; los parentesis indican el

inicio y final de los operadores `printf` y `scanf`, mientras que los operadores de C++ sólo utilizan un operador de inserción ó de extracción para indicar el inicio y final de estos.

UN EJEMPLO DE SALIDA IOSTREAM.

Este es un ejemplo interesante de salida `iostream`. Comenzamos con un doble click en THIN C++ dentro de un archivo llamado THIN Project Manager. Vamos a encontrar THIN Project Manager dentro del folder de desarrollo THIN C++, en el folder de THIN C++. THIN C++ tiene un prompt para abrir un archivo de proyecto (Figura 1.2.2.1). Vamos dentro del folder llamado THIN C++ Projects (se tiene que mover un nivel hacia arriba para encontrar esté), entonces dentro del Subfolder llamado `Chap 04.01-cout`, y abriendo el archivo de nombre `cout`.

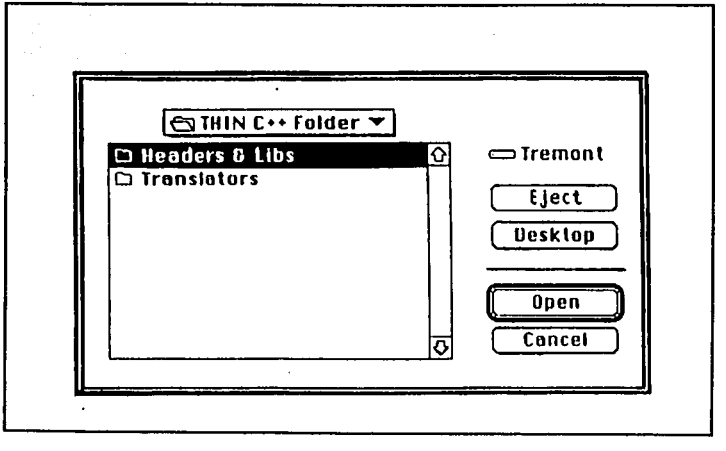


Fig. 1.2.2.1. Prompt THIN C++ para abrir un archivo de proyecto.

En este punto, aparecera una ventana de proyecto `count.`, como se muestra en la figura 1.2.2.2. Cada uno de los cuatro nombres en la ventana representan un archivo separado. `cout.cp` es el archivo que contiene el codigo fuente de C++. Los otros tres archivos son librerias que contienen varias rutinas de soporte. La libreria `iostreams` contiene todas las funciones de `iostream`, `CPlusLib` es un conjunto de rutinas de soporte para el compilador de C++, y ANSI++ es la libreria estandard de C con algunas modificaciones para soportar C++.

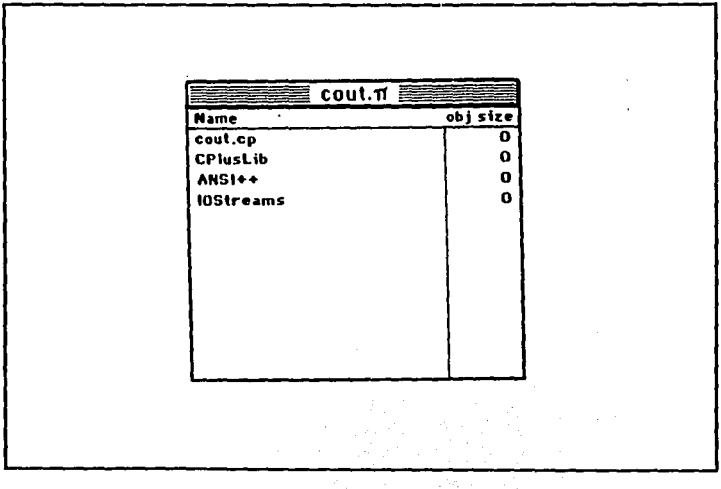


Fig. 1.2.2.2. Ventana de proyecto para cout.

Si tecleamos un doble-click sobre el nombre cout.cp en la ventana project, una nueva ventana va a aparecer, conteniendo el codigo fuente de cout, como sigue:

```
#include <iostream.h>
void main( void )
{
      char *name =  "Dr. Crusher";
      cout << "char:    " << name[ 0 ] << '\n'
      << "short:  " << (short) (name[ 0 ] ) << '\n'
      << "string: " << (unsigned long) name;
}
```

CORRIENDO COUT.

Seleccione RUN desde el menu Project. THIN C++ va a compilar su código fuente, entonces corre el programa compilado. Cuando la ventana de la pantalla aparezca, compare su salida con la mostrada en la figura 1.2.2.3. Para salir del programa, teclee return.

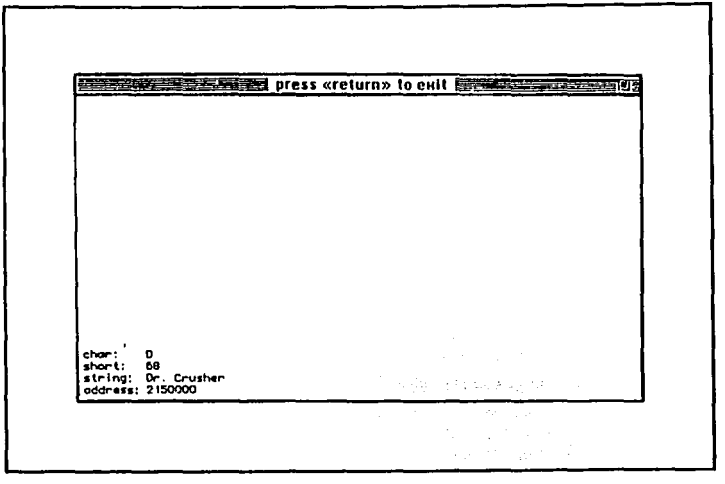


Fig. 1.2.2.3. Ventana de resultados.

LA CONTINUACION DEL CODIGO FUENTE.

El programa empieza inicializando el apuntador nombre de tipo char, apuntando a la cadena de texto "Dr. Crusher". Después viene una declaración grande caracterizada por once diferentes ocurrencias del operador <<. Esta declaración produce cuatro lineas de salida.

ESTA TESIS NO DEBE
SALIR DE LA BIBLIOTECA

El siguiente código arroja

```
cout << "char:    " << name[ 0 ] << '\n'
```

produce esta linea de salida:

```
char: D
```

Como se esperaba, imprimiendo name[0] produce el primer caracter del nombre, una D.

El siguiente código que se origina es

```
<< "short:  " << (short) (name[ 0 ] ) << '\n'
```

la linea de salida asociada con este código es la siguiente:

```
short: 68
```

Este resultado fué logrado para repartir el caracter 'D' a un short. En general, iostream despliega tipos (tal como short e int) como un entero. Como se espera, un float es desplegado en formato de punto flotante.

El siguiente código que arroja es

```
<< "string: " << name << '\n'
```

produce esta linea de salida:

```
string:     Dr. Crusher
```

Cuando el operador << encuentra un apuntador char, este asume que se requiere imprimir una cadena terminada en cero.

El código final en este ejemplo muestra otro camino para desplegar el contenido de un apuntador:

```
<< "address:  " << (unsigned long) name;
```

Otra vez, name se imprime, pero esta vez como un unsigned long. Este es el resultado.

```
address:    2150000
```

INICIALIZACION DE ARGUMENTOS POR DEFAULT.

C++ permite asignar valores por default (Conocido como Inicialización de argumentos por default) a argumentos de funciones. Por ejemplo, esta es una rutina diseñada para generar un tono una frecuencia especifica:

```
void  GenerateAtone( short frequency = 440 )
{
      // A frequency of 440 is equal to an A note
}
```

Se llama a esta función con un parametro, el valor anterior es usado. Por ejemplo, el llamado

```
GenerateAtone( 330 );
```

Va a generar un tono con una frecuencia de 330 beats por segundo, el cuál, en notación musical, es equivalente a una nota E. Si se llama a la función fuera especificando un valor, el valor por default es usado.

La llamada:

```
GenerateAtone( );
```

Va a generar un tono con una frecuencia de 440, el cuál representa una nota A.

EL ACERCAMIENTO TRADICIONAL.

Por convensión, todos los valores de los parametros por default estan especificados en la función prototipo antes que en la implementación de la función. Por ejemplo, la siguiente es una función prototipo, seguida por la misma función:

```
void  MyFunc( short param1 = 27 );
void  MyFunc( short param1 )
{
      // Body of the function...
}
```

VARIABLES POR REFERENCIA.

En C, todos los parametros son pasados por valor como oponiendose al pasado existente por referencia. Cuando se pasa un parametro a una función de C, el valor del parametro es pasado a la función. Cualquier cambio hecho a este valor no es acarreado de regreso al llamado de la función.

Este es un ejemplo:

```
void DoubleMyValue( short valueParam )
{
      valueParam *= 2;
}
void main( void )
{
      short number = 10;
      DoubleMyValue( number );
}
```

main( ) pone number a 10, entonces pasa este a la función DoubleMyValue( ). Desde que number es pasado por valor, el llamado a DoubleMyValue( ) no tiene efecto sobre number. Cuando DoubleMyValue( ) regresa, number todavía tiene un valor de 10.

LA ALTERNATIVA VARIABLES POR REFERENCIA.

C++ ofrece una buena alternativa para parametros basados en apuntadores. Variables por Referencia permite pasar un parametro por referencia, sin usar apuntadores.

Esta es otra versión del programa anterior, esta vez implementada con una variable por referencia.

```
void DoubleMyValue( short &referenceParam )
{
      referenceParam *= 2;
}
void main( void )
{
      short number = 10;
      DoubleMyValue( number );
}
```

Note que este código es justo como la primera versión, con una pequeña excepción. El parametro de DoubleMyValue( ) esta definido usando el operador &:

```
short &referenceParam
```

Las marcas & y referenceParam son una variable por referencia y llaman al compilador como referenceParam y este corresponde al parametro de entrada, number y el parametro son el mismo.

Al igual que en C para C++ podemos utilizar variables por referencia dentro de una función especifica, utilizando el operador &.

SOBRECARGANDO EL NOMBRE DE LA FUNCION.

La siguiente característica para discutir es, sobrecargando el nombre de la función, permite escribir varias funciones como parte del mismo nombre.

Suponemos que necesitamos una función que pueda imprimir el valor de una de sus variables, pueden ser de tipo long, short ó una cadena de texto. Se puede escribir una función que tenga cuatro parámetros:

```
Display( short whichType,
         long longParam,
         short shortParam,
         char *textParam );
```

El primer parámetro puede actuar como un switch, determinando cuál de los tres tipos fueron pasando e imprimiendose. El código main de la función puede observarse como:

```
if ( whichType == kIsLong )
      cout << "The long is:  " << longParam << "\n";
else if ( whichType == kIsShort )
      cout << "The short is:  " << shortParam << "\n";
else if ( whichType == kIsText )
      cout << "The text is:  " << text << "\n";
```

Otra solución es escribir tres funciones separadas, una para imprimir shorts, una para longs y una para cadenas de tipo texto:

```
void DisplayLong( long longParam );
void DisplayShort( short shortParam );
void DisplayText( char *text );
```

LOS OPERADORES NEW Y DELETE.

En C, la memoria asigna un llamado a `malloc( )` junto con un llamado a `free( )` cuando la memoria no necesita ampliarse. En C++, ocurre lo mismo para los operadores new y delete.

Se llama a new cuando se quiere asignar un bloque de memoria. Por ejemplo, el siguiente código asigna un bloque de 1024 caracteres:

```
char *buffer;
buffer = new char[ 1024 ];
```

new toma el tipo de un operando, asigna un bloque de memoria del mismo tamaño que el tipo, y regresa un apuntador al bloque. Regresa la memoria a el heap, usando el operador delete. El siguiente código libera la memoria asignada:

```
delete buffer;
```

new puede ser usado con cualquier tipo legal de C++, incluyendo los creados por nosotros. Son ejemplos:

```
struct Wobble

{
      short papaWobble;
      short mamaWobble;
      long  littleBabyWobble;
};
short       *shortPtr;
long double *longDoublePtr;
Wobble            *wobblePtr;
shortPtr = new short;
longDoublePtr = new long double;
wobblePtr = new Wobble;
```

Este es un ejemplo de un mal uso de *new*, garantizando la falla del compilador:

```
short *shortPtr;
shortPtr = new 1024;    // Will not compile!!!
```

Los operadores new y delete de C++, funcionan de la misma manera que malloc( ) y free( ) para C. Estos operadores funcionan juntos usando la zona de memoria libre que queda entre el programa y la parte alta de la pila para establecer y mantener una lista de almacenamiento disponible.

EL OPERADOR SCOPE.

La siguiente característica a examinar es el *operador scope* (::) de C++. El operador scope precede a la variable, indicando al compilador que observe fuera del bloque una variable del mismo nombre.

Suponemos que se declara una variable global y una variable local con el mismo nombre:

```
short number;
void main( void )
{
      short number;
      number = 5; // local reference
      ::number = 10;     // global reference
}
```

Dentro de *main( )*, la primera declaración se refiere a la definición local de number. La segunda declaración usa el operador scope para referirse a la definición global de number. Este código permite a number local con un valor de 5 y a number global con un valor de 10.

Este operador se utiliza en C++, para resolver conflictos entre nombres iguales definidos en una misma función; donde podemos tener definida una variable de tipo local y con el operador scope (::), podemos definir una variable de tipo global anteponiendo el operador a la variable. Este operador tambien es usado para conexión entre clases.

FUNCIONES EN LINEA.

Normalmente, cuando una función es llamada, el CPU ejecuta un conjunto de instrucciones que mueven el control desde la función llamando a la función llamada. Esta instrucción va al principio de la función y regresa a la dirección de return.

C++, proporciona *funciones en linea*, las cuales te permiten desviar estas instrucciones y salvar un bit de tiempo de ejecución. Así es como esta caracteristica trabaja.

Cuando se declara una función usando el comando *inline*, el compilador copia el cuerpo de la función dentro del llamado de la función, haciendo la copia de una parte de la instrucción de la función llamada como si esta hubiera sido escrita de ese modo originalmente.

Este operador se utiliza en C++, para resolver conflictos entre nombres iguales definidos en una misma función; donde podemos tener definida una variable de tipo local y con el operador scope (::), podemos definir una variable de tipo global anteponiendo el operador a la variable. Este operador tambien es usado para conexión entre clases.

FUNCIONES EN LINEA.

Normalmente, cuando una función es llamada, el CPU ejecuta un conjunto de instrucciones que mueven el control desde la función llamando a la función llamada. Esta instrucción va al principio de la función y regresa a la dirección de return.

C++, proporciona *funciones en linea*, las cuales te permiten desviar estas instrucciones y salvar un bit de tiempo de ejecución. Así es como esta caracteristica trabaja.

Cuando se declara una función usando el comando *inline*, el compilador copia el cuerpo de la función dentro del llamado de la función, haciendo la copia de una parte de la instrucción de la función llamada como si esta hubiera sido escrita de ese modo originalmente.

1.2.3 SURGIMIENTO DEL LENGUAJE VISUAL C++

PROGRAMACION VISUAL.

El éxito de Windows en el entorno PC ha obligado a los programadores a reciclarse para poder desarrollar aplicaciones que respeten el "look" y las necesidades de esta interface gráfica. La llamada "programación visual" supone un paso más en la búsqueda de nuevas herramientas que faciliten el desarrollo de software para Windows.

Empresas como Microsoft y Borland se han apresurado a lanzar al mercado versiones de sus productos con las palabras "visual", pero todavía no está muy claro el significado de este término. Se prodría decir que "programación visual" se entiende como el proceso de desarrollo de forma interactiva del entorno de usuario de las aplicaciones. La idea es que la "programación visual" permite que la mayor parte del proceso necesario para crear cualquier aplicación se pueda realizar en forma automática.

Así por ejemplo, si la mayoría de las aplicaciones de Windows disponen de menús desplegables, parece razonable desarrollar una herramienta para crearlos sin tener que recurrir a páginas de código indescifrable. Pero los beneficios no acaban aquí, porque todo lo mencionado hace tiempo que estaba disponible en los editores de recursos, ahora se cuanta con mayor integración entre el desarrollo de aplicaciones propiamente dicha y el diseño de interface de usuario.

Los editores de recursos que actualmente podemos encontrar en el mercado (en los compiladores de Pascal o C para Windows siempre viene alguno) siguen estas pautas:

Primero se crea el recurso deseado -que puede ser una ventana, un menú, un icono, algún tipo de letra, un botón, un campo de texto o cualquier otro elemento típico de Windows-, después se genera el código correspondiente al recurso y, por último, con más o menos pasos intermedios, se acaba uniéndolo al resto del código fuente de nuestro programa como se ve en la figura 1.2.3.1.

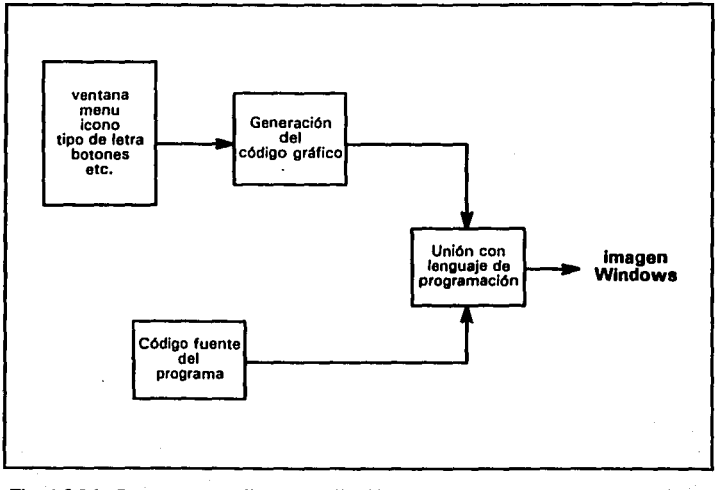


Fig. 1.2.3.1 Proceso para realizar una aplicación con imagen Windows.

Con la "programación visual" se pretende simplificar al máximo todo el proceso, evitando cuantos pasos intermedios sea posible. Lo normal es que se parta de la creación de un interface de usuario y se llegue a la escritura del código (si es necesario) en algún lenguaje de programación.

Por el momento, la "programación visual" ha dado un paso de gigante en la tarea de facilitar la vida a los informáticos; sin embargo, todavía no ha llegado el día en el que cualquier persona sea capaz de decirle a la computadora mediante un micrófono lo que quiere y que ésta lo haga. Hasta entonces tendremos que seguir trabajando con las herramientas disponibles, muchas de las cuales ya han comenzado su transición a la "programación visual".

Desde que la informática abandonó las tarjetas perforadas, la mayor preocupación de los programadores ha sido crear aplicaciones que "entrasen por los ojos"; es decir, que no sólo fueran fáciles de usar, sino también atrayentes. En pocos años hemos vivido una verdadera revolución en la manera de entender las aplicaciones; esta revolución ha venido de la mano de las nuevas y poderosas PCs, de los sistemas operativos y de periféricos tales como modems, impresoras o scáners.

Además, la proliferación de las interfaces gráficas de usuario -los llamados entornos de ventanas o GUI- no ha hecho más que acentuar el desfase que existía entre las técnicas de programación de los años ochenta y las actuales. Muchos han sido los desarrolladores que no han sabido adaptarse a los nuevos tiempos y se han quedado "fuera de juego"; sin embargo, aquellos que han actualizado sus conocimientos reconocen que ahora cuentan con mejores y más potentes herramientas para realizar su trabajo.

Al mismo tiempo, las herramientas de programación han evolucionado hasta convertirse en verdaderos entornos de desarrollo integrados que se basan en la misma filosofía de sencillez que gobierna los interfaces gráficos. En el campo de la integración de herramientas Borland fue pionera con su famoso Turbo Pascal, cuyas primeras versiones aparecieron para el sistema operativo CPM de la famosa familia de procesadores Z80 de Zilog. Por otra parte, Microsoft ha sido pionera con el concepto de "programación visual" con su popular Visual Basic, que ahora en su versión para Windows se ha convertido en un best-seller.

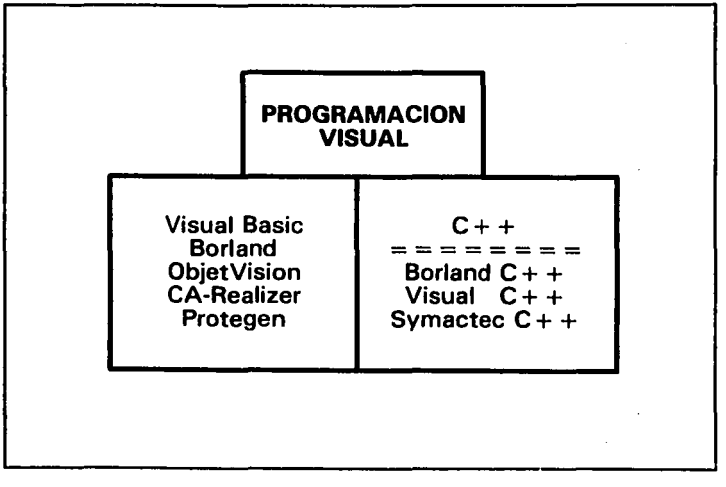


Figura 1.2.3.2 Paquetes y lenguajes que manejan la programación "Visual".

DIFICIL DEFINICION.

Anteriormente mencionamos que el término en sí era demasiado genérico y que procedía principalmente del nombre que las propias casas de software daban a sus IDE (*Integrated Development Environment*, entornos integrados de desarrollo) para lanzarlos publicitariamente con la idea de ser algo intuitivo y fácil de usar.

Sin embargo, con la aparición de los ultimos productos de desarrollo de aplicaciones bajo entornos gráficos se va haciendo necesaria una terminología adecuada que los clasifique de alguna manera.

Un entorno integrado de programación es aquel que permite desde un único programa realizar las tareas más habituales de desarrollo y depuración de una aplicación (incluido en algunos casos la optimización y realización automática de pruebas).

Si el entorno integrado es de "programación visual", estamos hablando de un producto que integra una serie de herramientas adicionales a las habituales y que son necesarias para el desarrollo de programas bajo una interface gráfica de usuario, todo ello con la mínima intervención del programador en las tareas más repetitivas y, por tanto, más automatizables.

Podemos decir que la "programación visual" se sitúa directamente por debajo de las llamadas herramientas CASE (Computer Aided Software Engeneering).

Los programas CASE permiten realizar el sueño (o pesadilla) de todo programador: pasar directamente del diseño de una aplicación (diagramas de flujo, estructuras de datos...) a producir la aplicación propiamente dicha. La pesadilla se puede hacer realidad para los

programadores cuando llegue el día en que cualquier usuario normal pueda crear, gracias a estas herramientas, su propia aplicación, prescindiendo, por lo tanto, de los servicios de un profesional.

PROGRAMAR LA INTERFACE

Con estas premisas, está claro que la "programación visual" ha pasado de ser una curiosa novedad metodológica, a constituirse como un conjunto de herramientas imprescindibles para el desarrollo de las complejas aplicaciones basadas en GUIs que se comercializan actualmente.

La complejidad de programar bajo un interface gráfica proviene del propio diseño de los entornos. De hecho, desarrollar el programa más sencillo para un entorno gráfico puede convertirse en una tarea desquiciante si no se dispone de las herramientas adecuadas.

Para afrontar el desarrollo de una aplicación para una interface gráfica (Windows u OS/" en el mundo PC) podemos seguir dos caminos. El primero consiste en armarnos de paciencia y enfrentarnos con los conceptos básicos del entorno, lo cual resulta una tarea tediosa y poco gratificante porque a veces no es efectiva ante las funcionalidades ya existentes (OLE) o los nuevos sistemas que se lanzarán en breve (por ejemplo, Windows Chicago). El segundo camino se simplifica al utilizar las mencionadas herramientas de "programación visual".

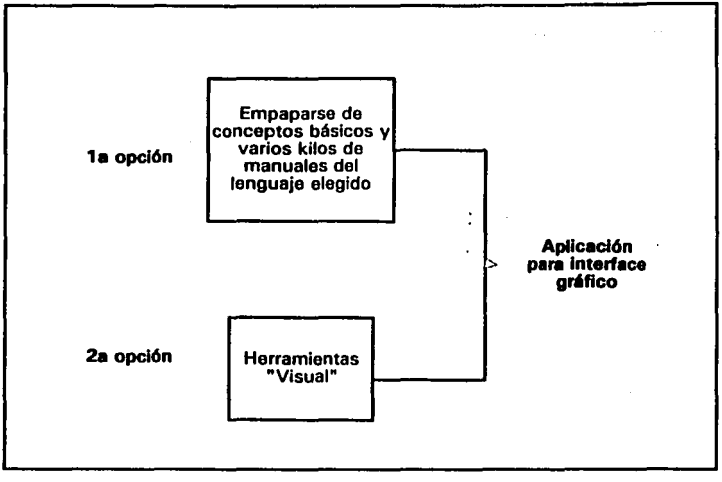


Fig. 1.2.3.3 Desarrollo de una aplicación para una interface gráfica,

Estos compiladores simplifican el diseño al proporcionarnos formatos, tipo o patrones que podemos aplicar a nuestras propias aplicaciones, al mismo tiempo que ofrecen una elevada confiabilidad porque se tratan de patrones probados en multitud de otros desarrollos. Todo ello redunda en una mayor eficiencia a la hora de programar y en un gran ahorro de tiempo de codificación.

VISUAL EN C.

Hasta hace poco los únicos que podían disfrutar de este tipo de ayuda en su trabajo eran los programadores de Basic y Pascal gracias al Visual Basic de Microsoft y al Turbo Pascal para Windows de Borland.

Sin embargo, sucede que la mayoría de los programadores profesionales prefieren el lenguaje C antes que el Basic o el Pascal. Su sencillez y poca rigidez sintáctica hacen que la construcción de compiladores sea muy sencilla, lo que facilita la portabilidad y la eficiencia de éstos, que llegan incluso a optimizar el código introducido por el programador.

Se puede decir sin temor a equivocarse que existen compiladores de C para TODAS las combinaciones posibles de microprocesadores, arquitecturas de hardware y sistemas operativos.

Si el C++ ha sido una evolución natural del C hacia la programación moderna, lo que aquí llamamos "programación visual" podría ser la evolución de los entornos de programación integrados para el lenguaje C que hasta hace poco no habían abandonado el modo texto.

Quede claro que no estamos hablando de una innovación en el campo del lenguaje propiamente dicho, sino de una inevitable evolución hacia formas más cómodas, y a la vez más seguras, de programar, gracias a la facilidad de uso de las interfaces gráficas.

MICROSOFT VISUAL C++

Aunque el lenguaje favorito de Microsoft es el Visual Basic para aplicaciones, son muchos los programadores que por muchos motivos prefieren el C o Pascal para sus aplicaciones. Para estos usuarios existe una herramienta muy poderoso "Microsoft Visual C++".

Hasta hace unos años, Microsoft no había adoptado la filosofía de los entornos integrados de programación en sus productos. Sus compiladores se invocaban dedes la línea de comandos, teniendo que incluir multitud de parámetros para conseguir los resultados deseados. Sin embargo, su entrada en este campo ha sido espectacular.

El nacimiento de Windows supuso un paso decisivo en general hacia el desarrollo de herramientas totalmente integradas e intuitivas, pero el más beneficiado en este aspecto ha sido Microsoft. Casi nadie esperaba que alcanzaría a la mismísima Borland -pionera en ambientes bajo DOS- en cuanto a potencia y facilidad de uso en sus entornos integrados de programación.

Visual C++ supone la competencia que Microsoft opone a Borland en este lenguaje que cuenta con tantos adeptos entre los programadores. Se supone que el inventor de Windows debe disponer de los mejores lenguajes de programación para las aplicaciones de este entorno.

1.3 Caracteristicas y funcionamiento de las Bases de Datos Relacionales

Una Base de Datos es una colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es la de servir a una o más aplicaciones; los datos son independientes de los programas que los usan; se emplean métodos bien determinados para incluir datos nuevos y modificar o extraer los datos almacenados.

El enfoque de la Base de Datos nos permite:

- Controlar la redundancia
- Mantener la consistencia
- Lograr la integración de los datos
- Compartir los datos entre las difrentes aplicaciones
- Cumplir con los estándares
- Tener facilidad en el desarrollo de aplicaciones
- Uniformar los controles de seguridad, privacia e integridad
- Independencia entre los datos y los programas
- Reducir el mantenimiento a los programas

Un sistema de manejo de Base de Datos (DBMS) consiste en un conjunto de datos relacionados entre si y un grupo de programas para tener acceso a esos datos. El objetivo primordial de un DBMS es crear un ambiente en que sea posible guardar y recuperar información de la bases de datos en forma conveniente y eficiente.

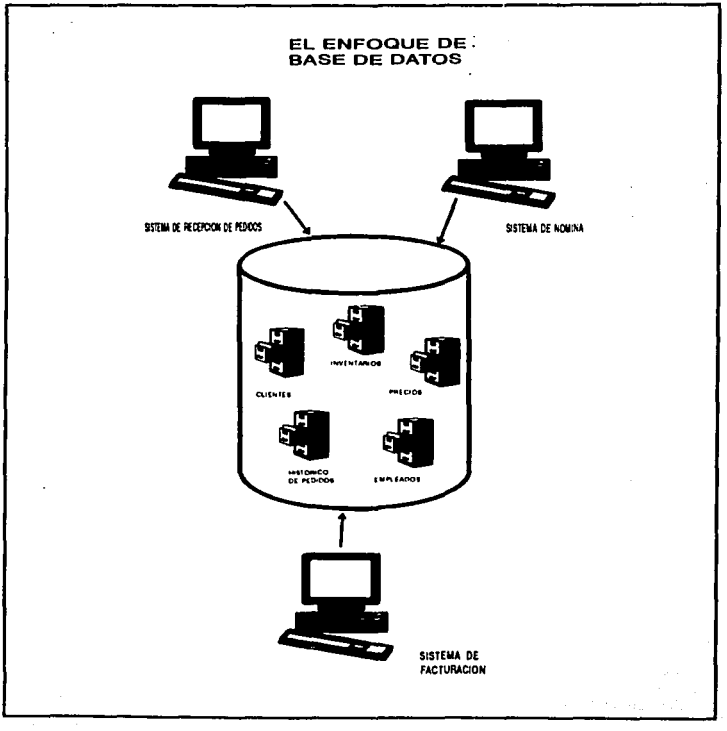


Fig. 1.3.1 El enfoque de Base de Datos.

TIPOS DE ENFOQUE DE BASES DE DATOS

Existen enfoques alternativos para visualizar y manejar datos a un nivel lógico independientemente de cualquier estructura física de soporte en que se basen,

Los modelos de base de datos que existen son:

- Modelo Jerárquico
- Modelo de Red
- Modelo Relacional

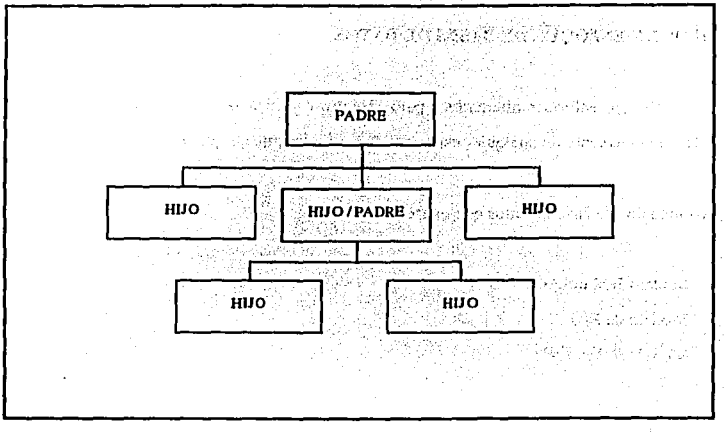
Modelo Jerárquico o de Arbol

La estructura lógica en la cual se sustenta la base de datos jerárquica es el árbol. Un árbol se compone de un nodo raíz y varios nodos sucesores, ordenados jerarquicamente. Cada nodo representa una entidad (tipo de registro) y las relaciones entre entidades son las conexiones entre los nodos.

El nodo colocado en la parte superior es llamado padre y los nodos inferiores son los hijos.

En el sistema jerárquico, las conexiones entre archivos no dependen de la información contenida en ellos; se definen al inicio y son fijos.

La característica sobresaliente de este modelo es el manejo de la conexión uno a muchos, entre un padre y varios hijos, en otras palabras, cada hijo sólo tiene un padre.



1.3.2 Enfoque Jerárquico.

Como ejemplo podemos tener 2 entidades: *cuentahabiente* y *cuenta*, relacionadas entre sí con una sola relación muchos a muchos, es decir, un cuentahabiente puede tener varias cuentas, y una cuenta puede pertenecer a varios cuentahabientes (fig. 1.3.3).

El tipo de registro *cuentahabiente* consiste de 3 campos: nombre, calle y ciudad. El tipo de registros cuenta tiene 2 campos: número y saldo.

Desventajas en el enfoque jeráquico:

- No modela sencillamente las relaciones M a M.
- Anomalias de inserción.
- Anomalias de barrado.
- Anomalias de actualización
- Se pueden dar consultas inconsistentes.

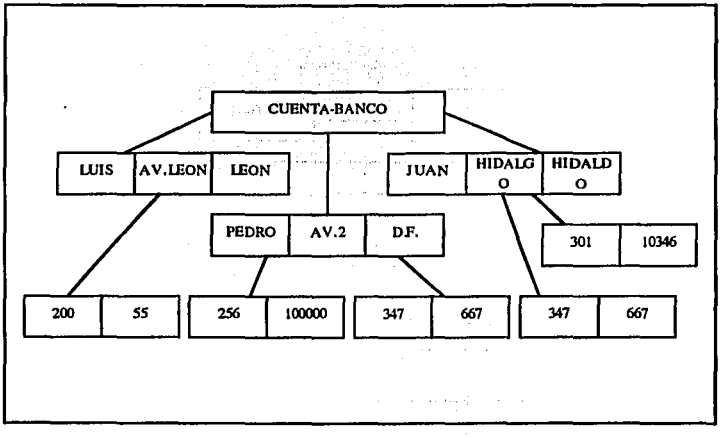


Fig. 1.3.3 Modelo Jerárquico.

Modelo de Red.

Los datos se representan como registros ligados formando un conjunto de datos intersectados.

La base de datos de red, a diferencia de las jerárquicas, permite cualquier conexión entre entidades, es decir, se pueden representar relaciones de muchos a muchos. En una red, un hijo puede tener varios padres y varios hijos a la vez.

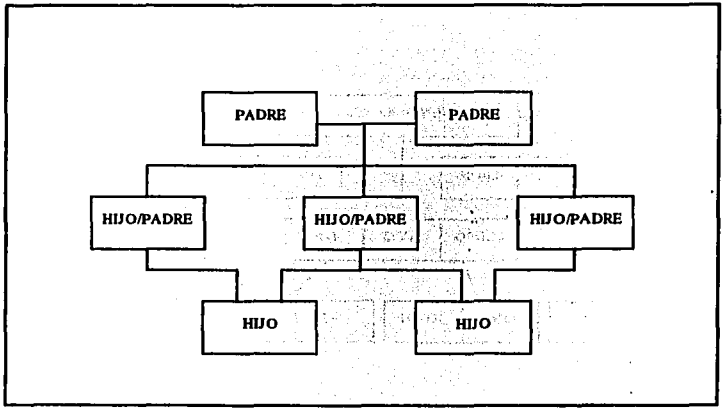


Fig. 1.3.4 Enfoque de Red.

Como ejemplo podemos tener 2 entidades: *cuentahabiente* y *cuenta*, relacionadas entre sí con una sola relación muchos a muchos, es decir, un cuentahabiente puede tener varias cuentas, y una cuenta puede pertenecer a varios cuentahabientes (fig. 1.3.5).

El tipo de registro *cuentahabiente* consiste de 3 campos: nombre, calle y ciudad. El tipo de registros cuenta tiene 2 campos: número y saldo.

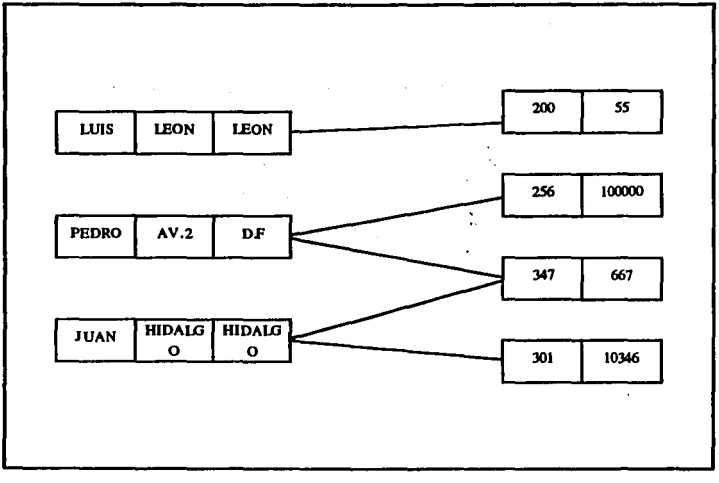


Fig. 1.3.5 Modelo de Red.

Desventajas en el modelo de red:

- Resulta difícil definir nuevas relaciones.
- Es complicado darle mantenimiento ya que cualquier cambio en la estructura requiere una descarga de datos.
- Representa desperdicio de recursos.
- Anomalías de inserción.
- Anomalías de borrado.

Modelo Relacional.

La estructura de una base de datos relacional está basada en la representación de entidades mediente tablas, las cuales constan de columnas (campos) y renglones (registros). Las relaciones entre tablas se llevan a cabo a través de un conjunto de columnas que se tengan en común, logrando una conexión dinámica entre un número ilimitado de ellas a través del contenido de esas columnas.

La ventaja de los sistemas relacionales es el poder modificar la información sin la preocupación de especificar las combinaciones entre registros.

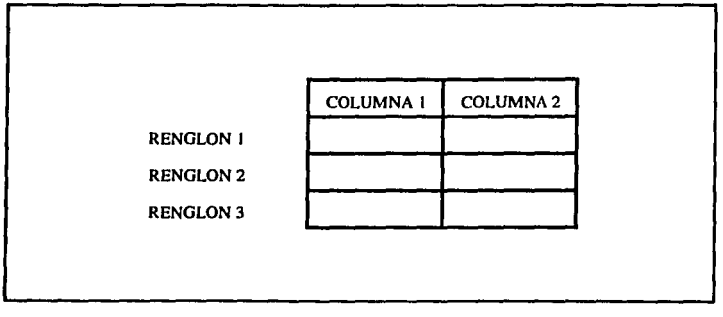
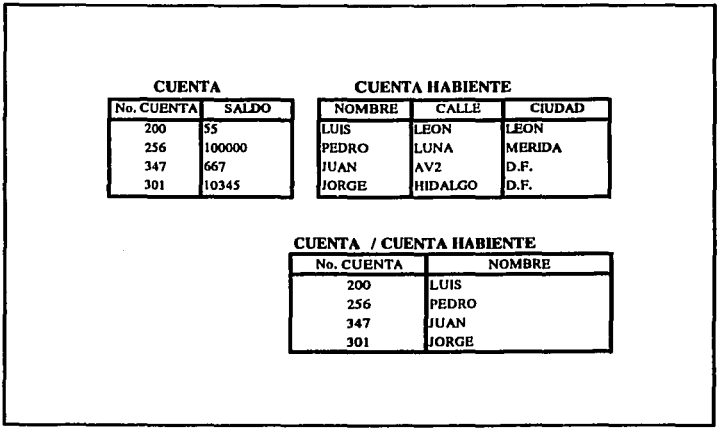


Fig. 1.3.6 Enfoque relacional.

Como ejemplo podemos tener 2 entidades: *cuentahabiente* y *cuenta*, relacionadas entre sí con una sola relación muchos a muchos, es decir, un cuentahabiente puede tener varias cuentas, y una cuenta puede pertenecer a varios cuentahabientes (fig. 1.3.7).

La tabla *cuentahabiente* consiste de 3 columnas: nombre, calle y ciudad. La tabla *cuenta* tiene 2 columnas: número y saldo.



CUENTA

No. CUENTA	SALDO
200	55
256	100000
347	667
301	10345

CUENTA HABIENTE

NOMBRE	CALLE	CIUDAD
LUIS	LEON	LEON
PEDRO	LUNA	MERIDA
JUAN	AV2	D.F.
JORGE	HIDALGO	D.F.

CUENTA / CUENTA HABIENTE

No. CUENTA	NOMBRE
200	LUIS
256	PEDRO
347	JUAN
301	JORGE

Fig. 1.3.7 Modelo relacional.

Modelo Entidad-Relación.

El Modelo Entidad-Relación es una técnica para definir las necesidades de información de cualquier empresa. Esta técnica involucra conceptos que se identifican con varios objetos de importancia para la empresa, a los cuales se les denomina ENTIDADES, a las características de dichos objetos se les denomina ATRIBUTOS y a como se relacionan estos objetos entre sí se le denomina RELACIONES.

Todos estos conceptos se modelan a través de cierto tipo de esquemas gráficos, los cuales muestran a los usuarios una manera más sencilla y práctica de visualizar sus necesidades de información.

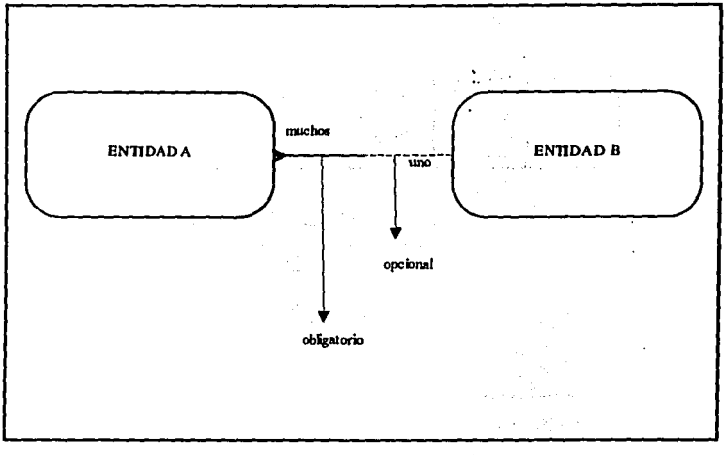


Fig. 1.3.8 Modelo Entidad-Relación.

Una entidad es una persona, cosa o lugar, que cae dentro del alcance del sistema, acerca de la cual el sistema debe mantener, correlacionar y desplagar información. La entidad se representa por medio de una caja con las esquinas redondeadas y dentro de ésta se escribe el nombre de la entidad, el cual debe estar en singular. Cada entidad debe tener un nombre único dentro del sistema, lo que indica que no pueden existir dos entidades dentro del mismo sistema con el mismo nombre.

La entidad dentro del modelo relacional se representa por medio de una tabla, donde el nombre de la tabla debe contener una columna que identifique de forma única a cada renglón de ésta. Esta columna recibe el nombre de llave primaria (PK), la cual no puede contener valores nulos, ni duplicados. Ejemplo Fig. 1.3.9.

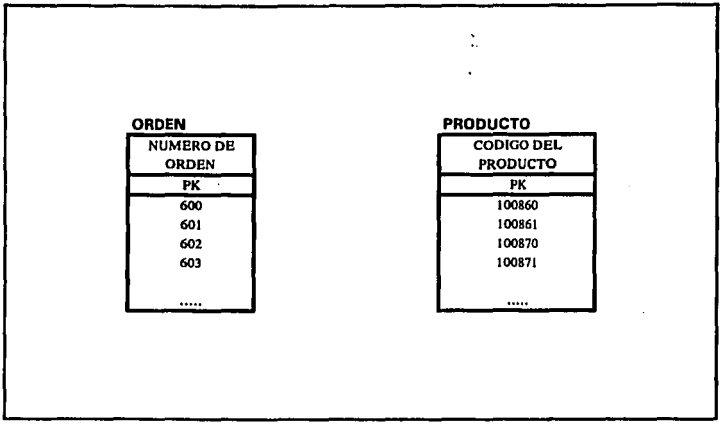
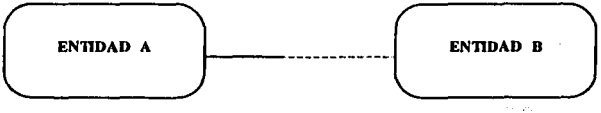


Fig. 1.3.9 Entidad.

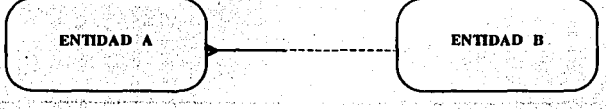
Una relación requiere de una o más entidades, la cual debe caer dentro del alcance del sistema, acerca de la cual el sistema debe mantener, correlacionar y desplegar información.

Las relaciones se presentan en varias modalidades:

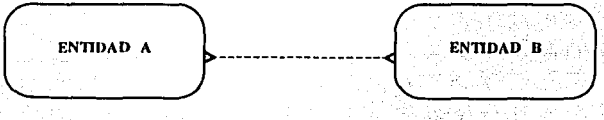
- Relaciones Uno a Uno con las combinaciones de obligatorio a opcional, opcional a opcional y obligatorio a obligatorio.



- Relaciones Uno a Muchos con las combinaciones de obligatorio a opcional, opcional a opcional y obligatorio a obligatorio.



- Relaciones Muchos a Muchos con las combinaciones de obligatorio a opcional, opcional a opcional y obligatorio a obligatorio.



Las RELACIONES dentro del modelo relacional se representan de la siguiente forma:

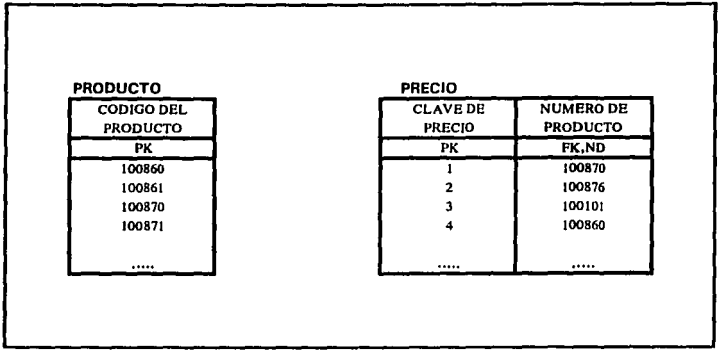


Fig. 1.3.10 Relación Uno a Uno.

- Relación Uno a Uno entre dos entidades se modela dibujando la llave primaria (PK) de la tabla en más renglones como llave foránea (FK) en la otra tabla.

- Una llave foránea es una o más columnas que son llave primaria en otra tabla, una llave foránea permite nulos y valores duplicados.

- Relación Uno a Muchos o Muchos a Uno entre dos entidades se modela dibujando la llave primaria (PK) de la tabla que tiene la correspondencia de uno como llave foránea (FK) en la otra tabla.

- Relación Muchos a Muchos entre dos entidades se modela dibujando una tercer tabla, la cual se compone de una llave primaria (PK) compuesta de dos columnas, las cuales son llaves primarias y foráneas a la vez.

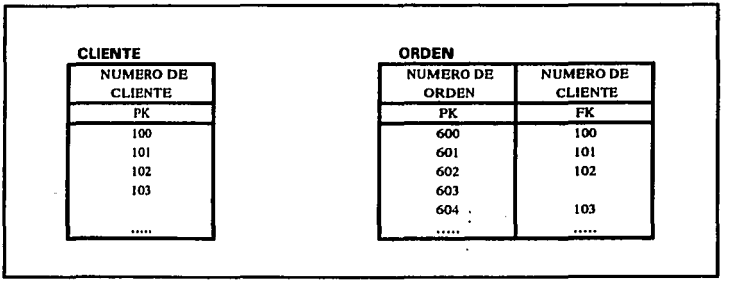


Fig. 1.3.11 Relación Uno a Muchos o Muchos a Uno.

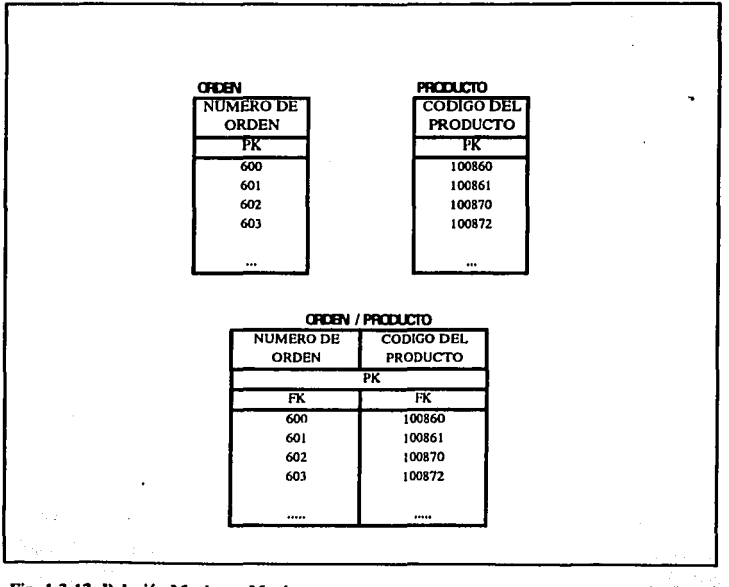


Fig. 1.3.12 Relación Muchos a Muchos.

Un atributo es una característica o cualidad de una entidad o relación, que cae dentro del alcance del sistema, acerca del cual el sistema debe mantener, correlacionar y desplagar información.

Para representar uno o varios atributos, se escribe el nombre del atributo dentro de la entidad.

Los atributos dentro del modelo relacional se presentan por medio de columnas dentro de una tabla, fig. 1.3.13.

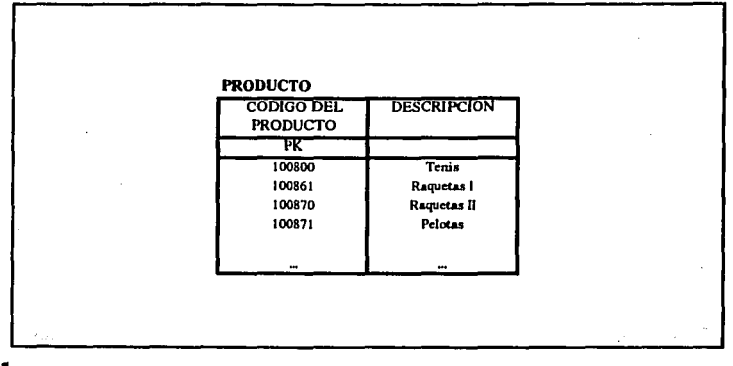


Fig. 1.3.13 Atributos.

EL PROCESO DE NORMALIZACION

El enfoque relacional posee bases matemáticas rigurosas que respaldan su teoría relacional, proporcionando simplicidad en las estructuras de datos utilizadas, facilitando su uso y modificaciones. Para poder obtener estas facilidades, el proceso de normalización es la clave.

Los objetivos del proceso de normalización son:

- Eliminar en lo posible todos los datos que mantegan anomalías
- Conservar toda la información
- Maximizar la flexibilidad
- La estructura debe ser tal que haya lugar para todos los datos requeridos.
- La redundancia que puede existir deberá ser usada por los elementos que son identificadores o llaves. Por lo que se debe tener cuidado de elegir aquellos que no estén sujetos a actualizaciones.
- Los efectos indeseables son las anomalías que pueden presentarse en las operaciones de actualización, inserción y eliminación.

Anomalía de Inserción: No se debe almacenar nueva información sobre una entidad en particular hasta que se establece su relación con otra entidad.

Anomalía de eliminación: La eliminación de un solo registro puede ocacionar la eliminación de toda una ocurrencia de una entidad.

Anomalía de actualización: si el valor de un atributo cambia, debe cambiar en los multiples sitios donde se encuentre definido.

- Esta capacidad de adaptabilidad de los cambios maximizan la independencia de uso particualr de los datos.

La normalización requiere tres acciones sobre un atributo de una entidad. Estas son las siguientes:

Primera forma normal

Segunda forma normal

Tercera forma normal

DEPENDENCIA FUNCIONAL

El poder definir si una relación se encuentra en la primera, segunda o tercera forma normal, se basa en las dependencias funcionales que existan entre los atributos y los dominios particulares a esa relación; las dependencias funcionales las determina directamente el significado o la semántica del contenido de la base de datos según la interpretación del diseñador de la base de datos.

El modelo relacional se basa en este concepto para establecer sus relaciones funcionales entre atributos. De esta forma, una definición formal de DEPENDENCIA FUNCIONAL en el modelo relacional sería la siguiente:

Dada la relación R se dice que el atributo B es funcionalmente dependiente del atributo A si en cualquier instante de tiempo cada valor de A no tiene más de un valor B asociado con él en la relación R. el indicar que B es funcionalmente dependiente de A es equivalente a indicar que A identifica o determina a B, lo cual se denota como A-->B.

esto último concuerda con la lógica matemática en la que, A-->B significa que A identifica a B, es decir, que si A tiene un cierto valor "a", entonces B debe tener un valor "b".

PRIMERA FORMA NORMAL

Uno de los objetivos del enfoque relacional es representar las bases de datos mediante relaciones planas o tablas. Por lo tanto cada idenfiticador en una relación debe poseer un solo valor de cada uno de los atributos y no mútiples valores de estos, si posee multiples valores se dice que existen grupos repetitivos.

Por ejemplo consideremos la relación INF_TUR que posee la siguiente estructura y se muestra en la Tabla 1.3.1

En esta relación se puede observar que existen multiples valores para el atributo CVE_GIRO que es el identificador, por lo tanto no es una relación plana.

Se dice que una relación está en **Primera Forma Normal** si para cada valor específico de un identificador existe uno y sólo un valor de cada atributo. Es decir no hay "grupos repetitivos". Pero se mantiene un alto grado de redundancia.

Para transformar la relación de la tabla 1.3.1. a la Primera Forma Normal aplicamos la conversión de la 1FN:

INF_TUR

CVE_GIRO	GIRO	TIPO_GIRO	NUM_FOLIO	NOM_COM	CALLE	NUMERO	COLONIA	C.P.	TEL	R.F.C	RAZON_SOC	CATEGORIA	PRN_SER
56	HOTEL	S	12H	HOTEL DORADO PACIFICO	PASEO IXTAPA	S/N	IXTAPA	40880	32025	OHD800408UR0	OPERADORA HOTELERA DORADO	5 ESTRELLAS	SERV. COMPL
56	HOTEL	S	5H	IXTAPA PALACE RESORT	PASEO DE LAS GARZAS ESQ. PASEO DEL RINCO	S/N	RESIDENCIAL III	40880	31359	APT981110R17	AMERICA PROMOTORA TURISTICA S.A. DE C.V.	5 ESTRELLAS	SERV. COMPL
56	HOTEL	S	23H	OMNI HOTEL	PASEO DEL PALOMAR	S/N	ZONA HOTELERA	40880	30003	PSI870529BKJ	PROPULSORA IXTAPA SUR, S.A. DE C.V.	5 ESTRELLAS	SERV. COMPL
15	RESTAURANTES	S	10R	PIZZERIA MAMMA NORM	PASEO IXTAPA	LOC. 8	IXTAPA	40880	30274	DINE420722DY2	DIBILDOX NIETO ENRIQUE	N/C	N/C
15	RESTAURANTES	S	4R	LAS MARGARITAS	PASEO IXTAPA	S/N	CENTRO	40880	30910	AIGS540802838	ALIJA GONZALEZ FRANCISCO	N/C	N/C
15	RESTAURANTES	S	8R	DA BAFFONE	PASEO IXTAPA	S/N	IXTAPA	40880	31122	RDA820107BKG	RESTAURANTE DA BAFONE, S.A.	N/C	N/C
73	DISCOTEQUES	S	3D	MAGIC CIRCUS	PASEO IXTAPA	S/N	IXTAPA	40880	31509	DMM000328LXA	OPIXTAPA MAGICO, S.A. DE C.V.	N/C	N/C
87	TIENDA DE ROPA	B	7TR	ACA JOE	PASEO IXTAPA	S/N	IXTAPA	40880	30302	ARO810717LHA	ACA ROPA S.A. DE C.V.	N/C	N/C

Tabla 1.3.1

CONVERSION DE LA PRIMERA FORMA NORMAL (1FN) :

La redundancia puede reducirse separándola en otro grupo aparte de la entidad, debiendo contener el identificador de la entidad original para mantenerse relacionadas.

Por lo tanto la relación INF_TUR normalizada en la 1FN quedará de la siguiente manera :

- Separando la redundancia en otro grupo aparte, obtendremos un catálogo denominado GIROS :

GIROS

CVE_GIRO	GIRO	TIPO_GIRO	CATEGORIA	PRIN_SER
56	HOTEL	S	5 ESTRELLAS	SERV. COMPL
56	HOTEL	S	5 ESTRELLAS	SERV. COMPL
56	HOTEL	S	5 ESTRELLAS	SERV. COMPL
15	RESTAURANTES	S	N/C	N/C
15	RESTAURANTES	S	N/C	N/C
15	RESTAURANTES	S	N/C	N/C
73	DISCOTEQUES	S	N/C	N/C
87	TIENDA DE ROPA	B	N/C	N/C

Tabla 1.3.2

El identificador de la relación GIIROS es CVE_GIRO.

La relación de la tabla 1.3.1 quedará tal y como se muestra en la tabla 1.3.3. con la conversión de la 1FN.

El identificador de la relación INF_TUR es FOLIO y CVE_GIRO para que pueda relacionarse con GIROS.

CVE_GIRO	NUM_FOLIO	NOM_COM	CALLE	NUMERO	COLONIA	C.P.	TEL	R.F.C	RAZON_SOC
58	12H	HOTEL DORADO PACIFICO	PASEO IXTAPA	S/N	IXTAPA	40880	32025	OHD800408UR0	OPERADORA HOTELERA DORADO
58	5H	IXTAPA PALACE RESORT	PASEO DE LAS GARZAS ESQ. PASEO DEL RINCON	S/N	RESIDENCIAL III	40880	31359	APT861110R17	AMERICA PROMOTORA TURISTICA S.A. DE C.V.
58	23H	OMNI HOTEL	PASEO DEL PALOMAR	S/N	ZONA HOTELERA	40880	30003	PIS8705286KJ	PROPULSORA IXTAPA SUR, S.A. DE C.V.
15	10R	PIZZERIA MAMMA NORMA	PASEO IXTAPA	LOC. 6	IXTAPA	40880	30274	DINE420722DY2	DIBILDOX NIETO ENRIQUE
15	4R	LAS MARGARITAS	PASEO IXTAPA	S/N	CENTRO	40880	30910	AIGS540802838	ALIJA GONZALEZ FRANCISCO
15	8R	DA-BAFFONE	PASEO IXTAPA	S/N	IXTAPA	40880	31122	RDA8201078KG	RESTAURANTE DA-BAFONE, S.A.
73	3D	MAGIC CIRCUS	PASEO IXTAPA	S/N	IXTAPA	40880	31599	DIM900328LXA	DPIXTAPA MAGICO, S.A. DE C.V.
87	7TR	ACA JOE	PASEO IXTAPA	S/N	IXTAPA	40880	30302	ARO910717LHA	ACA ROPA S.A. DE C.V.

Tabla 1.3.3

SEGUNDA FORMA NORMAL

La prueba para determinar si una entidad está en Segunda Forma Normal es: que el valor de cualquier atributo que no es llave dependa de todos los atributos que forman la llave.

CONVERSION A LA SEGUNDA FORMA NORMAL (2FN) :

Para normalizar una entidad en la Segunda Forma, se crea una nueva entidad de los atributos que dependen parcialmente de una llave, siendo parte del identificador de esta nueva entidad el atributo del cual depende para mantenerla relacionada con la original.

Siguiendo con el ejemplo anterior y aplicando la segunda forma normal, ahora obtendremos un segundo catálogo denominado FOLIOS como se muestra en la tabla 1.3.4, la cual contiene todos los atributos que dependen parcialmente de una llave.

En la tabla 1.3.5 se muestra como queda al final ya normalizada (con las 2 primeras normas) la relación INF_TUR.

NUM_FOLIO	NOM_COM	CALLE	NUMERO	COLONIA	C.P.	TEL	R.F.C	RAZON_SOC
12H	HOTEL DORADO PACIFICO	PASEO IXTAPA	S/N	IXTAPA	40880	32025	OHD800408UR0	OPERADORA HOTELERA DORADO
5H	IXTAPA PALACE RESORT	PASEO DE LAS GARZAS ESQ. PASEO DEL RINCON	S/N	RESIDENCIAL IXTAPA	40880	31359	APT881110R17	AMERICA PROMOTORA TURISTICA S.A. DE C.V.
23H	OMNI HOTEL	PASEO DEL PALOMAR	S/N	ZONA HOTELERA	40880	30003	PIS8705286KJ	PROPULSORA IXTAPA SUR, S.A. DE C.V.
10R	PIZZERIA MAMMA NORMA	PASEO IXTAPA	LOC. 8	IXTAPA	40880	30274	DIME420722DY2	DIBILDOX NIETO ENRIQUE
4R	LAS MARGARITAS	PASEO IXTAPA	S/N	CENTRO	40880	30910	AIGS540602838	ALUA GONZALEZ FRANCISCO
8R	DA-BAFFONE	PASEO IXTAPA	S/N	IXTAPA	40880	31122	RDA8201078KG	RESTAURANTE DA-BAFONE, S.A.
30	MAGIC CIRCUS	PASEO IXTAPA	S/N	IXTAPA	40880	31589	OIM900329LXA	OPIXTAPA MAGICO, S.A. DE C.V.
7TR	ACA JOE	PASEO IXTAPA	S/N	IXTAPA	40880	30302	ARO910717LHA	ACA ROPA S.A. DE C.V.

Tabla 1.3.4

INF_TUR

CVE_GIRO	NUM_FOLIO
56	12H
56	5H
56	23H
15	10R
15	4R
15	8R
73	3D
87	7TR

Tabla 1.3.5

TERCERA FORMA NORMAL

Se dice que una entidad se encuentra en Tercera Forma Normal si el valor de cada atributo depende de toda la llave y no de cualquier otro que no lo sea.

En esta forma normal, se buscan los atributos que están dependiendo de otro que no es una llave.

CONVERSION A LA TERCERA FORMA NORMAL (3FN) :

Para poner una entidad en Tercera Forma Normal, se crea una entidad con los atributos que no dependen de ningún atributo que forma la llave, siendo el identificador de la nueva entidad el atributo del cual era dependiente.

En el ejemplo, se observa que no existe este tipo de dependencia, por lo que se dice que cumple también con la tercera forma normal.

CAPITULO II

PLANTEAMIENTO DE LA PROBLEMATICA Y PROPUESTA DE SOLUCIÓN

2.1 Antecedentes.

Un *requerimiento* es una característica que debe incluirse en un nuevo sistema. Esta puede ser la inclusión de determinada forma para capturar o procesar datos, producir información, controlar una actividad de la empresa o brindar soporte a la gerencia. Es así como la determinación de requerimientos vincula el estudio de un sistema existente con la recopilación de detalles relacionados con él.

Dado que los analistas de sistemas no trabajan como gerentes o empleados en los departamentos de usuarios (como mercadotécnia, compras, producción o contabilidad), no tienen los mismos conocimientos, hechos y detalles que los usuarios y gerentes de esas áreas.

Por consiguiente, el primer paso del analista es comprender la situación. Ciertos tipos de requerimientos son tan fundamentales que son comunes en casi todas las situaciones. Dar respuesta a un grupo específico de preguntas, será de gran ayuda para comprender los requerimientos básicos.

También existe otra clase de requerimientos que depende de si el sistema está orientado hacia transacciones, toma de decisiones o se extiende por varios departamentos. Por ejemplo, la necesidad de informar al gerente de inventarios de un pedido insólitamente

grande que está por llegar subraya la importancia de eslabonar los departamentos de ventas, compras y almacén.

Esta etapa es la base en el planteamiento del diseño del sistema ya que la aceptación del sistema a desarrollar depende en un porcentaje muy valioso de una buena identificación de los requerimientos del usuario.

2.2 Requerimientos del usuario.

A continuación mencionaremos y describiremos algunos de los requerimientos del usuario para el desarrollo del sistema que nos hemos propuesto.

Los requerimientos específicos del usuario para el desarrollo del sistema son:

- Uso de un menú gráfico para el acceso a otros menús, utilizando para ello dos dispositivos, el mouse, y el teclado.

- Manejo de los mapas de las ciudades para la pronta localización de bienes y servicios, y lugares de recreación.

- Localización e información específica sobre un punto geográfico de interés deseado, utilizando tres radios de acción de diferente longitud.

- Información sobre bienes y servicios.

- Obtener información física del lugar e información económica (precios de las habitaciones de los hoteles, categoría de ellos, etc.).

- Obtención de un formato de quejas donde el interesado pueda indicar algún bien o servicio que no haya encontrado, o alguna característica que le gustaría que el sistema tuviera.

- Explicación breve del manejo del sistema.

- Obtener un sistema amigable y confiable.

- Sistema de tamaño pequeño (10 MB max.).

- El sistema funcione correctamente en cualquier PC (386 en adelante).

En conclusión los puntos antes mencionados nos dan un panorama de las necesidades que tiene el usuario para con el sistema (figura 2.2.1).

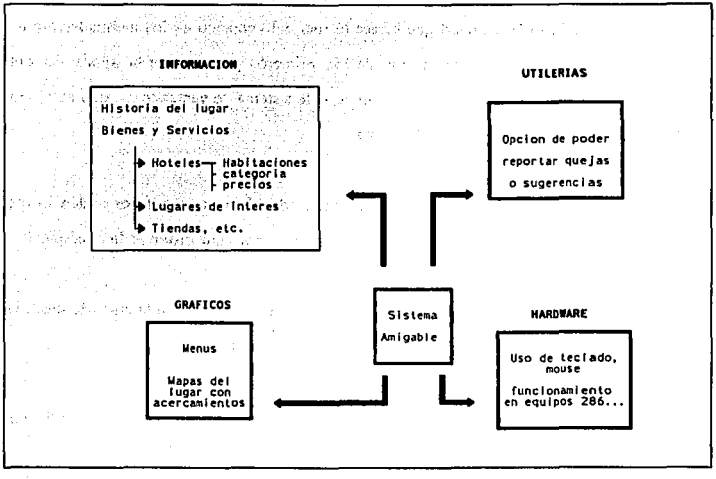


Fig. 2.2.1 Requerimientos del usuario.

Uno de los primeros puntos a tratar sería el del desarrollo de un sistema amigable, significando que no deberá causar problemas el uso del sistema para el usuario.

Para tomar este punto podríamos decir que el sistema deberá estar planteado para usuarios con poco conocimiento en computación a tal vez nulo, al tener esta aseveración en la línea como premisa del sistema, podremos garantizar que el uso del sistema no causará ningún problema para el usuario y tendremos la seguridad de que será una herramienta útil para el mismo.

En la actualidad la facilidad que ofrece el uso del ventaneo en los menús hace más agradable el manejo de los sistemas, uno de los primeros pioneros en la aplicación del ventaneo fue la compañía Microsoft Corporation, este sistema de ventaneo ha sido aplicado por la mayoría de los desarrolladores de software.

Los menús se deberán desarrollar en forma de bloques para tener orden en la selección de las opciones que nos proporciona el sistema y por ende orden en la operación.

Los colores también forman parte importante ya que de hacer una buena selección, el usuario no presentará síntomas de cansancio por el uso del mismo.

Las pantallas deberán ser legibles o sea que los conceptos deberán ser claros y entendibles por el usuario.

La ayuda en el sistema en forma inmediata ofrecerá una gran seguridad en el usuario, sin tener que distraerse en busca de manuales muy voluminosos y complejos.

El sistema debe tener un acceso sencillo sin tener que hacer muchas preguntas al usuario, salvo las necesarias para poder operarlo en una forma clara y concisa, ya que algunos programas tienen la particularidad de poner menús muy complejos y aunque dispongamos de ayuda de pantalla no debe de utilizarse para cada paso a ejecutar.

El sistema deberá tener capacidad de mantenimiento sencillo para futuras actualizaciones, ya que la información que se va a utilizar varía continuamente. Esto proporcionará al usuario información actualizada y veraz.

Es útil ver la determinación de requerimientos a través de tres grandes actividades: anticipación, investigación y especificación de requerimientos.

- ***Anticipación de requerimientos.*** Prever las características del sistema con base en la experiencia previa. Esto puede llevar al analista a investigar áreas y aspectos que de otra forma no serían tomados en cuenta.

- ***Investigación de requerimientos.*** Estudio y documentación del sistema actual utilizando para ello técnicas para hallar hechos, análisis de flujo de datos y análisis de decisión.

- ***Especificación de requerimientos.*** Análisis de los datos que describen el sistema para determinar qué tan bueno es su desempeño, qué requerimientosse deben satisfacer y las estrategias para alcanzarlos.

En nuestro caso, la información que requiere el turista, la mayoría de las veces es repetitiva, como es el caso de información de hoteles disponibles, lugares de interés, etc. y en otras constante, como lo es la historia del lugar, por lo tanto, un sistema de información computacional será de gran utilidad para la rápida y eficiente obtención de la información.

Como el sistema será usado tanto en agencias turísticas como en kioscos informativos en zonas de concentración turística, el sistema debe de contar tanto con información básica para rápida consulta como con información específica del lugar.

En este caso se debe tomar especial atención en el sentido de que debe de ser un sistema muy amigable, ya que, la mayoría de los usuarios serán personas que nunca ó muy pocas veces traten con una computadora. Así que las instrucciones y comandos deben ser sencillos de usar.

Un ambiente gráfico para un sistema turístico es de gran ayuda porque facilita la ubicación y la visualización de los sitios de interés.

No debemos perder de vista que el éxito del uso del sistema depende en gran parte de la interactividad que tenga con el usuario.

Una de las características más importantes para los usuarios de un sistema de información es la salida que éste produce. Si la salida no tiene calidad, entonces todo el sistema puede parecer a los usuarios tan poco necesario que evitarán usarlo, y esto posiblemente se convierta en causa del fracaso.

2.3 Estrategia de Solución

Este sistema se enfoca a la problematica a la que se enfrenta un turista al llegar a un sitio desconocido, en donde el tipo de información que requiere principalmente, es geográfica, económica, cultural y recreativa del lugar.

Por ejemplo, un turista que llega al puerto, puede hacer uso de este sistema si requiere información sobre las categorías de los hóteles que el puerto ofrece, el cual además le ofrecerá tanto la información económica, como sería el precio de las habitaciones así como la ubicación geográfica de los hoteles.

Por lo tanto, el principal objetivo del sistema es proporcionar información turística del puerto de Ixtapa-Zihuatanejo localizada en un mapa geográfico, y para ello primeramente se tiene que recabar toda la información correspondiente del lugar.

Para lograr lo anterior se buscarán fuentes de información lo más amplias posibles, como lo es el directorio telefónico de la ciudad, después se elaborarán diversos cuestionarios con preguntas que vayan de acuerdo con los elementos que se reúnan del directorio telefónico, también se consultarán revistas y folletos con publicidad turística del lugar para terminar de recabar la información, pensando conseguir éstas en las oficinas de la secretaría de turismo y en algunas agencias de viajes.

Una vez realizado lo anterior se iniciara el diseño de las bases de datos en las cuales se vaciará toda la información recabada en los cuestionarios correspondientes.

Después se iniciará el análisis y diseño del sistema que a grandes rasgos consistirá en:

Conseguir dos mapas respectivos de las ciudades para que entren en contacto directo con los diferentes bancos de datos que el sistema maneja, tales como los bancos de información económica, cultural y turística del puerto.

Los mapas se tomarán como una colección de puntos que se colocarán dentro de una estructuración determinada.

Para tal estructura se retomará el concepto de matriz con renglones y columnas en donde para cada pareja de puntos (X,Y), será una coordenada geográfica.

Las coordenadas geográficas estarán relacionadas con los diferentes bancos de datos y éstos a su vez lo estarán con un punto en particular del mapa.

Los bancos de datos también estarán relacionados entre si. Por ejemplo, si se señala un punto en el mapa, éste desplegará la información sobre el nombre del lugar seleccionado, el que a su vez estará en relación con la información económica, cultural y turística correspondiente, también se dará el caso contrario si se elige el nombre de un bien o servicio, el sistema se encargará de indicar la ubicación geográfica del lugar, mediante un punto luminoso sobre el mapa de la ciudad respectiva.

Como se puede observar se utilizará el concepto de función Uno a Uno, en donde un punto geográfico le corresponde un dato del banco de información y viceversa.

El sistema manejará tres radios de acción de diferente longitud 250, 500 y 1000 Kms para realizar los acercamientos sobre los mapas de las ciudades. Para lograr tal efecto se tomará el concepto de escalas, en donde a cada punto del mapa se le aplicará una "regla de tres" para hacer los aumentos y las disminuciones de los puntos geográficos.

Considerando que los mapas de las ciudades son mapas urbanos, es decir, son mapas que tienen definido el contorno de las calles y de las avenidas de la ciudad, a los cuales se le manejará como objetos que es un de puntos importantes de la programación orientada a objetos.

Después se realizarán los diversos módulos de programación los cuales estarán relacionados mediante diversas pantallas de consulta de información

Y por último se realizará un instructivo explicativo del uso del presente sistema.

2.3.1 PLAN DE TRABAJO

Considerando que el objetivo del sistema es proporcionar información turística del puerto de Iztapa-Zihuatanejo. Para elaborar el plan de trabajo primeramente se penso, en los aspectos más importantes que afectan a un turista, que son principalmente el conocimiento de los bienes y servicios que presta el lugar.

Para obtener información sobre ellos, se recurrió al directorio telefónico y a la sección amarilla del puerto, de ahí seleccionamos por orden alfabético, los bienes y servicios que a nuestro juicio consideramos más importantes:

Abarrotes y misceláneas
Aduanas
Agencias de Viaje
Angeles verdes
Arrendadora de vehículos
Arrendadora de vehículos acuáticos
Artesanías
Artículos deportivos
Artículos Fotográficos y revelado
Atractivos turísticos varios
Bancos
Cajeros Automáticos
Capitanía de puerto
Casas de bolsa
Centros comerciales
Centros nocturnos con variedad

Cerrajerías

Club de Golf

Club de tenis

Consulados

Cruz roja

Deportes acuáticos

Discoteques

Eléctrico Automotriz

Escuelas

Farmacias, perfumerías, boticas, droguerías

Gasolinerias

Gimnasios

Hospitales

Hoteles

Laboratorios Clínicos

Lineas de autobuses

Marinas

Mecánico Automotriz

Médicos

Mercados

Oficinas de lineas aéreas

Oficinas de migración

Playas

Playa Varadero

Playa Quieta

Playa Don Rodrigo

Playa Las Cuatas

Playa el Palmar

Playa Vista hermosa

Isla grande o de Ixtapa

Playa Cuachalalate

Playa del Coral

Playa Larga

Playa Linda

Policía y tránsito

Procuraduría de justicia

Procuraduría del consumidor

Restaurantes

Servicios postales

Servicios telefónicos

Servicios telegráficos

Tabaquerías

Tiendas, Autoservicio y Deptales

Tiendas de ropa casual

Tiendas de ropa formal

Zapaterías

2.3.2 Recopilación de la Información

Una vez que se tuvieron los elementos sobre los cuales buscar información, se elaboraron diversos cuestionarios, que contemplarán en forma general preguntas tanto económicas, culturales, recreativas y gráficas.

La información especifica sobre cada elemento se consiguió, consultando diversas, revista y folletos con publicidad turística, que nos fueron proporcionados por SECTUR (Secretaría de Turismo). Para la obtención de los mapas de las ciudades se consiguieron de la revista "Conozca México" en la cual por cada estado de la República se expide una colección de mapas de las ciudades que comprende, indicando las principales vías de comunicación de ellas.

Para buscar la información de los elementos en cuestión se formularon diversas preguntas dispuestas en cuatro cuestionarios que variaron según el tipo de bien o servicio que trataron, por ejemplo para recabar la información sobre los lugares de recreación, restaurantes, tiendas, centros comerciales, hoteles y otros servicios, se utilizaron los siguientes formatos:

Cuestionario de atractivos turísticos

Sitio de Atracción Turística : _____

Ubicación : _____

Descripción de lugar : _____

Información cultural : _____

_____

_____

Cuestionario de Tiendas y Departamentales

Nombre de la Tienda o Centro comercial: _____

Dirección : _____

Categoría : _____

Teléfonos: _____

Atención : _____

Descripción de la tienda y de lo que vende (Incluyendo marcas):

_____

_____

_____

_____

Cuestionario de Restaurantes

Nombre del restaurante : _____

Dirección : _____

Teléfonos: _____

Categoría : _____

Atención : _____

Horarios de servicio : _____

Tipo de comida : _____

_____

Cuestionario de Hoteles

Nombre del Hotel : _____

Dirección : _____

Categoría : _____

R.F.C. : _____

Propietario : _____

Servicios que presta : _____

_____

Teléfonos : _____

Horarios : _____

Tarifas por habitación o por persona : _____

_____

Cuestionario otros servicios

Nombre del bien o servicio : _____

Dirección : _____

Teléfonos : _____

Horarios de servicio : _____

Atención: _____

Descripción de los servicios que presta : _____

_____

_____

Tarifas/Servicio : _____

2.3.3 Clasificación de la Información

Una vez reunida la mayor cantidad de información económica, cultural y recreativa se optó por clasificarla en tres listas:

Lista de Bienes

Abarrotes y misceláneas

Artesanías

Artículos Fotográficos y revelado

Artículos deportivos

Centros comerciales

Farmacias, perfumerías, boticas, droguerías

Tabaquerías

Tiendas, Autoservicio y Deptales

Tiendas de ropa casual

Tiendas de ropa formal

Zapaterías

Lista de Servicios

Aduanas

Agencias de Viaje

Angeles verdes

Arrendadora de vehículos

Arrendadora de vehículos acuáticos

Atractivos turísticos varios

Bancos

Cajeros Automáticos

Capitanía de puerto

Casas de bolsa

Centros nocturnos con variedad

Cerrajerías

Club de Golf

Club de tenis

Consulados

Cruz roja

Deportes acuáticos

Discoteques

Eléctrico Automotriz

Escuelas

Gasolinerias

Gimnasios

Hospitales

Hoteles

Laboratorios Clínicos

Lineas de autobuses

Marinas

Mecánico Automotriz

Médicos

Mercados

Oficinas de lineas aéreas

Oficinas de migración

Policía y tránsito

Procuraduría de justicia

Procuraduría del consumidor

Restaurantes

Servicios postales

Servicios telefónicos

Servicios telegráficos

Lista de Atractivos Turísticos

Playa Varadero

Playa Quieta

Playa Don Rodrigo

Playa Las Cuatas

Playa el Palmar

Playa Vista hermosa

Isla grande o de Ixtapa

Playa Cuachalalate

2.4 ANALISIS.

2.4.1. ESPECIFICACIONES DEL ANALISIS.

El objetivo primario del análisis de datos es el de proporcionar las bases para el diseño de una base de datos, y un enfoque disciplinado al catalogar los datos existentes en términos de las entidades y las relaciones que presentan. Sin tal entendimiento de la parte de la organización que está siendo analizada, es más difícil establecer si la base de datos será instalada eficientemente. El análisis de datos proporciona un medio muy efectivo para comunicarse con usuarios que no son profesionales en el mundo computacional, ya que se dedica solamente a aquellos en que el usuario está familiarizado.

La fase de análisis es referida algunas veces como la formulación y análisis de requerimientos, lo cual involucra el establecimiento de los objetivos y la documentación de estos requerimientos.

El análisis de datos debe ser realizado por un equipo que contenga a los usuarios, el departamento de desarrollo de sistemas, y el grupo de administración de datos.

El equipo de análisis de datos puede no intervenir en la fase de análisis de requerimientos, si esta fase está limitada a entrevistas personales con diferentes niveles de gerencia y empleados claves en el procesamiento de bienes, servicios y organización de datos. El resultado de tales entrevistas deben ser los diagramas de proceso. Los objetivos específicos y los requerimientos de la base de datos deben obtenerse de un nivel más alto de la organización.

El equipo de análisis de datos debe identificar las entidades que son necesarias para resolver el problema definido por el usuario. Durante las etapas iniciales del análisis de datos es posible que no se conozcan todos los atributos de todas las entidades. Sin embargo, a medida que estos se determinen, el equipo debe documentar la definición del atributo y su papel en un diccionario de datos apropiado.

Elaboración del Modelo de Entidad

Durante la fase de análisis se determinan las entidades mayores y sus relaciones. Estas entidades y sus relaciones se representan en modelos llamados Modelos de Entidad. El modelo es un diagrama representativo de la relación entre las clases de entidades.

La representación nos permite incluir solo aquellas entidades que se requieren para resolver un problema particular del procesamiento de datos. El modelo de entidad es esencialmente una vista del mundo real de los datos de la organización en términos de entidades atributos y relaciones.

Durante la fase de modelaje se definen las clases y relaciones de entidad más significativas, sin embargo el modelo deberá ser revisado, modificado o extendido como

resultado del conocimiento sobre las nuevas entidades que se descubran. El modelo se usa para:

- Reducir la redundancia en las relaciones.
- Determinar cuales entidades son significativas al modelo y a los requerimientos del usuario
- Resolver las relaciones no binarias entre entidades.

Etapas en la integración de los Modelos de Entidad.

Las etapas requeridas para integrar los modelos de entidad son las siguientes:

- Identificar cada sinónimo u homónimo en los diferentes modelos. Esta tarea es más fácil si se usa un diccionario de datos. Los componentes con homónimos deben ser renombrados. Los componentes con sinónimos deben usar el mismo nombre.

- Los modelos de entidad para dos áreas de datos se integran superponiendo los tipos de entidad que sean idénticos o similares en los diferentes modelos de entidad, esto puede incrementar el número total de atributos del tipo de entidad, ya que las entidades idénticas pueden usar diferentes atributos.

- Como resultado de la integración, el modelo compuesto de entidad puede contener relaciones redundantes. Esta redundancia puede ser eliminada, sin embargo, determinar las relaciones que son directamente significativas y cuales son redundantes puede presentar dificultades que pueden ser resueltas solamente a través de un buen entendimiento del ambiente.

Derivación de los modelos de Entidad de Modelos tradicionales.

En realidad no existen reglas para esta derivación. La distribución de datos, con los cuales los archivos lógicos fueron construidos puede no ser la misma que se requiere para las clases de entidad respectivas. Sin embargo, las siguientes reglas pueden seguirse cuando se trate de convertir archivos planos al modelo de entidad de la aplicación.

- Listar todos los tipos de archivos en las programas relevantes.
- Listar todos los registros físicos en los archivos.
- Listar todos los datos en los registros.

Eliminar las redundancias e inconsistencias en los datos y los registros lógicos.

  - Listar todas las combinaciones posibles de clases de entidad de los registros lógicos. El nombre del registro es un indicador de la clase de entidad.
  - Listar todos los códigos en los registros que pueden dar las relaciones del modelo de entidad.

- Hacer un análisis preliminar de los datos .
- Acomodar los atributos con sus respectivas clases de entidad.

Este procedimiento debe dar como resultado un modelo de entidad que servirá como estructura para hacer otras revisiones, las cuales serán necesarias para un análisis más detallado de los datos.

Combinación de los de los Modelos de Entidad

Al convertir bases de datos existentes en su equivalente de modelo entidad, el diseñador puede llegar a diferentes modelos dependiendo de los programas o aplicaciones de los cuales los modelos fueron derivados. Se debe intentar remover las redundancias e inconsistencias al combinar los modelos de los varios programas para quedarse con un modelo integrado. Esta combinación permitirá al diseñador determinar lo siguiente:

- ¿Cuáles son las clases de entidad y los atributos comunes?
- Las inconsistencias en los nombres y uso de los atributos. Estas inconsistencias existen cuando se ve que dos entidades con diferentes nombres son la misma entidad.
- La eficacia del modelo en términos de satisfacer las necesidades del usuario.
- Si algunos atributos considerados en una entidad son realmente miembros de otra clase de entidad o de nuevas clases de entidad.
- La existencia de inconsistencias en las relaciones.

Este modelo combinado puede ser usado como la estructura para revisiones posteriores para llegar a un modelo de entidad integrado que sirva a un área de datos mayor, en lugar de varios modelos pequeños orientados a unas cuantas aplicaciones.

Agrupamiento de Clases de Entidades.

El agrupamiento de clases de entidades puede hacerse en la etapa de diseño físico o lógico. En la etapa del diseño físico esto puede hacerse basándose en consideraciones de desempeño. Las clases de entidades pueden juntarse o separarse en diferentes bases físicas dependiendo de los requerimientos de acceso.

El agrupamiento lógico de clases de entidades depende de la naturaleza de los datos y sus estructuras. Es necesario, pero no suficiente, decir que los atributos se agrupan en una clase de entidad porque estos identifican y describen la clase de entidad se hace enteramente para satisfacer los siguientes requisitos:

- El área que sirven los datos o de la cual se originan.
- La estructura de datos inherente.
- La vista del usuario.
- Los usos de los datos.
- Las consultas que se hacen de los datos.

- Las necesidades de procesamiento del usuario.

Diseño del Esquema logico y vistas de aplicación.

Las vistas de la aplicación pueden definirse como el conjunto de datos que son requeridos por una aplicación particular para satisfacer una necesidad específica de procesamiento de datos. Tenemos vistas de aplicación de:

- Una clase de entidad.
- Agrupamientos de clases de entidades.
- Agrupamientos de clases de entidad y bases de datos físicas.
- Agrupamiento de bases de datos físicas.

El esquema lógico puede ser definido como el mapeo del modelo de entidad en la construcción proporcionada por el manejador de la base de datos. En general el esquema lógico indica como se almacenará y accesará el modelo. En el diseño del esquema lógico tal vez sea necesario hacer algunos cambios al modelo para adecuarse al DBMS. El modelo de entidad tiene las siguientes características:

- Es una presentación de la vista de datos del mundo real.
- Proporciona las bases para continuar con el análisis y diseño de la base de datos.

- No está restringido a ningún sistema de manejador de bases de datos (DBMS).

- No es implementable directamente.

- Una estructura estable de referencia a la cual se puede agregar nuevas entidades, tributos y relaciones si la organización así lo requiere.

Transformación del Esquema logico a una Base de Datos Fisica.

Los detalles de esta fase dependen de las características del manejador escogido para el diseño.

El esquema lógico debe hacerse de manera que lo único que se deje a los diseñadores de la base física sea la selección de los métodos de acceso y los índices secundarios. Hubbard indica que deben seguirse las siguientes reglas durante el diseño físico:

- Cada clase de entidad debe ser tratada como una base de datos física.

- Si dos clases de entidades comparten una relación entre un atributo y la llave primaria por lo menos, entonces las estructuras deben consistir en dos bases de datos físicas con conexión virtual o física entre ellas.

- Las relaciones padre - hijo deben ser definidas en una sola base de datos física.

- Los segmentos que se usen frecuentemente deben ser mantenidos lo más cercanamente

posible a su raíz.

- Reducir el tiempo de búsqueda de grupos de datos grandes usando índices secundarios.
- Los segmentos de tamaños variables no deben ser colocados en el mismo grupo de datos si se hacen inserciones y borrados frecuentes.

En los siguientes puntos se aplicará lo anterior para nuestro diseño del sistema.

2.4.2 DIAGRAMA DE DESCOMPOSICION FUNCIONAL.

La estrategia de flujo de datos muestra el empleo de éstos en forma de gráfica. Las herramientas utilizadas al seguir esta estrategia muestran todoas las características escenciales del sistema y la forma en que se ajustan entre sí. Puede ser difícil comprender en su totalidad un proceso de la empresa si se emplea para ello sólo una descripción verbal; las herramientas para el flujo de datos ayudan a ilustrar los componentes esenciales de un sistema junto con sus interacciones.

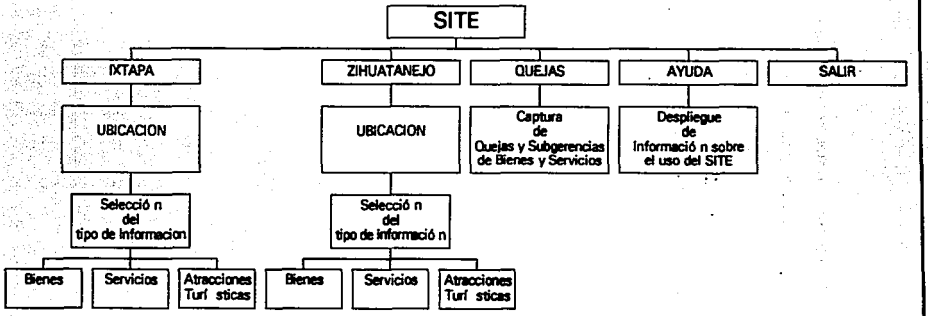
El diagrama de descomposición funcional muestra las partes fundamentales de la estructura del sistema. En el se muestran las diferentes entidades y como podemos llegar a ellas de acuerdo al flujo de información ya sea manual o automatizada, incluyendo procesos y retroceso de información.

Los usuarios y otras de la empresa que forman parte del proceso bajo estudio comprenden con facilidad anotaciones sencillas. Por consiguiente, los analistas pueden

trabajar con los usuarios y lograr que participen en el estudio de los diagramas del flujo de datos. Los usuarios pueden hacer sugerencias para modificar los diagramas con la finalidad de describir la actividad con mayor exactitud. Asi mismo pueden examinar las gráfica y reconocer con rapidez problemas; esto permite efectuar las correcciones necesarias antes de que comiencen otras tareas relacionadas con el diseño.

En la figura 2.4.2 se muestra el diagrama de descomposición de nuestro sistema. en él se pueden apreciar los diferentes módulos de funcionamiento y su flujo de datos.

Diagrama de Descomposición Funcional



En la primera parte podemos ver el módulo de entrada de variables en la cual el usuario escogerá la ciudad, el radio de definición de acción y el punto de interés; teniendo esta información el sistema desplegará las listas de información de bienes y servicios disponibles. De estas listas el usuario escogerá el bien o servicio de su interés. El sistema entonces desplegará la información especifica del lugar textual y graficamente. El usuario podrá tener la opción de imprimer la información obtenida para luego volver a solicitar información o salir del sistema.

2.5 OPCIONES DE SOLUCION.

CARACTERISTICAS, SELECCION DEL SOFTWARE DE BASE DE DATOS Y PAQUETE DE DESARROLLO DEL SISTEMA.

En esta época sobresalen los sistemas para el manejo de base de datos (DBMS), los archiveros de la edad electrónica. Si se busca entre el papeleo de cualquier organización en el mundo, seguramente se encontrará una gran carga de documentos solicitando el manejo de una base de datos: la lista de clientes para la sala de alquiler de videos, el inventario de cassettes en una tienda de música o los registros de personal de una empresa. De igual manera sobresalen los paquetes de computación que proporcionan las herramientas necesarias para el desarrollo de sistemas más amigables al usuario mediante un ámbiente gráfico.

A continuación se presenta una muestra típica de paquetes de bases de datos con capacidades semejantes. Es importante mencionar que lo que se va a apreciar en las descripciones no es necesariamente todo lo que puede obtenerse. En muchos casos, se pueden aplicar las capacidades básicas de las bases de datos. Además de mencionar dos paquetes de programación orientada a objetos para el desarrollo de sistemas mediante un ámbiente visual.

2.5.1 MANEJADORES DE BASES DE DATOS

CLIPPER.

Este paquete para el desarrollo de bases de datos es, definitivamente para los programadores. Si bien carece de algunas de las excelentes funciones de generación de códigos que tienen los demás paquetes, Clipper ofrece una riqueza de armas y capacidad que los programadores necesitan.

Dos de estas armas son un generador de reportes (RL) y una función (DBU) para la creación y manejo de los archivos de la base de datos, escritos en el propio lenguaje de programación de Clipper, y se incluye el código fuente, el cual puede utilizarse como referencia, o modificarlo para añadirlo a las aplicaciones de la base de datos.

La pantalla de DBU enlista las opciones a través de su parte superior, junto con las teclas de funciones asignadas. El resto de la pantalla está dedicado a una representación visual del panorama de la base de datos activa, el cual consiste en una columna partida en tres secciones. El primer grupo exhibe el nombre de la base de datos activa. Si al invocar el programa DBU se invoca un argumento, aparecerá aquí la base de datos o la vista especificada. En la siguiente sección aparecerán los índices activos relacionados con la base de datos activa, y el grupo al fondo muestra los nombres de los campos para la base de datos activa.

El generador de reportes, RL, no es tan flexible, no soporta un bosquejo WYSIWYG del reporte y tampoco permite ver la salida en forma preliminar conforme se vaya trabajando. La verdadera potencia de Clipper se basa en su codificación.

El compilador del programa es muy rápido además de tener algunas funciones muy interesantes, dentro de estas se destacan las siguientes: soporta los llamados bloques de código, pedacitos de código ejecutable que se pueden almacenar como variables, o pasar a otros programas como argumentos, para ejecutar un bloque de código, se utiliza una función EVAL( ); otra función útil es el uso hecho por Clipper de los archivos cerrados para hacer el seguimiento, a fin de determinar cuales archivos en un programa dependen de otros para operar correctamente y quedar al corriente.

Estando instalado este sistema, se puede invocar la función MiMake para llevar a cabo aquellas operaciones de compilación y enlace que se necesiten y mantener todos los archivos sincronizados. Clipper también soporta las funciones para leer los archivos binarios de DOS y escribir en ellos. También se aprecia el depurador de Clipper, el cual permite analizar el funcionamiento de código del programa, ejecutar comandos y revisar el estado de algunas variables, nombres de campos o expresiones en particular. Una ventanilla de estado enlista las bases de datos abiertas en todas las áreas de trabajo activas, como los valores de todos los comandos Set.

DBASE.

Ya pasaron a la historia los días cuando Dbase establecía las normas del manejo de las bases de datos en DOS. Dbase llegó a la cumbre principalmente por la fuerza de su poderoso lenguaje de programación, los retrasos abrieron la puerta para los paquetes con la misma potencia de programación, escondida bajo interfaces más sencillas, Dbase está llegando a ser más fácil de utilizar cada día.

El programa ya ofrece interfaces, manejado por menú que se conoce como control. Aunque el centro de control representa un mejoramiento para los usuarios intimidados por un sólo punto en una pantalla en blanco, la simple añadidura de menús accesibles no convierte esto en un producto de uso agradable.

Primero que nada, el paquete no soporta el ratón, es necesario valerse de las combinaciones de teclas Alt, teclas del cursor y teclas de las funciones para poder andar en este programa.

El centro de control no ha superado la necesidad de requerir una confirmación cada vez que salga de una operación y entre a otra. Aunque haya conservado los datos en la ventanilla y presione la tecla de Escape para regresar al menú principal, el centro de control le preguntará si realmente quiere hacer eso.

Al igual que otros muchos paquetes, inician con un bosquejo escueto de la estructura de la base de datos activa.

Se pueden añadir múltiples bases de datos a la consulta desde el menú, y permutar entre estas bases de datos, utilizando las teclas de las funciones. Apuntando a los nombres de campo en el bosquejo se podrán crear enlaces entre las bases de datos; especificar campos para ser incluidos en la consulta; organizar la base de datos con campos especificos y métodos de clasificación; y fijar las condiciones para la consulta. Una consulta también podrá desatar la actualización de múltiples registros; los resultados podrán ser conservados como vista o como una nueva base de datos.

El generador de formas no soporta la entrada a la base de datos sin algunas manipulaciones del código.

FOXPRO.

La interface de FoxPro incluye menús presentados junto con una ventanilla de comandos conveniente, para utilizar los menús o escribir los comandos sin que ninguno de los elementos de la interface interfieran con otro. Su soporte al ratón es el mejor de todos los paquetes basados en caracteres. Asimismo, la ventanilla de comandos mantienen una historia corrida de las intrucciones (al igual que Dbase III plus), esto facilita la repetición de los comandos utilizados a través de una sesión. También se puede selecionar parte de la historia de los comandos, y anexarla a sus aplicaciones.

La ventana presenta gráficamente todas las áreas de trabajo disponibles; se puede seleccionar un área de trabajo disponible, se puede seleccionar un área de trabajo y abrir en ella una base de datos, accionando un botón de comando. FoxPro ofrece una caja de diálogo con una lista de los campos principales.

El constructor de pantalla empieza como pantalla en blanco, en la cual se puede capturar el texto y colocar los campos, también se pueden crear botones de comando, casillas, marcar con "palomas", botones de "radio" y listas de extraer. Se pueden agregar pedacitos de código a cualquier objeto, incluso a los campos.

Desde la pantalla de establecimiento, se puede ejecutar código antes y después del programa de la pantalla de captura, cuando se genere el código para la pantalla diseñada, se pueden anexar otras pantallas a ella, lo cual ahorra tiempo una vez formulada una biblioteca de pantallas genéricas.

El funcionamiento de FoxPro es magnífico gracias, en parte, a su tecnología exclusiva de Rushmore, siendo su único inconveniente su voluminosa documentación, ya que es un problema localizar la información rápidamente, lo cual no obstante disminuye gracias a su fuerte función de ayuda en línea.

INFORMIX-SQL.

Informix SQL es cien por ciento una base de datos con lenguaje de consulta estructurado (SQL), el aspecto del programa es muy austero: no se encontrará con pantallas de colores múltiples con menús y ventanillas a la vista. Las pantallas del paquete, estilo Lotus, impulsadas por menús, automatizan las operaciones más significativas de la base de datos, tales como la creación de tablas, la definición y modificación de los campos, etc. Informix también incluye un generador de reportes, y un sistema para la ejecución de archivos de la definición de formas diseñadas.

En Informix se construye una forma, no moviendo un cursor sobre la pantalla con un ratón o teclas de fecha, sino escribiendo un archivo para la especificación de la forma, un tipo de definición de pantalla acompañado con instrucciones ejecutables.

Dicho archivo comprende cinco partes: una sección de tablas, que identifica cuáles tablas serán accedidas por la forma; una tabla de atribuciones, la cual describe cada campo exhibido por la forma, y una sección opcional de instrucciones, que define las operaciones que habrán de ser llevadas a cabo sobre los campos dentro de la forma.

Las secciones y tablas informan al sistema que debe presentar; la sección de pantalla indica en donde presentarlo, la sección de atribuciones indica como presentarlo y la sección de instrucciones le dice al sistema que hacer antes, mientras y después de presentarlo.

Se incluye con Informix-SQL varios paquetes de servicios: **BECHECK** verifica la integridad de los indices, si encuentra una discrepancia entre un archivo de datos y uno de sus indices, le permite reformar el indice; **DRLINK** y **DBLOAD** son de utilidad para el traslado de los datos entre Informix y el mundo exterior de archivos de Lotus 1-2-3, Dbase o ASCII; con **DBSCHEMA**, se pueden elaborar las intrucciones de SQL requeridas para crear una tabla o una base de datos.

PARADOX.

Una de las características más interesantes de **PARADOX** es su **velocidad**, destacándose su rapidisimo tiempo de respuesta para la lectura y edición de las tablas.

Paradox soporta los formatos de importación y exportación más importantes, pero existen problemas en la importación de información almacenada en el formato ASCII. Con Paradox, es necesario planear con cuidado antes de precipitarse a la construcción de una forma, aplicando esto principalmente con las formas que accesan a múltiples tablas. El diseño de formas con tablas múltiples exige la creación de una forma maestra, después, existe también la introducción, en dicha forma maestra de las formas incluidas en las otras tablas. Así que, para crear la forma maestra completa, deben diseñarse las formas que se incluirán primero.

El lenguaje para el manejo de la base de datos de PARADOX es PAL, aunque es erróneo representar a PAL como simplemente un DML. Lo que dificulta la programación con PAL es que maneja la transmisión de comandos a un robot sentado en un teclado operando Paradox; sin embargo, si simplemente no se quiere tener nada que ver con PAL, podrá instalarse el programador personal. Este programa es, esencialmente un constructor de aplicaciones que guía a través de la creación de una aplicación completa con menús y formas, ofreciendo el código PAL como su producto final.

2.5.2 MANEJADORES DE INTERFASES GRAFICAS (VISUAL)

VISUAL BASIC.

Visual Basic para Windows es un sistema de programación computarizado excitante. Visual Basic ha sido tremendamente aceptado, y Microsoft puso un poco de esfuerzo para integrarlo a la versión 3 de Windows. El poder, flexibilidad y velocidad de Visual Basic está ahora a la par con el lenguaje C unicamente, y en cuanto a productividad, Visual Basic es claramente superior.

Un programador medio necesita meses para tener una velocidad de programación para Windows usando el lenguaje C, pero con Visual Basic se pueden desarrollar aplicaciones para Windows con solo algunas horas de familiarizarse con el lenguaje tanto del paquete como de programación Basic. Se podra estar satisfecho al ver la primera aplicación desarrollada para Windows con un trabajo fácil y tan rápido como sea posible.

Visual Basic es altamente interactivo, manejando un lenguaje de programación, con el cuál permite aprender de ambas formas agradable y productivamente.

Visual Basic utiliza varias formas para sus presentaciones visuales de sistemas (En terminología de Visual Basic, una *forma* es una ventana asociada a controles, iconos, gráficos y código); la aplicación COLORBAR permite trazado de gráficas. La forma GETFILE permite al usuario seleccionar un archivo desde cualquier directorio en cualquier drive.

Mediante Visual Basic se puede generar un sistema que soporte la instalación y uso del ratón, para facilidad del propio sistema y del usuario; genera Cambios de Datos Dinamicos (DDE) para aplicaciones basadas en ventanas; trabaja con bases de datos en una interacción; y Objetos Ligados y Emprotados (OLE).

VISUAL C++.

El lenguaje visual C++ equivale a dos lenguajes Windows completos, en un solo paquete, los modelos de programación en Windows y los componentes de C++ trabajan juntos para realizar mejor las aplicaciones en un ambiente Windows.

Procesamiento de mensajes

Una gran diferencia entre los programas elaborados en C a los programas elaborados bajo Windows en C++ , es que los programas en lenguaje C llaman al sistema de operaciones para utilizar una salida, en cambio Windows utiliza las salidas a través de un mensaje.

Interface de dispositivos graficos (GDI)

Los programas elaborados en C son escritos directamente a la memoria de video y al puerto de impresión, la desventaja de esta técnica es que necesita un "driver" de software para la tarjeta de display y para la impresora. Windows y C++ introdujeron un nivel de abstracción llamado interface de dispositivos gráficos, para que así el programa no necesite conocer el tipo de tarjeta de display y que tipo de dispositivo de impresión se tiene. En lugar de localizar el hardware el programa llama a la función GDI la cual hace referencia a una estructura de dato llamada dispositivo de contexto, Windows convierte el dispositivo de contexto a un dispositivo físico y establece las instrucciones apropiadas de entrada/salida. El GDI es tan rápido como el acceso directo a video y permite diferentes aplicaciones mientras se escribe.

Programando las bases

Para programar los datos en lenguaje C se tienen que codificar los datos inicializandolos como constantes, o se debe dar datos separados para que pueda leer el programa. Cuando se programa en Windows y C++, se colocan los datos en un archivo fuente utilizando varios formatos, Windows une el archivo fuente dentro de un programa mediante un proceso llamado "binding". En el archivo fuente pueden estar incluidos mapas de bits, iconos, definiciones del menu. Con el editor de programas del Visual C++ se pueden editar la mayoría de los formatos.

Manejador de Memorias

La memoria convencional en el MS-DOS estaba limitada a 640 Kbytes limitando el tamaño de los programas, se pueden utilizar tecnicas para expander memoria y así permitir porgramas más largos pero ha la larga va ha ser insuficiente. Windows junto con C++ ofrece un manejador de memoria principal, el resultado es que la memoria ya no es un problema. Simplemente se coloca la memoria que se necesita y Windows se encarga de lo demás; ejecuta el programa, administra los recursos, automaticamente lo cambia al disco y despues lo coloca en memoria física, los cambios son tan buenos que la computadora va a tener mucha memoria disponible.

Uniones Dinamicas de Librerias (DLLs)

En los programas comunes todos los modulos de objetos en los programas se mantienen estáticos mientras se lleva a cabo el proceso de construcción. Windows y C++ permiten una interacción dinámica, lo cual quiere decir que librerías especialmente construidas pueden ser cargadas e interactuadas mientras se está corriendo un programa. La interacción dinamica incrementa la modularidad del programa debido a que se puede compilar y probar los DLLs separadamente. Algunas de las ventajas que estas librerías tienen es que se puede crear una propia, además de utilizar tanto iteracciones dinámicas como estáticas.

	C	Visual C++
Procesamiento de mensajes	Llaman al sistema de operación para utilizar una salida	Utiliza las salidas a traves de un mensaje
Programando las bases	Se utilizan los datos como constantes	Se une al archivo fuente dentro de un programa, en este se incluyen íconos, mapas de bit, etc.
Manejador de memoria	La memoria está limitada	Ofrece un manejador de memoria principal
Unlones dinámicas de librerias	Los programas se mantienen estáticos mientras se lleva a cabo el proceso de construcción	Se crean librerias especiales para permitir una interacción dinámica

Fig. 2.5.2.1 Ventajas de Visual C ++.

Cuando Microsoft cargo el C++, Windows estaba en el último estado de prueba, este nuevo sistema (C++) tiene avanzados sistemas de archivo con multilectura, con partes de seguridad, permite el acceso a las redes. Windows puede correr tanto el Windows basado en una aplicación de 16 bits, como en la nueva versión basada en una aplicación de 32 bits, la cual está desarrollandose, utilizando también la aplicación del sistema C++. Se pueden utilizar muchas herramientas del Visual C++ para convertir Windows a un estilo de programación más fácil.

El Worckbench de Visual es un huesped de Windows que directamente desciende del Microsoft Quick C para Windows. El Visual C++ esta disponible en dos versiones: La edición Estandar y la edición Profesional.

A.- Edición Estandar.- En la edición Estandar se provee de todos los componentes necesarios para producir tanto aplicaciones para Windows SDK y para aplicaciones del sistema.

B.- Edición Profesional.- En esta edición se tiene una impresión adicional de la documentación, una versión optimizada del compilador, algunas herramientas adicionales y la capacidad de producir aplicaciones del MS-DOS.

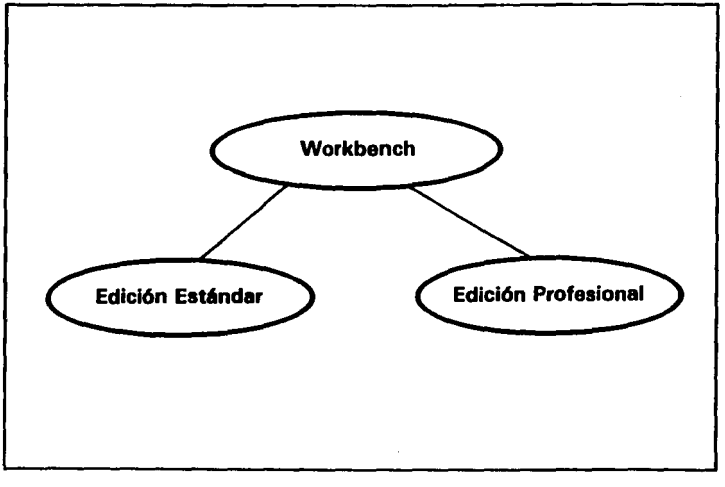


Fig. 2.5.2.2 Las dos versiones de Visual C++.

Con el original Windows SDK se tienen herramientas que editan dialogos, mapas de bits y fuentes; con el Visual C++ se puede utilizar el editor App Studio el cual incluye tanto menu de "lo que tu ves es lo que tu tienes" y una caja de dialogos que es

mucho muy superior a los dialogos elaborados en el Windows SDK, Otra ventaja es que cuando se utiliza App Studio se puede incertar los controles básicos del visual dentro de las cajones de dialogos, para despues poderlos conectar con el sistema C++. App Studio fué elaborado utilizando las herramientas de C ++, el cual tambien es capaz de elaborar sus propios recursos.

El Compilador C/C++

El compilador Visual C ++ puede procesar tanto los recursos del C como los recursos del C++, determina el lenguage observando el tipo de codigo que tiene la extensión del nombre del archivo. El compilador funciona también con el ANSI versión 2.1 y tiene una extensión de Microsoft adicional.

El Enlace

Para generar un archivo EXE, el Visual C++ procesa los archivos OBJ que el compilador produce. Para realizar el enlace se utiliza la librería de Microsoft para un mejor modelo de memoria.

Los recursos del Compilador

Los recursos del compilador del Visual C++ operan tanto en el modo de compilador como en el modo de enlace. En el modo de compilador un archivo en codigo ASCII del App Studio es compilado dentro de un archivo binario RES. En el modo de enlace un archivo RES es unido a un archivo ejecutable (EXE).

El Debugger

Si el programa trabaja la primera vez, no se necesita el debugger. El debugger del Visual C++ fue el primer huesped del Windows. Trabaja junto al Worckbench del Visual para asegurar con esto que los puntos de depuración sean salvados en disco.

Otra ventaja que tiene el Visual C++ es que con la Edición Profesional se puede tener un modo tipo caracter para el debugger en lugar de utilizar el debugger de Windows.

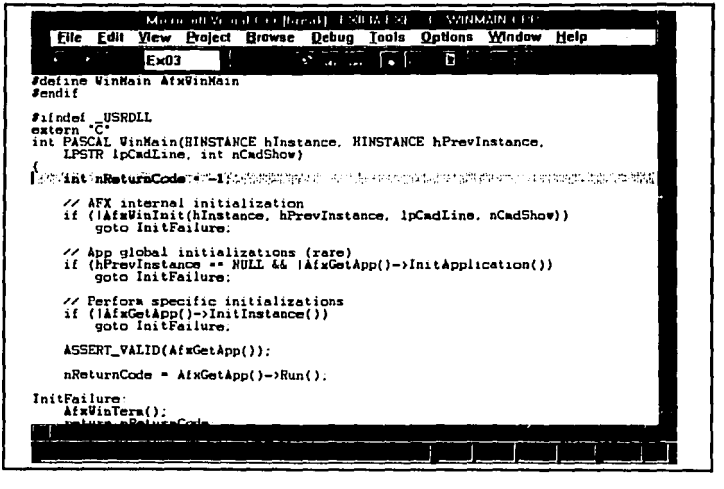


Fig. 2.5.2.3 La ventana del debugger de Visual C++.

Derivaciones de Clases

Visual C++ provee la habilidad de definir subclases, al describir como una nueva clase difiere de una vieja superclase. Reduce el tiempo del compilador y deja más espacio de memoria.

Los Recursos

Si se está escribiendo una aplicación de algún borrador, probablemente se debe de tener una buena memoria del dibujo original, de los codigos de archivo, de las clases, de las funciones, etc; pero si se está utilizando aplicaciones de otros recursos va ha llegar el momento que se necesite ayuda. El Visual C++ permite examinar una aplicación de una clase o de una función en lugar de examinarla de todo el archivo; esto se conoce con el nombre de "browser", el Browser tiene diferentes modos de examinado:

- Definiciones y referencias.- Se selecciona cualquier función, variable, tipo, macro o clase y despues se ve como está definida para usarla en cualquier proyecto.

- Llamado Grafico.- De un grupo de funciones se obtiene una representación gráfica de las funciones llamadas o las funciones que lo llamaron.

- Derivaciones de clases graficas/Base de clase gráficas.- Estas son gráficas de clase con diagramas jerárquicos. De una clase seleccionada se pueden ver las derivaciones de las clases. Se podrá controlar la expansión jerárquica con el mouse.

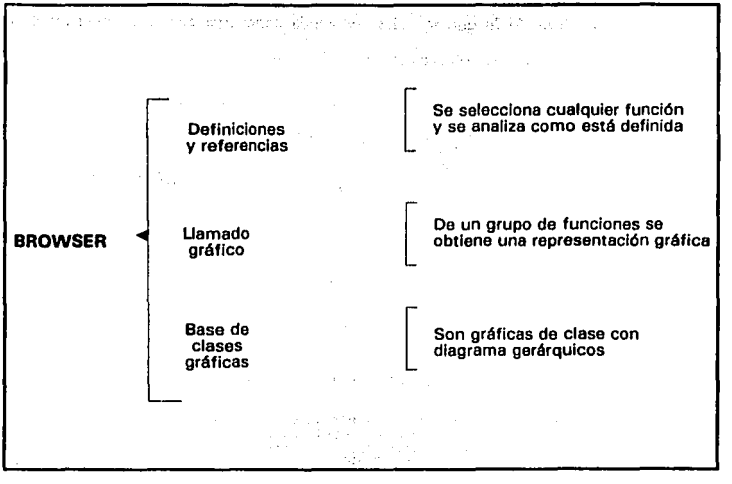


Fig. 2.5.2.4 Los recursos.

Lineas de Ayuda

Las líneas de ayuda también están disponibles en el Visual C++, tambien está disponible en el App Studio. Si se quiere obtener ayuda de una función simplemente se debe dar un click sobre la funcion en el Worckbench Visual y presionar F1, se verá una ventana de ayuda, como la que se muestra en la figura 2.5.2.5.

La ayuda de Visual C++ es util en la resolución de conflictos provocados entre la similitud que existe entre los nombres de las funciones del Windows SDK y las funciones que existen en la librería de Microsoft. Si se ha seleccionado una función que corresponde a muchas funciones en muchas clases, se podrá escoger la que mejor

convenga de una lista. Si lo que se quiere es ayuda sobre una clase, se verán muchas funciones y datos en una lista colocada en orden funcional.

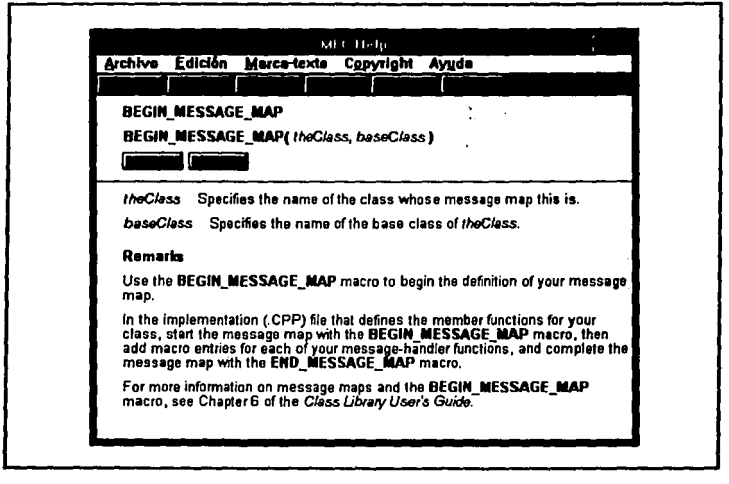


Fig. 2.5.2.5 La ventana de Ayuda de Visual C++.

Herramientas de Diagnostico de Windows

La Edición Profesional del Visual C++ contiene las mismas herramientas que están incluidas en el Windows SDK, cuando era un producto separado:

- SPY para observar los mensages de Windows,
- HEAPWALK para examinar la memoria,
- HC31 para compilar los archivos de ayuda y
- STRESS para limitaciones artificiales de memoria.

Tanto la Edición Estandar del Visual C ++ como la Edición Profesional incluyen la utilidad DBWIN, la cual pone en pantalla el diagnostico de las salidas, y el programa NMAKER que procesa los archivos. El NMAKE es usado para versiones noestandard de la librería de Microsoft.

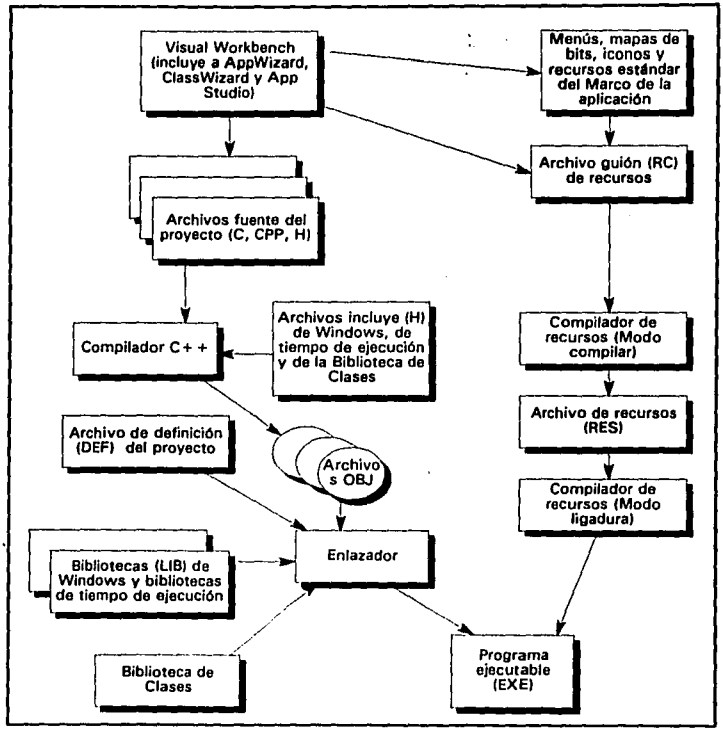


Fig 2.5.2.6 Proceso de Construcción de una aplicación con Visual C++.

OLE

Una de las opciones más importantes dentro de la definición de nuestro proyecto en el "AppWizard" es la incorporación de OLE (*Object Linking and Embedding*, unión e inclusión de objetos) pudiendo nuestra aplicación adoptar los papeles de cliente, servidor o ambas cosas. Esto permite el intercambio transparente y avanzado de archivos con otras aplicaciones que se acojan a las especificaciones OLE. No menos importante es la posibilidad de utilizar la ODBC (*Open DataBase Connectivity*, conectividad abierta de bases de datos) para manejar archivos y cabeceras de los programas de bases de datos más utilizados.

Una vez definidos todos los parámetros de nuestro proyecto, el "AppWizard" nos permitirá visualizar qué archivos van a ser creados y nos mostrará un resumen de las opciones que hemos incorporado. Estos archivos serán, básicamente:

* Un archivo *make* (extensión .mak).
* Un archivo -clw, que será usado por el "ClassWizard" para la edición de las clases.
* Un archivo .def, que contendrá la información que se debe pasar a Windows para que el programa funcione correctamente.
* Un archivo .rc, que almacenará los recursos que hemos incluido en muestra aplicación (tales como barras de menús o ventanas de diálogo), y que podremos modificar con el "AppStudio".
* Un archivo .cpp, que contiene el código fuente del programa.
* Y un archivo .h que contiene las cabeceras necesarias para su funcionamiento.

Además, se crea un subdirectorio RES, que contiene otros recursos como iconos o recursos no editables por el "AppStudio". Y además un archivo README. TXT, que contiene información sobre los archivos que se han creado.

Más Magos

Una vez realizados los primeros pasos en el desarrollo de una aplicación, se intentará conseguir que se comporte como tengamos previsto. Para ello, se editarán directamente los archivos fuente. Se trata de un proceso indispensable para incluir nuestras rutinas y utilizar las herramientas de que disponemos.

Una de las herramientas que se utilizan en el "ClassWizard", que nos permite visualizar en una ventana todas las clases que incluye nuestro programa. Esto hará posible editar en todo momento el código fuente correspondiente a la definición de una clase con sólo hacer un doble-clic con el ratón. Esto resulta especialment útil, ya que el manejo de OLE y de archivos de bases de datos se ha organizado mediante la definición de clases especiales. Así, se pueden añadir o modificar las funciones contenidas en ellas con bastante facilidad.

Recursos

Otra herramienta para ir dando forma a una aplicación en el "AppStudio", que nos permitirá editar y crear los recursos que se han incluido en el proyecto. El "AppStudio" ha sido desarrollado pensando en dotar al programador de una herramienta muy intuitiva y fácil de usar. Los tipos de recursos que podemos manejar son: teclas rápidas o aceleradores, ficheros de mensaje (especialmente útiles para traducir mensajes), ventanas de diálogo, menús y gráficos (iconos, cursores o bitmaps).

La edición de las ventanas de diálogo y de los menú se realiza sobre el resultado final; es decir, en todo momento nos movemos dentro de la ventana que se va a visualizar, pudiendo mover elementos o editar sus propiedades simplemente mediante el doble click del ratón. Una vez que hayamos hecho una modificación, ésta aparecerá inmediatamente en la ventana, visualizando en la pantalla el aspecto que va tener dentro de la aplicación.

La edición de bitmaps y de iconos utiliza un pequeño pero sofisticado editor gráfico que nos permite incluir otros archivos de imágenes mediante enlace OLE, o simplemente copiando y pegando desde otras aplicaciones.

Cuando se editan las características de un recurso cualquiera, podemos dejar activa la ventana que visualiza los datos correspondientes a una zona determinada. por ejemplo una opción de menú. De este modo, nos podremos mover dentro de la ventana y, según sea la zona a la que estamos apuntando, irán cambiando los parámetros y podremos modificarlos.

También es posible invocar el "ClassWizard", pues los recursos están definidos mediante clases. Posteriormente, se pueden editar de la manera que ya se mencionó.

QuickWin

Visual C++ permite producir aplicaciones que se ejecuten en una ventaja de Windows, cuyas funciones de entrada/salida sean idénticas a los estándares para DOS (a saber: (print(), scanf(), y para C++ cin y cout). Esta posibilidad dota a las aplicaciones desarrolladas para DOS de una portabilidad inmediata, ya que no se hacen llamadas directas al BIOS.

La última herramienta que puede resolvernos más de un problema es el depurador, que también está integrado con el resto de las herramientas. Se trata de una versión "Codeview" que incorpora todas las funciones básicas de un depurador, aprovechando las posibilidades del entorno. Con el depurador se puede ejecutar el programa paso a paso, inspeccionar variables, expandirlas si son estructuras de datos, y hasta detener otras aplicaciones Windows mientras el depurador está funcionando.

La biblioteca Foundation Class en Visual C++.

El paso siguiente de Microsoft C/C++7 y su biblioteca Foundation Class se presenta con Visual C++ en donde su principal característica es su interface gráfica para manejar todo bajo Visual, o sea ambiente Windows, con bibliotecas más potentes y grandes, todo bajo la programación orientada a objetos.

Microsoft en Visual C++ nos presenta la versión 2.0 de la biblioteca Foundation Class con aplicación en framework, que constituye su herramienta principal.

Estos son los beneficios de la biblioteca de clases:

Biblioteca Foundation Class versión 2.0 de el C++ Microsoft Windows API. La aceptación de este depende de su comprensión del Lenguaje C++. Si se tiene, entonces, será de manera natural para Windows tener una interface de programación C++.

Aplicaciones framework en la estructura estándar. Cualquier programador que empieza un gran proyecto de desarrollo piensa en la estructura del código. El problema es que cada estructura del programador es diferente y para un nuevo equipo de trabajo es difícil aprender la estructura. La biblioteca Foundation Class con aplicación de framework

incluye su propia estructura de aplicaciones que han sido probadas en muchos ambientes de software y en muchos proyectos. Si se escribe un programa usando la biblioteca de clases se puede enviarlo a cualquier parte del mundo sabiendo que su subordinado puede fácilmente mantenerlo y aumentar el código.

Pero para nada las estructuras de la biblioteca de clases es inflexible para realizar otros programas. Con la biblioteca de clases sus programas pueden realizar cualquier programa que Windows SDK puede hacer.

Las aplicaciones framework son pequeñas y rápidas. Función por función los programas de la biblioteca de clases son casi igual de pequeños que los programas de Windows SDK. Al igual la velocidad en algunas circustancias es actualmente más rápida que su equivalente Windows SDK.

La biblioteca de clases de framework tiene valiosas características. La bilblioteca Foundation Class versión 1.0 con que cuanta la versión Microsoft C/C++ versión 7.0, donde esencialmente se programa con una interface para Windows que contiene:

- Clases de propósito general (no Windows necesariamente) incluyendo:

  - Coleccion de clases para listas, arrays, y mapas.
  - Una útil y eficiente clases de cadenas.
  - Tiempos, duración de tiempos y clases de fechas.
  - Clases de acceso a archivos para operación de sistemas independientes.
  - Soporte para almacenar y recuperar objetos sistemáticos y de disco.
  - Jerarquía de clases.

incluye su propia estructura de aplicaciones que han sido probadas en muchos ambientes de software y en muchos proyectos. Si se escribe un programa usando la biblioteca de clases se puede enviarlo a cualquier parte del mundo sabiendo que su subordinado puede fácilmente mantenerlo y aumentar el código.

Pero para nada las estructuras de la biblioteca de clases es inflexible para realizar otros programas. Con la biblioteca de clases sus programas pueden realizar cualquier programa que Windows SDK puede hacer.

Las aplicaciones framework son pequeñas y rápidas. Función por función los programas de la biblioteca de clases son casi igual de pequeños que los programas de Windows SDK. Al igual la velocidad en algunas circustancias es actualmente más rápida que su equivalente Windows SDK.

La biblioteca de clases de framework tiene valiosas características. La bilblioteca Foundation Class versión 1.0 con que cuanta la versión Microsoft C/C++ versión 7.0, donde esencialmente se programa con una interface para Windows que contiene:

- Clases de propósito general (no Windows necesariamente) incluyendo:

  - Coleccion de clases para listas, arrays, y mapas.
  - Una útil y eficiente clases de cadenas.
  - Tiempos, duración de tiempos y clases de fechas.
  - Clases de acceso a archivos para operación de sistemas independientes.
  - Soporte para almacenar y recuperar objetos sistemáticos y de disco.
  - Jerarquía de clases.

- Actualizado soporte de aplicaciones para interface MDI (Multiple Document Interface).
- Efectivo soporte para OLE (Object Linking and Embedding).

La biblioteca Foundation Class versión 2.0 de Visual C++ conserva muchas características de la versión 1.0 que se encuentran basadas en aplicaciones Windows. Con la aplicación de la arquitectura framework se suman importantes características nuevas:

- Gran soporte para herramientas de menú como: *abrir Archivo, Salvar, y Salvar como;* con la más recientemente lista de archivos.
- Presentación preliminar y especificar impresora.
- Movimiento iterativo de pantallas y partición de pantalla.
- Barra de herramientas y control de barras.
- Acceso a los controles de Microsoft Visual Basic.
- Ayuda
- Procesamiento automático de entrada de datos en ventana de diálogo.
- Fácil programación para la interface OLE.
- Soporte DLL.

Las herramientas de Visual C++ reducen el código complicado. "AppStudio", "AppWizard", y "ClassWizard" estos reducen significativamente el tiempo necesario para escribir el código de una aplicación. Por ejemplo, "AppStudio" crea archivos de encabezados que contienen valores asignados para constantes *#define*. "AppWizard" genera la estructura de código para una aplicación completa, y "ClassWizard" genera prototipos y cuerpos de funciones para mensajes de encabezado.

Una aplicación framework es "una colección integrada de componentes de software orientado a objetos que ofrece todas las necesidades para generar una aplicación". Una aplicación framework es superior a una biblioteca de clases. Una biblioteca de clases ordinaria está incomunicada del diseño de clases para ser incorporada dentro de cualquier programa, pero una aplicación framework define la estructura del programa en sí mismo. Windows desarrolla además bibliotecas de clases incluyendo Microsoft Foundation Class Library versión 1.0, Borland OWL, y Microsoft Foundation Class Library versión 2.0, que son considerados aplicaciones frameworks.

Sin embargo Microsoft Foundation Class Library versión 2.0 de Visual C++ ofrece significativamente más características que los otros sistemas.Con las aplicaciones framework se puede decir que C++ contiene lo necesario para llamarse Visual.

No hay duda que el Microsoft Visual C++ es una herramienta de programación muy potente, y que se ha conseguido un compilador capaz de realizar aplicaciones muy sofisticadas con cierta facilidad, gracias a la última versión de las librerías de clases de Microsoft, las bibliotecas Foundation Classes 2.0.

Lo más sobresaliente el producto es la posibilidad de incluir las especificaciones OLE dentro de nuestras propias aplicaciones. OLE se está convirtiendo en una opción casi indispensable para cualquier aplicación que necesite un intenso intercambio de datos con otras aplicaciones sin perder la transparencia del característico "pinchar y arrastrar" de Windows.

También hay que mencionar las librerías ODBC que nos permiten crear fácilmente aplicaciones compatibles con archivos Access. Btrieve, dBase, Excel, SQL y la mayoría de las bases de datos del mercado con la facilidad que proporciona el entorno.

2.5.3 SELECCION DE HERRAMIENTAS DE SOFTWARE

Elección del Manejador de Bases de Datos.

En cuanto al manejador de bases de datos; se pretende utilizar Paradox for DOS; cuyas caracteristicas y el porqué de la elección(información obtenida de Benchmarks) se describen a continuación:

Con la nueva versión de *Paradox for DOS*, el usuario trabaja inmediatamente y a todo vapor, ya que incluye *Expertos* integrados en su propia computadora, quienes le muestran la manera más fácil de realizar su trabajo con un solo click del ráton.

Paradox for DOS cuenta con *Expertos en Enlaces*, con lo cual el usuario puede llegar a dominar una base de datos fácilmente, debido a que podrá crear enlaces de manera visual, lo único que tiene que hacer es dibujar líneas entre las mismas y el *Experto de Enlaces* le muestra automáticamente cómo enlazar las tablas y realizar la intrincada labor de relacionarlas, no importando qué tan compleja sea la relación.

Los *Expertos en Formatos y Reportes* ayudan al usuario a elaborar bocetos con especificaciones precisas instantáneamente. Con un click del ratón, el usuario le pide al *Experto* que vaya cambiando los diseños hasta encontrar el que más le agrade, por ejemplo, el *Experto en Formatos o Reportes* podrá colocar sus campos horizontal o verticalmente, seleccionar o incluir únicamente los campos que el usuario elija o bien desplegar registros tales como una tabla dentro del diseño instantáneamente y con un click del ratón, el Experto puede acomodar sus campos en forma de etiqueta de correo si así lo solicita, cualquier cambio se puede realizar también de manera instantánea.

El *Experto en Consultas (Query Expert)*, le permite elaborar consultas basadas en sus formatos y reportes existentes. El usuario elige el formato o reporte que desee y el Experto recuperará los datos de una o más tablas y las colocará todas dentro del *"query"* o consulta, creando los enlaces automáticamente; basta con marcar la información que usted desee, pulsar luego un botón y correr la consulta.

PRUEBAS DE BENCHMARK.

Para evaluar la ejecución de las bases de datos multiusuarios, la revista PC Magazine desarrolló un test que prueba cada límite del producto, a través de poner en un ambiente multiusuario muy demandante. Como comparación previa, la revista Magazine escoge utilizar un subgrupo del AS3AP (ANSI SQL Standard Scalable and Portable). Las pruebas para sistemas de bases de datos relacionales, desarrollados por Carolyne Turbyfill, Dina Bitton, y Cyril Orji en la Universidad de Cornell. AS3AP es una industria aceptada de Pruebas que cubren un amplio espectro de operaciones comunes de bases de datos y es también escalable, los resultados válidos son proporcionados en plataformas de PC's a mainframes.

Este año, PCLabs hizo unas cuantas modificaciones a su prueba bed. Mientras que las pruebas del año pasado se enfocaron casi exclusivamente en resultados multiusuario ("bases de datos multiusuarios: listas para compartir", PCMagazine, Marzo 31 de 1992), Las pruebas de este año incluyen varias pruebas para evaluar la ejecución de selecciones, úne y actualiza una sola estación de trabajo. La brueba bed fué actualizada también. Para las pruebas de uso único, se usaron una Compaq Deskpro 486/33M con 8MB de RAM y 310MB de discoduro. El servidor fué una Compaq Systempro 486/33 con 32MB de RAM (sube de 16MB para el último año) y 1.6GB de alta ejecución del que almacena el disco duro mediante el uso del controlador IDA (Intelligent Drive Array) de Compaq.

Para incrementar la resistencia de la prueba de red mientras se evitan los cuellos de botella del hardware, PCLabs probó arriba de 36 estaciones de trabajo y balanceó el tráfico de red simétricamente mediante el uso de los adaptadores Token-Ring EISA de 32 bits en el archivo del servidor. Las estaciones de trabajo fueron idénticamente equipadas con computadoras 386SX/16 de Compaq Deskpro con 5MB de RAM. Se realizaron todas las pruebas usando NetWare 3.11 en una red Token-Ring a 16megabit por segundo.

A cada fabricante se le dió una prueba de especificación detallada para correr nuestras pruebas, y se requirió de un código optimizado de módulos desarrollados en el análisis del manejador de la base de datos (DBMS) para completar cada operación. Los fabricantes no permitieron usar cualquier programa desarrollado o compilado con herramientas que no están incluídos en sus productos. Además un representante de cada fabricante fué presentado en PCLabs por una semana de pruebas para asegurar que los productos eran propiamente instalados y probados.

Todos los productos fueron instalados en una configuración multiusuario. Se instaló la revisión productos DOS en el archivo del servidor de la red, mientras los productos basados en Windows fueron instalados localmente en cada estación de trabajo, con Microsoft Windows 3.1.

La **Carga e Indice** de prueba mide cómo cada paquete de base de datos puede importar un índice a 4 tablas de base de datos con 100000 registros por tabla. Dos tablas contienen cada una dos índices, la tercera tabla contiene 4 índices y la cuarta 5. Cada tabla tiene sus propias características distintas de distribución de datos: una contiene valores únicos por cada columna, otra contiene 10 valores únicos porcentuales por cada columna, la tercera contiene 100 valores distintos por cada columna y la cuarta tiene varios usos de distribuciones y es usada para las pruebas que implican actualización.

Los archivos importantes están localizados en un volumen de NetWare y disco de canal físico que están separados del destino de la base de datos, y la importación e indexado son ejecutados desde una estación de trabajo 486/33.

Antes de la prueba, se permitió a cada fabricante especificar su preferencia por archivos desmarcados, archivos ASCII delimitados por coma ó formatos ASCII de longitud compuesta.

Ambos Microsoft FoxPro para DOS y Microsoft FoxPro para Windows superan en la carga e indice de prueba, completando ambos pasos en, alrededor de 20 minutos cada uno menos que la mitad del tiempo del siguiente-lugar del producto, Parodox para DOS. DataEase fué por mucho el producto más bajo en esta prueba. Sin embargo, DataEase fué cargado por 4 tablas en menos de una hora, el lento indexado fué producido en un tiempo total de más o menos 5.5 horas.

Paradox para Windows requiere un segundo paso para proceder a importar los datos. Mientras que la versión de DOS tubo un comando delimitado por Append al agregar datos a una tabla existente y pudo copiar los datos del archivo original de ASCII. Paradox para Windows requirió de un importante operación, seguida de una tabla de operación de adición, logrando así una anotación similar. Esto duplica el tiempo de procesamiento, cada registro tuvo que ser procesado dos veces.

Cuando se importó cada una de las cuatro tablas de pruebas, descubrimos un virus que resultó en diferentes grados de datos perdidos durante este proceso. Desde que el producto no fué exitosamente cargado las tablas e indices exactos, en su puntación en la carga e indexado de prueba fué "erronea".

La prueba **Select** mide qué tan rápido cada paquete de base de datos puede ejecutar tablas únicas de preguntas donde regresa el 10% de los registros. Ejecutamos dos tipos de preguntas. La selección numerica usa la llave primaria de la tabla y un rango secuencial como registro de criterio. La pregunta alfanumérica ejecuta una pareja-exacta en un campo de texto usando un índice secundario.

Tres de los productos probados -Microsoft Access, FoxPro para DOS y FoxPro para Windows- regresaron una lista de apuntadores a los datos seleccionados en memoria y no el valor actual de las columnas especificadas. Sin embargo, estos productos ejecutan la pregunta que es requerida casi instantaneamente, ellos deben gastar tiempo adicional para recuperar el dato real si una operación más lejana va a ser ejecutada en el conjunto de resultados. Los productos como Paradox y R:Base regresan los datos; mientras se ejecuta cada pregunta, ellos no requieren este acceso adicional a la base de datos.

Access usa un fondo de preguntas de procesamiento y regresa el control después de la primera pantalla de datos disponible, se emite un comando dentro del tiempo de la prueba que es enviada por Access al final de ésta.

Sin embargo, ambas versiones de Paradox regresan en forma rápida la porción númerica de la prueba Select, la selección de tiempos alfanumérica sufrió de un uso deficiente del índice secundario. Con Paradox, si el resultado de una pregunta es menor del 8%, aproximadamente, de la tabla entera, el índice secundario ayuda a la ejecución; el índice estorba la ejecución. Nuestra pregunta alfanumérica regresó 10% de la tabla.

Para la prueba de **Join**, se ejecutaron varias de las uniones especificadas en el AS3AP.

Sin embargo, las uniones regresaron pequeños resultados que, generalmente se procesaban rápidamente, las dos tablas de uno a muchos produjó un amplió rango de funcionamiento, dependiendo del plan de optimización utilizado.

DataEase, KnowledgeMan y Paradox para DOS optimizaron la pregunta usada en esta unión. En contraste, Paradox para Windows y Access se tomaron en tiempos más bajos. A diferencia de Paradox para DOS, Paradox para Windows usa la ingeniería de base de datos de Borland, la cual, en su primera emisión no es retornada para la ejecución.

Para la prueba **Update**, ejecutamos volúmenes actualizados, inserciones y supresiones (Modify, Append y Delete, respectivamente). Primero salvamos 1000 registros de la tabla a una tabla temporal. En seguida, se actualizó el rango correspondiente a la mitad de la tabla de prueba y cambiando el valor de la llave primaria la cuál forza a la base de datos a actualizar su índice. Para medir la rápidez de los productos cuando se están insertando los registros en la tabla de prueba. Finalmente se borrarón 1000 registros de la tabla.

La más difícil de las 3 operaciones para la mayoría de los productos fué actualizar (update). Advanced Revelation y KnowledgeMan ambos tomaron un largo tiempo para hacer esto, debido al índice requerido de organización. Paradox para DOS demostró ser eficiente en la ejecución de las 3 operaciones completando todas ellas en 5 segundos. Ambas versiones de FoxPro también demostraron una rápida manipulación de la tarea Modify (modificación).

Pruebas de Ejecución: Bases de Datos Relacionales.

La prueba **Random Read** (Lectura Aleatoria) dá una idea del número máximo de ocurrencias que cada paquete puede manejar. En esta prueba, cada estación de trabajo casualmente selecciona hileras de la misma tabla y las descarta dirigiendo la salida al mecanismo NUL. Los registros son accesados en el modo de browse para maximizar la cantidad de ocurrencias.

Se corre la prueba por 2 hr 10 min. Para determinar dónde ocurre la mejor etapa, incrementamos el número de estaciones de trabajo gradualmente, adicionando estaciones en intervalos de 10 minutos hasta 36 estaciones de trabajo que están corriendo o hasta que el servidor no pueda soportar más conexiones. La etapa está calculada en transacciones por segundo por cada 10 min. de intervalo.

De nuevo, no hay "tiempo para pensar" insertando entre cada repetición de la prueba, la red cargada es muchas veces lo que el mismo número de estaciones de trabajo produciría en una situación de la vida real.

Los 3 productos -FoxPro para DOS, FoxPro para Windows y Paradox para DOS- destacarón; cada uno alcanzó una etapa por encima de 3000 transacciones por segundo con una carga de 36 estaciones de trabajo. Sin embargo, Paradox para DOS llegó a 32 estaciones de trabajo, la curva de la etapa para ambas verisones de FoxPro quedó esencialmente lineal a las 36 estaciones de trabajo, indicando que el máximo momento para el producto no ha sido todavía alcanzado.

Access también funcionó consistentemente en la prueba de Random Red, un alcance de 846 transacciones por segundo con 36 estaciones de trabajo. Paradox para

Windows funcionó en nuestra de Random Read en forma desalentadora, punteando en 6 estaciones de trabajo y luego estabilizando cerca de 60 transacciones por segundo.

Para la prueba de **Report Generation** (Generación de Reporte), una estación de trabajo imprime un reporte mientras otras ocho estaciones de trabajo ejecutan la prueba de Random Read en el fondo. Para producir el reporte el paquete tiene que unir dos tablas de 100000 hileras y generar un total de 1000 hileras usando calculos de fechas, valores mínimos y máximos, subtotales y totales. Los resultados son impresos en un archivo ASCII en la estación de trabajo que está generando el reporte.

De nuevo, ciertos productos funcionan considerablemente mejor que el resto. Ambas versiones de FoxPro, Paradox para DOS y Access producen el reporte considerablemente más rápido que como lo hicieron los otros productos. Cada versión de FoxPro completó la prueba en menos de dos minutos. En el otro extremo del espectro, KnowledgeMan y Paradox para Windows cada uno tomó más de media hora en completar el reporte y R:Base fué el más lento, tomando 46 minutos para completar el reporte.

La prueba de **Random Write** (Escritura Aleatoria) da una idea del máximo número de ocurrencias actualizadas que el paquete puede ejecutar. En esta prueba, cada estación de trabajo aleatoriamente actualiza hileras individuales de la misma tabla. Se modificó el valor de un campo indexado así que ambas hileras seleccionadas y el índice deben ser actualizados. Las hileras son accesadas en modo update, el cual permite compartir el dato en la tabla, pero no admite que cualquier otra estación de trabajo tenga acceso a una hilera que está siendo actualizada.

Se corrió la prueba por 2 hr 10 min. Para determinar donde el punteo de la etapa ocurre, incrementamos el número de estaciones de trabajo gradualmente, adicionando estaciones en intervalos de 10 minutos hasta que 36 estaciones de trabajo estuvieran corriendo o hasta que el servidor no soporte por más tiempo más conexiones. La etapa es calculada en transacciones por segundo cada uno con 10 min. de intervalo. La red de carga es muchas veces lo que el mismo número de estaciones podría producir en una situación de la vida real.

Ambas versiones de FoxPro se colocaron muy aparte del paquete. La gráfica de ejecución muestra que su etapa estaba justo nivelando 36 estaciones de trabajo con cerca de 770 transacciones por segundo -casi 3 veces el nivel en la mejor ejecución en esta prueba, DataEase.

Ambas versiones de Paradox funcionan penosamente en esta prueba, debido al alto número de requerimientos simultáneos. Para obtener registros protegidos en la misma tabla, Paradox estación de trabajo debe adquirir "sección critica de protección" en el mismo archivo de protección. Para ocurrencias de requerimientos de protección a una sola tabla los clientes de Paradox deben de accesar a este mismo archivo de semáforo en un proceso serial, creando una ejecución sustancial de cuello de botella. Paradox para DOS se punteó en 3 estaciones de trabajo con un total de 42 transacciones por segundo, mientras Paradox para Windows punteó solo 5 transacciones por segundo durante la prueba.

Superbase también dió una pobre actuación en nuestra prueba de Random Write, porque estabamos obligados a usar tablas de protección en lugar de hileras de protección.

Como complemento de la prueba Random Write, PC Labs corrió una rutina de verificación para asegurar que cada registro actualizado fuerá exitosamente completado en la base de datos. Debido a que un virus en R:Base envolvió la protección de contención del producto, no fué exitosamente completado en todas las actualizaciones, así que no se presentan sus resultados en esta gráfica. Microrim ha hecho una cura disponible para corregir este problema, pero no estaba disponible en tiempo para estas pruebas.

Con las compañías más grandes de software ahora haciendo las base de datos una parte central de sus negocios, la competencia es muy grande. El ganador más grande de todos es el desarrollador de la base de datos. Las bases de datos relacionales están captando con otras principales corrientes de productos de software la facilidad de uso, mientras que al mismo tiempo se convierten en más poderosas y sofisticadas.

Tres entradas sobresalen de la multitud y ganar nuestra distinción de la elección del editor : Microsoft FoxPro 2.5 para DOS, Microsoft FoxPro 2.5 para Windows y Paradox para Windows. Estos tres productos alcanzan un buen balance entre la accesibilidad y el poder, el cual les dá una gran atracción. Los usuarios finales y desarrolladores profesionales encontrarán en ellos útiles herramientas. Si usted tiene una herencia del código de XBase o si la rapidez es de principal importancia, entonces usted querrá echar un vistazo a uno de los paquetes de FoxPro. Pero si usted quiere empezar una nueva base de datos, una garantia para tomar ventaja de las nuevas herramientas de hoy de POO, entonces Paradox es el indicado.

FoxPro fué siempre rápido, pero ahora es aún más rápido. FoxPro también proporcionó fuertes herramientas de usuarios finales. La versión de Windows mantiene la poderosa, pero amigable tradición del producto de DOS, mientras toma una buena ventaja de las características especiales de Windows, tales como DDE y OLE. Su plataforma

transportadora le permite mover fácilmente aplicaciones entre DOS y Windows, y la nueva capacidad de compilación permite a los desarrolladores mantener un sólo código de base para ambas versiones de sus aplicaciones.

Paradox no tiene la rapidez de FoxPro, pero trae bases de datos relacionales a un nivel completo y nuevo de sofisticación, combinando una consistente interfase de orientación de objetos, e integridad referencial, y acceso transparente a los archivos de formato múltiple. En futuras versiones, Paradox se volverá aún más fuerte.

Otros productos que revisamos son menos atrayentes pero dichos de ser notados. Superbase toma una gran ventaja sobre la interfase de Windows y es fácil de usar, sin embargo, le encontramos carencias en los caballos de fuerza para mantenerlo en un volumen alto en el ambiente de multiusuarios. DataEase, en forma experta, combina una poderosa aplicación constructora con facilidad de uso, pero otra vez a expensas de la rapidez.

Dadas las pruebas de Benchmark podemos concluir que el manejador de bases de datos que utilizaremos será Paradox, debido a que podemos emplear las herramientas de esté, sobre Programación Orientada a Objetos, indispensable para el desarrollo de nuestro sistema en cuestión.

Tanto Paradox para DOS, como Paradox para Windows destacan entre los primeros manejadores, que no son los mejores como las dos versiones de FoxPro, pero que tienen las ventajas de la Programación Orientada a Objetos y servira de alguna manera para la captura de información de los principales prestadores de bienes y servidores del lugar en particular y del manejo de las mismas bases de datos, para actualizar y corregir su información.

En cuanto al paquete de programación que realizará la integración de todo el sistema será el Visual C+ + , ya que nos proporciona (entre otras cosas) un ambiente gráfico produciendo un sistema de calidad para el usuario y facilidad en el manejo de la información.

CAPITULO III

DESARROLLO DEL SISTEMA

3.1 Diseño

3.1.1 ESPECIFICACIONES DEL DISEÑO.

Tomando en cuenta que el sistema será realizado en un lenguaje de programación visual, en donde la interface con el usuario será gráfica, entonces estamos hablando de que el sistema será manejado bajo ambiente Windows. Así, el usuario podrá obtener información turística del puerto de Ixtapa-Zihuatanejo, mediente el uso de iconos y de ventanas.

Para elaborar el presente sistema nos basaremos en las características de la Programación Orientada a Objetos, basándonos principalmente en los conceptos de Abstracción, (el cual hace más simple la escritura de programas grandes), la encapsulación (facilita el mantenimiento de un programa) y finalmente el concepto de jerarquías de clases, (que es una herramienta de clasificación muy poderosa que hace que un programa sea extensible fácilmente). Una vez definido lo anterior, el trabajo de diseño consistirá primeramente en:

a) Hacer un análisis del problema planteado, definiendo las variables de entrada y de salida de información.

b) Identificar las clases que el sistema necesita. Las clases deben estar basadas en las actividades centrales del sistema.

c) Una vez que se identificaron las clases, la siguiente tarea es determinar que responsabilidades presentan, a esto se le llama **"Asignación de atributos y comportamientos"**. Las responsabilidades de una clase caen dentro de dos categorías:

- La información que un objeto de esa clase debe contener.
- Las operaciones que un objeto puede ejecutar o que pueden ser ejecutados.
- (¿Qué puede hacer este objeto?)

d) Encontrar relaciones entre las clases, para poder identificar tales relaciones se debe de considerar como una clase ejecuta el comportamiento que le ha sido asignado.

e) - Organizar las clases en jerarquías. Esto es una extensión del segundo paso, pero requiere la información ganada del tercer y cuarto paso. El proceso de asignar atributos y comportamientos a las clases logra que se tenga una mejor idea de sus similitudes y de sus diferencias; al identificar las relaciones entre clases se observa que clases necesitan ser incorporadas a la funcionalidad de otras.

f) - Recabar y capturar la información turística correspondiente de las bases de datos que el sistema manejará. Las bases de datos contendrán información de todos los servicios que el puerto ofrece, por ejemplo el nombre de aeropuertos, central de autobuses, hoteles, hospitales, restaurantes, discotecas, delegaciones, bancos, aduanas, casas de cambio, escuelas, capitanías, etc.; junto con toda la información concerniente a tales sitios, como su dirección, precio, categoría y teléfono.

También se manejarán bases de datos de imágenes; especialmente en los hoteles, pensando en ayudar al turista en hacer una elección del hotel en donde se desee hospedar.

g) - Una vez hecho lo anterior, se dispondrá a digitalizar los mapas de las ciudades de Ixtapa y Zihuatanejo; esto se hará mediante una tableta digitalizadora, con la cual se obtendrán, una serie de puntos que se manejarán como coordenadas "*x, y*" que serán almacenadas en un arreglo, con un formato "autocad", y de ahí se hará la conversión de formato a archivos PCX para que el lenguaje lo pueda manejar.

h) - Después se realizarán el diseño de las pantallas de consulta y de información del sistema, tales como la pantalla de presentación y las de los menús junto con todas las cajas de diálogo y edición que el sistema manejará.

i) - Entonces se procederá a realizar los módulos de programación para que se puedan relacionar las pantallas que contienen los mapas de las ciudades, con las diferentes bases de datos y entre ellas.

j) - Una vez hecho lo anterior se elaborará el código de programación que se encargará de ligar los módulos de programación anteriores.

k) - Se eleborará en el sistema una opción de ayuda del usuario, en donde se describirán los pasos que se requieren para ejecutar el sistema.

3.1.2 DIAGRAMA DE FLUJO DE DATOS

La estrategia de flujo de datos muestra el empleo de éstos en forma gráfica. Las herramientas utilizadas al seguir esta estrategia muestran todas las características escenciales del sistema y la forma en que se ajustan entre sí. Puede ser difícil comprender en su totalidad un proceso del sistema si se emplea para ello sólo una descripción verbal; las herramientas para el flujo de datos ayudan a ilustrar los componentes esenciales de un sistema junto con sus interacciones.

Un diagrama de flujo de datos se emplea para describir y analizar el movimiento de datos a través de un sistema, ya sea que éste fuera manual a automatizado, incluyendo procesos, lugares para almacenar datos y retrasos en el sistema. Los diagramas de flujo de datos son la herramienta más importante y la base sobre la cual se desarrollan otros componentes, La transformación de datos de entrada en salida por medio de procesos puede describirse en forma lógica e independiente de los componentes físicos asociados al sistema.

El diagrama de flujo de nuestro sistema se muestra en la figura 3.1.2.1 el cual consta de 4 grandes procesos:

- Proceso de Ubicación.
- Proceso de Información Turística.
- Proceso de Quejas.
- Proceso de Ayuda.

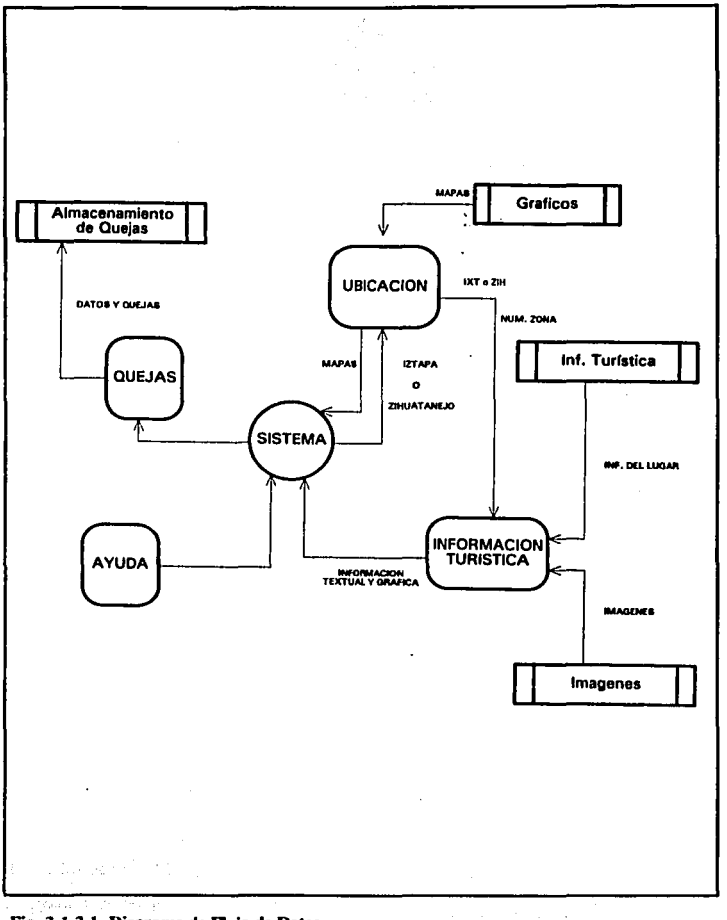


Fig. 3.1.2.1 Diagrama de Flujo de Datos.

Proceso de Ubicación

En este proceso los usuarios envian hacia ubicación su opción (Iztapa o Zihuatanejo). Ubicación necesita esta información para poder obtener de la base de mapas las coordenadas de la ciudad elegida para procesar el mapa y desplegar en pantalla el mapa y mandar la ubicación al proceso de información turistica.

Proceso de Información Turistica

Una vez recibidos los datos de ubicación, el proceso de Información Turistica recibe por parte del usuario la opción de la cual necesita información; con esto el proceso obtiene de la base de información turistica la información del lugar y de la base de imagenes las graficas disponibles de los bienes y servicios del lugar y las despliega en pantalla.

Proceso de Quejas

Aquí el usuario manda en un archivo typo texto su queja, la cual será almacenada en la base de almacenamiento de quejas.

Proceso de Ayuda

En todo momento el usuario del sistema podrá solicitar ayuda del funcionamiento del sistema mandando su pregunta y el proceso de Ayuda se encargará de desplegar el tema solicitado.

3.1.3 DISEÑO FISICO Y CONSTRUCCION DE LA BASE DE DATOS.

En cuanto al diseño físico del sistema podemos decir que este estara integrado por una serie de imagenes; que representaran las pantallas del mismo y figuras tales como mapas de los lugares en estudio, en este caso de Ixtapa-Zihuatanejo. Para tales imágenes se describirá brevemente cuales son los tipos de formatos de imágenes que existen y las características de estos para el manejo de la información. La elección del formato de las imágenes se integrará finalmente al sistema.

Para la construcción de la base de datos se tomarán en cuenta: como principal requisito, los requerimientos del usuario, fundamentales para la elaboración del diseño del sistema y posteriormente para el desarrollo del mismo. La recopilación y clasificación de la información. El manejo de las bases de datos se realizará mediante el DBMS (Data Base Manager System), Manejador de Base de Datos de Paradox para DOS; el cual nos ayudará a crear y generar las bases de datos, realizar las relaciones entre ellas, generar los indices necesarios y de esta forma conjuntar las imágenes con la información almacenada en bases de datos para desarrollar el sistema.

DESCRIPCION DE LOS FORMATOS "TIFF", "GIF" Y "PCX".

FORMATO TIFF (TAG IMAGE FILE FORMAT).

TIFF es un formato que fue diseñado para promover el intercambio de datos de imágenes digitales. Los campos fueron definidos primeramente pensando en "desktop publishing" y aplicaciones relacionadas.

El escenario general para el cual TIFF fue inventado, asume que el software de aplicación para "scanear" o "pintar" crea un archivo TIFF, el cual puede ser leído e incorporado dentro de un documento o publicación tal como un paquete de "desktop publishing".

La intención de TIFF es organizar y codificar practicas existentes con respecto a la definición y uso de datos digitales. TIFF fue diseñado para ser un formato de intercambio muy extensible.

TIFF no es un lenguaje de impresora o lenguaje de descripción de páginas, tampoco es su propósito ser un standard general para el intercambio de documentos. Puede ser de utilidad para algunas aplicaciones de edición de imágenes. El objeto de diseño primario fue el proveer de un medio ambiente dentro del cual la permuta de datos e imágenes entre programas de aplicación pueda ser realizado. TIFF es por lo tanto planeado para ser un super conjunto de formatos de archivos de imágenes existentes para scanners de oficina (y programas de pintura y cualquier otro que produzca imágenes con pixeles).

TIFF esta propuesto para ser independiente de sistemas operativos especificos, sistemas, compiladores y procesadores. La única hipótesis significante es que el medio de almacenamiento soporte algo como un archivo, definido como una secuencia de bytes de 8-bits, donde los bytes son numerados de 0 a N. El archivo TIFF más largo es de 2**32 bytes. Dado que los apuntadores son usados libremente, un archivo TIFF es más fácilmente leído desde un dispositivo de acceso aleatorio, aunque es posible leer y escribir archivos TIFF en un medio secuencial como cintas magnéticas.

La extensión de archivos recomendada por MS-DOS para archivos TIFF es .TIF. El filetype recomendado por Macintosh es TIFF. Convenciones en otros ambientes computacionales no han sido aun establecidos.

ESTRUCTURA

En TIFF, campos individuales son identificados con una etiqueta única. Esto permite que campos particulares puedan estar presentes o ausentes en un archivo según sea requerido por la aplicación.

Algunos archivos TIFF tendrán solo unos pocos campos en ellos; otros tendrán varios. El software que cree archivos TIFF deberá escribir tantos campos como crea que serán necesarios y útiles (y no más). El software que lea archivos TIFF deberá hacer lo mejor que pueda con los campos que encuentre ahí.

Existen muchas formas en las cuales un esquema de formato de archivos orientados a etiquetas puede ser implementado. TIFF usa la siguiente aproximación:

Existen tres partes principales en un archivo TIFF. La primera es un pequeño encabezado de archivo imagen. El siguiente es un directorio de todos los campos que serán encontrados en este archivo. Finalmente, tenemos los datos para los campos.

Un archivo TIFF comienza con una pequeña cantidad de datos posicionales, conteniendo la siguiente información:

Bytes 0-1:

La primera palabra del archivo sirve para especificar el orden de los bytes usados dentro del archivo. Los valores definidos actualmente son:

LL (hex 4949)
MM (hex 4D4D)

En el formato LL, el orden de los bytes es siempre del menos significativo al más significativo, para enteros de 16 y 32 bits.

En el formato MM, el orden de los bytes es siempre del más significativo al menos significativo, para enteros de 16 y 32 bits.

Bytes 2-3:

La segunda palabra del archivo es el número de versión de TIFF. Este número no debe cambiar. La actual descripción se refiere a la versión 42, por lo cual 42 (2A en hex) debe estar almacenado en esta palabra.

Bytes 4-7:

Esta palabra contiene el offset (en bytes) del primer Directorio del Archivo de Imagen (Image File Directory = IFD). El directorio debe estar en cualquier localidad en el archivo después del encabezado. (El termino byte offset será siempre usado para referirse a una localidad con respecto al principio de el archivo. El primer byte en el archivo tiene un offset de 0).

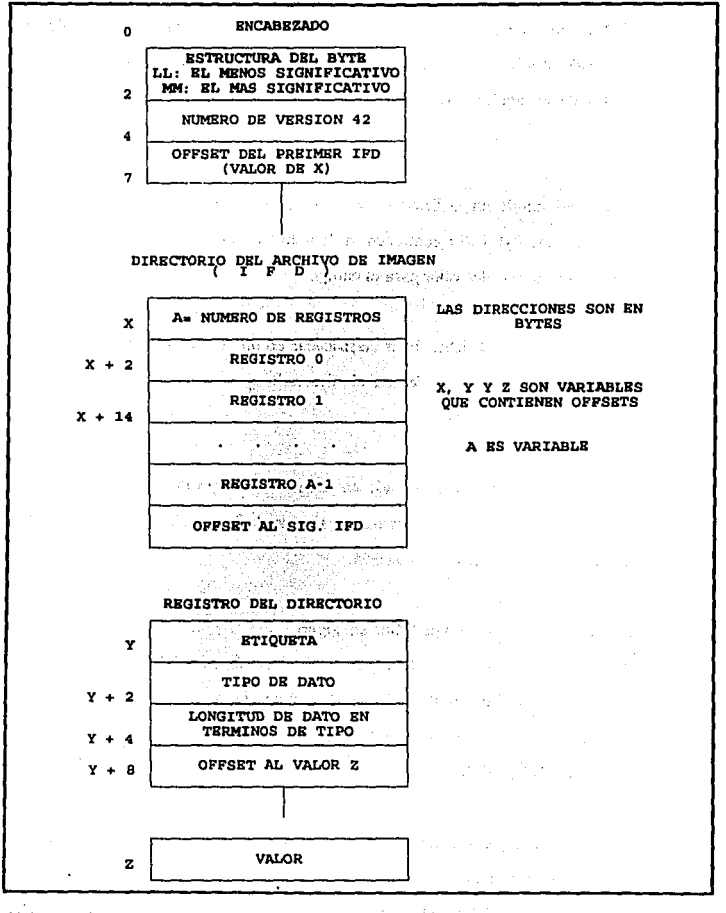


Tabla 3.1.3.1 Formatos tipo TIFF.

Un IFD consiste de 2 bytes para contar el número de entradas (por ejemplo el número de campos), seguidos de una secuencia de campos de 12 bytes, seguidos por un offset de 4 bytes de el siguiente IFD (o 0 si no hay). Cada campo tiene el siguiente formato:

Los bytes 0-1 contienen la Etiqueta para el campo. Los bytes 2-3 contienen el Tipo de el campo. Los bytes 4-7 contienen la longitud de el campo. Los bytes 8-11 contienen el offset (en bytes) del valor para el campo.

Las entradas en un IFD deben estar clasificadas en orden ascendente por etiqueta. Los valores para los cuales las entradas en el directorio apuntan no necesitan estar en algún orden particular en el archivo.

La parte de Longitud es especificada en términos del tipo de dato. Una palabra simple de 16 bits (SHORT) tiene una longitud de 1, y no de 2, por ejemplo. Los tipos de datos y su longitud se describen a continuación:

1=BYTE	entero de 8 bits sin signo
2=ASCII	byte de 8 bits que almacenan códigos ASCII
3=SHORT	entero de 16 bits (2 bytes) sin signo
4=LONG	entero de 32 bits (4 bytes) sin signo
5=RATIONAL	2 LONGs

Note que puede haber más de un IFD. Cada IFD define un subarchivo. Un uso potencial de archivos subsecuentes es el describir una sub-imágen que esta de algún modo relacionada con la imágen principal. Otro uso es el representar múltiples páginas de imágenes, por ejemplo, un documento de facsímil requiere más de una página.

FORMATO GIF (GRAPHICS INTERCHANGE FORMAT)

GIF es un estandard de CompuServe para la definición de imágenes a color. Este "formato de intercambio gráfico" permite, alta calidad, alta resolución gráfica para ser desplegada en una variedad de hardware gráfico y esta propuesto como un mecanismo de despliegue e intercambio de imágenes gráficas.

FORMATO GENERAL DE UN ARCHIVO

GIF SIGNATURE (FIRMA GIF):

Identifica a los datos como una imagen GIF valida. Consiste de los siguientes seis caracteres :

G I F 8 7 a

Los últimos tres caracteres "87a" pueden ser vistos como un número de versión para una definición particular de GIF.

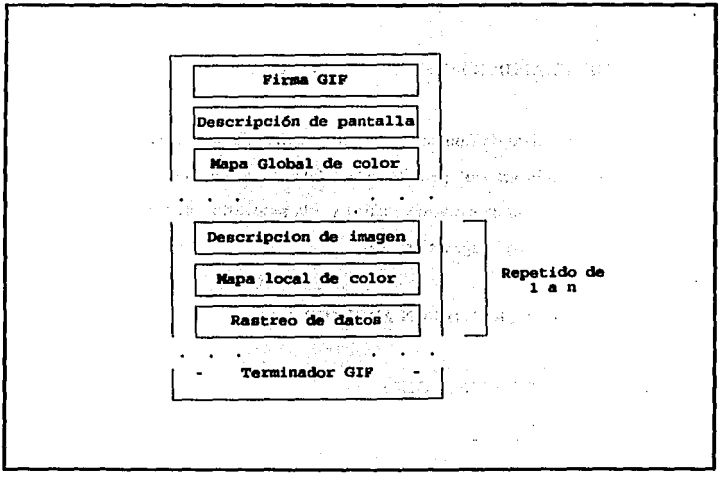


Tabla 3.1.3.2 Formato tipo GIF.

SCREEN DESCRIPTOR (DESCRIPTOR DE PANTALLA):

Describe el total de parámetros para las imágenes GIF. Define todas las dimensiones de un espacio de imágen o la pantalla lógica requerida, la existencia de información de mapeo de colores, color del fondo de la pantalla, e información de la intensidad del color.

Esta información es almacenada en una serie de bytes de 8-bits como sigue:

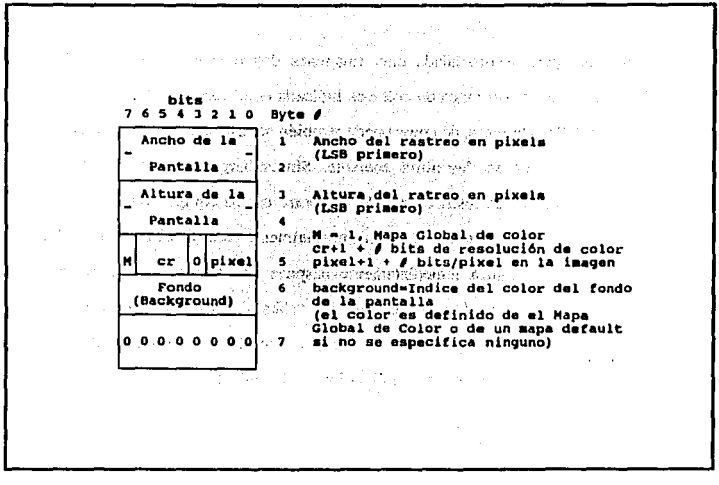


Tabla 3.1.3.3 Descriptor de pantalla.

El ancho y largo de la pantalla lógica pueden ser más largos que el despliegue físico. Como las imágenes más largas que el despliegue físico son manejadas es una implementación que depende y puede tomar ventaja de las características del hardware.

El valor de "pixel" también define el máximo número de colores dentro de una imágen. El rango de valores para "pixel" es de 0 a 7 lo cual representa 1 a b bits. Esto representa un rango de 2 (B & W) a 256 colores. El bit 3 de la palabra 5 esta reservada para futuras definiciones y debe ser cero.

GLOBAL COLOR MAP (MAPA GLOBAL DE COLOR):

Es opcional pero recomendada para imágenes donde la exactitud del color es deseada. La existencia de este mapa de color es indicada en el campo "M" del byte 5 del Descriptor de Pantalla. Un mapa de color puede también ser asociado con cada imágen en un archivo GIF tal como se describirá adelante. Sin embargo este mapa global será normalmente usado dadas las restricciones de hardware en los equipos disponibles hoy. En una imágen individual la bandera "M" será normalmente cero. Si el mapa global esta presente, su definición seguirá inmediatamente después del descriptor de pantalla. El número de entradas en el mapa de color es igual a 2**(bits por pixel), donde cada entrada consiste de tres valores que representan la relativa intensidad de rojo, verde y azul respectivamente. La estructura de el mapa de color es el siguiente :

bits 7 6 5 4 3 2 1 0	Byte #	
Intensidad rojo	1	Valor del rojo para el Indice 0
Intensidad verde	2	Valor del verde para el Indice 0
Intensidad azul	3	Valor del azul para el Indice 0
Intensidad rojo	4	Valor del rojo para el Indice 1
Intensidad verde	5	Valor del verde para el Indice 1
Intensidad azul	6	Valor del azul para el Indice 1
		(Continua para los colores restantes)

Tabla 3.1.3.4 Mapa Global de Color.

El blanco será representado como (255,255,255), el negro como (0,0,0) y el amarillo como (180,180,0).

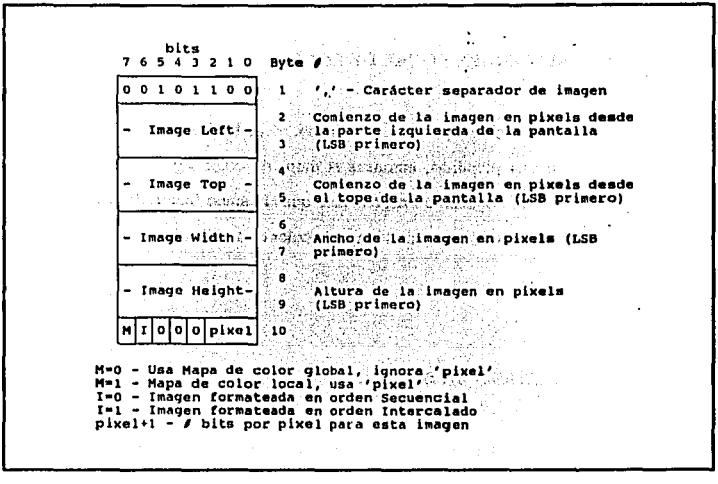


Tabla 3.1.3.5 Descriptor de Imagen.

IMAGE DESCRIPTOR (DESCRIPTOR DE IMAGEN):

El descriptor de imagen define la colocación actual y la extensión de la siguiente imagen dentro de el espacio definido en el descriptor de pantalla. Cada descriptor de imagen es introducido por un carácter separador de imagen. El rol del separador de

imagen es simplemente proveer un carácter de sincronización para introducir un descriptor de imagen. Esto es deseable si el archivo GIF contiene más de una imagen. Este carácter es definido como 0x2C hex o ',' (coma). Cuando este carácter es encontrado entre imágenes, el descriptor de imagen seguirá inmediatamente.

LOCAL COLOR MAP (MAPA LOCAL DE COLOR):

Es opcional y se define aquí para uso futuro. Si el bit "M" de el byte 10 de el descriptor de imagen esta prendido, entonces el mapa de color sigue al Descriptor de imagen que aplica solo a la siguiente imagen, note que el campo 'pixel' de el byte 10 de el descriptor de imagen es usado solo si el mapa de color esta indicado.

RASTER DATA (RASTREO DE DATOS):

El formato de la actual imagen esta definido como una serie de valores de pixeles que crean la imagen. Los pixeles son almacenados de derecha a izquierda secuencialmente para un renglón de la imagen. El primer paso escribe cada 8 renglones, empezando con el renglón del tope de la imagen. El segundo paso escribe cada 8 renglones empezando en el quinto renglón desde el tope. El tercer paso escribe cada 4 renglones empezando en el tercer renglón desde el tope. El cuarto paso completa la imagen, escribiendo cada renglón, empezando en el segundo renglón desde el tope. Una descripción gráfica de este proceso es el siguiente:

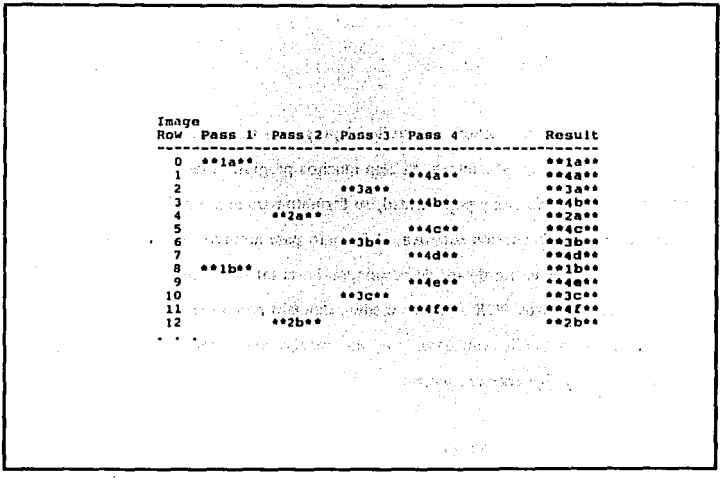


Image Row	Pass 1	Pass 2	Pass 3	Pass 4	Result
0	**1a**				**1a**
1				**4a**	**4a**
2			**3a**		**3a**
3				**4b**	**4b**
4		**2a**			**2a**
5				**4c**	**4c**
6			**3b**		**3b**
7				**4d**	**4d**
8	**1b**				**1b**
9				**4e**	**4e**
10			**3c**		**3c**
11				**4f**	**4f**
12		**2b**			**2b**

. . .

Tabla 3.1.3.6 Rastreo de datos.

GIF TERMINATOR (TERMINADOR DE GIF):

Para proveer una sincronización para la terminación de un archivo de imagen GIF, un decodificador GIF deberá procesar el final del modo GIF cuando el carácter 0xB hex o ';' sea encontrado después de que una imagen ha sido procesada. Por convención el software decodificador pausara y esperara una acción indicando que el usuario esta listo para continuar. Esta puede ser el introducir el retorno de carro en el teclado o un 'click' del mouse.

FORMATO PCX

Hace algunos años, ZSoft desarrollo el programa para PC Paintbrush como una respuesta al MacDraw de Macintosh. Si bien muchos programas de gráficos rivalizaron a Paintbrush en características y popularidad, su formato para archivos PCX ha trascendido al programa. Con cada versión sucesiva, el formato para archivos PCX se ha adaptado a los nuevos estandares en hardware de despliegue hasta tal punto que muchos programas DOS ahora usan archivos PCX como un medio estandard para intercambiar gráficas.

Los archivos PCX empiezan con un encabezado (header) de 128 byte. El encabezado tiene la siguiente estructura:

CAMPO	TIPO
pcxManufacturer	BYTE
pcxVersion	BYTE
pcxEncoding	BYTE
pcxBitsPixel	BYTE
pcxXmin, pcxYmin	INTEGER
pcxXmax, pcxYmax	INTEGER
pcxHres, pcxVres	INTEGER
pcxPalette[16]	PCXRGB
pcxReserved	BYTE
pcxPlanes	BYTE
pcxBytesLine	INTEGER
pcxPaletteInfo	INTEGER
pcxFiller[58]	BYTE

El campo pcxManufacturer es siempre \xA. Este es un "numero mágico" usado para identificar un archivo PCX. El campo pcxVersion identifica que versión de Paintbrush PC o que programa compatible creó este archivo. Los identificadores de las versiones se muestran en la Tabla 3.1.3.7:

ID	PC	Paintbrush Version
0	2.5	
2	2.8	With palette
3	2.8	Without palette
5	3.0	

Tabla 3.1.3.7 ID de Version PCX.

El campo pcxEncoding debe tener el valor de 1. El campo pcxBitsPerPixel nos indica cuantos bits consecutivos en el archivo representa un pixel en la pantalla, ver Tabla 3.1.3.8.

BITS/PIXEL	Color (Display Mode)
1	Monochorome, EGA/VGA
	16 COLOR
2	CGA 4 COLOR
8	MCGA/VGA 256 COLOR

Tabla 3.1.3.8 Bits por Pixel.

Los siguientes cuatro campos (pcxXmin, pcxYmin, pcxXmax y pcxYmax) definen los limites de la imagen. Las dimensiones de la imagen serán pcxXmax - pcxXmin + 1 por pcxYmax - pcxYmin + 1. Los limites mas bajos de pcxXmin y pcxYmin son usualmente 0.

Los campos pcxHres y pcxVres especifican la resolución de la pantalla que será usada para crear el archivo. Estos campos pueden ser ignorados. El campo pcxPalette contiene el color de la paleta. El campo pcxReserved usualmente contiene el modo de video del BIOS pero no esta documentado como tal. Puede ser ignorado.

El campo pcxPlanes dice cuantos planos de color tiene la tarjeta de video. Por ejemplo, el empiezo de una imagen PCX en modo EGA 16-color aparece en la figura 3.1.3.1. La imagen tiene dos pixels azules, tres pixels verdes, dos pixeles rojos, y un pixel magenta (rojo + azul). La imagen tiene de ancho 640 pixels (80 bytes = 640 pixels / 8 bytes/pixel). El número actual de bytes en el archivo por scanline es probablemente menor que 80 bytes debido a la compresión.

El campo pcxBytesLine indica cuantos bytes existen en cada scanline. Este valor dependerá de el numero de bits por pixel y las dimensiones de la imágen. El campo pcxPaletteInfo es para imágenes VGA. El valor es 1 para escalas de gris y 2 para imágenes en color.

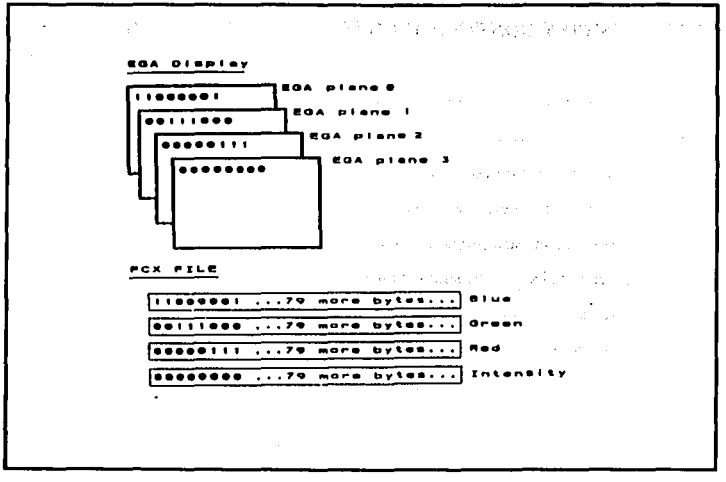


Fig. 3.1.3.1 Representación de una imagen PCX en modo EGA COLOR-16.

El resto de los 128-bytes del encabezado esta reservado para futuras adiciones a el formato PCX.

DISEÑO FISICO Y CONSTRUCCION DE LA BASE DE DATOS.

Se ha mencionado en capítulos anteriores, que el manejador de bases de datos (DBMS) que utilizaremos será Paradox para DOS; la elección fue hecha por ser el más apegado a estudios de confiabilidad en cuanto al manejador de bases, a la rápidez de respuesta para un número elevado de registros, a la relación entre las diferentes bases de datos y para fines de nuestro propio sistema.

El diseño físico se generará tomando en cuenta los datos recabados en la elaboración de cuestionarios.

Toda la información se capturará en diferentes bases de datos clasificandolas en clases distintas para Prestadores y Bienes de Servicio. Estas serán las dos grandes clasificaciones que utilizaremos y de las cuales surgirán las distintas clases y sus respectivos objetos para la manipulación de la información.

Otros puntos importantes que utilizaremos para la construcción de la base de datos serán: las relaciones entre clases que resultan al generar el Diagrama de Flujo de Datos, así como del Diagrama de Entidad-Relación que nos ayudará a determinar los indices de relación entre clases, para poder accesar a una u otra.

Esto facilitará la construcción de no solo una sino varias bases de datos dependiendo del número de clases que generemos de la información obtenida y del número de prestadores de bienes y servicios.

También, otro dato relevante será la realización del Diccionario de Datos que contendrá cada uno de los campos que utilizaremos: el nombre del campo, el tipo de campo y la longitud de dichos campos. Con tal información se generarán los registros necesarios y las relaciones para cada una de nuestras bases.

No debemos olvidar que todo el manejo de la información en cuanto a bases de datos se refiere será manipulada por el manejador de bases de datos de Paradox para DOS. Y que el desarrollo del sistema a nivel de programas y ambiente visual será mediante Visual C++.

NORMALIZACIÓN DE LA BASE DE DATOS EN PARADOX.

Dentro del módulo de Información Turística, se considerá la información de los diferentes Prestadores de Bienes y Servicios.

Para llevar acabo la normalización utilizaremos los catálogos de giros, información general, productos y precios.

Para la primera forma normal dentro de los catálogos del sistema y de esté modulo en particular tomaremos los Catálogos de Giros y el de Información General.

Así tomaremos los campos siguientes del catálogo de giros:

CAMPO	DESCRIPCIÓN
cve_giro	La clave que le designaremos a cada uno de los giros ya sea para Bienes o Servicios.
ngiro	Nombre de cada uno de los giros.
what	Campo que generamos para diferenciar un Bien o un Servicio, asignando la letra "B" o "S", respectivamente.

Para el catálogo de información general tomaremos los campos siguientes:

CAMPO	DESCRIPCION
cve_giro	Clave del giro que puede ser un Bien o Servicio.
nombre_com	Nombre comercial del giro.
categoria	Campo particular para el giro de Hoteles y especificar la categoría de éstos.
rfc	Registro Federal de Causantes.
colonia	Colonia del giro (establecimiento: Hotel, Restaurante, Abarrotes, etc.).
calle	Calle del establecimiento.
numero	Número del establecimiento.
c.p.	Código Postal del establecimiento.
tel	Teléfono del establecimiento.
horario	Horario en el cual trabaja el establecimiento.
Prin_Ser_1	Principal(es) servicio(s) que presta el establecimiento en particular.
Razon_soc	Razón Social del establecimiento.
folio	Número del folio para cada una de las encuestas, que corresponde a un establecimiento en particular.

La llave para relacionar a estos dos catálogos será la clave del giro, y la normalización quedaría de la siguiente forma:

Nombre del Giro	Hoteles
Bien o Servicio	S
Nombre Comercial	Hotel Dorado Pacífico

Categoría	5 Estrellas
R.F.C.	OHD800408URO
Colonia	Ixtapa
Calle	Paseo Ixtapa
Número	---
C.P.	40880
Teléfono	32025
Horario	---
Principal Servicio	Servicios Complementarios 5 Estrellas
Razón Social	Operadora Hotelera Dorado Pacífico

De esta forma, generamos un nuevo catálogo al cual denominaremos catálogo A. Con este catálogo A y el catálogo de productos generaremos la segunda forma normal.

Dentro del catálogo de productos tomaremos los siguientes campos:

CAMPO	DESCRIPCION
cve_giro	Clave del giro.
cve_produ	Clave del producto, que servirá para diferenciar a cada uno de éstos.
producto	Nombre del tipo de producto.
consec	Número consecutivo del tipo de producto.
marca_prod	Marca del producto.
presen	Presentación del producto.

El campo llave para estos catálogos será nuevamente la clave del giro. La normalización quedará de la siguiente forma :

Nombre del Giro	Hoteles
Bien o Servicio	S
Nombre Comercial	Hotel Dorado Pacífico.
Clave del Producto	501
Producto	Habitación
Consecutivo	4
Marca del Producto	J. Suite
Presentación	---

A este catálogo lo denominaremos catálogo B.

Finalmente, para obtener la tercera forma normal utilizaremos el catálogo B y el catálogo de precios. Y los campos que utilizaremos de éste último serán:

CAMPO	DESCRIPCION
cve_giro	Clave del giro.
cve_produ	Clave del producto.
producto	Nombre del tipo de producto.
consec	Número consecutivo del tipo de producto.
marca_prod	Marca del producto.
presen	Presentación del producto.
precio	Precio del Producto.
folio	Número de folio para cada una de las encuestas que corresponde a un giro en particular.

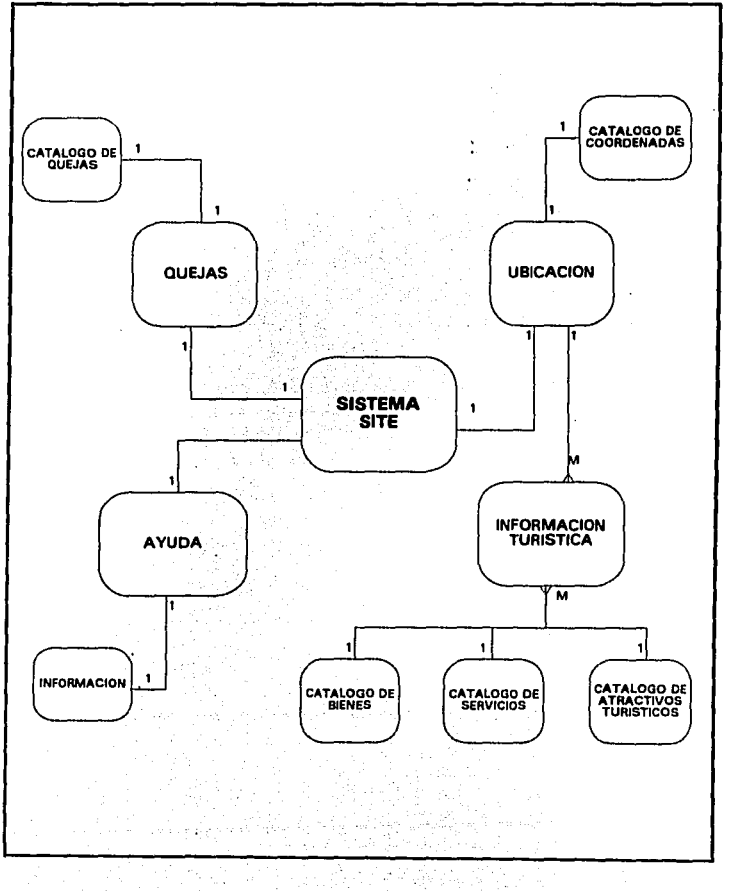
El campo llave en este caso será la clave del giro y el folio, para relacionar ambos catálogos, obteniendo finalmente el tipo de prestador de Bien o Servicio, el nombre del giro, la información general del giro, los productos que maneja y el precio de cada uno de ellos de la siguiente manera:

Nombre del Giro	Hoteles
Bien o Servicio	S
Nombre Comercial	Hotel Dorado Pacífico
Razón Social	Operadora Hotelera Dorado Pacífico
Clave del Producto	501
Producto	Habitación
Consecutivo	4
Marca del Producto	J. Suite
Presentación	---
Precio	580.00
folio	55

Podemos concluir en esta etapa de la normalización que obtenemos información importante de cada una de los prestadores, pero además, de igual forma el precio de cada uno de los productos que se ofrecen al usuario.

El código de la normalización en Paradox para DOS se incluye en el Apéndice B.

3.1.4 DIAGRAMA DE ENTIDAD-RELACION.



3.1.5 DICCIONARIO DE DATOS

El diccionario contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenidos y organización. También identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información. Por lo tanto, sirve como punto de partida para identificar los requerimientos de las bases de datos durante el diseño del sistema.

ESTRUCTURA DE LOS ARCHIVOS I_GIRO.DBF Y Z_GIRO.DBF

Estos archivos se encargan almacenar todos los bienes y servicios que el sistema ofrecerá al usuario, incluyendo una clave de giro que relacionará estos archivos con los archivos i_info.dbf y z_info.dbf.

CLAVE GIRO	CVE_GIRO	N	3
GIRO	GIRO	C	30
TIPO DE GIRO	TIPO-GIRO	C	1

Clave del giro	Nombre del Bien o Servicio	Tipo
1	Abarrotes	B
6	Farmacias	B
7	Labs. Clínicos	S
9	Electrico Automotríz	S

ESTRUCTURA DE LOS ARCHIVOS I_INFO.DBF Y Z_INFO.DBF

Estos archivos se encargan almacenar las coordenadas de los puntos en donde se ubican los bienes y servicios que el sistema ofrecerá al usuario, incluyendo una clave de giro que relacionará estos archivos con los archivos i_giro.dbf y z_giro.dbf.

REGISTRO	REG	N	4
COORDENADAX	X	N	4
COORDENADAY	Y	N	4
CLAVE DEL GIRO	CVE-GIRO	N	2

Registro	X	Y	Clave del giro
10	12	50	1
20	20	48	6
30	25	47	7

ESTRUCTURA DE LOS ARCHIVOS I_DATOST.DBF Y Z_DATOST.DBF

Estos archivos se encargan almacenar las coordenadas relacionandolas con los nombres de los bienes, servicios y atracciones turísticas que el sistema ofrecerá al usuario.

COORDENADA X	X	N	4
COORDENADA Y	Y	N	4
NOMBRE DEL LUGAR	LUGAR	C	30

Campo X	Campo Y	Nombre del Lugar
36	12	Hotel Ixtapa
50	72	Casa de bolsa
Xn	Yn	Lugar n

ESTRUCTURA DE LOS ARCHIVOS I_LUGT.DBF Y Z_LUGT.DBF

Estos archivos se encargan almacenar los nombres de las atracciones turísticas con los archivos de tipo TXT que almacenan la información cultural de dichas atracciones.

NOMBRE DEL LUGAR	LUGAR	C	20
NOMBRE DEL ARCHIVO	ARCHIVO	C	20

Nombre del lugar	Archivo en que se encuentra la información
Playa del Coral	ICORAL.TXT
Playa Cuachalalate	ICUACHA.TXT
Playa las Cuatas	ICUATAS.TXT

ARCHIVOS I_DATOS.DBF Y Z_DATOS.DBF

Estos archivos almacenan las coordenadas(x,y) de los puntos que forman los mapas de las ciudades de Ixtapa y Zihuatanejo.

Nombre	Nombre del campo	Tipo	Ancho
COORDENADA X	CAMPOX	N	4
COORDENADA Y	CAMPOY	N	4

ARCHIVOS I_SIT.DBF Y Z_SIT.DBF

Estos archivos almacenan la información turística de las dos ciudades.

Nombre	Nombre del Campo	Tipo	Ancho
Registro	Reg	N	5
Coordenada X	X	N	4
Coordenada Y	Y	N	4
Giro	Giro	N	2
Rázón Social	Razon_soc	C	40
Colonia	Colonia	C	40
Calle	Calle	C	5
Número	Numero	C	5
Código Postal	CP	C	5
Nombre Completo	Nombre_com	C	40
RFC	RFC	C	13
Propietario	Propiet	C	40
Teléfono	Tel	C	10
Horario 1	Horario1	C	10
Horario 2	Horario2	C	4
Horario 3	Horario3	C	30
Horario 4	Horario4	C	30
Principal servicio	Prin_serv	C	30
Categoria	Categoria	C	20

3.1.6 DISEÑO E IMPLEMENTACION DE LOS DIVERSOS MODULOS DE PROGRAMACION.

Para comenzar con la realización del proyecto se tomaron en cuenta sus requerimientos y elementos básicos con los que va a contar, tales como, la elaboración de un sistema gráfico, es decir, el ambiente visual que se le va a presentar al usuario, y los componentes que lo formarán, por ejemplo: dado que este sistema será de tipo informativo y que se ejecutará bajo ambiente Windows utilizará el concepto de ventanas, ya que mediante ellas se proporcionará la información respectiva que el usuario desee. Ahora bien, de este punto tenemos que considerar que probablemente por el tamaño de la información no se despliegue completamente en la ventana, asi que tenemos que implementar un objeto que nos permita visualizar poco a poco la información, y este objeto es la llamada barra de desplazamiento o de Scroll que nos permitirá movernos a través de la ventana.

Después de contemplar como se desplegará la información, consideramos la forma de accesar a esas ventanas y para ello se contará con una barra de menú con todos los comandos necesarios para su manejo, además de que el sistema también tendrá disponible una barra de herramientas la cual es un menú gráfico compuesto de iconos que son dibujos relativos a la función que realizan dentro del sistema.

Para tener acesso tanto a la barra de menú como a la barra de herramientas se utilizarán los dos dispositivos de entrada de la computadora, el teclado y el ratón. Por ejemplo, para utilizar la barra de menú mediante el teclado, el usuario presionará la tecla Alt para activar la barra y después con las teclas de navegacion moviendose de izquierda

a derecha; se colocará sobre el nombre del comando deseado para ejecutarlo. Para accesar con el Ratón a la barra de herramientas; la cual realiza la misma función que la barra de menú; se colocará el apuntador del ratón sobre el icono deseado y se presionará el botón izquierdo para ejecutar la acción.

Para desarrollar los requerimientos anteriormente descritos a cada uno de ellos se les colocará dentro del concepto de **Clases** para implementar sus funciones.

Para saber como podemos definir y formar dichas clases, utilizaremos los siguientes pasos:

- Identificación de las clases básicas
- Asignar atributos y comportamientos
- Encontrar las relaciones entre las clases
- Colocar las clases dentro de Jerarquías

Identificación de las clases

Para la identificación de las **clases básicas** que el sistema manejará; nos tenemos que hacer la siguiente pregunta ¿Qué clases son necesarias para este sistema?, una respuesta inmediata es la **clase ventana** ya que es el obje[redacted] se utilizará con más frecuencia. Entonces la primera clase es llamada **Ventana.**

Ahora bien; el sistema maneja las acciones del usuario con los dos dispositivos de entrada, el ratón y el teclado. Estas acciones son significativas para el sistema, porque ellas cambian el estado de una ventana; y de este punto, surge la segunda **clase** llamada **Evento** que describe tales estados de la ventana. Por ejemplo cuando un usuario presiona el botón del ratón sobre una ventana, una señal del **objeto Evento** es pasada al **objeto**

FALLA DE ORIGEN

Ventana, y este responde adecuadamente. Esto es un ejemplo de una clase que describe las cosas que le pasan a los objetos. Pero en el caso de que existan varias ventanas a la vez, debe de existir una clase que se encargue de mantener la información sobre todas las ventanas y las relaciones entre ellas y el usuario. Esta es la tercera clase y es llamada **Administrador de ventanas.** Dicha clase no modela una entidad mencionada en la descripción de nuestro sistema, es creada porque existe información que debe ser mantenida por el sistema, aunque esa información no será utilizada por otras clases. Es por esta razón que este sistema frecuentemente maneja posiciones, y para representar a estas es conveniente utilizar una cuarta clase llamada **Punto** para que las controle. Análogamente el sistema utiliza ventanas en forma rectangular, entonces se declara, la última clase básica, **la clase Rectángulo** que se encarga de manejar la forma de las ventanas.

Atributos y comportamientos

Para realizar este punto nos hacemos la siguiente pregunta ¿Qué información manejan las clases? por ejemplo la clase Administrador de ventanas controla los eventos generados por el usuario.

La clase Ventana se encarga de conocer el tamaño y la posición de las pantallas del sistema, además de almacena toda la información que debe ser desplegada por una ventana que se desplace, incluyendo el texto de afuera de la ventana. La ventana conoce su posición, la cual relaciona al texto completo con la barra de desplazamiento.

Por ejemplo consideremos el comportamiento de la barra de desplazamiento más de cerca. Si el usuario se mueve hacia el final del documento, el texto se mueve una linea a la vez en esa dirección.

FALLA DE ORIGEN

En el parráfo anterior se describe un conjunto de acciones bastante complejo para ser ejecutadas por una sola ventana. Más aún, si sabemos que una ventana puede tener dos barras de desplazamiento, una vertical y una horizontal, esto trae como consecuencia que para la realización de este proceso se requiere repetir muchas veces el código para que responda a la señal del ratón. En vez de hacer que una sola ventana sea responsable del comportamiento de la barra de desplazamiento, se puede definir una clase separada que se encargue del funcionamiento de estas barras. Con esto se logra que una ventana sea la responsable de la interpretación de la señal del ratón sobre el texto, y que la barra de desplazamiento sea la responsable de la interpretación de la señal del ratón en su superficie. Y asi; los objetos de esas dos clases tanto la barra como la ventana interactuan para ejecutar el desplazamiento en el texto. Una de las funciones principales de la barra es conocer el tamaño de la ventana y la posición sobre la pantalla.

Y de esto encontramos que el propósito de la información que almacena el objeto Evento es determinar el tipo de información que maneja; por ejemplo, si la señal de entrada viene del teclado, entonces la información relevante es la tecla que fue presionada. Si la señal de entrada viene del ratón, la información relevante es la localización del cursor del ratón y de los botones del ratón.

Estos procesos, son dos distintos tipos de eventos, que se pueden implementar en dos clases separadas, llamadas OnLButtonDown y OnRButtonDown
¿Qué hacen las clases OnLButtonDown y OnRButtonDown?. Estas dos clases no hacen mucho, excepto transportar la información que contienen.

Identificación de las relaciones entre las clases

Una ventana puede tener una o dos barras de desplazamiento. Cada barra esta definida como una ventana hija y cada ventana es la madre de las barras de

desplazamiento, esto puede ser representado mediante un diagrama de relaciones, donde la clase Ventana contiene una o dos instancias de la barra de desplazamiento.

La interacción entre las ventanas y las barras requieren dos formas de comunicación, una es utilizando las barras de desplazamiento que afectan a la ventana y la otra es moviendo el cursor de la ventana para que afecten a la barra. Como resultado, la barra debe conocer el tamaño de las ventanas.

Así los eventos son pasados para interactuar con las diferentes ventanas o barras de desplazamiento, entonces el interactor debe tener una función que acepte un objeto evento como un parámetro.

Interfaces

La jerarquía ventana consiste de una clase base que interactua con las clases derivadas, Ventana y Barra de desplazamiento. Vamos a considerar estas clases moviéndonos desde el principio de la jerarquía hacia abajo.

Las características que tienen en común todos los interactores, incluyendo las ventanas de texto y las barras de desplazamiento, son que tienen un tamaño y una posición relativa y esta información debe ser accesible. Ya que ellas deben de saber si un par de coordenadas dado, cae dentro de sus bordes, indicando con esto que deberán responder a una señal del ratón. En resumen todos los interactores deben tener la capacidad para desplegarse ellos mismos y para responder a los eventos.

Colocar las clases dentro de jerarquías

Las jerarquías son muy sencillas en este punto, así que no se necesita reestructurarlas. Este paso se puede realizar hasta que termine el proceso de diseño, sin embargo es apropiado decidir que características pertenecen a la clase base y que características pertenecen a las clases derivadas. Esto requiere de una examinación más profunda de los atributos y del comportamiento de las clases, así que primeramente se tiene que hacer un análisis de las clases con interfaces públicas, que son las siguientes:

Clase Ventana

La jerarquía ventana consiste de una clase base que interactua con las clases derivadas ***Ventanas*** y ***Barra de desplazamiento***. Vamos a considerar estas clases comenzando, de arriba hacia abajo.

Las características que tienen en común todos los interactores incluyendo las barras de desplazamiento y las ventanas de texto, es que todas ellas tienen un tamaño y una posición relativa, y esta información debe ser accesible, ya que deberán registrar cuando un par de coordenadas dado, caiga dentro de sus bordes indicando que deben responder a una señal del ratón. En resumen, todos los interactores tienen la capacidad para desplegarse ellos mismos y de responder a los eventos (señales del ratón). Sin embargo como un interactor no es un objeto genérico que pueda desplegarse el mismo o de responder a los eventos, esas funciones deben ser declaradas como funciones virtuales puras.

Ventanas de texto

Una ventana de texto almacena toda la información que va a ser desplegada. Un arreglo puede ser usado para manejar texto, algunas veces de tamaño variable como en

las listas ligadas, para hacerlo más flexible, pero en este caso cada ventana tiene una cantidad de texto fija.

Cada ventana puede contener una o dos barras de desplazamiento o ninguna de ellas, dependiendo de la longitud y el ancho del texto que va a ser desplegado.

Barras de Desplazamiento(scroll)

Una barra de desplazamiento conoce su función; ya sea si es vertical u horizontal, y la información que representa. Conoce la posición de la caja de desplazamiento la cual puede ser representada como un número entre 1 y 100.

Una barra de desplazamiento debe tener el ancho de un caracter si es vertical y el alto de un caracter si es horizontal. Y no puede estar separada de una ventana de texto, es más la ventana de texto debe de existir antes que la barra de desplazamiento sea creada.

También existe otras funciones que manejan la barra de desplazamiento por ejemplo si el usuario desplaza una ventana de texto moviendo el cursor con las teclas de navegación, la barra de desplazamiento de la caja debe de moverse también. No es muy apropiado manejar desde el teclado esta barra, ya que esto rompe con su propósito ademas de ser muy incomodo, la ventana de texto es la encargada de manejarla. La barra de desplazamiento debe sin embargo tener una función que fije la posición de la caja la cual se llama ***Ventana.***

Cuando el botón del ratón es presionado sobre la barra de desplazamiento, el texto en la ventana tambien se desplaza. El objeto ***Barra de desplazamiento*** dice que se asocie con el objeto ***Ventana*** para ejecutar la acción de desplazamiento, puede hacerse esto

enviando un nuevo tipo de evento, un ***DesplazamientoEvento***, que contiene la dirección (hacia atrás y hacia adelante) y la cantidad de texto (una línea o una página).

Una barra de desplazamiento puede mover su propia caja de desplazamiento, ¿pero que tanto puede moverla?; primero se tiene que considerar la longitud del documento y el ancho de la ventana. Por ejemplo si el texto es cuatro veces tan largo como la ventana, desplazar una página hacia abajo significa mover la caja de desplazamiento ¼ de la barra de desplazamiento hacia abajo. Sin embargo si el texto es 20 veces tan largo como la ventana, desplazar una pagina hacia abajo significa mover la caja de desplazamiento 1/20 de la barra de desplazamiento hacia abajo. Desplazar una línea a la vez es un proceso similar. Esto es, calcular la distancia que la caja de desplazamiento puede mover la barra entonces, deberá de conocer la longitud del documento y la altura de la ventana (o el porcentaje correspondiente del desplazamiento horizontal). Esto podría duplicar la responsabilidades, porque esos valores son almacenados por la clase Ventana, sin embargo la barra de desplazamiento no mueve su caja de desplazamiento cuando el ratón es accionado, únicamente ajusta la caja cuando la clase Ventana llama la función setSlider. La clase Ventana es responsable de computar apropiadamente la posición de la caja de desplazamiento.

El administrador de ventanas

La clase administrador de ventanas registra el orden de las ventanas de texto y los eventos hechos por el usuario.

Sobre la pantalla, las ventanas aparecen como un pila de hojas de papel sobre el escritorio, siendo la ventana de la cima la única ventana activa, esto sugiere una clase que implemente una pila como estructura de datos, sin embargo una pila permite accesar únicamente al elemento de la cima. El Administrador de ventanas debe accesar a las otras

ventanas sin desactivar la actual ventana. Alguna clase de lista permite accesar a todos los elementos que sean necesarios.

El administrador de ventanas debe accesar estas para ver el orden en que reciben una señal del ratón. Más que una ventana debe registrar la posición donde el ratón fue accionado pero únicamente la ventana expuesta es la que recibe la señal. Para encontrar la ventana expuesta, el administrador de ventanas debe de probar el orden de las ventanas comenzando desde la ventana de la cima de la pila hacia abajo. Sin embargo, la pantalla debe refrescarse sobreponiendo las ventanas otra vez. Esta vez, el administrador debe accesar las ventanas comenzando desde la ventana de abajo de la pila hacia arriba. Así, el administrador de ventanas necesita una clase que permita la iteración de sus elementos en ambas direcciones.

Por consiguiente, una clase Lista que permite las operaciones en ambas direcciones es la apropiada, tal lista deberá ser implementada con una doble lista ligada o en un arreglo. La clase administrador de ventanas puede tener un objeto miembro de tipo lista.

Consideramos como las clases Administrador de ventanas, Ventana, y Barra de desplazamiento trabajan juntas. Cuando el usuario acciona el ratón sobre la barra de desplazamiento y sobre la ventana de texto. El objeto Evento contiene la locación donde la señal del ratón ocurre la cual es enviada al objeto administrador de ventanas ya que esta clase es la encargada de organizar la ventanas y busca cual de ellas recibió la señal del ratón esta pasa la señal al objeto Ventana para verificar si las barras de desplazamiento están siendo accionadas.

Todo el proceso anterior que describe la comunicación entre las ventanas y el usuario se ilustra en la siguiente figura 3.1.6.1.

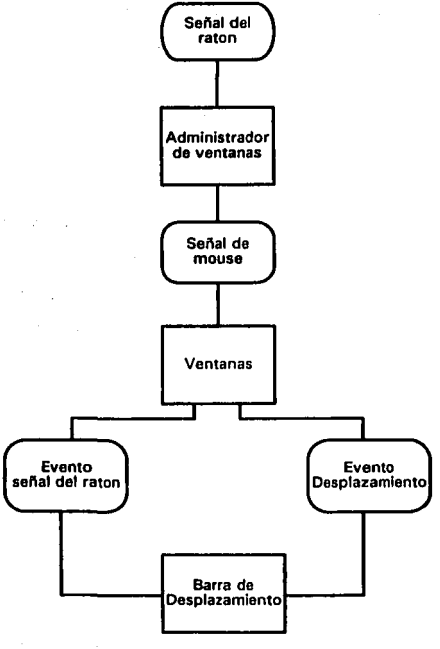


Fig. 3.1.6.1 Administrador de Ventanas.

Uso de App Wizard

Una vez que sabemos como ir identificando las clases y su comportamiento en nuestro sistema, procederemos a hacer uso de una de las herramientas que el lenguaje nos ofrece para definirlas de una manerá fácil. El lenguaje Visual C++ utiliza tres poderosas herramientas que nos ayudan en la creación de una aplicación de Windows y una de ellas es la utileria App Wizard (las otras dos son Class Wizard y App Studio) que se emplea para la creación de codigo fuente de manera automatica, con esto se genera el código básico del programa, el cual se puede introducir ya sea manualmente o utilizando las herramientas Class Wizard y App Studio. La utileria App Wizard emite una serie de archivos que contienen varias clases que construyen la aplicación. La utileria Class Wizard nos permite usar el código emitido por App Wizard, para generar otras clases que complementan la aplicación, la utileria App Studio a su vez también nos permite usar varios de los archivos generados por App Wizard para generar otros componentes del sistema.

La rutina que se sigue para generar estos archivos es la siguiente:

1. Se entra a la aplicación del Visual C++ y se invoca la opción **App Wizard** del menú ***project***.

   Se teclea Site como nombre del proyecto y se selecciona el directorio donde se almacenará este último.
2. Se presiona el botón con la etiqueta ***Options***, de este paso aparece la caja de dialogo "Options" en la cual marcaremos la opción de barra de herramientas inicial, que va a contener nuestra aplicación.
3. Se presiona el botón de ***OK*** de la caja de diálogo Options para regresarnos a la pantalla del App Wizard.
4. Se presiona el botón con la etiqueta ***Classes*** para abrir su caja de diálogo la cual lista la siguientes clases:

- La clase CSiteApp, la cual modela la aplicación SITE
- La clase CMainFrame, la cual maneja la estructura de las ventanas
- La clase CSiteDoc que modela el documento de aplicación
- La clase CSiteWiew la cual representa la apariencia de la aplicación

5. Se presiona el botón ***OK*** para regresar a la pantalla de App Wizard
6. Aparece una ventana con el resumen de todas las clases que se van a generar, entonces se oprime el botón con la etiqueta ***Create*** para construir el código fuente del archivo que genera el proyecto.

La generación de archivos termina y se puede listar los archivos abriendolos seleccionando la opción de abrir ya sea en la barra de menú o en la barra de herramientas de la pantalla principal del Visual C++.

Los archivos que generó App Wizard son los siguientes:

MAINFRM.CPP

MAINFRM.H

RESOURCE.H

SITEDOC.CPP

SITEDOC.H

SITEWIEW.CPP

SITEWIEW.H

SITE.DEF

SITE.H

SITE.MAK

SITE.RC

STDAFX.CPP

STDAFX.H

Los archivos con la extensión .CPP son los archivos de implementación de las clases, los archivos con la extensión .H contienen la definición de las estructuras llamadas ID´s que son las variables que se encargan de ligar las ventanas con los comandos del menú principal. El archivo con la extensión .DEF define el módulo de parámetros que la aplicación maneja declararando el nombre del archivo y la descripción del mismo. El archivo con la extensión .MAK es el archivo que se genera cuando se ligan todos los archivos para construir el sistema. El archivo con la extensión .RC mantiene la informacion que controla apariencia visual del sistema.

Para construir el archivo ejecutable se selecciona la opcion ***Build Execute*** del menú ***project***. Entonces el archivo que se genera contiene un menú y una barra de herramientas inicial con diferentes opciones, las cuales nosotros modificamos para diseñar nuestra propia aplicación.

Para ello se crearon otras clases para complementar el proposito del sistema y para lograrlo nos valimos de otras de las herramientas del lenguaje que es **Class Wizard**, esta se utiliza para ir declarando el nombre de otras clases en el archivo CSiteView. La implementación de la función de cada una de estas clases se hizo manualmente.

Los siguientes párrafos se basarán en la explicación de las funciones de las clases dentro del algoritmo del sistema, además de que se anexará el código de dichas funciones.

Algoritmo del sistema

Primeramente se hará una declaración de las variables públicas y de los archivos que las clases van a utilizar más adelante y el código es el siguiente:

```
1.  / siteview.cpp :Implementación  de la clase CSiteView
2.  #include "stdafx.h"
3.  #include "site.h"
4.  #define MARCA 9999
5.  #include "sitedoc.h"
6.  #include "siteview.h"
7.  #include "dradio.h"
8.  #include "dcuadro.h"
9.  #include "dselecc.h"
10. #include "dbienser.h"
11. #include "dinfo.h"
12. #ifdef _DEBUG
13. #undef THIS_FILE
14. static char BASED_CODE THIS_FILE[] = __FILE__;
15. #endif
16. typedef struct    {     entero x;
17. entero y;
18. char titulo[31]; }      desc_sit;
19. typedef struct   {     entero reg;
20. entero x;
21. entero y;
22. byte   giro; } lugar_sit;
23. entero *a_apt;                          // Apuntador
```

```
24.  entero *a_lin;                          // puntos del mapa
25.  int ancho = 4;                          // relleno de los puntos rojos
26.  double acerca = .05;                    // inicialización del aumento
                                                del sistema en 5%
27.  byte ciudad = 0, limite = 1,            // inicialización de variables
     giro_act = 0, examinar=0, Amplificar=1;
28.  long max_x, max_y, min_x, min_y;        //declaración de la variables
                                             que controlan el tamaño de la
                                             ventanas y el mapa
29.  long *stack, *a_stack;                  // declaración de la pila
30.  double derecho, abajo, dif_x,dif_y;
31.  char *na_datos = "i_datos.dbf",         //declaración de las bases de
                                             datos
     *na_info = "i_info.dbf",
     *na_datost = "i_datost.dbf";
32.  char *na_lugt = "i_lugt.dbf",
     *na_giro = "i_gir.dbf",
     *na_sit = "i_sit.dbf";
33.  CFile fsit;
34.  desc_sit  *a_dat;                       // sitios en el mapa
35.  lugar_sit *a_inf;                      // Relación de Bienes y Servicios en
                                                el archivo
36.  lugt_sit  *a_lugt;                     // lista de lugares turísticos
37.  desc_giro *a_giro;                     // Relación de giros
```

A) En primer lugar el usuario seleccionará la ciudad sobre la cual desea obtener información, ya sea Ixtapa o Zihuatanejo.

Para que el sistema diferencie entre las dos ciudades, se tomará el estado de una variable llamada **ciudad** cuyo valor variará entre 0 y 1, inicilalizada primeramente con el valor de 0, el cual corresponde a la ciudad de Ixtapa y el valor de 1 le corresponde a la ciudad de Zihuatanejo. El estado de esta variable permitirá abrir los archivos de texto (archivos con la extensión TXT) para que se lea la información cultural de las ciudades.

Los siguientes codigos ejecutan el proceso anterior:

```
1.  void CSiteView::OnCiudadIxtapa()
2.  {
3.  CClientDC dc( this );
4.  CRect rect;
5.  GetClientRect( rect );
6.  ciudad = 0;                          //variable ciudad
7.  na_datos  = "i_datos.dbf";
8.  na_info   = "i_info.dbf";
9.  na_datost = "i_datost.dbf";
10. na_lugt   = "i_lugt.dbf";
11. na_giro   = "i_gir.dbf";
12. na_sit    = "i_sit.dbf"; // archivo que contiene la ubicación
                             de los sitios de Ixtapa
13. free(a_inf);
14. free(a_lin);
```

```
15.  free(a_dat);
16.  free(a_lugt);
17.  free(a_giro);
18.  giro_act = 0, examinar=0, Amplificar=1;
19.  leer();
20.  fsit.Close();
21.  fsit.Open(na_sit,CFile::modeRead);    //abre el archivo i_sit.dbf
22.  InvalidateRect( rect ,TRUE);
23.  }


 1.  void CSiteView::OnCiudadZihuatanejo()
 2.  {
 3.  CClientDC dc( this );
 4.  CRect rect;
 5.  GetClientRect( rect );
 6.  ciudad = 1;                  //variable ciudad
 7.  na_datos  = "z_datos.dbf";
 8.  na_info    = "z_info.dbf";
 9.  na_datost = "z_datost.dbf";
10.  na_lugt    = "z_lugt.dbf";
11.  na_giro   = "z_gir.dbf";
12.  na_sit    = "z_sit.dbf";  //archivo  que contiene la ubicación
                                //de los sitios de Zihuatanejo
13.  free(a_inf);
14.  free(a_lin);
15.  free(a_dat);
```

```
16.   free(a_lugt);
17.   free(a_giro);
18.   giro_act = 0, examinar=0, Amplificar=1;
19.   leer();
20.   fsit.Close();
21.   fsit.Open(na_sit,CFile::modeRead); // abre el archivo z_sit.dbf
22.   InvalidateRect( rect,TRUE);
23.   }
```

Dependiendo del estado de la bandera **ciudad** ya sea si es 0 o 1 se abre el archivo de texto **"zihuata.txt"o"ixtapa.txt"**, que contiene la información cultural correspondiente. El código que despliega ésta es el siguiente:

```
1.    void CSiteView::OnSiteConocer()
2.    {
3.
4.    DCuadro dlg;              // ventana que despliega la información
5.    CFile temp;
6.    double tam;
7.    char *buffer;
8.temp.Open((ciudad)?"zihuata.txt":"ixtapa.txt",CFile::modeRead); //abre
                                                    archivo de texto
9.    tam = temp.GetLength();
10.   buffer = (char*) malloc(tam+1);   // reserva espacio de memoria
                                          //para los archivos de texto
11.   temp.Read(buffer,tam);
12.   temp.Close();
```

```
13.   *(buffer+(int)tam) = '\0';
14.   dlg.m_titulo = (ciudad)?"ZIHUATANEJO":"IXTAPA"; //despliega el
                                                    //titulo en la ventana
                                                    //que se abre
15.   dlg.m_texto = buffer;
16.   dlg.DoModal(); // declaración de que tipo de ventana que se abre
17.   free(buffer); //cuando termina libera la memoria reservada
18.   }
19.   }
```

El procedimiento que lee los datos y que es llamado en las dos clases anteriores es el siguiente. El procedimiento se llama leer().

```
1.    void leer() {
2.    CFile temp;
3.    char *buff, *c_apt4, *c_apt5, *c_apt2;
4.    entero *a_apt, x, y, i, n_apt;
5.    desc_sit *p_dat;
6.    lugar_sit *i_apt;
7.    lugt_sit *l_apt;              //Apuntadores de las bases de datos
8.    desc_giro *g_apt;
9.    buff = (char*) malloc(200);
10.   c_apt4 = (char*) malloc(5); *(c_apt4+4) = '\0';//reserva de memoria
11.   c_apt5 = (char*) malloc(6); *(c_apt5+5) = '\0';//reserva de memoria
12.   c_apt2 = (char*) malloc(3); *(c_apt2+2) = '\0';//reserva de memoria
```

B) Dependiendo de la ciudad elegida se desplegará el mapa respectivo de la ciudad. Esto se hace mediante un procedimiento de lectura de datos de los archivos **"i_datos.dbf"** o **"z_datos.dbf"** que son los archivos que contienen los puntos en coordenadas X,Y que forman los mapas de las ciudades.

El siguiente código ejecuta el proceso anterior:

```
1.  // Lectura de puntos del mapa
2.  temp.Open(na_datos,CFile::modeRead);  //Esta instrucción abre el
                                          //archivo de datos i_datos.dbf o
                                          //z_datos.dbf
3.  temp.Read(buff,4);                   // salta 4 bytes
4.  temp.Read(&n_apt,2);   // Pregunta por la cantidad de registros del
                           //archivo .DBF
5.  temp.Read(buff,2);          // salta dos bytes
6.a_lin = (entero*) malloc((n_apt*2+1)*sizeof(entero));  // Esta
                                          //instrucción reserva memoria para
                                          //los puntos del mapa
7.  a_apt = a_lin;
8.  temp.Read(buff,90);
9.  for(i=1;i<=n_apt;i++) {//Inicio del bloque de lectura de datos
10. temp.Read(c_apt4,1);
11. temp.Read(c_apt4,4); x = atoi(c_apt4);
12. temp.Read(c_apt4,4); y = atoi(c_apt4);
13. *a_apt++ = x;
14. *a_apt++ = y;
15. };
```

```
16.   *a_apt++ = FINAL;
17.   temp.Close();                    //Final del bloque de lectura
```

C) Una vez que el mapa de la ciudad elegida se visualice sobre la pantalla, el usuario seleccionará el tipo de informacion que desea obtener, ya sea un bien, servicio o atracción turística. Para diferenciar entre estos, el sistema lee dos archivos **"z_giro.dbf"** o **"i_giro.dbf"** ya que estos almacenan toda la información de cada ciudad. En la estructura de estos archivos se encuentra un campo llamado **Tipo** que es el que distingue entre el tipo de información ya sea si es un Bien o un Servicio, y lo que hace es almacenar el valor de 0 cuando se refiere a Bienes y almacena el valor de 1 cuando se trata de Servicios y estos a su vez estan asociados a un nombre del lugar y a una clave que se le asigna a cada lugar. La selección de Atracciones Turísticas se encuentra implementada en otra clase que se explica en el paso 6.

La estructura de este archivo es la siguiente:

Clave del giro	Nombre del Bien o Servicio	Tipo
1	Abarrotes	0
6	Farmacias	0
7	Labs. Clínicos	1
9	Electrico Automotríz	1

La elección del tipo de información la realizará el siguiente código:

```
1.  // Lectura de Relación de giros
2.  temp.Open(na_giro,CFile::modeRead); //distingue entre el tipo
                                        //de información
3.  temp.Seek(4,CFile::begin);
4.  temp.Read(&n_apt,2);
5.  temp.Seek(2,CFile::current);
6.  a_giro = (desc_giro*) malloc((n_apt+1)*sizeof(desc_giro));
7.  g_apt = a_giro;
8.  temp.Seek(121,CFile::current);
9.  for(i=1;i<=n_apt;i++) {
10. temp.Seek(1,CFile::current);
11. g_apt->marca = 0;
12. temp.Read(c_apt2,2);          g_apt->giro = atoi(c_apt2);
13. temp.Read(g_apt->nombre,36);          g_apt->nombre[36] = '\0';
14. temp.Read(c_apt4,1);          g_apt->tipo = *c_apt4 - '0';
15. *g_apt++;
16. };
17. g_apt->marca = FINAL;
18. temp.Close();
```

D) Al seleccionar el tipo de información, se abrirá una ventana que despliega la lista con todos los nombres de los Bienes, Servicios y Atracciones Turísticas que las ciudades ofrecen. De esa lista el usuario seleccionará el nombre de un lugar y al hacerlo, sobre el mapa se visualizará, con puntos rojos, la ubicación del sitio o sitios donde se encuentran dichos lugares. Este proceso lee los archivos **"i_info.dbf" o "z_info.dbf"**,

ya que estos almacenan el número de registro, la ubicación en cordenadas X,Y y la clave del giro ya que esta se asocia con el nombre de la ubicación del sitio el cual se encuentra en los archivos "i_giro.dbf" y "z_giro.dbf" anteriormente descritos. La estructura de esos archivos es la siguiente:

Registro	X	Y	Clave del giro
10	12	50	1
20	20	48	6
30	25	47	7

El siguiente código ejecuta el proceso anterior:

```
1.  //Lectura ubicacion en el archivo de Bienes y Servicios
2.  temp.Open(na_info,CFile::modeRead);
3.  temp.Read(buff,4);
4.  temp.Read(&n_apt,2);
5.  temp.Read(buff,2);
6.  a_inf = (lugar_sit*) malloc( (n_apt+1)*sizeof(lugar_sit));
7.  i_apt = a_inf;
8.  temp.Read(buff,153);
9.  for(i=1;i<=n_apt;i++) {
10. temp.Read(c_apt5,1);
11. temp.Read(c_apt5,5); i_apt->reg  = atoi(c_apt5);
```

```
12.   temp.Read(c_apt4,4); i_apt->x    = atoi(c_apt4);
13.   temp.Read(c_apt4,4); i_apt->y    = atoi(c_apt4);
14.   temp.Read(c_apt2,2); i_apt->giro = atoi(c_apt2);
15.   i_apt++;
16.   };
17.   i_apt->x = FINAL;
18.   i_apt->y = FINAL;
      temp.Close();
```

E) Sobre los puntos rojos aparece un letrero que indica el nombre de ese sitio. La colocación de esos letreros se hace mediante un proceso de lectura de los archivos **"i_datost.dbf"** y **"z_datost.dbf"**, ya que estos se encargan de relacionar el punto (coordenada (x,y)) con el nombre que corresponde a ese sitio.

La estructura de esos archivos es la siguiente:

Campo X	Campo Y	Nombre del Lugar
36	12	Hotel Ixtapa
50	72	Casa de bolsa
Xn	Yn	Lugar n

El siguiente código ejecuta el proceso anterior:

```
1.   // Lectura sitios en el mapa
```

```
2.   temp.Open(na_datost,CFile::modeRead); //abre el archivo
                                              //datost.dbf
3.   temp.Read(buff,4);
4.   temp.Read(&n_apt,2);
5.   temp.Read(buff,2);
6.   a_dat = (desc_sit*) malloc((n_apt+1)*sizeof(desc_sit));
7.   p_dat = a_dat;
8.   temp.Read(buff,121);
9.   for(i=1;i<=n_apt;i++){
10.  temp.Read(c_apt5,1);
11.  temp.Read(c_apt4,4); p_dat->x = atoi(c_apt4);
12.  temp.Read(c_apt4,4); p_dat->y = atoi(c_apt4);
13.  temp.Read(p_dat->titulo,30);
14.  p_dat->titulo[30] = '\0';
15.  p_dat++;
16.  };
17.  p_dat->x = FINAL;
18.  p_dat->y = FINAL;
19.  temp.Close();
```

F) Al elegir uno de los nombres de la lista de Atracciones Turísticas, se desplegará una ventana con la información cultural de esa atracción en especial. Para obtener dicha información, el sistema abrirá los archivo "i_lugt.dbf" o "z_lugt.dbf" ya que estos almacenan todos los nombres de los lugares asociadas con los nombres de los archivos de texto que contienen la información cultural de dichos lugares. La estructura de los archivos es la siguiente:

Nombre del lugar	Archivo en que se encuentra la información
Playa del Coral	ICORAL.TXT
Playa Cuachalalate	ICUACHA.TXT
Playa las Cuatas	ICUATAS.TXT

El siguiente código ejecuta el proceso anterior:

```
1.  // Lectura de lugares turísticos
2.  temp.Open(na_lugt,CFile::modeRead);
3.  temp.Read(buff,4);
4.  temp.Read(&n_apt,2);
5.  temp.Read(buff,2);
6.  a_lugt = (lugt_sit*) malloc((n_apt+1)*sizeof(lugt_sit));
7.  l_apt = a_lugt;
8.  temp.Read(buff,89);
9.  for(i=1;i<=n_apt;i++) {
10. temp.Read(c_apt5,1);
11. l_apt->marca = 0;
12. temp.Read(l_apt->nombre,36);
13. temp.Read(l_apt->archivo,12); // lectura del archivo.TXT que
                                  //contiene la información
14. l_apt->nombre[36] = '\0';
15. l_apt->archivo[12] = '\0';
16. l_apt++;
17. };
```

```
18.   l_apt->marca = FINAL;
19.   temp.Close();
20.   free(buff);
21.   free(c_apt5);
22.   free(c_apt4);
23.   free(c_apt2);
24.   }
25.   void trans(long x, long y, long *xp, long *yp) {
26.   *xp = ((x-min_x)/dif_x) * derecho;
27.   *yp = (1 - ((y-min_y)/dif_y)) * abajo;
28.   }
```

G)Se establece si se desea hacer un acercamiento (zoom) teniendo 3 diferentes opciones de aumento 5%, 25% y 50% . Esto se logra colocando el apuntador del ratón sobre la zona que se quiera observar con más detalle y presionado el botón del mismo.

El siguiente código describe como el sistema diferencia entre los tipos de aumento que el usuario selecciona.

```
1.   void CSiteView::OnUbicarRadiodeaccin()
2.   {
3.   DRadio dlg;
4.   if (acerca <= 0.05)    // como el 5% es el aumento inicial. Se hace
                            //a partir de ese valor la comparación
5.   dlg.m_radio1 = 0;
6.   else if (acerca <= 0.25) //Pregunta si se escogió el 25% de aumento
7.   dlg.m_radio1 = 1;
8.   else if (acerca <= 0.50) //Pregunta si se escogio el 50% de aumento
9.   dlg.m_radio1 = 2;
```

```
10.  if (dlg.DoModal()==1) {
11.  if (dlg.m_radio1 == 0)
12.  acerca = 0.05;
13.  else if (dlg.m_radio1 == 1)
14.  acerca = 0.25;
15.  else if (dlg.m_radio1 == 2)
16.  acerca = 0.50;
17.  };
```

Anteriormente se mencionó que una de las principales funciones del sistema, es desplegar los mapas de las respectivas ciudades, los cuales van a responder a una señal del ratón o del teclado. Para lograr lo anterior se manejaran dos archivos con formato DBF (la extensión .DBF siginifica archivo de base de datos) que almacenarán la información con los puntos que forman los dos mapas en coordenadas (X,Y), de ahí a cada punto del mapa se le asignará la ubicación de algún sitio en particular, pudiendo ser este algún Bien, Servicio o Atracción Turística y a cada uno de estos lugares se almacenarán en dos bases de datos siendo estas, "i_datost.dbf" y "z_datos.dbf."

Los archivos "i_datost.dbf" y "z_datost.dbf" tendrán la siguiente estructura :

Coordenadas Xd	Coordenadas Yd	Tipo	Descripción
Xd1	Yd1	Base de datos servicios	Banco
Xd2	Yd2	Base de datos Bienes	Hotel
Xd3	Yd3	Base de datos Atracciones Turísticas	Playa el Palmar
Xdn	Ydn		Banco n

donde:

n: número de puntos

Tipo: especificar el nombre de la base de datos en que se encuentra ese punto

Descripción : nombre del lugar al que corresponde ese punto

Y así a cada sitio que se seleccione sobre el mapa, inmediatamente el sistema identificará la posición que ocupa en la ventana y buscará en los archivos **"i_datost.dbf"** y **"z_datost.dbf"** a que punto corresponde ese sitio del mapa. Cuando lo haya hecho, buscará el archivo que contiene la base de datos en que se encuentra ese lugar y desplegará una ventana de texto con la información respectiva de ese sitio.

Para realizar el cambio de aumento se hará una transformación de coordenadas para la graficación de los mapas.

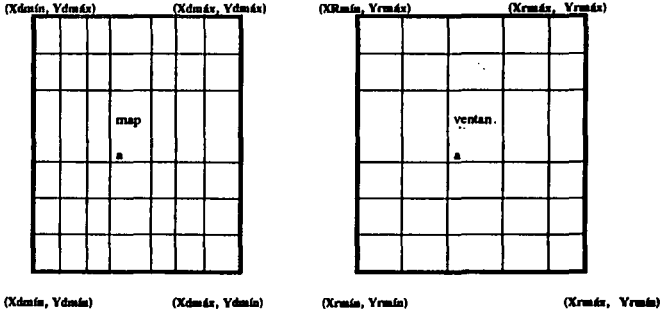
El proceso anterior se define de la siguiente manera:

Se considerarán dos objetos una ventana y un mapa, los cuales estarán representados como dos matrices formadas de renglones llamados (X) y columnas llamadas (Y). Para diferenciar los renglones y las columnas que forman el mapa y la ventana, a los renglones y a las columnas del mapa les llamaremos (Xd,Yd) respectivamente y a los renglones y a las columnas de la ventana les llamaremos (Xr,Yr) también respectivamente.

Los rangos de los renglones y las columnas serán:

Renglones Xd = Xdminima . . .a Xdmaxima;
Columnas Yd = Ydminima . . . a Ydmaxima;
Renglones Xr = Xrminima . . . a Xrmaxima;
Columnas Yr = Yrminima . . . .a Yrmaxima;

En la siguiente figura se ilustra como quedarían formadas las matrices:



Donde:

(Xd, Yd) : Representan los datos de los puntos del mapa

y

(Xr, Yr) : Representan los puntos de la ventana.

Para poder hacer la transformación de coordenadas entre las ventanas y los mapas se utilizarán las siguientes ecuaciones:

$$\frac{Xr - Xrmin}{Xrmax - Xrmin} = \frac{Xd - Xdmin}{Xdmax - Xdmin} \quad \text{.....(1)}$$

$$Xr = \left(\frac{Xd - Xdmin}{Xdmax - Xdmin}\right)(Xrmax - Xrmin) + Xrmin \quad \text{.....(2)}$$

$$\frac{Yr - Yrmin}{Yrmax - Yrmin} = \frac{Yd - Ydmin}{Ydmax - Ydmin} \quad \text{.....(3)}$$

$$Yr = \left(\frac{Yd - Ydmin}{Ydmax - Ydmin}\right)(Yrmax - Yrmin) + Yrmin \quad \text{.....(4)}$$

En las ecuaciones (1) y (3) se hace una igualdad entre los datos del mapa y la ventana, con el propósito de hacer corresponder el tamaño de la ventana con el tamaño del mapa.

Las ecuaciones (2) y (4) realizan el despeje del punto mínimo o máximo en coordenadas X,Y para obtener su valor.

Un acercamiento se producirá cuando incrementemos los puntos mínimos y decrementemos los puntos máximos del mapa.

Las relaciones dadas por las ecuaciones (1) y (2) se deben de conservar para no deformar la figura del mapa dentro de la ventana, la cual se formará al restar y sumar los puntos máximos y mínimos de las matrices.

Realizando el proceso anterior, entonces se podrá relacionar la clase mapa con las otras clases para obtener la información respectiva de ese punto seleccionado en particular, pudiendo realizar la función de imprimir dicha gráfica del punto con su respectiva información.

El siguiente código es el que se encarga de ejecutar todo el proceso anterior. Para realizar los acercamientos (aumentos) se presinará el botón izquierdo del ratón:

```
1.  void CSiteView::OnLButtonDown(UINT nFlags, CPoint point)
2.  {
3.  CView::OnLButtonDown(nFlags, point);
4.  long xp, yp;                          //, x_inf, y_inf;
5.  long x, y, inc_x, inc_y, mid_x, mid_y;
6.  double fact_m;
8.  unsigned int cont = 0, largo;
9.  lugar_sit *i_inf;
10. DInfo dlg;
11. char *nom_com, *calle, *numero, *colonia, *cp, *ho1, *ho2;
12. char *ho3, *ho4, *pser, *cat, *tel, *rz, *prop, *rfc;
13. CClientDC dc( this );
14. CRect rect;
15. GetClientRect( rect );
16. xp = point.x;
```

```
17.  yp = point.y;
18.  x = (xp/derecho)*dif_x + min_x   ;
19.  y = (1 - yp/abajo)*dif_y + min_y ;
20.  if (Amplificar) {
21.  mid_x = (long) dif_x/2.0;     // implementación de la ecuaciones 1,
                                   //2 , 3,y 4
22.  mid_y = (long) dif_y/2.0;
23.  inc_x=mid_x*(acerca/2.0);
24.  inc_y=mid_y*(acerca/2.0);
25.  if (x - (mid_x - inc_x) < min_x) {
26.  max_x = min_x + (dif_x - 2*inc_x);
27.  } else {
28.  if (x + (mid_x - inc_x) > max_x) {
29.  min_x = max_x - (dif_x - 2*inc_x);
30.  } else {
31.  min_x = x - (mid_x - inc_x);
32.  max_x = x + (mid_x - inc_x);
33.  }
34.  };
35.  if (y - (mid_y - inc_y) < min_y) {
36.  max_y = min_y + (dif_y - 2*inc_y);
37.  } else {
38.  if (y + (mid_y - inc_y) > max_y) {
39.  min_y = max_y - (dif_y - 2*inc_y);
40.  } else {
41.  min_y = y - (mid_y - inc_y);
42.  max_y = y + (mid_y - inc_y);
```

```
43.   }
44.   };
45.   dif_x=max_x-min_x;
46.   dif_y=max_y-min_y;
47.   abajo=rect.bottom;
48.   derecho=rect.right;
49.   fact_m=dif_x/dif_y;
50.   if (abajo*fact_m>derecho)
51.   abajo= derecho/fact_m;
52.   else
53.   derecho=abajo*fact_m;
54.   *a_stack++ = (long) min_x;
55.   *a_stack++ = (long) max_x;
56.   *a_stack++ = (long) min_y;
57.   *a_stack++ = (long) max_y;
58.   limite = 0;
59.   InvalidateRect( rect ,TRUE);
60.   } else {
61.   i_inf = a_inf;
62.   while (i_inf->x!=FINAL) {
63.   if ((i_inf->giro==giro_act)    &&
64.   (abs((i_inf->x)-x)<=ancho) &&
65.   (abs((i_inf->y)-y)<=ancho)    ) {
66.   fsit.Seek(579+337*cont,CFile::begin);
67.   fsit.Seek(7,CFile::current);
68.   rz = (char*) malloc((largo=40)+1);
69.   fsit.Read(rz,largo);
```

```
70.   *(rz+largo) = '\0';
71.   dlg.m_rz = rz;
72.   colonia = (char*) malloc((largo=40)+1);
73.   fsit.Read(colonia,largo);
74.   *(colonia+largo) = '\0';
75.   dlg.m_colonia = colonia;
76.   calle = (char*) malloc((largo=40)+1);
77.   fsit.Read(calle,largo);
78.   *(calle+largo) = '\0';
79.   dlg.m_calle = calle;
80.   numero = (char*) malloc((largo=15)+1);
81.   fsit.Read(numero,largo);
82.   *(numero+largo) = '\0';
83.   dlg.m_numero = numero;
84.   cp = (char*) malloc((largo=5)+1);
85.   fsit.Read(cp,largo);
86.   *(cp+largo) = '\0';
87.   dlg.m_cp = cp;
88.   nom_com = (char*) malloc((largo=40)+1);
89.   fsit.Read(nom_com,largo);
90.   *(nom_com+largo) = '\0';
91.   dlg.m_nom_com = nom_com;
92.   rfc = (char*) malloc((largo=13)+1);
93.   fsit.Read(rfc,largo);
94.   *(rfc+largo) = '\0';
95.   dlg.m_rfc = rfc;
96.   prop = (char*) malloc((largo=40)+1);
```

```
97.   fsit.Read(prop,largo);
98.   *(prop+largo) = '\0';
99.   dlg.m_prop = prop;
100.  tel = (char*) malloc((largo=10)+1);
101.  fsit.Read(tel,largo);
102.  *(tel+largo) = '\0';
103.  dlg.m_tel = tel;
104.  ho1 = (char*) malloc((largo=4)+1);
105.  fsit.Read(ho1,largo);
106.  *(ho1+largo) = '\0';
107.  dlg.m_ho1 = ho1;
108.  ho2 = (char*) malloc((largo=4)+1);
109.  fsit.Read(ho2,largo);
110.  *(ho2+largo) = '\0';
111.  dlg.m_ho2 = ho2;
112.  ho3 = (char*) malloc((largo=4)+1);
113.  fsit.Read(ho3,largo);
114.  *(ho3+largo) = '\0';
115.  dlg.m_ho3 = ho3;
116.  ho4 = (char*) malloc((largo=4)+1);
117.  fsit.Read(ho4,largo);
118.  *(ho4+largo) = '\0';
119.  dlg.m_ho4 = ho4;
120.  pser = (char*) malloc((largo=50)+1);
121.  fsit.Read(pser,largo);
122.  *(pser+largo) = '\0';
123.  dlg.m_pser = pser;
```

```
124.  cat = (char*) malloc((largo*20)+1);
125.  fsit.Read(cat,largo);
126.  *(cat+largo) = '\0';
127.  dlg.m_cat = cat;
128.  dlg.DoModal();
   // lectura de las variables que desplegaran las ventanas de información
129.  free(nom_com);    free(calle);    free(numero); free(colonia);
130.  free(cp);         free(ho1);      free(ho2);    free(ho3);
131.  free(ho4);        free(pser);     free(cat);    free(tel);
132.  free(rz);         free(prop);     free(rfc);
133.  };
134.  i_inf++;
135.  cont++;
136.  };
137.  };
138.  }
```

El siguiente código es el que se encarga de ejecutar las acciones del botón derecho del ratón cuya función es la de realizar los alejamientos y para poderlo hacer, pregunta si esta activada la función de Amplificar (activa los alejamientos) y desactivada la función de Examinar (activa el despliegue de las ventanas de información).

```
1.  void CSiteView::OnRButtonDown(UINT nFlags, CPoint point)
2.  {
3.  double fact_m;
4.  CView::OnRButtonDown(nFlags, point);
5.  CClientDC dc( this );
```

```
6.   CRect rect;
7.   if (examinar) return;     //pregunta si esta activada la opción de
                               //examinar
8.   GetClientRect( rect );
  //pregunta por la posición de los puntos máximos y minimos dentro de la
  //pila
9.   if (a_stack-4<=stack) {
10.  return;
11.  };
  // Colocación de los puntos maximos y minimos dentro de la pila
12.  a_stack-=4;
13.  min_x = (long) *(a_stack-4);
14.  max_x = (long) *(a_stack-3);
15.  min_y = (long) *(a_stack-2);
16.  max_y = (long) *(a_stack-1);
17.  dif_x= (double) max_x-min_x;
18.  dif_y= (double) max_y-min_y;
19.  abajo= (double) rect.bottom;
20.  derecho= (double) rect.right;
21.  fact_m=dif_x/dif_y;
22.  if (abajo*fact_m>derecho)
23.  abajo= derecho/fact_m;
24.  else
25.  derecho= abajo*fact_m;
26.  limite = 0;
27.  InvalidateRect( rect ,TRUE);
28.  }
```

Para regresar al aumento inicial de los mapas automaticamente, se toma el valor original del Máximo y Mínimo, inicializandolos y volviendo a dibujar. Para ello se utiliza el siguiente código:

```
1.  void CSiteView::OnUbicarCiudad()
2.  {
3.  CClientDC dc( this );
4.  CRect rect;
5.  GetClientRect( rect );
6.  limite = 1;          // inicializa las variables a su valor original
7.  InvalidateRect( rect ,TRUE);
8.  }
```

H) Para obtener la información que corresponde a los sitios, se selecciona antes, la opcion **examinar** del menú **Ubicar radio de acción;** esto es, para poder desplegar la ventana que contiene dicha información y es entonces cuando se coloca el apuntador del ratón sobre uno de los puntos rojos (sitios) del mapa presionando el botón del mismo. Y si después se quiere hacer un aumento sobre el sitio seleccionado se tiene que seleccionar la opción de **Amplificar** del menú **Ubicar radio de acción.**

El siguiente codigo efectua el proceso anterior:

```
1.  void CSiteView::OnUbicarciudadAmplificar()
2.  {
4.  amplificar = 1;   // se activa la opción de amplificar
5.  examinar = 0;   // se desactiva la opcion de examinar
6.
7.  }
```

```
 8.  void CSiteView::OnUbicarciudadExaminar()
 9.  {
11.  amplificar = 0; // se desactiva la opción de amplificar
12.  examinar = 1;      //se activa la opción de examinar
13.
14.  }
15.  void CSiteView::OnInformacintursticaAtractivosturs ticos()
   // visualización de la información de atracciones turísticas
16.  {
17.  DSelecc dlg;
18.  DCuadro dlg2;
19.  CFile temp;
20.  double tam;
21.  char *buffer;
22.  lugt_sit *l_apt;
23.  dlg.m_sitio = (ciudad)?"ZIHUATANEJO":"IXTAPA";
24.  dlg.m_lugt = a_lugt;
25.  if (dlg.DoModal()!=1) return;
26.  l_apt = a_lugt;
27.  while ((strcmp(dlg.m_lista,l_apt->nombre)!=0) && (l_apt-
     >marca!=FINAL))
28.  l_apt++;
29.  if (l_apt->marca==FINAL) return;
30.  temp.Open( l_apt->archivo  ,CFile::modeRead);
31.  tam = temp.GetLength();
32.  buffer = (char*) malloc(tam+1);
33.  temp.Read(buffer,tam);
```

```
34.   temp.Close();
35.   *(buffer+(int)tam) = '\0';
36.   dlg2.m_titulo = dlg.m_lista;
37.   dlg2.m_texto = buffer;
38.   dlg2.DoModal();
39.   free(buffer);
40.   }
```

El siguiente código activa la información de Bienes:

```
41.   void CSiteView::OnInformacintursticaBienes()
42.   {
43.
44.   DBienSer dlg;
45.   desc_giro *g_apt;
46.   CClientDC dc( this );
47.   CRect rect;
48.   dlg.m_tipo = 0;
49.   dlg.m_giro = a_giro;
50.   dlg.m_titulo = (ciudad)?"ZIHUATANEJO - BIENES":"IXTAPA - BIENES";
51.   if (dlg.DoModal()!=1) return;
52.   g_apt = a_giro;
53.   while ((strcmp(dlg.m_lista,g_apt->nombre)!=0) && (g_apt-
      >marca!=FINAL))
54.   g_apt++;
55.   if (g_apt->marca==FINAL) return;
56.   giro_act = g_apt->giro;
```

```
57.   GetClientRect( rect );
58.   limite = 0;
59.   InvalidateRect( rect ,TRUE);
60.   }
```

El siguiente código activa la información de Servicios:

```
61.   void CSiteView::OnInformacintursticaServicios()
62.   {
63.
64.   DBienSer dlg;
65.   desc_giro *g_apt;
66.   CClientDC dc( this );
67.   CRect rect;
68.   dlg.m_tipo = 1;
69.   dlg.m_giro = a_giro;
70.   dlg.m_titulo = (ciudad)?"ZIHUATANEJO - SERVICIOS":"IXTAPA -
      SERVICIOS";
71.   if (dlg.DoModal()!=1) return;
72.   g_apt = a_giro;
73.   while ((strcmp(dlg.m_lista,g_apt->nombre)!=0) && (g_apt-
      >marca!=FINAL))
74.   g_apt++;
75.   if (g_apt->marca==FINAL) return;
76.   giro_act = g_apt->giro;
77.   GetClientRect( rect );
78.   limite = 0;
```

```
79.    InvalidateRect( rect ,TRUE);
       }
```

I) Para limpiar la pantalla del mapa se inicializan las variables originales. Para ello se utilizó el siguiente código.

```
1.void CSiteView::OnUbicarLimpiar()
2.{
3.CClientDC dc( this );
4.CRect rect;
5.GetClientRect( rect );
6.giro_act = 0;        // regresa las variables a su valor original
7.GetClientRect( rect );
8.limite = 0;
9.InvalidateRect( rect ,TRUE);
```

J) Para dibujar el mapa se hace uso de una pila cuya función es la de almacenar los valores máximos y mímos del mapa, esto con el propósito de ir comparando sí los datos que se estan leyendo de los archivos "i_datos.dbf" o "z_datos.dbf" se encuentran dentro de la pila para entonces graficarlos. Para empezar a gráficar se coloca en la pila un valor minimo para X,Y que es la cordenada 10,20.

El método que se sigue para dibujar el mapa es unir una serie de puntos(coordenadas) con rayas, cuando se empieza a dibujar una nueva serie de puntos se pregunta sí el valor de la variable **Marca** es igual **9999,** para entonces comenzar a dibujar una nueva linea, este valor se lee en los archivos "i_datos.dbf" o "z_datos.dbf". El código que se encarga de dibujar el mapa es el siguiente:

```
1.  // CSiteView drawing
2.  void CSiteView::OnDraw(CDC* pDC)
3.  {
4.  CSiteDoc* pDoc = GetDocument();
5.  CClientDC dc( this );
6.  CRect rect, sitio;
7.  CBrush      sit_brush;
8.  entero *a_apt;
9.  long xp, yp, x, y;
10. double fact_m;
11. byte flag = 1;
12. desc_sit *p_dat;
13. lugar_sit *i_inf;
14. GetClientRect( rect );
15. dc.SetTextAlign( TA_BASELINE | TA_CENTER );
16. dc.SetTextColor( ::GetSysColor( COLOR_WINDOWTEXT ) );
17. dc.SetBkMode(TRANSPARENT);
18. a_apt = a_lin;
19. if (limite) {
20. min_x = *a_apt++;
21. max_x = *a_apt++;
22. min_y = *a_apt++;
23. max_y = *a_apt++;
24. dif_x=max_x-min_x;
25. dif_y=max_y-min_y;
26. abajo=rect.bottom;
27. derecho=rect.right;
```

```
28.   fact_m=dif_x/dif_y;
29.   if (abajo*fact_m>derecho)
30.   abajo=derecho/fact_m;
31.   else
32.   derecho=abajo*fact_m;
33.   a_stack = stack;                  // Estas instrucciones guardan los
                                        //limites en el tope de la pila
34.   *a_stack++ = (long) min_x;
35.   *a_stack++ = (long) max_x;
36.   *a_stack++ = (long) min_y;
37.   *a_stack++ = (long) max_y;
38.   } else {
39.   a_apt+=4;
40.   limite = 1;
41.   };
42.   while (*a_apt != FINAL) {
43.   x = *a_apt++;
44.   y = *a_apt++;
45.   if ((x!=MARCA) && (y!=MARCA))  // se pregunta si el valor de X y Y
                                        // es igual a 9999,  para comenzar a
                                        //dibujar una nueva serie de puntos
{
46.   flag = 1;
47.   } else {
48.   x = *a_apt++;
49.   y = *a_apt++;
50.   flag = 0;
```

```
51.   };
52.   trans(x,y,&xp,&yp);
53.   if (flag) {dc.LineTo( (int) xp, (int) yp);}
54.   else  {dc.MoveTo( (int) xp, (int) yp);};
55.   };
56.   /*
57.   POINT pt;
58.   pt.x = 10;                    //valor X mínimo inicial
59.   pt.y = 20;                 // valor Y mínimo inicial
60.   CClientDC dc(this);
61.   dc.SetPixel(pt.x,pt.y,RGB(255,0,0));
62.   */
63.   //dc.SelectStockObject(GRAY_BRUSH);
64.   p_dat = a_dat;
65.   while (p_dat->x!=FINAL) {
66.   x     = p_dat->x;
67.   y     = p_dat->y;
68.   if (min_x<=x && x<=max_x && min_y<=y && y<=max_y)  // pregunta si
                                 //el punto se encuentra dentro de la
                                 //pantalla para entonces graficarlo.
{
69.   trans(x,y,&xp,&yp);
70.   dc.TextOut( (int) xp, (int) yp, p_dat->titulo, 30 ); // despliega
                                                  //el titulo del lugar
71.   };
72.   p_dat++;
73.   };
```

```
74.  i_inf = a_inf;
75.  while (i_inf->x!=FINAL) {
76.  x = i_inf->x;
77.  y = i_inf->y;
78.  if (i_inf->giro==giro_act && min_x<=x && x<=max_x &&//pregunta si
                                  //el punto se encuentra dentro de la
                                  //pantalla entonces se rellena el punto rojo
                                  //que indica la ubicación.
79.  min_y<=y && y<=max_y) {
80.  trans(x,y,&xp,&yp);
81.  dc.Arc( (int) xp-ancho, (int) yp-ancho, (int) xp+ancho,
82.  (int) yp+ancho, (int) xp-ancho, (int) yp,
83.  (int) xp-ancho, (int) yp);
84.  };
85.  i_inf++;
86.  };
87.  sit_brush.CreateSolidBrush(RGB(255,60,70));
88.  dc.SelectObject(&sit_brush);
89.  i_inf = a_inf;
90.  while (i_inf->x!=FINAL) {
91.  x = i_inf->x;
92.  y = i_inf->y;
93.  if (i_inf->giro==giro_act && min_x<=x && x<=max_x &&
94.  min_y<=y && y<=max_y) {
95.  trans(x,y,&xp,&yp);
96.  dc.FloodFill( (int) xp, (int) yp,RGB(0,0,0));
97.  };
```

```
98.   i_inf++;
99.   }
100.  /*
101.  dc.SetPixel(xp,abajo-yp,RGB(255,0,0));
102.  dc.TextOut( ( rect.right / 2 ), ( rect.bottom / 2 ),s,
      s.GetLength() );
103.  CSiteView::OnDraw(&dc);
```

K) Las instrucciones que se utilizaron para unir los comandos del menú y de los botones de la barra de herramientas con las funciones implementadas en las clases son las siguientes:

```
////////////////////////////////////////////////////////////////////////////
// CSiteView message handlers
1.    // CSiteView
2.    IMPLEMENT_DYNCREATE(CSiteView, CView)
3.    BEGIN_MESSAGE_MAP(CSiteView, CView)
4.    //{{AFX_MSG_MAP(CSiteView)
5.    ON_COMMAND(ID_UBICAR_CIUDAD, OnUbicarCiudad)
6.    ON_WM_LBUTTONDOWN()
7.    ON_COMMAND(ID_CIUDAD_IXTAPA, OnCiudadIxtapa)
8.    ON_COMMAND(ID_CIUDAD_ZIHUATANEJO, OnCiudadZihuatanejo)
9.    ON_WM_RBUTTONDOWN()
10.   ON_COMMAND(ID_UBICAR_RADIODEACCIN, OnUbicarRadiodeaccin)
11.   ON_COMMAND(ID_SITE_CONOCER, OnSiteConocer)
12.   ON_COMMAND(ID_INFORMACINTURSTICA_ATRACTIVOSTURSTICOS,
      OnInformacinturasticaAtractivosturasticos)
```

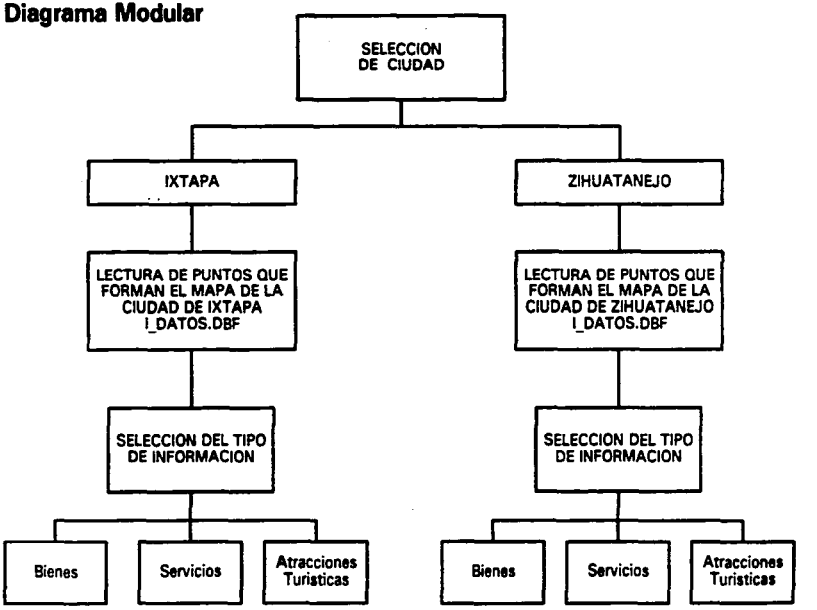
```
13.   ON_COMMAND(ID_INFORMACINTURSTICA_SERVICIOS,
      OnInformacintursticaServicios)
14.   ON_COMMAND(ID_INFORMACINTURSTICA_BIENES,
      OnInformacintursticaBienes)
15.   ON_COMMAND(ID_UBICAR_LIMPIAR, OnUbicarLimpiar)
16.   ON_COMMAND(ID_UBICARCIUDAD_AMPLIFICAR,    OnUbicarciudadAmplificar)
17.   ON_COMMAND(ID_UBICARCIUDAD_EXAMINAR, OnUbicarciudadExaminar)
18.   //}}AFX_MSG_MAP
20.   ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
21.   ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()
```

El diseño de las ventanas que despliegan la información se describirá en el siguiente punto de este capítulo.

La implementación del archivo CSiteView completo se puede consultar el apendice que se encuentra en la parte final de este capítulo.

Los siguientes cuadros describen de manera modular la realización de este sistema.

Diagrama Modular



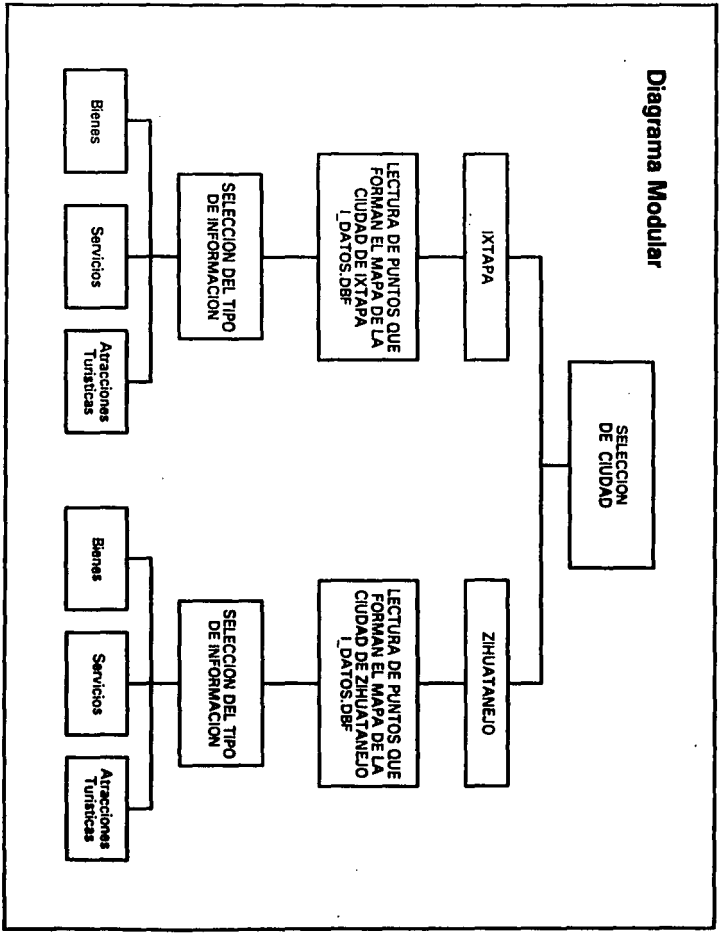


Diagrama Modular
SELECCION DE CIUDAD
IXTAPA
ZIHUATANEJO
LECTURA DE PUNTOS QUE FORMAN EL MAPA DE LA CIUDAD DE IXTAPA I_DATOS.DBF
LECTURA DE PUNTOS QUE FORMAN EL MAPA DE LA CIUDAD DE ZIHUATANEJO I_DATOS.DBF
SELECCION DEL TIPO DE INFORMACION
SELECCION DEL TIPO DE INFORMACION
Bienes
Servicios
Atracciones Turisticas
Bienes
Servicios
Atracciones Turisticas

CONTINUACION N

MODULO DE OBTENCION DE INFORMACION

Modulo para Ampliar
Acercamiento del 5%
Acercamiento del 25%
Acercamiento del 50%
Modulo de Examinar
Modulo de Examinar
Modulo de Examinar
Obtención de Información
Obtención de Información
Obtención de Información
¿Otra Consulta?
¿Otra Consulta?
¿Otra Consulta?
Modulo de selección de ciudad
Modulo de selección de tipo de Información
Modulo de selección de ciudad
Modulo de selección de tipo de Información
Modulo de selección de ciudad
Modulo de selección de tipo de Información
Salir
Salir
Salir
Salir
Salir
Salir

3.1.7 DISEÑO DE LAS PANTALLAS DE CONSULTA.

En este capítulo se ilustra como fueron realizadas las pantallas que el sistema va a presentar al usuario y con las que va a interactuar. Para el diseño de estas, se hizo uso principalmente de una de las herramientas que provee Visual C++, que es **App Studio**. Esta aplicación nos permite desarrollar la presentación del sistema, la cual se compone de una barra de menú, una barra de herramientas compuesta de iconos, que son botones con las figuras y dibujos que expresan la función que realizan, y las ventanas que van a desplegar la información.

Para el diseño de la pantalla principal, que es la que va a desplegar las barras de menú y herramientas se utilizaron los siguientes procedimientos:

Barra de menú

1.- Primeramente del archivo que genero App Wizard Site.rc (Resource Script) que es el archivo encargado de mantener toda la información de la presentación del sistema, se teclea el código que forma la barra de menú. El símbolo ampersan (&) se coloca antes de una letra del nombre del comando (&Ayuda) para que en la pantalla que se le presente al usuario aparezca el nombre del comando con una de sus letras subrayadas lo que le indica cual es la llave de acceso rápido al comando, por ejemplo en el comando llamado **Ayuda** se encuentra subrayada la letra A lo cual indica que el usuario accesa a ese comando simplemente presionando la tecla ALT y la tecla que contiene la letra A.

El código es el siguiente:

```
IDR_MAINFRAME MENU PRELOAD DISCARDABLE

BEGIN
    POPUP "&Site"
    BEGIN
        MENUITEM "&Acerca",
  ID_APP_ACERCA
        MENUITEM "&Conocer",
  ID_SITE_CONOCER
        MENUITEM SEPARATOR
        MENUITEM "&Salida",
  ID_APP_SALIR
    END
    POPUP "&Ciudad"
    BEGIN
        MENUITEM "&Ixtapa",
  ID_CIUDAD_IXTAPA
        MENUITEM "&Zihuatanejo",
  ID_CIUDAD_ZIHUATANEJO
    END
    POPUP "&Ubicar"
    BEGIN
        MENUITEM "&Ciudad",
  ID_UBICAR_CIUDAD
        MENUITEM "&Radio de Acción",
  ID_UBICAR_RADIODEACCIN
        MENUITEM SEPARATOR
```

```
            MENUITEM "&Limpiar",
        ID_UBICAR_LIMPIAR
          END
          POPUP "Información &Turística"
          BEGIN
              MENUITEM "&Bienes",
       ID_INFORMACINTURSTICA_BIENES
              MENUITEM "&Servicios",
    ID_INFORMACINTURSTICA_SERVICIOS

              MENUITEM SEPARATOR
              MENUITEM "&Atractivos Turísticos",
                        ID_INFORMACINTURSTICA_ATRACTIVOSTURSTICOS
        END
        MENUITEM "&Quejas",                     65535
        MENUITEM "&Ayuda",                      65535
    END
    endif #
```

2.- Después hay que compilarlo en la opción Compile del menú Project.

3.- Y por último se tiene que ligar con los demás archivos del programa (archivos con las extensiones .cpp , .h, y .def) para elaborar el archivo ejecutable(.Exe).

```
        MENUITEM "&Limpiar",
    ID_UBICAR_LIMPIAR
      END
      POPUP "Información &Turística"
      BEGIN
        MENUITEM "&Bienes",
    ID_INFORMACINTURSTICA_BIENES
        MENUITEM "&Servicios",
  ID_INFORMACINTURSTICA_SERVICIOS

        MENUITEM SEPARATOR
        MENUITEM "&Atractivos Turísticos",
                ID_INFORMACINTURSTICA_ATRACTIVOSTURSTICOS
      END
      MENUITEM "&Quejas",                    65535
      MENUITEM "&Ayuda",                     65535
  END
  endif #
```

2.- Después hay que compilarlo en la opción Compile del menú Project.

3.- Y por último se tiene que ligar con los demás archivos del programa (archivos con las extensiones .cpp , .h, y .def) para elaborar el archivo ejecutable(.Exe).

Barra de Herramientas

La barra de herramientas es también un menú como el anterior, pero en forma gráfica ya que consta de iconos (botones que muestran mediante un dibujo una función específica del sistema), esto con el motivo de que se prevee que el sistema será utilizado tanto por turismo nacional como extranjero, entonces se pretende hacer un sistema que sea entendible a cualquier individuo no importando el idioma que hable. Para realizar la barra se hizo lo siguiente:

1.- Desde la pantalla Workbench del Visual C++ se elige el comando AppStudio que se encuentra en el menú Tools.

2.- En la pantalla que despliega, se elige la opción Open del menú File y se escoge el archivo con la extensión rc. Ahí se presenta una pantalla que despliega dos botones de los cuales elegimos el botón de New para elaborar los iconos que van a formar la barra de herramientas.

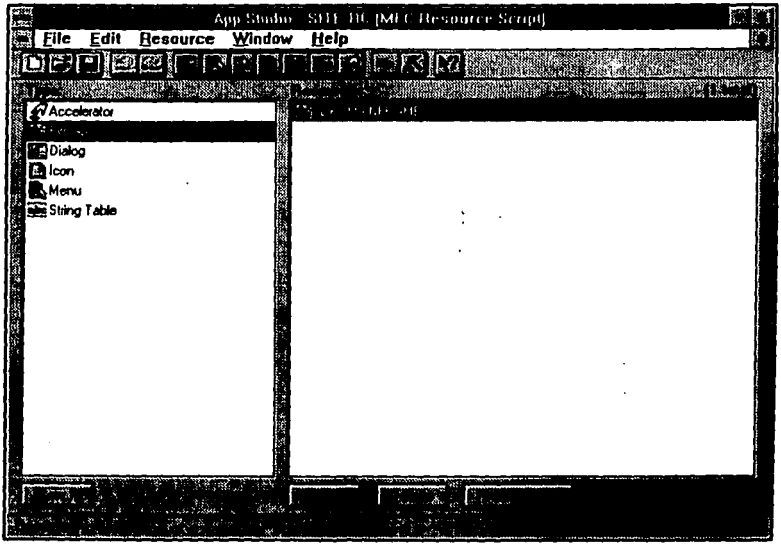


Fig. 3.1.7.1 Pantalla de App Studio.

3.- Se procede a dibujar cada uno de los iconos de la barra, cuidando que la imagen sea alusiva a la función que queramos que realice, para este proceso utilizamos la paleta de herramientas que el Visual C++ nos proporciona para ayudarnos a realizar el dibujo.

4.- Una vez terminado el proceso anterior, se salva el archivo y se añade la siguiente estructura en el archivo MainFrame.cpp

```
static UINT BASED_CODE buttons[] =
// lista de las funciones que realiza el sistema
        {
  ID_ABRIR
  ID_CERRAR
  ID_LIMPIAR
  ID_FUNCION_ESPECIFICA
  ...
  ...
  ...
};
```

Tal estructura relaciona los botones de las imágenes con sus respectivos comandos definidos como estructuras ID 's.

Nota: Hay que cuidar el orden en que se agregan los comandos porque los va relacionando de acuerdo a como están ordenados los iconos en la barra de herramientas.

5.- Finalmente se compila el archivo y se reconstruye todo el programa en la opción Rebuilt del menú Project, para elaborar el archivo ejecutable.

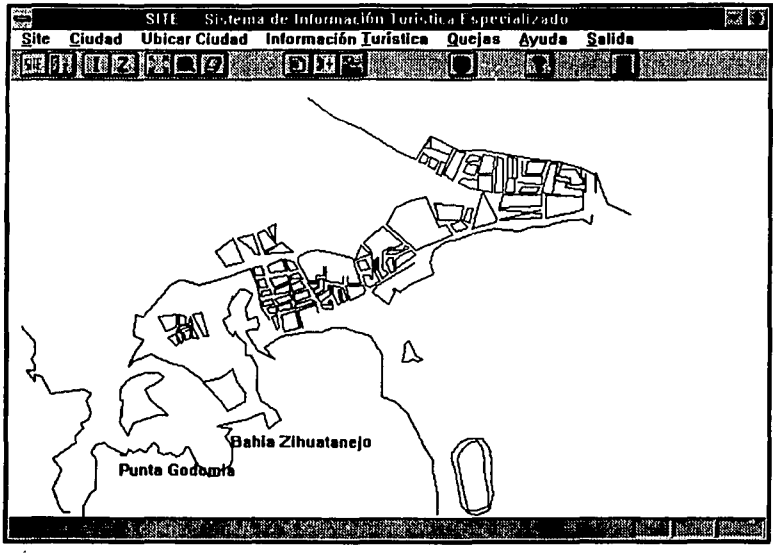


Fig. 3.1.7.2 Visualización de la barra de herramientas

Cajas de dialogo

El término 'Cajas de diálogo" es un nombre erróneo ya que estas son ventanas especiales que contienen controles que sirven para desplegar información o entrada de datos. Las aplicaciones de Windows utilizan las cajas de dialogo para intercambiar información con el usuario.

Existen dos tipos de cajas de dialogo las Modal y Modeless, las cajas de dialogo Modal requieren que se cierre la caja antes de continuar con la aplicación, ya que de no hacerlo significaría ejecutar un intercambio de datos critico, de hecho las cajas de

dialogo desactivan su ventanas anteriores mientras ellas están abiertas, en cambio las cajas de dialogo Modeless no necesitan estar cerradas para continuar utilizando la aplicación.

Existe una biblioteca de MFC declarada como clase CDialog que maneja ambas cajas de dialogo la Modeless y Modal. La clase CDialog es una clase derivada de la clase Cventana, tiene un constructor y un número de funciones miembro, incluyendo la función Create. La clase CDialog esta declarada como protegida para obligarnos a crear nuestras propias cajas de dialogo de tipo Modeless. Estas clases derivadas pueden declarar ciertas clases para suplementar nuestras cajas de dialogo como son:

- Declarar únicamente la clase constructor como publica ya que esta invoca a la función CDialog::Create, esta característica crea la caja de dialogo en un paso.
- Declarar ambas funciones la constructor y la create. Este esquema crea instancias de la caja de dialogo en dos pasos, primero llamando la clase constructor y después invocando la función Create.

La declaración de la función CDialog::Create es:

BOOL Create (LPCSTR lpTemplateName, CVentana* pParentVentana = NULL);

BOOL Create (UINT nIDTemplate, CVentana* pParentVentana = NULL);

Los parámetros LpTempalte y nIDTemplate son de la caja de dialogo y el comando ID respectivamente. El parámetro pParentVentana es el apuntador a la ventana raíz.

Ejecución de las cajas de dialogo Tipo Modal

Comúnmente, las cajas de dialogo tipo Modal son creadas más frecuentemente que las cajas de dialogo Modeless y mucho más frecuentemente que las ventanas, para la ejecución de las cajas de dialogo modal se involucran los siguientes pasos:

1.- Crear una instancia de la caja de dialogo utilizando la caja de dialogo de la clase constructor.

2.- Llamar a la función DoModal(), declarada en la clase CDialog, para traer la caja de dialogo. Comúnmente, la caja de dialogo contiene dos botones por defecto que son OK y Cancel estos botones tienen predefinido a los comandos ID's IDOK e IDCANCEL, respectivamente, se pueden utilizar los controles de los botones con diferentes mensajes que son OK y CANCEL, sin embargo es mejor mantener IDOK e IDCANCEL con esos nombres de los botones. Utilizar los IDs permite tomar ventaja de las estructuras ya que responde automáticamente a IDOK e IDCANCEL provistas por las funciones OnOK y OnCancel definidas en la clase de la caja de dialogo. Presionando el botón OK comúnmente indica que se acepta el dato corriente en la caja de dialogo. Y en caso contrario presionando el botón de cancel indica que se esta en desacuerdo con el dato corriente y por lo tanto no lo efectúa el proceso con el cual esta relacionado. La declaración de la función DoModal( ) es int DoModal ( ); la función regresa un entero que representa la salida. Comúnmente los valores de salida son IDOK o IDCANCEL.

3.- Se comparan los resultados de la función DoModal con IDOK(o con IDCANCEL). La salida de esta comparación determina los pasos que toma, tales pasos involucra el acceso de datos que se introdujeron en los controles de las caja de dialogo.

Las funciones OnInitDialog, OnOk y OnCancel manejan la ejecución de la caja de dialogo del tipo Modal. Esos parámetros están declarados como funciones virtuales. La función OnInitDialog sirve para inicializar la caja de dialogo y sus controles. La declaración de OnInitDialog es:

```
virtual BOOL OnInitDialog();
```

La función regresa el resultado boleano para indicar la inicialización de la caja de dialogo. Comúnmente la función OnInitDialog inicializa los controles de la caja de dialogo. Esta inicialización usualmente copia los datos de los buffer.

La función OnOk se maneja presionando el botón de Ok. La declaración de la función OnOk es:

```
virtual void OnOk( );
```

La función OnOk sirve para copiar datos de los controladores de la caja de dialogo a los buffers. Utilizando la función EndDialog, la función OnOk permite proveer la función DoModal regresando su valor. Usualmente la ultima estructura en la función OnOk la definición es EndDialog(IDOK). Después la función llamada origina que la función DoModal regresa a IDOK.

La función OnCancel es manejada presionando el botón cancel. La declaración de la función OnCancel es

```
virtual void OnCancel ( );
```

La función OnCancel sirve para limpiar antes de que la caja de dialogo sea cerrada, la cual involucra el cierre de archivos, por ejemplo, usualmente la última estructura de la función OnCancel es EndDialog (IDCANCEL). Después la función llamada origina que la función DoModal regrese a IDCANCEL.

Interactuar con ventanas frecuentemente involucra utilizar cajas de dialogo que contienen varios tipos de controles, tales como caja de edición y botones de control(pushbutton). Esos controles pueden ser incluidos en ventanas o más frecuentemente en cajas de dialogo. Los controladores pueden ser implementados como se explica a continuación:

Control de texto

El control de texto provee una ventana o una caja de dialogo con texto estático. La clase Cstatic implementa ese control.

La clase Cstatic es derivada de la clase ventana y ofrece un texto estático que esta definida por un área de display, texto a un display y atributos de texto.

Sintaxis:

```
BOOL Create(const char FAR* lpText,          // Control de texto
            DWORD dwStyle,                   // Control de estilo
            const RECT& rect,                // Control de área
            Cventana* pParentVentana,        // Ventana Raíz
            UINT nID = 0Xffff),              // Control ID
```

El parámetro lpText especifica el texto del texto de control estático, el parámetro dwStyle designa el control de estilo. Los argumentos del estilo típicamente incluyen los estilos WS_CHILD y WS_VISIBLE. La creación de esos controles incluyen uno o más de esos estilos, el parámetro Rect especifica el área que se va a ocupar de tamaño rectangular, el parámetro pParentVentana es el apuntador a la ventana raíz, el parámetro nID especifica el control ID. La función Create asigna un parámetro por defecto el 0xffff, porque un control de texto estático comúnmente no envía mensajes a su ventana raíz.

El Control Pushbutton

El control PushButton es quizá sociológicamente el control más poderoso, ya que por costumbre sabemos que cuando presionamos un botón de control intuimos que va suceder algo.

Existen básicamente dos tipos de botones, los botones por defecto y los botones de nodefecto. Para manejar estos dentro den las ventanas hay que colocarlos primero dentro de una clase llamada Cbutton que es una clase derivada de la clase ventana, y en su implementación se hace uso de dos funciones que son de una relevante importancia.

La función Create

```
BOOL Create(const char FAR* lpCaption,     //Etiqueta del botón
            DWORD dwStyle,                 // Control de estilo
            const RECT& rect,              // Control de área
            Cventana* pParentVentana,      // Ventana Raíz
            UINT nID );                    // Control ID
```

El parámetro lpCaption especifica la etiqueta del botón es decir, el nombre de la función que ejecuta, el parámetro dwStyle designa el tipo de botón. Los estilos comúnmente utilizados son WS_CHILD, WS_VISIBLE, BS_PUSHBTTON Y WS_TABSTOP, el parámetro Rect especifica el área que se va a ocupar de tamaño rectangular y la posición, el parámetro pParentVentana es el apuntador a propia ventana, el cual puede estar junto a una ventana o a una caja de dialogo, el parámetro nID especifica el control ID.

El Visual C++ incluye la aplicación App Studio que permite crear cajas de dialogo y los botones de control de estas, pero es conveniente primero aprender como funciona el archivo con la extensión RC para hacer más fácil el trabajo con AppStudio.

La implementación del archivo CMainframe se encuentra en el apéndice B al finalizar este capítulo.

3.2 Documentacion.

3.2.1 MANUAL DEL USUARIO

SITE Sistema de Información Turística Especializado. Este sistema nos proporciona Información Turística del puerto de Ixtapa Zihuatanejo.

Para empezar con Windows y SITE

Para comenzar a trabajar con **SITE**, siga estos pasos:

1. Escriba WIN en el indicador de comandos de DOS (Por lo general C: >) y oprima Enter.
2. Active la ventana del grupo donde se encuentra **SITE** Para Windows. Esta ventana es la del grupo en donde se haya creado el elemento del programa de **SITE**.

La figura 3.2.1 muestra la ventana del grupo **SITE** con el elemento del programa **SITE** para Windows seleccionados. La ventana del grupo puede estar abierta en su pantalla, como la del grupo Main (Principal) y la del grupo **SITE** de la figura 3.2.1, o bien cerrada, como los iconos de los cuatro grupos en la esquina inferior de la ventana del Program Manager (administrador de programas) de la figura 3.2.1.

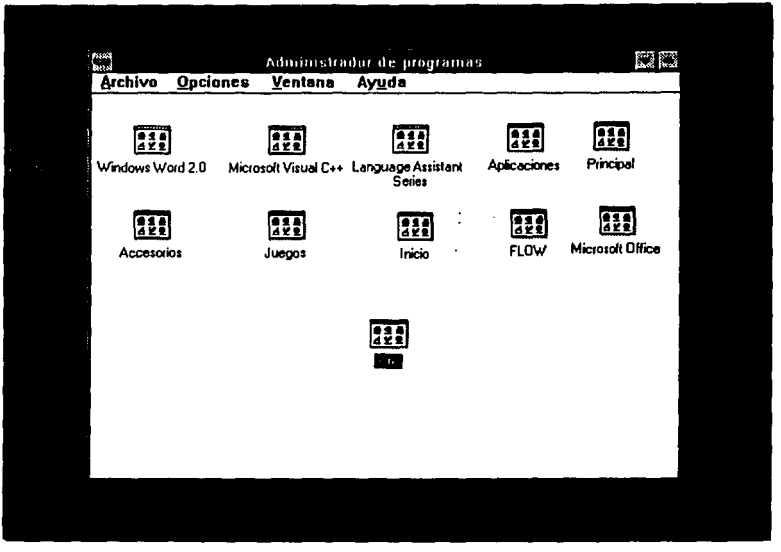


Fig. 3.2.1 Ventana del grupo SITE

Ratón: Pulse sobre la ventana abierta del grupo o pulse dos veces sobre el grupo cerrado de iconos que contenga el elemento del programa de **SITE para Windows.**

Teclado: Oprima Ctrl +Tab hasta que se active la ventana abierta del grupo que se desea o hasta que resulte el título del icono del grupo cerrado. Presione Enter para llevar el icono del grupo cerrado. Presione para llevar el icono del grupo a la ventana.

3. Inicie **SITE** para Windows.

   Ratón: Pulse dos veces sobre el icono del elemento del programa **SITE** para Windows.

   Teclado: Oprima las teclas de dirección hasta que resalte el icono del elemento del programa **SITE** para Windows y en seguida oprima Enter.

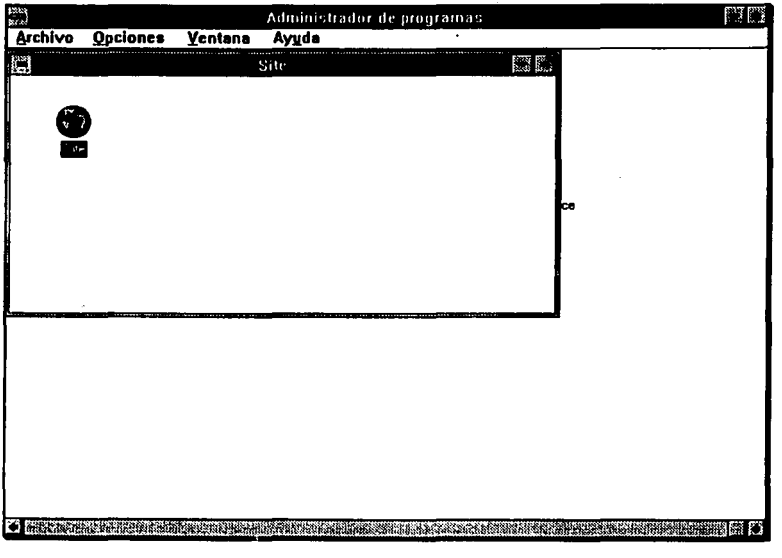


Fig. 3.2.2 Iniciación de SITE.

Si está en el indicador de comandos de DOS (por lo general C:>) y desea iniciar Windows y **SITE** para Windows al mismo tiempo, haga lo siguiente:

a. Escriba WIN C:\SITE\SITE.EXE

b. Oprima Enter

Para comprender la pantalla de SITE

Si utiliza por primera vez las aplicaciones de Windows, debe aprenderse las partes que forman la pantalla. Estudie la figura 3.2.3 y la tabla 3.2.1. para conocer las partes de aplicación y de documento de la pantalla .

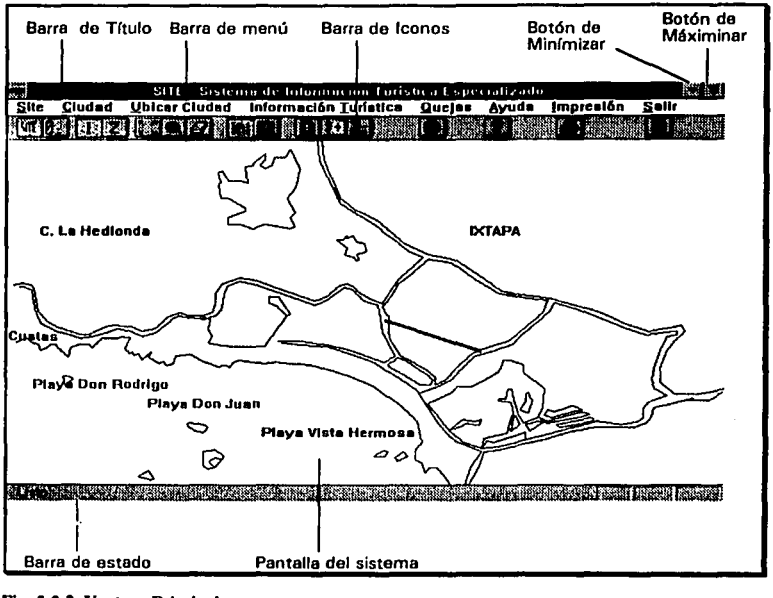


Fig. 3.2.2 Ventana Principal.

Tabla 3.2.1 Componentes de la pantalla de SITE

Término	Aplicación
Ventana de Aplicación	La Ventana dentro de la que corre **SITE**. Si esta utilizando Windows, puede ejecutar varias aplicaciones de Windows, cada una en su propia ventana. Una ventana de aplicación puede ocupar toda la ventana.
Ventana de documento	La ventana en la que aparece cada documento.
Menú de control de la aplicación	El menú utilizado para el manejo de la ventana de aplicación. Para activar su menú, oprima Alt, barra espaciadora o sobre el icono de control.
Menú del control del documento	El menú que se utiliza para el manejo de la ventana del documento activo. Para activar su menú, oprima Alt, guión (-) u oprima el botón del ratón sobre el icono de control.
Barra de Título	La barra de la parte superior de una ventana, que contiene el título de la aplicación o del documento.
Ventana Activa	La ventana que acepta entradas y comandos. La ventana cuya barra de título aparece sólida es por lo general la ventana del frente.

Ventana Inactiva	Cualquier ventana con la que no se este trabajando, cuya barra de título aparece sombreada o en tono gris.
Apuntador del ratón	Apuntador en la pantalla que muestra la posición del ratón.
Punto de Inserción	Punto donde aparece el texto cuando se escribe.
Barra de menúes	Lista horizontal de nombres de menúes, exhibida detrás de la barra del título de la aplicación
Barra de iconos	Barra de contiene botones y cajas de texto, que indican la dos ciudades a escoger, los acercamientos, las opciones de bienes, servcios y atracciones turísticas.
Icono para minimizar	El equivalente simbolico del comando Minimize, que reduce una ventana a un icono.
Icono para Maximizar	El equivalente simbolico del comando Maximize, que aumenta el tamaño de una ventana a toda la pantalla.
Barra de recorrido	Barra horizontal o vertical con flechas, se utiliza para recorrer el documento con el control del apuntador del ratón.
Barra de estado	Barra en la parte inferior de la pantalla, que muestra los icono o indicadores de comandos

Cómo usar la barra de iconos

La barra de iconos le da acceso directo a la mayoría de los comandos de uso más frecuente con el ratón sin pasar por los menúes.

La tabla 3.2.2 resume la funciones de cada uno de los botones que aparecen en la barra de iconos.

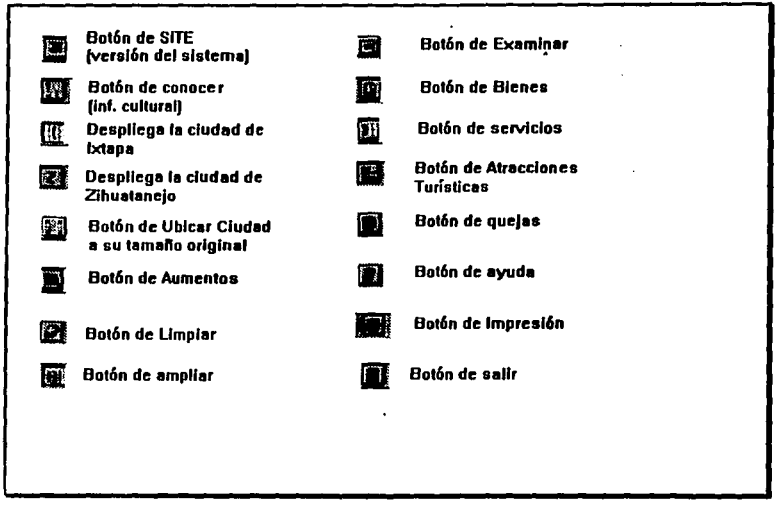


Tabla 3.2.2.

Cómo usar el ratón

El ratón o mouse es un dispositivo manual, que al moverse desplaza un apuntador en la pantalla. Los botones permiten hacer indicaciones a **SITE** cuando el ratón esta bien colocado en un menú o comando. Cuando oprime uno de los botones del ratón, se produce alguna acción correspondiente a la posición del apuntador.

Cómo mover el ratón

El ratón que se muestra en la figura 3.2.3 es un pequeño dispositivo que acomoda a la palma de su mano. Al mover el ratón en su escritorio, el apuntador del ratón, que por lo general es una flecha, se mueve en la misma dirección relativa en la pantalla. Notará que el uso del ratón para "apuntar" a un elemento de la pantalla resulta un proceso natural.

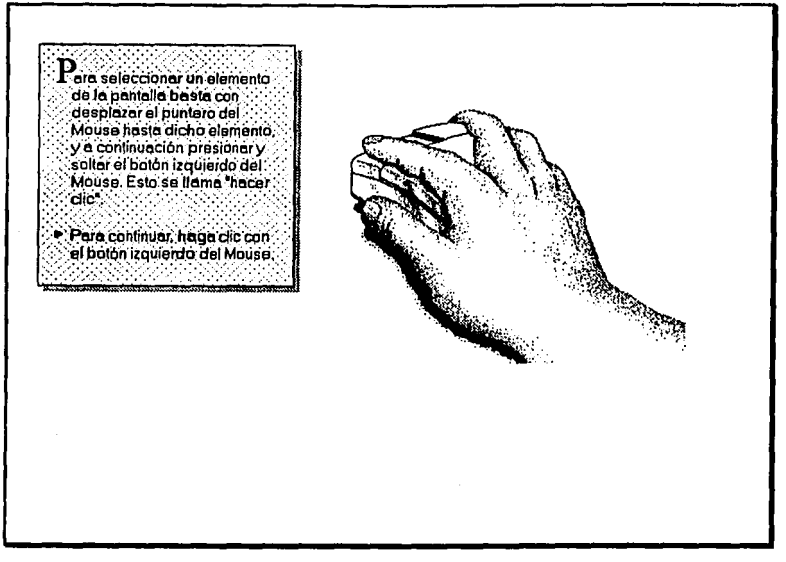


Fig. 3.2.3 El ratón.

Cuando utilice el ratón, el cable debe quedar hacia adelante, lejos de su brazo, de modo que los botones estén debajo de sus dedos. Aseguresé de mantener el aparato orientado, de manera que al alejar el ratón de usted, el apuntador se mueva hacia arriba en la pantalla. Este método hace que el ratón sea fácil de usar. Un ratón tiene dos o tres botones; sin embargo **SITE** sólo utiliza dos. El botón izquierdo será el de uso más frecuente.

Cómo hacer una selección con el ratón

Se utilizan dos técnicas básicas para la selección de elementos con el ratón:

- Se oprime el botón una vez para seleccionar un elemento, como un menú, comando, celda u objeto gráfico. Al oprimir el botón en alguna posición de la pantalla, como una palabra, se obtiene el acceso rápido a un comando.

- Oprima dos veces el botón para seleccionar una opción de una caja de diálogo y elegir al mismo tiempo la opción de aprobación. Este procedimiento, realizado en algunas regiones de la pantalla, genera un acceso rápido a una selección. Para esto, mueva el apuntador del ratón sobre el elemento y oprima de manera rápida dos veces el botón izquierdo.

Para elegir un comando con el ratón, oprima el botón en el nombre del menú en la barra de menúes y después sobre el nombre del comando en el menú descendente.

Uso del teclado

SITE para Windows no se limita al uso del ratón. Con el teclado puede ejecutar comandos o hacer selecciones de dos maneras:

- Oprimir la teclas de dirección para elegir menúes y comandos.
- Oprimir las teclas de función.

Cómo elegir comandos de menú con el teclado

En la barra de menués, el nombre de cada menú tiene una letra subrayada que se utiliza para activarlo desde el teclado. Cada comando en un menú también tiene sólo una letra subrayada, que se utiliza para activarlo.

Para elegir un comando desde un menú con el método de las combinaciones de teclas, siga estos pasos:

1. Oprima la tecla Alt para activar la barra de menúes.
2. Oprima la letra subrayada del menú que desee abrir.
3. Oprima la letra subrayada del comando en el menu descendente.

Después de activar la barra también se puede pasar de un menú o comando a otro al oprimir las teclas de dirección. Para elegir un comando nuevo se debe oprimir las teclas de dirección. Para elegir un comando con este método, seleccione el menú que desee pulsando Alt y la letra apropiada, oprima la tecla de dirección hacia abajo para resaltar el comando que piense elegir y presione Enter.

Sugerencia

Oprima Esc o el botón del ratón en el nombre del menú para salir de él sin tener que hacer una elección. Presione Esc o el botón del ratón en el botón de cancelar para regresar de una caja de diálogo sin hacer una elección.

Cómo usar SITE

Una vez que tenemos abierta la ventana principal de **SITE**, en la cual se nos muestra la barra de título, la barra de menú, la barra de iconos y la barra de estado en la parte inferior de la ventana; notamos que sobre la pantalla se encuentra desplegado el mapa de la ciudad de **IXTAPA**. Si observamos la barra de iconos, notamos, que el icono que contiene la letra **I** se encuentra sumido. Y esto nos indica que tenemos seleccionada la ciudad de Ixtapa y que podemos obtener la información Turística de ella.

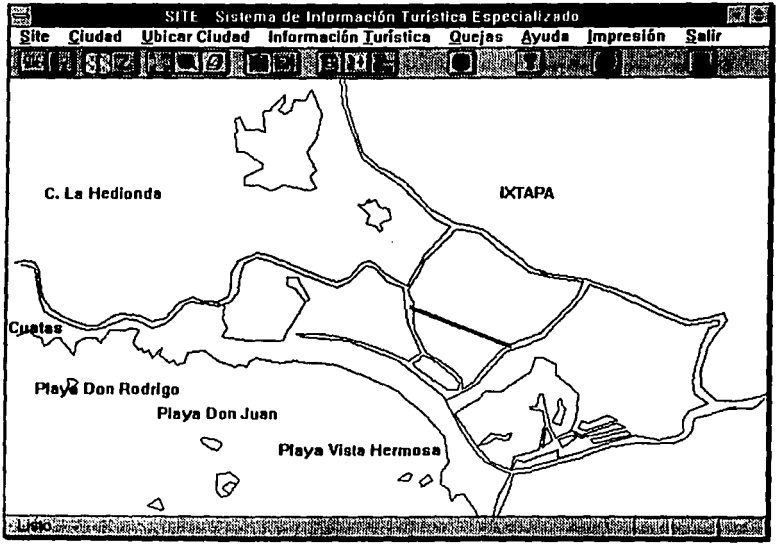


Fig. 3.2.4 Ventana de SITE.

Cómo cambiar de Ciudad

Si queremos cambiar a la ciudad de **ZIHUATANEJO,** utilizando el ratón o el teclado hacemos lo siguiente:

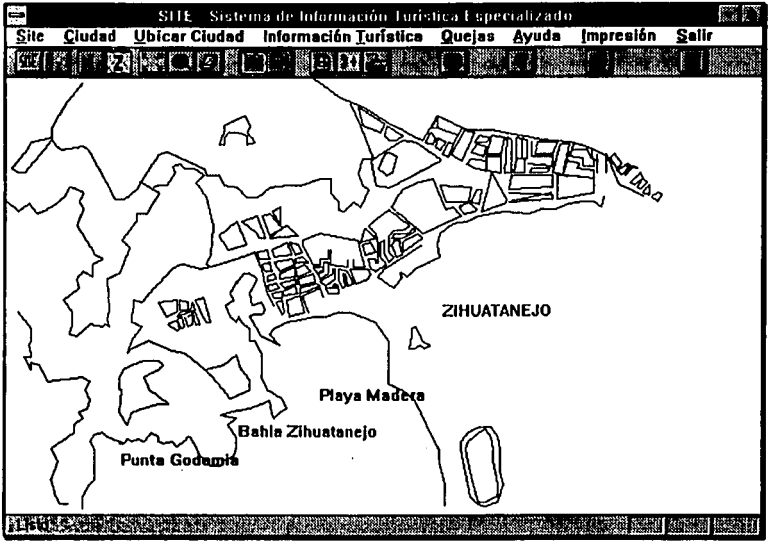


Fig. 3.2.5 Cambio de Ciudad.

Ratón: Colocando el apuntador del ratón sobre el botón de la barra de iconos que contiene la letra **Z** (se sume el botón), se presiona este, para activar la ventana que contiene el mapa de la ciudad de Zihuatanejo para poder obtener la nformación turística correspondiente.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **C** subrayada del menu **Ciudad** y se presiona la letra subrayada **Z** del menú descendente **Zihuatanejo**, para activar la ventana que contiene el mapa de la ciudad de Zihuatanejo para poder obtener la información turística correspondiente.

Si queremos cambiar a la ciudad de **IXTAPA**, utilizando el ratón o el teclado hacemos lo siguiente:

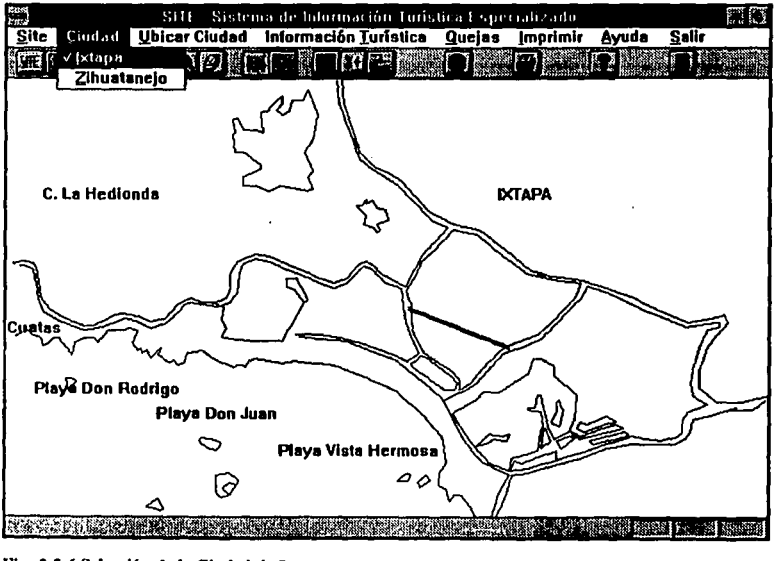


Fig. 3.2.6 Selección de la Ciudad de Ixtapa.

Ratón: Oprimimos el botón de la barra de iconos que contiene la letra I (se sume el botón), y con esto se activa la ventana que contiene el mapa de la ciudad de Ixtapa para poder obtener la información turística correspondiente.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **C** subrayada del menú **Ciudad** y se presiona la letra subrayada **I** del menú descendente **Ixtapa**, para activar la ventana que contiene el mapa de la ciudad de Zihuatanejo para poder obtener la información turística correspondiente.

Cómo utilizar el comando Conocer

Dependiendo de la ciudad que tengamos seleccionada ya sea Ixtapa o Zihuatanejo podemos obtener información cultural de la ciudad, y para ello se selecciona el comando **Conocer** que se encuentra en el menú **Site** ya sea de la barra de iconos o de la barra de menú.

Ratón: Oprimimos el botón de la barra de iconos que contiene el dibujo de una antorcha encendida (se sume el botón), y con esto se activa la ventana que contiene la información cultural de esa ciudad.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **S** subrayada del menú **Site** y se presiona la letra subrayada **C** del menú descendente **Conocer**, para activar la ventana que contiene la información cultural de esa ciudad.

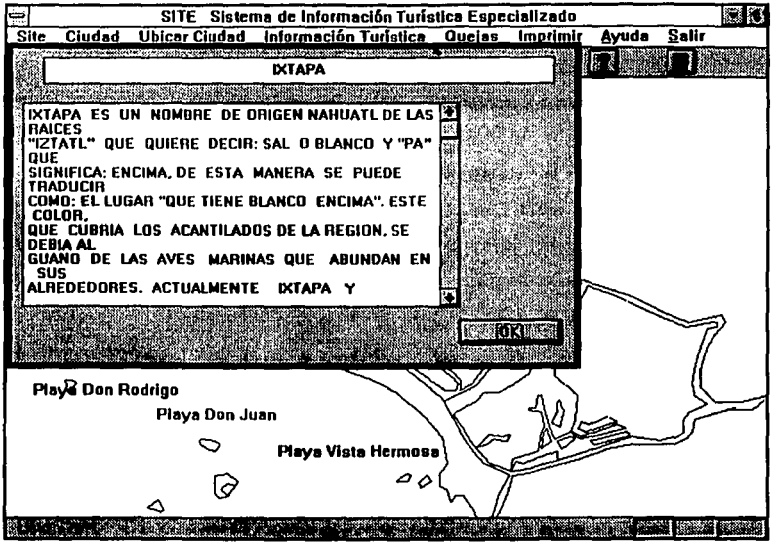


Fig. 3.2.7 Selección del comando conocer.

Cómo utilizar el menú Ubicar Ciudad

El menú **Ubicar Ciudad** nos permite seleccionar los comandos **Radio de acción y Limpiar.**

Comando Radio de Acción

El comando Radio de Acción nos permite efectuar los acercamientos que queremos efectuar sobre el mapa respectivo de la ciudad seleccionada, teniendo tres opciones de acercamiento 5%, 25% y 50%.

Ratón: Oprimimos el botón de la barra de iconos que contiene el dibujo de una lupa (se sume el botón), y con esto se activa la ventana que contiene la caja de diálogo que nos permite seleccionar una de tres opciones de aumento, ya sea 5%, 25% y 50%. A lado de cada una de las opciones se encuentra un circulo vacío. Para activar la opción, por ejemplo del 25%, colocamos el apuntador del ratón sobre el circulo vacío del letrero que dice 25% y se presiona el botón izquierdo del ratón, al hacer esto se rellena el circulo de la opción de 25% después se presiona el botón **OK** que se encuentra en la parte derecha de la caja de dialogo para indicar que estamos de acuerdo con este 25% de aumento.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **U** subrayada del menú **Ubicar Ciudad** y se presiona la letra subrayada **R** del menú descendente **Radio de acción**, y con esto se activa la ventana que contiene la caja de diálogo que nos permite seleccionar una de tres opciones de aumento, ya sea 5%, 25% y 50%. A lado de cada una de las opciones se encuentra un circulo vacío. Para activar la opción, por ejemplo del 25%, pulsamos la tecla **TAB** hasta colocarnos sobre el circulo vacío del letrero que dice 25%, al hacer esto se rellena el circulo de la opción de 25% y se presiona **Enter** para indicar que estamos de acuerdo con este 25% de aumento.

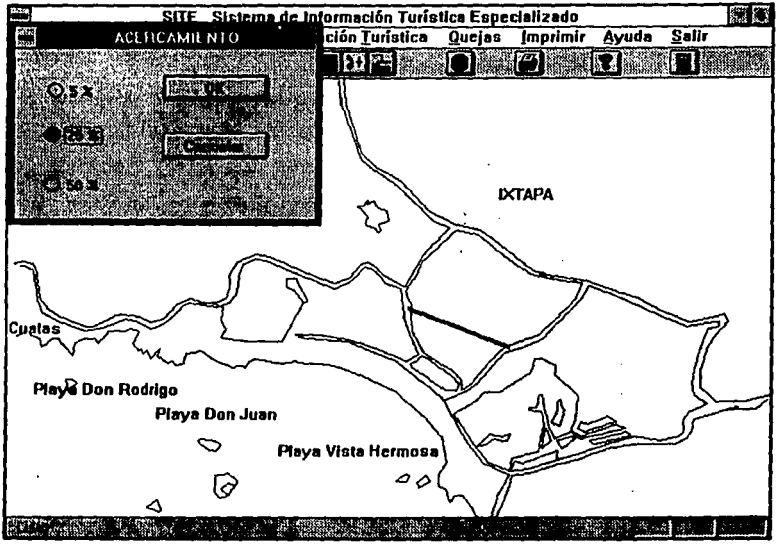


Fig. 3.2.8 Selección del Menú Ubicar Ciudad.

Para realizar el aumento sobre los mapas, basta con colocar el cursor del Ratón sobre el área del mapa que queremos visualizar mejor y pulsamos el botón izquierdo del ratón. Con esto se observa un aumento del 25% cada vez que pulsamos el ratón sobre el área respectiva.

Comando Ubicar

El comando Ubicar nos permite regresar al tamaño inicial de los mapas.

Ratón: Oprimimos el botón de la barra de iconos que contiene el dibujo de cuatro flechas que se expanden (se sume el botón), y con esto se regresa al tamaño inicial de la ventana.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **U** subrayada del menú **Ubicar Ciudad** y se presiona la letra subrayada **C** del menú descendente **Ciudad**, y con esto se regresa al tamaño inicial de la ventana.

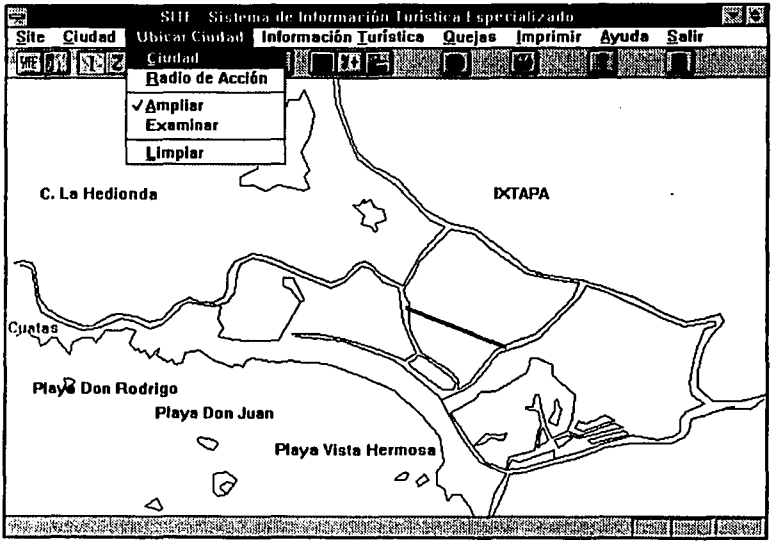


Fig. 3.2.9 Selección del comando ubicar.

Comando Limpiar

El comando Limpiar nos permite limpiar los mapas de las selecciones que hayamos realizado sobre ellos.

Ratón: Oprimimos el botón de la barra de iconos que contiene el dibujo de un borrador o goma de borrar (se sume el botón), y con esto se borran todas las selecciones anteriores.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **L** subrayada del menú **Limpiar** y se presiona la letra subrayada **L** del menú descendente **Limpiar**, y con esto se borran todas las selecciones anteriores.

Cómo usar el menú Información Turística

Dependiendo de la ciudad seleccionada, este menú nos permite accesar a los comandos de Bienes, Servicios y Atracciones Turísticas que son los que nos proporcionan la Información Turística del puerto de Ixtapa - Zihuatanejo.

Comando Bienes

Este comando nos permite escoger un nombre de un Bien entre la lista de **Bienes** de las ciudades de Ixtapa o Zihuatanejo.

Ratón: Colocando el apuntador del ratón sobre el botón de la barra de iconos que contiene la letra A (se sume el botón), se presiona este para activar la ventana que contiene la lista de todos los bienes de cada ciudad.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **I** subrayada del menú **Información Turística** y se presiona la letra subrayada **B** del menú descendente **Bienes**, para activar la ventana que contiene la lista de todos los bienes que ofrece cada ciudad.

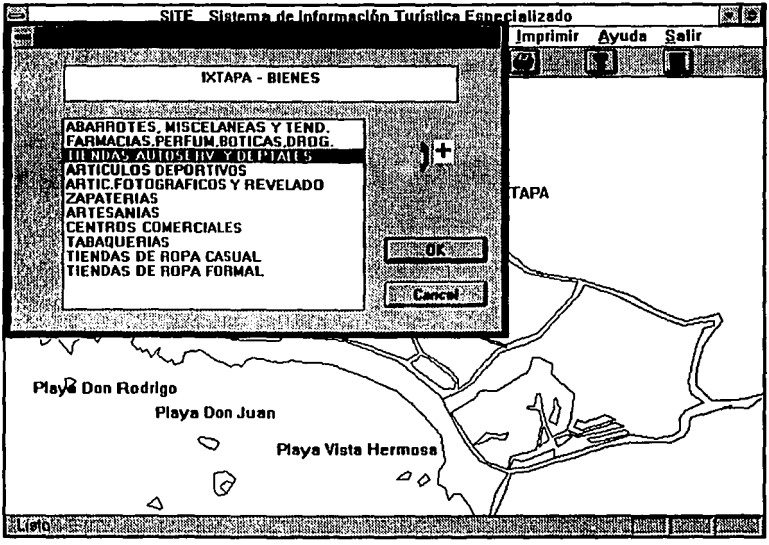


Fig. 3.2.10 Selección del comando Bienes

En la ventana que se abre se despliega una lista con todos los nombres de los Bienes que la ciudad ofrece. Use esta lista para seleccionar el nombre del Bien del que desee obtener información. Este tipo de ventana permanece abierta siempre de forma que pueda ver la lista. Puede recorrer la lista al oprimir el botón del ratón en las flechas hacia arriba y

hacia abajo, en la barra de recorrido que se ubica a la derecha de la caja de lista o bien al oprimir las teclas de dirección hacia arriba o hacia abajo. Se puede desplazar a una alternativa con rapidez al oprimir una vez el botón en la caja y escribir la primera letra de la alternativa deseada. La lista se recorre hasta los nombres que comienzan con esa letra. Cuando vea el elemento que desea seleccionar, oprima el botón sobre él, o bien oprima la tecla de dirección hacia arriba o hacia abajo hasta que el elemento quede resaltado.

Comando Servicios

Este comando nos permite escoger entre la lista de **Servicios** de las ciudades de Ixtapa o Zihuatanejo.

Ratón: Colocando el apuntador del ratón sobre el botón de la barra de iconos que contiene los dibujos de un teléfono y cruz roja (se sume el botón), se presiona este, para activar la ventana que contiene la lista de todos los servicios de cada ciudad.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **I** subrayada del menú **Información Turística** y se presiona la letra subrayada **S** del menú descendente **Servicios**, para activar la ventana que contiene la lista de todos los bienes que ofrece cada ciudad.

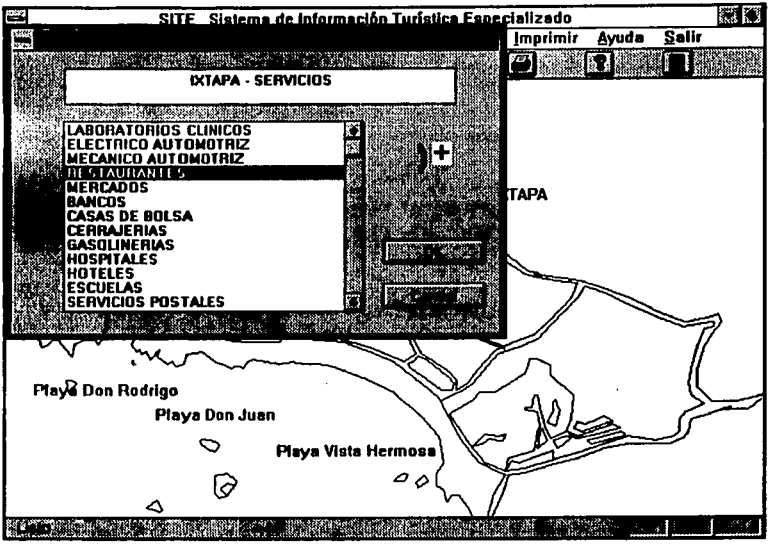


Fig. 3.2.11 Selección del comando Servicios

En la ventana que se abre, se despliega una lista con todos los nombres de los Servicios que la ciudad ofrece. Use esta lista para seleccionar el nombre del servicio del que desee obtener información. Este tipo de ventana permanece abierta siempre de forma que pueda ver la lista. Puede recorrer la lista al oprimir el botón del ratón en las flechas hacia arriba y hacia abajo en la barra de recorrido que se ubica a la derecha de la caja de lista o bien al oprimir las teclas de dirección hacia arriba o hacia abajo. Se puede desplazar a una alternativa con rapidez al oprimir una vez el botón en la caja y escribir la primera letra de la alternativa deseada. La lista se recorre hasta los nombres que comienzan con esa letra.

Cuando vea el elemento que desea seleccionar, oprima el botón sobre él, o bien oprima la tecla de dirección hacia arriba o hacia abajo hasta que el elemento quede resaltado.

Comando Atracciones Turísticas

Este comando nos permite escoger entre la lista de **Atracciones Turísticas** de las ciudades de Ixtapa o Zihuatanejo.

Ratón: Colocando el apuntador del ratón sobre el botón de la barra de iconos que contiene el dibujo de dos playas (se sume el botón), se presiona este para activar la ventana que contiene la lista de todas las Atracciones Turísticas de cada ciudad.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **f** subrayada del menú **Información Turística** y se presiona la letra subrayada **A** del menú descendente **Atracciones Turísticas**, para activar la ventana que contiene la lista de todas las Atracciones Turísticas que ofrece cada ciudad.

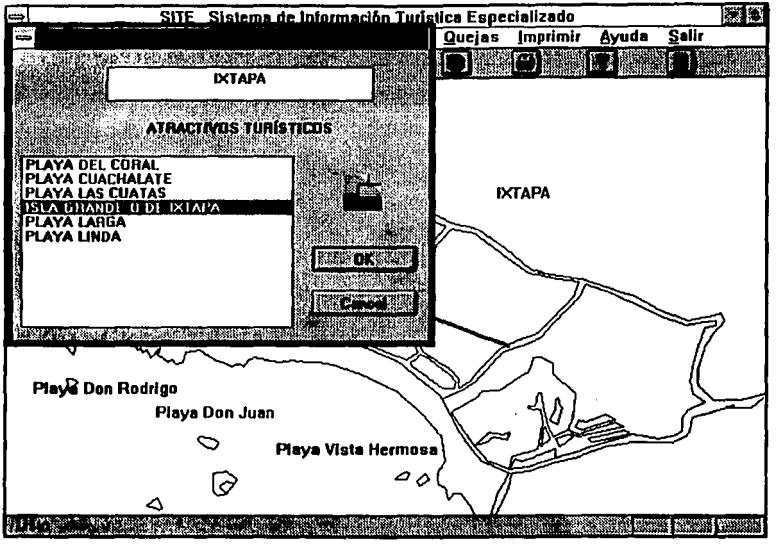


Fig. 3.2.12 Selección del comando Atracciones Turísticas

En la ventana que se abre, se despliega una lista con todos los nombres de las Atracciones Turísticas que la ciudad ofrece. Use esta lista para seleccionar el nombre de la Atracción Turística de la que desee obtener información. Este tipo de ventana permanece abierta siempre de forma que pueda ver la lista. Puede recorrer la lista al oprimir el botón del ratón en las flechas hacia arriba y hacia abajo en la barra de recorrido que se ubica a la derecha de la caja de lista o bien al oprimir las teclas de dirección hacia arriba o hacia abajo. Se puede desplazar a una alternativa con rapidez al oprimir una vez el botón en la caja y escribir la primera letra de la alternativa deseada. La lista se recorre hasta los nombres que

comienzan con esa letra. Cuando vea el elemento que desea seleccionar, oprima el botón sobre él, o bien oprima la tecla de dirección hacia arriba o hacia abajo hasta que el elemento quede resaltado.

Cómo Obtener Información

Una vez que se haya seleccionado el nombre del Bien, Servicio o Atracción Turística, sobre el mapa de la ciudad se dibujaran una serie de puntos rojos, cada punto indica la ubicación de un sitio de la ciudad. Para obtener información sobre ese sitio, hay que seleccionar primeramente de la barra de iconos el botón verde que contiene la letra **E** que corresponde al comando de examinar o de la barra de menu hay que oprimir la tecla **Alt** para activar la barra de menú, se oprime la letra **U** subrayada del menu **Ubicar Ciudad** y se presiona la letra subrayada **E** del menú descendente **Examinar.**

Una vez hecho esto ya se puede obtener la información de ese sitio al unicamente tener que colocar el apuntador del ratón sobre un punto rojo y presionar su botón izquierdo. Al hacer esto se desplegará una ventana con la información correspondiente de ese sitio, como es la dirección del lugar, el tipo de servicio que ofrece, su horario de atención, su teléfono, su categoría o la información cultural de ese lugar.

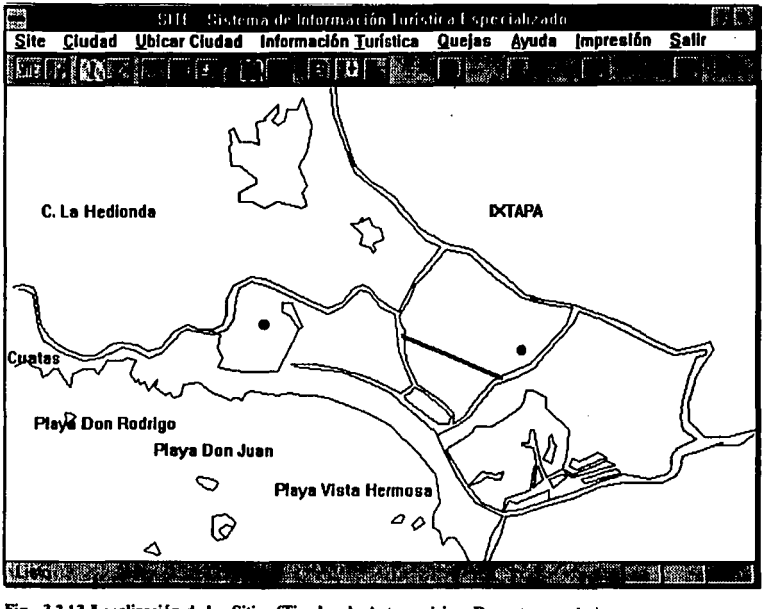


Fig. 3.2.13 Localización de los Sitios (Tiendas de Autoservicio y Departamentales).

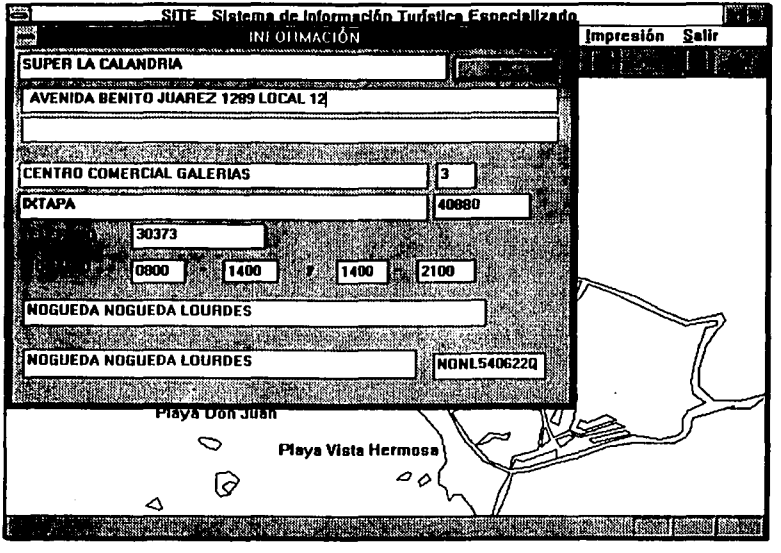


Fig. 3.2.14 Despliegue de información.

Si se desea hacer un acercamiento sobre algún sitio en particular se tendrá que activar el comando Amplificar de la barra de iconos, presionando el botón verde que contiene la tecla A o de la barra de menú oprimiendo la tecla **Alt** para activar la barra de menú, se oprime la letra U subrayada del menu **Ubicar Ciudad** y se presiona la letra subrayada A del menú descendente **Amplificar.** Después se realiza el procedimiento de **Cómo usar el menú Ubicar Ciudad** anteriormente explicado.

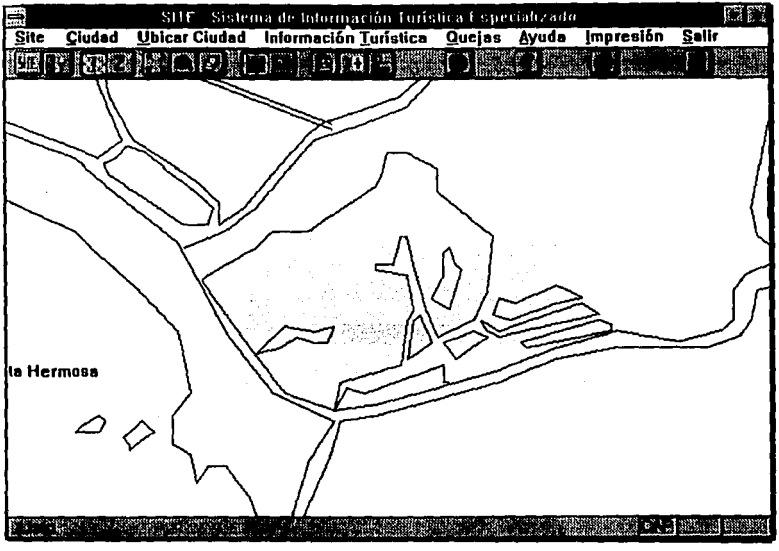


Fig. 3.2.15 Selección del comando Amplificar.

Cómo usar el Comando Quejas

Este comando nos permite accesar a un formato donde el usuario puede manifestar alguna sugerencia o queja respecto a Site. Se puede accesar a este comando ya sea usando la barra de iconos o desde la barra de menú.

Ratón: Oprimimos el botón de la barra de iconos que contiene el dibujo de una cara enojada (se sume el botón), y con esto se activa la ventana que contiene el formato de quejas de Site.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **Q** subrayada del menu **Quejas** y con esto se activa la ventana que contiene el formato de quejas de Site.

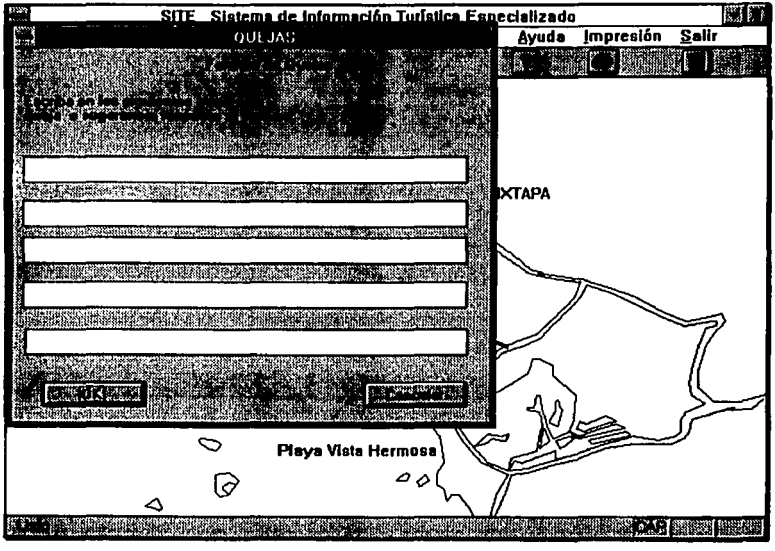


Fig. 3.2.16 Selección del Formato de Quejas.

Cómo usar el comando Ayuda

Este comando nos permite accesar a un formato donde el usuario puede obtener Ayuda respecto al funcionamiento del sistema Site. Se puede accesar a este comando ya sea usando la barra de iconos o desde la barra de menú.

Ratón: Oprimimos el botón de la barra de iconos que contiene el dibujo de un signo de interrogación (se sume el botón), y con esto se activa la ventana que contiene la información de Site.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **A** subrayada del menu **Ayuda** y con esto se activa la ventana que contiene la información de Site.

Cómo usar el menu de Impresión

El menú de impresión nos permite obtener una gráfica del sitio impresa con la información respectiva del lugar. Para usar el menu de impresión siga estos pasos:

Elija el comando Impresión de la barra de iconos o de la barra de menú.

Ratón: Oprimimos el botón de la barra de iconos que contiene el dibujo de una Impresora.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **I** subrayada del menu **Impresión.**

Cómo salir de Site Para Windows

Para salir de Site para Windows, siga estos pasos:

Elija el comando salir de la barra de iconos o de la barra de menú.

Ratón: Oprimimos el botón de la barra de iconos que contiene el dibujo de una puerta.

Teclado: Presionamos la tecla **Alt** para activar la barra de menú, se oprime la letra **S** subrayada del menu **Salir**

□

3.3 Mantenimiento

Actualización y depuración de la infromación

En cuanto al mantenimiento del sistema, éste se realizará unicamnete para las bases de datos, para lo cual, se actualizará la información dentro de un módulo diseñado para cumplir con tal función, dicho módulo contará con las funciones básiscas de actualización tales como: altas, bajas, cambios y consultas. De igual forma se pretende depurar toda la información dependiendo de los tiempos que se establezcan para realizar ésta. El código que genera Paradox para DOS, que lleva acabo el mantenimiento de la información ésta contenido en el Apéndice B.

CONCLUSIONES

Conclusiones

La terminación del sistema se logró principalmente al empleo de las técnicas Orientadas a objetos, implementadas mediante el lenguaje de programación Visual C + +. Gracias a la utilización de tales técnicas se obtuvieron las siguientes conclusiones:

- Se logró optimizar el proceso de información turística en el puerto de Ixtapa-Zihuatanejo.

- El software que se realizo tuvo el propósito de que su uso con una interfaz usuario gráfica fuera lo más sencillo posible.

- El sistema resulta ser amigable para cualquier persona, mexicana o extranjera que no posea ningún conocimiento de computación, ya que proporciona acceso mediante símbolos gáficos de tipo universal, con los cuales no es necesario leer las opciones del menú principal, sino únicamente presionar un botón y así obtener la información.

- El sistema permite al usuario conocer la gran variedad de Bienes, Servicios y Atracciones Turísticas que el puerto de Ixtapa Zihuatanejo ofrece, además de los lugares donde puede adquirirlos, los precios de venta, sin tener que moverse del sitio donde esta efectuando la consulta, para su mayor comodidad

- Se logró crear una pantalla sugestiva al usuario de manera que este, sólo tiene que apuntar a iconos o elementos de un menú desplegado, que están relacionados con los objetos. Tomando en cuenta que el proceso de ver y apuntar es más fácil que recordar y escribir. El software de los futuros sistemas se debe de diseñar, teniendo presente que

contarán con una interfaz usuario gráfica, para así poder trabajar con las diversas interfaces dominantes de este tipo (como la Macintosh).

- El diseño de software se generó, a partir de reglas, formatos y clases basadas en depósitos previamente establecidos.

- El código no se generó de forma manual, en la medida de los posible. La aplicación se realizó a partir de componentes ya existentes. Muchos de los cuales están construidos de modo que se pueden adaptar a un diseño particular. Las nuevas herramientas de programación deben proporcionar la máxima capacidad posible para la generación de código.

- En el diseño de las pantallas se logró de crearlas de buena calidad, puesto que se integrarón a partir de componentes que han sido probados y verificados varias veces.

- La programación se volvió más sencilla ya que los programas se conjuntaron a partir de piezas pequeñas, cada una de las cuales, en general se pueden crear varias veces.

- Este sistema ayudará a incrementar el nivel turístico del sitio de interés, ya que al contemplar diversos giros comerciales, el usuario tendrá mayores opciones a elegir y por lo tanto, mayor número de negocios aumentarán sus ingresos.

Mientras más pronta sea la difusión del análisis y el diseño orientado a objetos, más rápidamente pasará la creación del software de ser una industria de campo a una disciplina de la ingeniería.

BIBLIOGRAFIA

BIBLIOGRAFIA

Eckel, Bruce
-Aplique C++

Gurewich. Ori & Gurewich, Nathan
-Paradox 4.5 for Windows Unleashed

James Martin, Jame J. Odell
-Analisis y Diseñó Orientado a Objetos

Kryuglinski David J.
-Inside Visual C++

Kryuglinski David J.
-Progrese con Visual C++.

Murray, Willian H.

Microsoft C/C++7 Manual de Referencia.

Padwick, Gordon

-Paradox For Windows.

Saison, Ted
-Programación Orientada a Objetos con Borland C++

Setrag Khoshafian, Razmik Abnous
-Object Orientation
Concepts, Languages, Databases, User Interfaces

Stroustrup, Bjarne
-Turbo C/C++ The Complete Reference

Roetxheim, William
-Programming Windows with Borland C++

Wiener, Richard S., Pinson, Lewis J.
-An Introduction to Objets Oriented Programming and C++

Timothy Budd
-An Introduction to Object Oriented Programming

Mark Mullin
-Object Oriented Program Design with examples in C ++

James A. Senn.
-Análisis y Diseño de Sistemas de Información

Henry F. Korth.
-Fundamentos de Bases de Datos

REVISTAS

-PC Magazine en Español
Volumen 3-Número 12
Diciembre 1992. Pag. 96,100
"Base de Datos"

-PC Magazine en Español
Volumen 5-Número 2
Febrero de 1994. Pag. 10
"Visual C++"

-PC World
Junio 1994. Pags, 11,12,14,15,16,17,18,20,21,22
"La Base de Datos Correcta para Usted"

-PC World
Mayo 1994. Pag. 10
"Biblioteca gráfica C++ para Windows 3.1"

-PC Actual

"Programación Visual."

Madrid, Marzo 1994, Año V, Nº 51.

pag. 102 - 130.

-PC Actual.

"Compiladores de C++"

Madrid, Mayo 1994, Año V, Nº 53.

pag. 138 - 148.

Apendice

A

ARCHIVO MAINFRM.CPP

```
// mainfrm.cpp : implementation of the CMainFrame
class
//
#include "stdafx.h"
#include "site.h"
#include "mainfrm.h"
#include <stdlib.h>
#include <malloc.h>
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame,
CFrameWnd)
BEGIN_MESSAGE_MAP(CMainFrame,
CFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
        ON_WM_CREATE()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////
// arrays of IDs used to initialize control bars

// toolbar buttons - IDs are command buttons
static UINT BASED_CODE buttons[] =
{
        // same order as in the bitmap 'toolbar.bmp'
        ID_APP_ABOUT,
        ID_SITE_CONOCER,
                ID_SEPARATOR,
        ID_CIUDAD_IXTAPA,
        ID_CIUDAD_ZIHUATANEJO,
                ID_SEPARATOR,
        ID_UBICAR_CIUDAD,
        ID_UBICAR_RADIODEACCIN,
        ID_UBICAR_LIMPIAR,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_UBICARCIUDAD_BUSCAR,
        ID_UBICARCIUDAD_EXAMINAR,
                ID_SEPARATOR,
                ID_SEPARATOR,
ID_INFORMACINTURSTICA_BIENES,
ID_INFORMACINTURSTICA_SERVICIOS,
ID_INFORMACINTURSTICA_ATRACTIVOS
        ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_APP_QUEJAS,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_APP_AYUDA,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_APP_AYUDA,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_APP_EXIT,
};
static UINT BASED_CODE indicators[] =
{
        ID_SEPARATOR,                   //
status line indicator
        ID_INDICATOR_CAPS,
        ID_INDICATOR_NUM,
        ID_INDICATOR_SCRL,
};
/////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
        // TODO: add member initialization code
here
}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT
lpCreateStruct)
{
        if (CFrameWnd::OnCreate(lpCreateStruct)
== -1)
                return -1;

        if (!m_wndToolBar.Create(this) ||
        !m_wndToolBar.LoadBitmap(IDR_MAINFR
AME) ||

        !m_wndToolBar.SetButtons(buttons,
         sizeof(buttons)/sizeof(UINT)))
        {
                TRACE("Failed to create
toolbar\n");
                return -1;          // fail to create
        }

        if (!m_wndStatusBar.Create(this) ||
```

```
        !m_wndStatusBar.SetIndicators(indicators,
                sizeof(indicators)/sizeof(UINT)))
        {
                TRACE("Failed to create status
bar\n");
                return -1;          // fail to create
        }

        return 0;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
        CFrameWnd::AssertValid();
}

void CMsinFrame::Dump(CDumpContext& dc) const
{
        CFrameWnd::Dump(dc);
}

#endif //_DEBUG
```

ARCHIVO SITEDOC.CPP

```
// sitedoc.cpp : implementation of the CSiteDoc class
//
#include "stdafx.h"
#include "site.h"
#include "sitedoc.h"
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSiteDoc
IMPLEMENT_DYNCREATE(CSiteDoc, CDocument)

BEGIN_MESSAGE_MAP(CSiteDoc, CDocument)
        //{{AFX_MSG_MAP(CSiteDoc)
                // NOTE - the ClassWizard will add
and remove mapping macros here.
                //    DO NOT EDIT what you see in
these blocks of generated code !
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////


BOOL CSiteDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;
        // TODO: add reinitialization code here
        // (SDI documents will reuse this document)
        return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CSiteDoc serialization
void CSiteDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                // TODO: add storing code here
        }
        else
        {
                // TODO: add loading code here
        }
}


/////////////////////////////////////////////////////////////////////////////
// CSiteDoc diagnostics

#ifdef _DEBUG
void CSiteDoc::AssertValid() const
{
        CDocument::AssertValid();
}

void CSiteDoc::Dump(CDumpContext& dc) const
{
        CDocument::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CSiteDoc commands
```

ARCHIVO SITEWIEW.CPP

```
// siteview.cpp : implementation of the CSiteView class
//
#include "stdafx.h"
#include "site.h"
#define MARCA 9999
#include "sitedoc.h"
#include "siteview.h"
#include "dradio.h"
#include "dcuadro.h"
#include "dselecc.h"
#include "dbienser.h"
#include "dinfo.h"
#include "dquejas.h"
#include "dayuda.h"
#ifdef _DEBUG
#undef THIS_FILE
```

```
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif
//////////////////////////////////////////////////////////////////////
// CSiteView
IMPLEMENT_DYNCREATE(CSiteView, CView)
BEGIN_MESSAGE_MAP(CSiteView, CView)
        //{{AFX_MSG_MAP(CSiteView)
        ON_COMMAND(ID_UBICAR_CIUDAD,
OnUbicarCiudad)
        ON_WM_LBUTTONDOWN()
        ON_COMMAND(ID_CIUDAD_IXTAPA,
OnCiudadIxtapa)
        ON_COMMAND(ID_CIUDAD_ZIHUATAN
EJO, OnCiudadZihuatanejo)
        ON_WM_RBUTTONDOWN()
        ON_COMMAND(ID_UBICAR_RADIODEA
CCIN, OnUbicarRadiodeaccin)
        ON_COMMAND(ID_SITE_CONOCER,
OnSiteConocer)
        ON_COMMAND(ID_INFORMACINTURST
ICA_ATRACTIVOSTURSTICOS,
OnInformacintursticaAtractivostursticos)
        ON_COMMAND(ID_INFORMACINTURST
ICA_SERVICIOS, OnInformacintursticaServicios)
        ON_COMMAND(ID_INFORMACINTURST
ICA_BIENES, OnInformacintursticaBienes)
        ON_COMMAND(ID_UBICAR_LIMPIAR,
OnUbicarLimpiar)
        ON_COMMAND(ID_UBICARCIUDAD_BU
SCAR, OnUbicarciudadBuscar)
        ON_COMMAND(ID_UBICARCIUDAD_EX
AMINAR, OnUbicarciudadExaminar)
        ON_UPDATE_COMMAND_UI(ID_CIUDA
D_IXTAPA, OnUpdateCiudadIxtapa)
        ON_UPDATE_COMMAND_UI(ID_CIUDA
D_ZIHUATANEJO, OnUpdateCiudadZihuatanejo)
        ON_UPDATE_COMMAND_UI(ID_UBICA
RCIUDAD_BUSCAR, OnUpdateUbicarciudadBuscar)
        ON_UPDATE_COMMAND_UI(ID_UBICA
RCIUDAD_EXAMINAR,
OnUpdateUbicarciudadExaminar)
        ON_COMMAND(ID_APP_QUEJAS,
OnAppQuejas)
        ON_COMMAND(ID_APP_AYUDA,
OnAppAyuda)
        //}}AFX_MSG_MAP
        // Standard printing commands
        ON_COMMAND(ID_FILE_PRINT,
CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIE
W, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

typedef struct      {       entero x;
                            entero y;
                            char titulo[31]; }
        desc_sit;

typedef struct      {       entero reg;
                            entero x;
                            entero y;
                            byte   giro; } lugar_sit;

entero *a_apt;
entero *a_lin; // puntos del mapa
int ancho = 4;
double acerca = .05;
byte ciudad = 0, limite = 1, giro_act = 0, examinar=0,
buscar=1;
long max_x, max_y, min_x, min_y;
long *stack, *a_stack;
double derecho, abajo, dif_x,dif_y;
char *na_datos = "i_datos.dbf", *na_infosi =
"i_infosi.dbf", *na_datost = "i_datost.dbf";
char *na_lugt = "i_lugt.dbf", *na_giro = "i_gir.dbf";
CFile fsit;
desc_sit  *a_dat;  // sitios en el mapa
lugar_sit *a_inf;  // Relación de Bienes y Servicios en
el archivo
lugt_sit  *a_lugt; // lista de lugares turísticos
desc_giro *a_giro; // Relación de giros

//////////////////////////////////////////////////////////////////////
void leer() {
        CFile temp;
        char *buff, *c_apt4, *c_apt5, *c_apt2;
        entero *a_apt, x, y, i, n_apt;
        desc_sit *p_dat;
        lugar_sit *i_apt;
        lugt_sit *l_apt;
        desc_giro *g_apt;

        buff  = (char*) malloc(200);

c_apt4 = (char*) malloc(5); *(c_apt4+4) = '\0';
c_apt5 = (char*) malloc(6); *(c_apt5+5) = '\0';
c_apt2 = (char*) malloc(3); *(c_apt2+2) = '\0';


        // Lectura de puntos del mapa

        temp.Open(na_datos,CFile::modeRead);

        temp.Read(buff,4);
        temp.Read(&n_apt,2);
        temp.Read(buff,2);
        a_lin = (entero*)
malloc((n_apt*2+1)*sizeof(entero));
        a_apt = a_lin;
        temp.Read(buff,90);

  for(i=1;i<=n_apt;i++) {
  temp.Read(c_apt4,1);
temp.Read(c_apt4,4); x = atoi(c_apt4);
temp.Read(c_apt4,4); y = atoi(c_apt4);

        *a_apt++ = x;
        *a_apt++ = y;
  };
  *a_apt++ = FINAL;
```

```
    temp.Close();

            // Lectura de Relación de giros

            temp.Open(na_giro,CFile::modeRead);

            temp.Seek(4,CFile::begin);
            temp.Read(&n_apt,2);
            temp.Seek(2,CFile::current);
            a_giro = (desc_giro*)
malloc((n_apt+1)*sizeof(desc_giro));
            g_apt = a_giro;
            temp.Seek(121,CFile::current);

    for(i=1;i<=n_apt;i++) {
                        temp.Seek(1,CFile::current);
                        g_apt->marca = 0;
                        temp.Read(c_apt2,2);
                        g_apt->giro = atoi(c_apt2);
                        temp.Read(g_apt->nombre,36);
                        g_apt->nombre[36] = '\0';
                        temp.Read(c_apt4,1);
                        g_apt->tipo = *c_apt4 - '0';

                        *g_apt++;
    };
    g_apt->marca = FINAL;
    temp.Close();


//Lectura ubicacion en el archivo de Bienes y Servicios

            temp.Open(na_infosi,CFile::modeRead);
            temp.Seek(4,CFile::begin);
            temp.Read(&n_apt,2);
            temp.Seek(2,CFile::current);
            a_inf = (lugar_sit*) malloc(
            (n_apt+1)*sizeof(lugar_sit));
            i_apt = a_inf;
            temp.Seek(634,CFile::current);

    for(i=1;i<=n_apt;i++) {
    temp.Read(c_apt5,1);
    temp.Read(c_apt5,5); i_apt->reg  = atoi(c_apt5);
temp.Read(c_apt4,4); i_apt->x    = atoi(c_apt4);
temp.Read(c_apt4,4); i_apt->y    = atoi(c_apt4);
temp.Read(c_apt2,2); i_apt->giro = atoi(c_apt2);
temp.Seek(329,CFile::current);
            i_apt++;
    };
    i_apt->x = FINAL;
    i_apt->y = FINAL;

    temp.Close();

// Lectura sitios en el mapa

            temp.Open(na_datost,CFile::modeRead);
            temp.Read(buff,4);
            temp.Read(&n_apt,2);
            temp.Read(buff,2);
            a_dat = (desc_sit*)
malloc((n_apt+1)*sizeof(desc_sit));
            p_dat = a_dat;
            temp.Read(buff,121);

    for(i=1;i<=n_apt;i++) {
    temp.Read(c_apt5,1);
temp.Read(c_apt4,4); p_dat->x    = atoi(c_apt4);
temp.Read(c_apt4,4); p_dat->y    = atoi(c_apt4);

temp.Read(p_dat->titulo,30);
            p_dat->titulo[30] = '\0';
            p_dat++;
    };

    p_dat->x = FINAL;
    p_dat->y = FINAL;
    temp.Close();


// Lectura de lugares turísticos

            temp.Open(na_lugt,CFile::modeRead);
            temp.Read(buff,4);
            temp.Read(&n_apt,2);
            temp.Read(buff,2);
            a_lugt = (lugt_sit*)
malloc((n_apt+1)*sizeof(lugt_sit));
            l_apt = a_lugt;
            temp.Read(buff,89);

    for(i=1;i<=n_apt;i++) {
                        temp.Read(c_apt5,1);
                        l_apt->marca = 0;

                        temp.Read(l_apt->nombre,36);
                        temp.Read(l_apt->archivo,12);

                        l_apt->nombre[36] = '\0';
                        l_apt->archivo[12] = '\0';
                        l_apt++;
    };

    l_apt->marca = FINAL;
    temp.Close();
            free(buff);
            free(c_apt5);
            free(c_apt4);
            free(c_apt2);
}

void trans(long x, long y, long *xp, long *yp) {
            *xp = ((x-min_x)/dif_x) * derecho;
            *yp = (1 - ((y-min_y)/dif_y)) * abajo;
}
////////////////////////////////////////////////////////////////////////
// CSiteView construction/destruction

CSiteView::CSiteView()
```

```
{
        // TODO: add construction code here
        stack = (long*) malloc( 500*sizeof(long));
        leer();
        fsit.Open(na_infosi,CFile::modeRead);

}

CSiteView::~CSiteView()
{
        fsit.Close();
}

/////////////////////////////////////////////////////////////////////////////
// CSiteView drawing

void CSiteView::OnDraw(CDC* pDC)
{
        CSiteDoc* pDoc = GetDocument();

        // TODO: add draw code here
        CClientDC dc( this );
        CRect rect, sitio;
        CBrush   sit_brush;
        entero *a_apt;
        long xp, yp, x, y;
        double fact_m;
        byte flag = 1;
        desc_sit *p_dat;
        lugar_sit *i_inf;

        GetClientRect( rect );

        dc.SetTextAlign( TA_BASELINE |
TA_CENTER );
        dc.SetTextColor( ::GetSysColor(
COLOR_WINDOWTEXT ) );
        dc.SetBkMode(TRANSPARENT);

        a_apt = a_lin;

        if (limite) {
                min_x = *a_apt++;
                max_x = *a_apt++;
                min_y = *a_apt++;
                max_y = *a_apt++;

                dif_x=max_x-min_x;
                dif_y=max_y-min_y;
                abajo=rect.bottom;
                derecho=rect.right;

                fact_m=dif_x/dif_y;
          if (abajo*fact_m>derecho)
                abajo=derecho/fact_m;
          else
                derecho=abajo*fact_m;

                a_stack = stack;
                *a_stack++ = (long) min_x;
                *a_stack++ = (long) max_x;
                *a_stack++ = (long) min_y;
                *a_stack++ = (long) max_y;

        } else {
                a_apt+=4;
                limite = 1;
  };

        while (*a_apt != FINAL) {

                x = *a_apt++;
                y = *a_apt++;

                if ((x!=MARCA) &&
(y!=MARCA)) {
                        flag = 1;
                } else {
                        x = *a_apt++;
                        y = *a_apt++;
                        flag = 0;
                };
                trans(x,y,&xp,&yp);

                if (flag) {dc.LineTo( (int) xp, (int)
yp);}
                        else  {dc.MoveTo( (int)
xp, (int) yp);};

        };


/*
        POINT pt;
        pt.x = 10;
        pt.y = 20;
        CClientDC dc(this);
        dc.SetPixel(pt.x,pt.y,RGB(255,0,0));
*/
        //dc.SelectStockObject(GRAY_BRUSH);

        p_dat = a_dat;
        while (p_dat->x!=FINAL) {
                        x   = p_dat->x;
                        y   = p_dat->y;

                        if (min_x<=x &&
x<=max_x && min_y<=y && y<=max_y) {

        trans(x,y,&xp,&yp);
                                dc.TextOut(
(int) xp, (int) yp, p_dat->titulo, 30 );
                        };

                        p_dat++;
        };
```

```
        i_inf = a_inf;
        while (i_inf->x!=FINAL) {
                x = i_inf->x;
                y = i_inf->y;

        if (i_inf->giro==giro_act && min_x<=x &&
x<=max_x &&
         min_y<=y && y<=max_y) {
        trans(x,y,&xp,&yp);
dc.Arc( (int) xp-ancho, (int) yp-ancho, (int) xp+ancho,
 (int) yp+ancho, (int) xp-ancho, (int) yp,
 (int) xp-ancho, (int) yp);
                        };

        i_inf++;
        };

        sit_brush.CreateSolidBrush(RGB(255,60,70)
);
        dc.SelectObject(&sit_brush);

        i_inf = a_inf;
        while (i_inf->x!=FINAL) {
                        x = i_inf->x;
                        y = i_inf->y;

        if (i_inf->giro==giro_act && min_x<=x &&
        x<=max_x &&
        min_y<=y && y<=max_y) {

        trans(x,y,&xp,&yp);
        dc.FloodFill( (int) xp, (int) yp,RGB(0,0,0));
       };

        i_inf++;
        }

/*
dc.SetPixel(xp,abajo-yp,RGB(255,0,0));
dc.TextOut( ( rect.right / 2 ), ( rect.bottom / 2 ),s,
s.GetLength() );
  CSiteView::OnDraw(&dc);
*/


}

////////////////////////////////////////////////////////////
// CSiteView printing

BOOL CSiteView::OnPreparePrinting(CPrintInfo*
pInfo)
{
        // default preparation
        return DoPreparePrinting(pInfo);
}

void CSiteView::OnBeginPrinting(CDC* /*pDC*/,
CPrintInfo* /*pInfo*/)
{
        // TODO: add extra initialization before
printing
}

void CSiteView::OnEndPrinting(CDC* /*pDC*/,
CPrintInfo* /*pInfo*/)
{
        // TODO: add cleanup after printing
}



////////////////////////////////////////////////////////////
// CSiteView diagnostics

#ifdef _DEBUG
void CSiteView::AssertValid() const
{
        CView::AssertValid();
}

void CSiteView::Dump(CDumpContext& dc) const
{
        CView::Dump(dc);
}

CSiteDoc* CSiteView::GetDocument() // non-debug
version is inline
{
        ASSERT(m_pDocument-
>IsKindOf(RUNTIME_CLASS(CSiteDoc)));
        return (CSiteDoc*) m_pDocument;
}

#endif //_DEBUG

////////////////////////////////////////////////////////////
// CSiteView message handlers

void CSiteView::OnUbicarCiudad()
{

        // TODO: Add your command handler code
here
        CClientDC dc( this );
        CRect rect;

        GetClientRect( rect );

        limite = 1;
        InvalidateRect( rect ,TRUE);

}

void CSiteView::OnLButtonDown(UINT nFlags,
CPoint point)
{
        // TODO: Add your message handler code
here and/or call default
```

```
CView::OnLButtonDown(nFlags, point);

long xc, yc, xp, yp; //, x_inf, y_inf;
long x, y, inc_x, inc_y, mid_x, mid_y;
//unsigned short int *apt;
double fact_m;
unsigned int cont = 0, largo;
lugar_sit *i_inf;
DInfo dlg;
char *nom_com, *calle, *numero, *colonia,
*cp, *ho1, *ho2;
char *ho3, *ho4, *pser, *cat, *tel, *rz, *prop,
*rfc;

CClientDC dc( this );
CRect rect;

GetClientRect( rect );

xc = point.x;
yc = point.y;

if (buscar) {
x = (xc/derecho)*dif_x + min_x  ;
        y = (1 - yc/abajo)*dif_y + min_y ;

        mid_x = (long) dif_x/2.0;
        mid_y = (long) dif_y/2.0;
        inc_x=mid_x*(acerca/2.0);
        inc_y=mid_y*(acerca/2.0);

        if (x - (mid_x - inc_x) < min_x) {
        max_x = min_x + (dif_x - 2*inc_x);
        } else {
        if (x + (mid_x - inc_x) > max_x) {
        min_x = max_x - (dif_x - 2*inc_x);
                } else {
        min_x = x - (mid_x - inc_x);
        max_x = x + (mid_x - inc_x);
                }
        };

        if (y - (mid_y - inc_y) < min_y) {
        max_y = min_y + (dif_y - 2*inc_y);
        } else {
        if (y + (mid_y - inc_y) > max_y) {
        min_y = max_y - (dif_y - 2*inc_y);
                } else {
        min_y = y - (mid_y - inc_y);
        max_y = y + (mid_y - inc_y);
                }
        };

        dif_x=max_x-min_x;
        dif_y=max_y-min_y;
        abajo=rect.bottom;
        derecho=rect.right;

        fact_m=dif_x/dif_y;

        if (abajo*fact_m>derecho)
                abajo= derecho/fact_m;
        else
                derecho=abajo*fact_m;

                *a_stack++ = (long) min_x;
                *a_stack++ = (long) max_x;
                *a_stack++ = (long) min_y;
                *a_stack++ = (long) max_y;

                limite = 0;
                InvalidateRect( rect ,TRUE);

        } else {
                i_inf = a_inf;
                while (i_inf->x!=FINAL) {
                x = i_inf->x;
                y = i_inf->y;
                trans(x,y,&xp,&yp);
                if ((i_inf->giro==giro_act)    &&
                (abs(xp-xc)<=ancho) &&
                (abs(yp-yc)<=ancho)    ) {

fsit.Seek(643+345*cont,CFile::begin);

fsit.Seek(15,CFile::current);

        rz = (char*) malloc((largo=40)+1);
        fsit.Read(rz,largo);
         *(rz+largo) = '\0';
         dlg.m_rz = rz;

        colonia = (char*) malloc((largo=40)+1);

fsit.Read(colonia,largo);
        *(colonia+largo) = '\0';
        dlg.m_colonia = colonia;

        calle = (char*) malloc((largo=40)+1);
        fsit.Read(calle,largo);
        *(calle+largo) = '\0';
        dlg.m_calle = calle;

        numero = (char*) malloc((largo=15)+1);

fsit.Read(numero,largo);
         *(numero+largo) = '\0';
        dlg.m_numero = numero;

        cp = (char*) malloc((largo=5)+1);
        fsit.Read(cp,largo);
         *(cp+largo) = '\0';
        dlg.m_cp = cp;

        nom_com = (char*) malloc((largo=40)+1);

        fsit.Read(nom_com,largo);
        *(nom_com+largo) = '\0';
        dlg.m_nom_com = nom_com;
```

```
rfc = (char*) malloc((largo=13)+1);
fsit.Read(rfc,largo);
*(rfc+largo) = '\0';
dlg.m_rfc = rfc;

prop = (char*) malloc((largo=40)+1);
fsit.Read(prop,largo);
*(prop+largo) = '\0';
dlg.m_prop = prop;

tel = (char*) malloc((largo=10)+1);
fsit.Read(tel,largo);
*(tel+largo) = '\0';
dlg.m_tel = tel;

ho1 = (char*) malloc((largo=4)+1);
fsit.Read(ho1,largo);
*(ho1+largo) = '\0';
dlg.m_ho1 = ho1;

ho2 = (char*) malloc((largo=4)+1);
fsit.Read(ho2,largo);
*(ho2+largo) = '\0';
dlg.m_ho2 = ho2;

ho3 = (char*) malloc((largo=4)+1);
fsit.Read(ho3,largo);
*(ho3+largo) = '\0';
dlg.m_ho3 = ho3;

ho4 = (char*) malloc((largo=4)+1);
fsit.Read(ho4,largo);
*(ho4+largo) = '\0';
dlg.m_ho4 = ho4;

pser = (char*) malloc((largo=50)+1);
fsit.Read(pser,largo);
*(pser+largo) = '\0';
dlg.m_pser = pser;

cat = (char*) malloc((largo=20)+1);
fsit.Read(cat,largo);
*(cat+largo) = '\0';
dlg.m_cat = cat;

dlg.DoModal();
free(nom_com);    free(calle);
free(numero); free(colonia);
free(cp); free(ho1); free(ho2);
free(ho3);  free(ho4);
free(pser); free(cat);
free(tel); free(rz); free(prop);
free(rfc);
                    };

i_inf++;
cont++;
          };
};
}

void CSiteView::OnCiudadIxtapa()
{
        // TODO: Add your command handler code
here
        CClientDC dc( this );
        CRect rect;

        GetClientRect( rect );

        ciudad = 0;
        na_datos  = "i_datos.dbf";
        na_info = "i_info.dbf";
        na_sit = "i_sit.dbf";
        na_datost = "i_datost.dbf";
        na_lugt  = "i_lugt.dbf";
        na_giro  = "i_gir.dbf";

        free(a_inf);
        free(a_lin);
        free(a_dat);
        free(a_lugt);
        free(a_giro);
        giro_act = 0, examinar=0, buscar=1;
        leer();
        fsit.Close();
        fsit.Open(na_info,CFile::modeRead);
        InvalidateRect( rect ,TRUE);
}

void CSiteView::OnCiudadZihuatanejo()
{
        // TODO: Add your command handler code
here
        CClientDC dc( this );
        CRect rect;

        GetClientRect( rect );

        ciudad = 1;
        na_datos  = "z_datos.dbf";
        na_infosi = "z_info.dbf";
        na_sit = "z_sit.dbf";
        na_datost = "z_datost.dbf";
        na_lugt  = "z_lugt.dbf";
        na_giro  = "z_gir.dbf";
        free(a_inf);
        free(a_lin);
        free(a_dat);
        free(a_lugt);
        free(a_giro);
        giro_act = 0, examinar=0, buscar=1;
        leer();
        fsit.Close();
        fsit.Open(na_info,CFile::modeRead);
        InvalidateRect( rect,TRUE);
}

void CSiteView::OnRButtonDown(UINT nFlags,
CPoint point)
```

```
{
        // TODO: Add your message handler code
here and/or call default

        double fact_m;
        CView::OnRButtonDown(nFlags, point);
        CClientDC dc( this );
        CRect rect;
        if (examinar) return;
        GetClientRect( rect );

        if (a_stack-4<=stack) {
                return;
        };

        a_stack-=4;
        min_x = (long) *(a_stack-4);
        max_x = (long) *(a_stack-3);
        min_y = (long) *(a_stack-2);
        max_y = (long) *(a_stack-1);

        dif_x= (double) max_x-min_x;
        dif_y= (double) max_y-min_y;
        abajo= (double) rect.bottom;
        derecho= (double) rect.right;

        fact_m=dif_x/dif_y;
  if (abajo*fact_m>derecho)
        abajo= derecho/fact_m;
  else
        derecho= abajo*fact_m;

        limite = 0;
        InvalidateRect( rect ,TRUE);

}

void CSiteView::OnUbicarRadiodeaccin()
{
        // TODO: Add your command handler code
here

        DRadio dlg;

        if (acerca <= 0.05)
                dlg.m_radio1 = 0;
        else if (acerca <= 0.25)
                dlg.m_radio1 = 1;
        else if (acerca <= 0.50)
                dlg.m_radio1 = 2;

        if (dlg.DoModal()==1) {
                if (dlg.m_radio1 == 0)
                        acerca = 0.05;
                else if (dlg.m_radio1 == 1)
                        acerca = 0.25;
                else if (dlg.m_radio1 == 2)
                        acerca = 0.50;
        };

}

void CSiteView::OnSiteConocer()
{
        // TODO: Add your command handler code
here
        DCuadro dlg;
        CFile temp;
        double tam;
        char *buffer;

        temp.Open((ciudad)?"zihuata.txt":"ixtapa.txt"
,CFile::modeRead);
        tam = temp.GetLength();
        buffer = (char*) malloc(tam+1);
        temp.Read(buffer,tam);
        temp.Close();
        *(buffer+(int)tam) = '\0';

        dlg.m_titulo =
(ciudad)?"ZIHUATANEJO":"IXTAPA";
        dlg.m_texto = buffer;
        dlg.DoModal();
        free(buffer);
}

void
CSiteView::OnInformacintursticaAtractivosturisticos()
{
        // TODO: Add your command handler code
here
        DSelecc dlg;
        DCuadro dlg2;
        CFile temp;
        double tam;
        char *buffer;
        lugt_sit *l_apt;

        dlg.m_sitio =
(ciudad)?"ZIHUATANEJO":"IXTAPA";
        dlg.m_lugt = a_lugt;

        if (dlg.DoModal()!=1) return;

        l_apt = a_lugt;

        while ((strcmp(dlg.m_lista,l_apt-
>nombre)!=0) && (l_apt->marca!=FINAL))
                l_apt++;

  if (l_apt->marca==FINAL) return;

        temp.Open( l_apt->archivo
,CFile::modeRead);
        tam = temp.GetLength();
        buffer = (char*) malloc(tam+1);
        temp.Read(buffer,tam);
        temp.Close();
        *(buffer+(int)tam) = '\0';
```

```
        dlg2.m_titulo = dlg.m_lista;
        dlg2.m_texto = buffer;
        dlg2.DoModal();
        free(buffer);

}


void CSiteView::OnInformacinturisticaBienes()
{
        // TODO: Add your command handler code
here
        DBienSer dlg;
        desc_giro *g_apt;
        CClientDC dc( this );
        CRect rect;

        dlg.m_tipo = 0;
        dlg.m_giro = a_giro;
        dlg.m_titulo = (ciudad)?"ZIHUATANEJO -
BIENES":"IXTAPA - BIENES";

        if (dlg.DoModal()!=1) return;

        g_apt = a_giro;

        while ((strcmp(dlg.m_lista,g_apt-
>nombre)!=0) && (g_apt->marca!=FINAL))
                g_apt++;

  if (g_apt->marca==FINAL) return;

        giro_act = g_apt->giro;
        GetClientRect( rect );
        limite = 0;
        InvalidateRect( rect ,TRUE);
}

void CSiteView::OnInformacinturisticaServicios()
{
        // TODO: Add your command handler code
here
        DBienSer dlg;
        desc_giro *g_apt;
        CClientDC dc( this );
        CRect rect;

        dlg.m_tipo = 1;
        dlg.m_giro = a_giro;
        dlg.m_titulo = (ciudad)?"ZIHUATANEJO -
SERVICIOS":"IXTAPA - SERVICIOS";

        if (dlg.DoModal()!=1) return;

        g_apt = a_giro;

        while ((strcmp(dlg.m_lista,g_apt-
>nombre)!=0) && (g_apt->marca!=FINAL))
                g_apt++;

  if (g_apt->marca==FINAL) return;

        giro_act = g_apt->giro;
        GetClientRect( rect );
        limite = 0;
        InvalidateRect( rect ,TRUE);

}

void CSiteView::OnUbicarLimpiar()
{
        // TODO: Add your command handler code
here
        CClientDC dc( this );
        CRect rect;
        GetClientRect( rect );

        giro_act = 0;
        GetClientRect( rect );
        limite = 0;
        InvalidateRect( rect ,TRUE);


}

void CSiteView::OnUbicarciudadBuscar()
{
        // TODO: Add your command handler code
here
        buscar = 1;
        examinar = 0;

}

void CSiteView::OnUbicarciudadExaminar()
{
        // TODO: Add your command handler code
here
        buscar = 0;
        examinar = 1;

}

void CSiteView::OnUpdateCiudadIxtapa(CCmdUI*
pCmdUI)
{
        // TODO: Add your command update UI
handler code here
        pCmdUI->SetCheck(!ciudad);

}

void
CSiteView::OnUpdateCiudadZihuatanejo(CCmdUI*
pCmdUI)
{
        // TODO: Add your command update UI
handler code here
        pCmdUI->SetCheck(ciudad);
}
```

Apendice

B

MODULO DE MANTENIMIENTO EN PARADOX.

Para este módulo; la aplicación se correrá en el ambiente de DOS, en el cual se deberá tener acceso al subdirectorio de Paradox, tecleando el siguiente comando desde la raíz del disco duro :

C:\PDOX35\PPROG\PPROG [ENTER]

El archivo PPROG.EXE, es el ejecutable que permite que la aplicación se lleve acabo en Paradox. Una vez tecleado este comando, Paradox abrirá un menú de opciones en la pantalla del monitor; dentro de ese menú principal se deberá escoger la opción :

INICIAR [ENTER]

Al tomar esta opción el paquete preguntará por el nombre de la aplicación a correr, se deberá teclear el nombre de :

MANTE [ENTER]

Este nombre corresponde a la aplicación del módulo de mantenimiento. Una vez que se teclea éste, aparecerá en la pantalla el menú principal del módulo de mantenimiento de las bases de datos del sistema de información turística.

Dentro del menú principal del módulo, aparecerán las siguientes opciones :

Altas Bajas Modificaciones Consultas Salir

Al elegir la opción de ***ALTAS***, el sistema asume que se trata de un proceso de altas de información en cada una de las bases de datos del sistema, que en este caso, serán aquéllas en las cuales dicha información necesite actualizarse y darse de alta. El programa desplegará otro submenú dentro de esta opción, el cual contendrá las siguientes opciones:

Giros **Información-Gral.** **Productos** **Precios**

En cada una de estas opciones, los catálogos de *GIROS*, *INFORMACION-GRAL.*, *PRODUCTOS* y *PRECIOS*, se generará un registro dependiendo de la opción que se elija, y para dicho registro se deberá introducir la información que solicite cada una de éstas.

Para salir de cada una de las opciones y no realizar la operación se debe hacer con la tecla de **[ESC]**.

Al elegir la opción de ***BAJAS*** del menú principal, el sistema asume que se trata de un proceso de baja de información en cada una de las bases de datos del sistema, que en este caso, serán aquéllas en las cuales dicha información necesite actualizarse y darse de baja. El programa desplegará otro submenú dentro de esta opción, el cual contendrá las siguientes opciones:

Giros **Información-Gral.** **Productos** **Precios**

En cada una de estas opciones, los catálogos de *GIROS*, *INFORMACION-GRAL.*, *PRODUCTOS* y *PRECIOS*, se borrará un registro dependiendo de la opción que se elija.

Para salir de cada una de las opciones y no realizar la operación se debe hacer con la tecla de **[ESC]**.

Al elegir la opción de ***MODIFICACIONES*** del menú principal, el sistema asume que se trata de un proceso de modificación de información en cada una de las bases de datos del sistema, que en este caso, serán aquéllas en las cuales dicha información necesite actualizarse y modificarse. El programa desplegará otro submenú dentro de esta opción, el cual contendrá las siguientes opciones:

Giros **Información-Gral.** **Productos** **Precios**

En cada una de estas opciones, los catálogos de *GIROS*, *INFORMACION-GRAL.*, *PRODUCTOS* y *PRECIOS*, se modificarán los campos de un registro dependiendo de la opción que se elija.

Para salir de cada una de las opciones y no realizar la operación se debe hacer con la tecla de **[ESC]**.

Al elegir la opción de ***CONSULTAS*** del menú principal, el sistema asume que se trata de un proceso de consulta de información en cada una de las bases de datos del sistema, que en este caso, serán aquéllas en las cuales dicha información necesite consultarse. El programa desplegará otro submenú dentro de esta opción, el cual contendrá las siguientes opciones:

Giros **Información-Gral.** **Productos** **Precios**

En cada una de estas opciones, los catálogos de *GIROS*, *INFORMACION-GRAL.*, *PRODUCTOS* y *PRECIOS*, se consultará un registro dependiendo de la opción que se elija.

Para salir de cada una de las opciones y no realizar la operación se debe hacer con la tecla de **[ESC]**.

Al elegir la opción de ***SALIR*** del menú principal, el programa desplegará otro submenú dentro de esta opción, el cual contendrá las siguientes opciones:

NO **SI**

Para la opción de ***NO***, el sistema regresará a la opción de salir y se mantendrá dentro de éste.

Para la opción de ***SI***, el sistema saldra de la aplicación y regresará al Sistema Operativo o a la aplicación de Paradox, dependiendo de la ruta de salida que le demos.

MODULO DE MANTENIMIENTO.
CODIGO GENERADO POR PARADOX.

```
; Mante

if (sysmode() <> "MenúPr") then
  Message "La aplicación sólo puede arrancar desde el modo principal de Paradox"
  Sleep 3000
  return
endif

Echo Off
Clear
Reset
Cursor Off



; preguntar la contraseña de la aplicación; esta contraseña determina
;  el acceso a las tablas en la aplicación permitido para
;  usuario activo de la aplicación.

@ 0, 0
Style Attribute SysColor(0)
?? fill(" ",160)
@ 1, 0
?? "Introduzca la ctraseña. de la aplicación; [Esc] cancelar; [Intro] sin contraseña"
@ 0, 0
?? "Contraseña: "
Cursor Normal
zzzcolor = int(SysColor(0) / 16)
Style Attribute ((zzzcolor * 16) + zzzcolor)
Accept "a50" To pword
Style
EscEnter = not retval
Cursor Off

if (EscEnter) then
  Message "Cancelando la aplicación"
  Sleep 2000
  Clear
  return
endif
```

```
if (pword <> "") then
  Password pword
endif


; fijar el procedimiento de error para la aplicación

ReadLib "Manteutl" ApplicErrorProc
ErrorProc = "ApplicErrorProc"
ApplicErrorRetVal = FALSE


; Arrancar la aplicación

ReadLib "Mante1" Mante1Menu
Mante1Menu()
Release Procs Mante1Menu

Clearall
if (pword <> "") then
  UnPassword pword
endif


 ; Mante1



AppLib = "Mante1"
if (not isfile(AppLib + ".lib")) then
  Createlib AppLib
endif


proc Mante1Menu()
private x, escape, zzzmexit, zzzzexit, pword

  zzzzexit = FALSE
  x = "Altas"
  while (TRUE)
    Clear

    ShowMenu
       "Altas": "Alta de registros en los Catalogos",
       "Bajas": "Bajas de Información de registros en Catalogos",
```

```
    "Modificaciones": "Modificaciones a registros de Catalogos",
    "Consultas": "Consultas a los Catalogos",
    "Salir": "Salir de la aplicación"
  Default x
   To x

 switch
  case x = "Altas":
   ReadLib "Mante1" Mante2Menu
   escape = Mante2Menu()
   escape = not escape
   Release Procs Mante2Menu

  case x = "Bajas":
   ReadLib "Mante2" Mante4Menu
   escape = Mante4Menu()
   escape = not escape
   Release Procs Mante4Menu

  case x = "Modificaciones":
   ReadLib "Mante2" Mante5Menu
   escape = Mante5Menu()
   escape = not escape
   Release Procs Mante5Menu

  case x = "Consultas":
   ReadLib "Mante1" Mante3Menu
   escape = Mante3Menu()
   escape = not escape
   Release Procs Mante3Menu

  case x = "Salir":
 ShowMenu
    "No": "No salir de la aplicación",
    "Sí": "Salir de la aplicación"
  To zzzmexit

 zzzzexit = (zzzmexit = "Sí")
 escape = (zzzmexit = "Esc")

  case x = "Esc":
   escape = FALSE
 endswitch

 Reset
```

```
; reincializar el valor de ErrorProc
   ErrorProc = "ApplicErrorProc"
   ApplicErrorRetVal = FALSE

   if (zzzzexit) then
    return TRUE
   endif

   if (not escape) then
    x = "Altas"
   endif
  endwhile
endproc

Writelib AppLib Mante1Menu
Release Procs Mante1Menu


 ; Mante2


AppLib = "Mante1"
if (not isfile(AppLib + ".lib")) then
  Createlib AppLib
endif


proc Mante2S1()
private opResult

  Readlib "Manteutl" EntryTable, KECheck, ToggleForm,
      EdFldView, HelpKey, EntryCancel, EntryDoIt,
      RenamePre, RenameSet, SaveList, CreateList,
      PrintList

  opResult = EntryTable("Catgirm", "", "1", FALSE)

  Release Procs EntryTable, KECheck, ToggleForm,
      EdFldView, HelpKey, EntryCancel, EntryDoIt,
      RenamePre, RenameSet, SaveList, CreateList,
      PrintList

  return opResult
endproc
```

```
Writelib AppLib Mante2S1
Release Procs Mante2S1

proc Mante2S2()
private opResult

 Readlib "Manteutl" EntryTable, KECheck, ToggleForm,
     EdFldView, HelpKey, EntryCancel, EntryDoIt,
     RenamePre, RenameSet, SaveList, CreateList,
     PrintList

 opResult = EntryTable("Inform", "", "F", FALSE)

 Release Procs EntryTable, KECheck, ToggleForm,
     EdFldView, HelpKey, EntryCancel, EntryDoIt,
     RenamePre, RenameSet, SaveList, CreateList,
     PrintList

 return opResult
endproc

Writelib AppLib Mante2S2
Release Procs Mante2S2

proc Mante2S3()
private opResult

 Readlib "Manteutl" EntryTable, KECheck, ToggleForm,
     EdFldView, HelpKey, EntryCancel, EntryDoIt,
     RenamePre, RenameSet, SaveList, CreateList,
     PrintList

 opResult = EntryTable("Cprodum", "", "F", FALSE)

 Release Procs EntryTable, KECheck, ToggleForm,
     EdFldView, HelpKey, EntryCancel, EntryDoIt,
     RenamePre, RenameSet, SaveList, CreateList,
     PrintList

 return opResult
endproc

Writelib AppLib Mante2S3
Release Procs Mante2S3
```

```
proc Mante2S4()
private opResult

 Readlib "Manteutl" EntryTable, KECheck, ToggleForm,
     EdFldView, HelpKey, EntryCancel, EntryDoIt,
     RenamePre, RenameSet, SaveList, CreateList,
     PrintList

 opResult = EntryTable("Precim", "", "F", FALSE)

 Release Procs EntryTable, KECheck, ToggleForm,
     EdFldView, HelpKey, EntryCancel, EntryDoIt,
     RenamePre, RenameSet, SaveList, CreateList,
     PrintList

 return opResult
endproc

Writelib AppLib Mante2S4
Release Procs Mante2S4

proc Mante2Menu()
private x, escape

 x = "Giros"
 while (TRUE)
  Clear

  ShowMenu
     "Giros": "Alta de registros del Catalogo de Giros",
     "Información_Gral.": "Alta de Información General de Prestadores Bienes-Servicios",
     "Productos": "Alta de registros del catálogo de productos",
     "Precios": "Alta de registros del catálogo de precios"
    Default x
    To x

  switch
   case x = "Giros":
    ReadLib "Mante1" Mante2S1
    escape = Mante2S1()
    escape = not escape
    Release Procs Mante2S1

   case x = "Información_Gral.":
```

```
      ReadLib "Mante1" Mante2S2
      escape = Mante2S2()
      escape = not escape
      Release Procs Mante2S2

     case x = "Productos":
      ReadLib "Mante1" Mante2S3
      escape = Mante2S3()
      escape = not escape
      Release Procs Mante2S3

     case x = "Precios":
      ReadLib "Mante1" Mante2S4
      escape = Mante2S4()
      escape = not escape
      Release Procs Mante2S4

     case x = "Esc":
      return FALSE
    endswitch

; reincializar el valor de ErrorProc
    ErrorProc = "ApplicErrorProc"
    ApplicErrorRetVal = FALSE

    if (not escape) then
     return TRUE
    endif
   endwhile
  endproc

  Writelib AppLib Mante2Menu
  Release Procs Mante2Menu


  ; Mante3



  AppLib = "Mante1"
  if (not isfile(AppLib + ".lib")) then
   Createlib AppLib
  endif
```

```
proc Mante3S1()
private opResult, tbl

 Play "Manteq1"    ; poner la consulta en el área de trabajo
 if (ApplicErrorRetVal) then
  ClearAll
  return FALSE
 endif

 Readlib "Manteutl" QueryDoIt
 rt = QueryDoIt()
 Release Procs QueryDoIt

 if (not rt) then
  return FALSE
 endif

 Readlib "Manteutl"  ViewTable, ToggleForm, VwFldView,
     HelpKey

 opResult = ViewTable("Solucion", "Mantes1", "F", FALSE)

 Release Procs ViewTable, ToggleForm, VwFldView,
     HelpKey

 return opResult
endproc

Writelib AppLib Mante3S1
Release Procs Mante3S1

proc Mante3S2()
private opResult, tbl

 Play "Manteq2"    ; poner la consulta en el área de trabajo
 if (ApplicErrorRetVal) then
  ClearAll
  return FALSE
 endif

 Readlib "Manteutl" QueryDoIt
 rt = QueryDoIt()
 Release Procs QueryDoIt

 if (not rt) then
```

```
  return FALSE
 endif

 Readlib "Manteutl"  ViewTable, ToggleForm, VwFldView,
     HelpKey

 opResult = ViewTable("Solucion", "Mantes2", "F", FALSE)

 Release Procs ViewTable, ToggleForm, VwFldView,
     HelpKey

 return opResult
endproc

Writelib AppLib Mante3S2
Release Procs Mante3S2

proc Mante3S3()
private opResult

 Readlib "Manteutl"  ViewTable, ToggleForm, VwFldView,
     HelpKey

 opResult = ViewTable("Cprodum", "Cprodum", "1", FALSE)

 Release Procs ViewTable, ToggleForm, VwFldView,
     HelpKey

 return opResult
endproc

Writelib AppLib Mante3S3
Release Procs Mante3S3

proc Mante3S4()
private opResult

 Readlib "Manteutl"  ViewTable, ToggleForm, VwFldView,
     HelpKey

 opResult = ViewTable("Precim", "Precim", "1", FALSE)

 Release Procs ViewTable, ToggleForm, VwFldView,
     HelpKey
```

```
 return opResult
endproc

Writelib AppLib Mante3S4
Release Procs Mante3S4

proc Mante3Menu()
private x, escape

 x = "Giros"
 while (TRUE)
  Clear

  ShowMenu
    "Giros": "Consultas al catálogo de giros",
    "Información_Gral.": "Consultas al catálogo de información gral. de prestadores",
    "Productos": "Consultas al catálogo de productos",
    "Precios": "Consultas al catálogo de precios"
   Default x
   To x

  switch
   case x = "Giros":
    ReadLib "Mante1" Mante3S1
    escape = Mante3S1()
    escape = not escape
    Release Procs Mante3S1

   case x = "Información_Gral.":
    ReadLib "Mante1" Mante3S2
    escape = Mante3S2()
    escape = not escape
    Release Procs Mante3S2

   case x = "Productos":
    ReadLib "Mante1" Mante3S3
    escape = Mante3S3()
    escape = not escape
    Release Procs Mante3S3

   case x = "Precios":
    ReadLib "Mante1" Mante3S4
    escape = Mante3S4()
    escape = not escape
    Release Procs Mante3S4
```

```
    case x = "Esc":
      return FALSE
    endswitch

; reincializar el valor de ErrorProc
    ErrorProc = "ApplicErrorProc"
    ApplicErrorRetVal = FALSE

    if (not escape) then
      return TRUE
    endif
  endwhile
endproc

Writelib AppLib Mante3Menu
Release Procs Mante3Menu


 ; Mante4



AppLib = "Mante2"
if (not isfile(AppLib + ".lib")) then
  Createlib AppLib
endif


proc Mante4S1()
private opResult

  if (isempty("Catgirm")) then
    Message "No hay registros que editar"
    Sleep 3000
    return FALSE
  endif

  if (ApplicErrorRetVal) then
    return FALSE
  endif

  Readlib "Manteutl" EditTable, ToggleForm, EdFldView,
      HelpKey, EditCancel, SEditDoIt, SEditDelNoIns
```

```
 opResult = EditTable("Catgirm", "Catgirm", "", "3", FALSE,
       "SEditDoIt", "SEditDelNoIns",
       "[Supr] Borrar registro",
       TRUE, FALSE, FALSE)

 Release Procs EditTable, ToggleForm, EdFldView,
     HelpKey, EditCancel, SEditDoIt, SEditDelNoIns


 return opResult
endproc

Writelib AppLib Mante4S1
Release Procs Mante4S1

proc Mante4S2()
private opResult

 if (isempty("Inform")) then
  Message "No hay registros que editar"
  Sleep 3000
  return FALSE
 endif

 if (ApplicErrorRetVal) then
  return FALSE
 endif

 Readlib "Manteutl" EditTable, ToggleForm, EdFldView,
     HelpKey, EditCancel, SEditDoIt, SEditDelNoIns

 opResult = EditTable("Inform", "Inform", "", "2", FALSE,
       "SEditDoIt", "SEditDelNoIns",
       "[Supr] Borrar registro",
       TRUE, FALSE, FALSE)

 Release Procs EditTable, ToggleForm, EdFldView,
     HelpKey, EditCancel, SEditDoIt, SEditDelNoIns


 return opResult
endproc

Writelib AppLib Mante4S2
Release Procs Mante4S2
```

```
proc Mante4S3()
private opResult

 if (isempty("Cprodum")) then
  Message "No hay registros que editar"
  Sleep 3000
  return FALSE
 endif

 if (ApplicErrorRetVal) then
  return FALSE
 endif

 Readlib "Manteutl" EditTable, ToggleForm, EdFldView,
     HelpKey, EditCancel, SEditDoIt, SEditDelNoIns

 opResult = EditTable("Cprodum", "Cprodum", "", "3", FALSE,
       "SEditDoIt", "SEditDelNoIns",
       "[Supr] Borrar registro",
       TRUE, FALSE, FALSE)

 Release Procs EditTable, ToggleForm, EdFldView,
     HelpKey, EditCancel, SEditDoIt, SEditDelNoIns


 return opResult
endproc

Writelib AppLib Mante4S3
Release Procs Mante4S3

proc Mante4S4()
private opResult

 if (isempty("Precim")) then
  Message "No hay registros que editar"
  Sleep 3000
  return FALSE
 endif

 if (ApplicErrorRetVal) then
  return FALSE
 endif
```

```
 Readlib "Manteutl" EditTable, ToggleForm, EdFldView,
     HelpKey, EditCancel, SEditDoIt, SEditDelNoIns

 opResult = EditTable("Precim", "Precim", "", "3", FALSE,
       "SEditDoIt", "SEditDelNoIns",
       "[Supr] Borrar registro",
       TRUE, FALSE, FALSE)

 Release Procs EditTable, ToggleForm, EdFldView,
     HelpKey, EditCancel, SEditDoIt, SEditDelNoIns


 return opResult
endproc

Writelib AppLib Mante4S4
Release Procs Mante4S4

proc Mante4Menu()
private x, escape

 x = "Giros"
 while (TRUE)
  Clear

  ShowMenu
    "Giros": "Bajas a registros del catalogo de giros",
    "Información_Gral.": "Bajas a registros de información general",
    "Productos": "Bajas a registros de catálogo de productos",
    "Precios": "Bajas a registros de catálogo de precios"
   Default x
   To x

  switch
   case x = "Giros":
    ReadLib "Mante2" Mante4S1
    escape = Mante4S1()
    escape = not escape
    Release Procs Mante4S1

   case x = "Información_Gral.":
    ReadLib "Mante2" Mante4S2
    escape = Mante4S2()
    escape = not escape
    Release Procs Mante4S2
```

```
    case x = "Productos":
      ReadLib "Mante2" Mante4S3
      escape = Mante4S3()
      escape = not escape
      Release Procs Mante4S3

    case x = "Precios":
      ReadLib "Mante2" Mante4S4
      escape = Mante4S4()
      escape = not escape
      Release Procs Mante4S4

    case x = "Esc":
      return FALSE
   endswitch

; reincializar el valor de ErrorProc
   ErrorProc = "ApplicErrorProc"
   ApplicErrorRetVal = FALSE

   if (not escape) then
    return TRUE
   endif
  endwhile
endproc

Writelib AppLib Mante4Menu
Release Procs Mante4Menu


; Mante5


AppLib = "Mante2"
if (not isfile(AppLib + ".lib")) then
  Createlib AppLib
endif


proc Mante5S1()
private opResult

  if (isempty("Catgirm")) then
```

GLOSARIO

GLOSARIO

ANSI SQL.- Standard Scalable and Portable.

ANSI++.- Librería Estandar de C.

ANSI-SPARC.- (American National Standards Institute/Systems Planning and Requirements Committee). Se le conoce principalmente por su propuesta de crear una arquitectura de bases de datos, según la cual éstas se definen en tres niveles: esquema conceptual, esquema externo y esquema interno.

ANSI.- (American National Standards Institute) Intituto Nacional Americano de Normalización. Organización que determina las normas relativas al hardware, en puntos tales como: protocolos de nivel de enlace, posiciones y significado de las pastillas en los chips, registro en cinta y disco, y algunas normas para el software.

ARGC.- Argumento de C y C++.

ARGV.- Argumento por valor de C y C++.

BENCHMARK.- Prueba de características; evaluación de rendimiento; punto de referencia en comparar la medida del rendimiento con la de otros (hardware y software) que hayan sido sometidos a la misma prueba de características.

CASE.- Computer Aided Software Engineering. Ingeniería de Software Asistida por Computadora. Software que se utiliza en cualquiera o en todas las fases del desarrollo de un sistema de información, incluyendo análisis, diseño y programación. Por ejemplo, los diccionarios de datos y herramientas de diagramación ayudan en las fases de análisis y diseño, mientras que los generadores de aplicaciones aceleran la fase de programación.

CHARACTER (CHAR).- Elemento de un conjunto dado de caracteres, subdivisión de una palabra de máquina, que comprende, generalmente, seis, siete u ocho bits.

CADASYL.- (Conference on Data Systems Languages). Conferencia sobre lenguajes de sistemas de datos. Se fundó en una reunión convocada en el pentagonó en 1959. Su objetivo era, primeramente, el desarrollo de un lenguaje normalizado de programación para el proceso de datos.

DAG.- Gráfica Aciclica Dirigida.

DBMS.- (Data Base Management System). Sistema Manejador de Base de Datos.

DBSCHEMA.- Esquema de la base de datos.

ENTIDAD.- Es un objeto que existe y es distinguible de otros objetos.

GIF (Formato).- Graphics Interchange Format. Estandard de compuserve para la definición de imágenes a color.

GLOBAL COLOR MAP.- Mapa Global de Color. Recomendada para imágenes donde la exactitud del color es deseada.

HFS.- Estructura de árbol jerárquico de directorios y archivos.

ICONO.- Una diminuta representación pictórica de un objeto, tal como una aplicación, archivo o unidad de disco, que se utiliza en interfaces gráficas de usuario (GUI).

IDA.- Intelligent Drive Array.

IMAGE DESCRIPTOR.- Descriptor de Imágen. Define la colocación actual y la extensión de la siguiente imágen dentro del espacio definido en el descriptor de pantalla.

INFORMIX-SQL.- Manejador de base de datos Secuential Query Language.

INTERFACE.- Una conexión e interacción entre hardware, software y usuario.

MAIN( ).- Función de C y C++.

OBJETO.- Unidad única en la que se encapsulan código y datos.

OLE.- Objetos Ligados y Empotrados.

ORDEN.- De un árbol, es el número de descendientes que puede tener un nodo.

OUTLETS.- Técnica de corolario para lograr la comunicación.

PATH.- Ruta que tiene un directorio o archivo dentro de la memoria de la computadora.

PIXEL.- Término derivado de picture element: elemento de imagen. Uno de los elementos en orden (o matriz) grande que contiene información gráfica. Guarda datos que representan el brillo y posiblemente, el color de una pequeña región de la imagen.

RANDOM READ.- Lectura Aleatoria.

RASTER DATA.- Rastreo de Datos

RELACION.- Es una asociación entre varias entidades.

SCREEN DESCRIPTOR.- Descriptor de pantallas. Describe el total de parámetros para las imégenes GIF.

SQL.- Standar Query Language: Lenguaje Normalizado de Consulta.

TIFF (Formato).- Tag image File Format.

TObject.- Clase abstracta de un lenguaje de programación