29
17

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DE UN SISTEMA GENERADOR DE SISTEMAS UTILIZANDO LENGUAJES DE CUARTA GENERACION

T E S I S

QUE PARA OBTENER EL TITULO DE

Ingeniero en Computación

P R E S E N T A

SALVADOR GIL SOLORIO

Director de Tesis: ING. ROBERTO MALDONADO MAZA

MEXICO, D. F. 1989

TESIS CON
FALLA DE ORIGEN

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.

INDICE

INTRODUCCION

Los grandes avances que se han dado en la computación, tanto en software como en hardware, motivan al diseño de nuevas tecnologías que puedan contribuir con tal fin.

En el área de software se dieron grandes pasos, los cuales van desde los lenguajes de máquina, hasta los lenguajes de cuarta generación y los de inteligencia artificial. Como complemento a esto, se han desarrollado diversas metodologías que ayudan a la rápida, fácil y eficiente implementación de sistemas de información.

Una de las tendencias más notorias en la industria de cómputo es el incremento relativo del costo del software ,en relación con el costo de los equipos, ya que en los principios de la computación el hardware representaba más del 70% del costo total de una solución y en la actualidad esta proporción es inversa. Esto es debido al alto contenido de mano de obra dentro de la programación del software, reducción de costos y a los avances en la microelectrónica que, son por consecuencia los principales problemas que originan el encarecimiento relativo del software.

Lo expuesto en el párrafo anterior y el lento crecimiento en la productividad del software, dan origen a lo que se conoce como "La crisis de la programación"(#1). Esto ha motivado que algunos países industrializados dediquen grandes recursos de investigación y desarrollo para generar técnicas y herramientas que logren hacer más barato y rápido el desarrollo de programas; por ejemplo en Japón se tienen fábricas de software que han alcanzado un alto nivel de productividad en los últimos años.

Existe otra tendencia originada por las casas de software, que consiste en desarrollar paquetes comerciales que se puedan aplicar a diversas empresas, con un poco de adaptaciones por parte de éstas. En general, esto resulta benéfico para las empresas, pues adoptan tecnologías modernas en la automatización de

---

(#1) ComputerWorld Nov 9, 1987. Ricardo Zermeño

sus oficinas ó procesos de manufactura. En algunos casos, dependiendo de la calidad del paquete comercial, no se llegan a cumplir con los requerimientos necesarios para un buen funcionamiento del proceso de automatización de las empresas, lo cual va en perjuicio directo de los intereses de las mismas. Por lo cual es importante realizar una buena auditoria de estos paquetes, para elegir la alternativa que resuelva mejor el problema.

Lo más importante de la crisis de la programación es que la programación representa en la actualidad el principal cuello de botella del desarrollo de sistemas. Los equipos son cada dia más poderosos y versátiles, sin embargo, no existe el personal capacitado ni los sistemas adecuados.

Han sido varios los intentos por facilitar al máximo el desarrollo de sistemas a personas con pocos conocimientos de computación, o en su defecto con desconocimiento del lenguaje de programación adecuado para el desarrollo del sistema.

Este trabajo de tesis es un intento más por facilitar el desarrollo de sistemas. En los dos primeros capitulos, se pretende dar una breve historia de los avances que ha tenido el software hasta la fecha, tecnologías actuales y algunos paquetes comerciales. En los dos capitulos siguientes , se trata de conceptualizar y explicar las caracteristicas y alcances del Sistema Generador de Sistemas, exponiéndose algunos conceptos técnicos que ayudaron en el desarrollo. En el quinto capitulo, se presentan las aplicaciones y perspectivas que le veo al Sistema Generador de Sistemas. Y por último, presento las conclusiones de este trabajo de tesis.

I.— ANTECEDENTES DE LOS LENGUAJES DE PROGRAMACION.

1).- Evolución de los instrumentos de cálculo.

Desde épocas remotas el hombre se ha enfrentado a diferentes tipos de problemas, en los cuales se requeria una serie de cálculos para llegar a la solución. En la mayoria de los casos, dicha solución consistia en procedimientos de cálculo relativamente complejos, que para algunas personas resultaba dificil, fastidioso, o bien consumía mucho tiempo al resolverlo. Por esta razón, el hombre siempre se ha preocupado por inventar técnicas que faciliten la solución de este tipo de problemas.

Las primeras técnicas utilizadas fuerón las de conteo, dentro de estas tenemos:

a) Conteo con los dedos u otros objetos(piedras, varas, etc.).
b) El ábaco(China 2,600 A.C.).
c) Tabla de logaritmos(Escocia, Jhon Napier 1614).
d) Regla de cálculo(Inglaterra, William Oughtred 1632).
e) Máquina de Pascal(Francia, Blaise Pascal 1642).
f) Máquina de Jacquard(Josseph Marie J. 1801).
g) Máquina de diferencias(Charles Babbage 1812).
h) Calculadoras de tarjeta perforada(EUA, Herman Hollerith).

2).- Primeras computadoras electromecánicas y electrónicas.

En 1937 el profesor Howard Aiken, de la Universidad de Harvard, utilizando los principios de Babbage y de Hollerith, construye un mecanismo automático de cálculo. Con esto, surge una nueva tecnologia que dará como fruto el origen de la computación. Conforme avanza el tiempo, se presentan nuevos avances que originan la evolución de las computadoras, la cual se ilustra a través de las sigientes generaciones de computadoras digitales:

a) Primera generación.

Utilizó tubos al vacio y componentes electrónicos básicos, que como caracteristicas tenian un costo elevado,

gran volumen, consumo de fuerza considerable, calentamiento, retardo de lógica y una enorme cantidad de fallas en las redes(en comparación con las actuales).

b) Segunda generación

Se utilizaron elementos como transistores, diodos y componentes en base a semiconductores, los cuales eran montados en tarjetas de circuitos impresos.

c) Tercera generación.

Utiliza circuitos integrados(Chips), los cuales son elementos pequeños de aproximadamente 1 a 5 cm^3, los cuales pueden contener más de 50,000 elementos y tener miles de circuitos, por lo cual también se les da el nombre de microcircuitos.

Algunos adelantos surgidos con esta generación son:

- Auto control y ejecución continua de procesos.
- Auto diagnóstico e informe de errores.
- Multiprogramación.
- Multiproceso.
- Gran velocidad de entrada-proceso-salida.
- Evolución de los lenguajes de programación.

d) Cuarta generación.

Persiguen como meta principal, obtener una mayor velocidad de procesamiento de información, así como compactar sus diseños. Para esto utilizan tecnologías avanzadas de diseño y microcircuitos de muy larga escala de integración(VLSI).

e) Quinta generación.

La tendencia es diseñar computadoras inteligentes que integren funciones de software al hardware, que permitan al usuario aprovechar las ventajas de la computación con un esfuerzo mínimo de su parte. Con ellas, el usuario no requerirá una capacitación concienzuda, pues la computadora le indicará las pasos a seguir en el desarrollo de sus tareas. Serán completamente amigables, muy flexibles y sus configuraciones serán adecuadas a las necesidades del usuario.

3).- Aplicaciones de las computadoras.

Existen 5 áreas de aplicación de la computación:

a) Cientifica.
b) Procesamiento de datos.
c) Procesamiento de texto.
d) Inteligencia artificial.
e) Programación de sistemas.

a) Aplicaciones cientificas.

Se caracterizan principalmente por la manipulación de números y arreglos de estos, utilizan principios matemáticos y estadisticos como base de sus algoritmos. Estos algoritmos tratan problemas tales como: exámenes estadisticos, programación lineal, análisis de regresión y aproximaciones numéricas para la solución de ecuaciones diferenciales e integrales, asi como una diversa gama de problemas ingenieriles.

Los problemas cientificos suelen tener una considerable complejidad matemática, razón por la cual, los programadores deberán conocer los principios matemáticos necesarios para abordar correctamente los problemas o hacer refinamientos, pues solo asi se tendrán soluciones adecuadas.

Los problemas cientificos requieren en la mayoria de los casos, de mayor trabajo por parte del procesador central de una computadora, que de los dispositivos de entrada/salida, esto es debido a que la solución es en base a cálculos, con un minimo de informes durante el proceso.

b) Aplicaciones de procesamiento de datos.

Los problemas de procesamiento de datos tienen como interes predominante, la creación, mantenimiento, extracción y compendio de datos en los archivos. Se incluyen dentro de estas aplicaciones, algunas funciones de oficina tales como:

. Nómina
. Contabilidad
. Facturación
. Inventarios

. Producción
. Pedidos y ventas, etc.

Un programa tipico de proceso de datos consume la mayor parte de su tiempo en operaciones de entrada/salida, ya sea localizando, actualizando y mostrando ó imprimiendo información, según la frecuencia requerida(diaria, semanal, mensual, anual, etc.).

Existen dos maneras de realizar el procesamiento de datos. La primera consiste en la ejecución por lotes(batch), la cual, no tiene comunicación directa con el usuario, o en su defecto es mínima. La segunda es interactivamente, en la cual los procesos tienen una comunicación constante con los usuarios, ya sea para solicitar, o bien para proporcionar información.

En las aplicaciones de procesamiento de datos, debe considerarse la integridad de los datos, queriéndose decir con esto que los programas deberán proteger la información de los archivos, tratando de evitar posibles contaminaciones en la información y en general darle confiabilidad y precisión a la aplicación.

c) Aplicaciones de procesamiento de texto.

Se caracterizan por manipular el texto en lenguaje natural, olvidándose del proceso de números. Dentro de esta área, existen diversas modalidades, algunas son más versátiles que otras, ya sea por un mejor formateo, o bien por permitir funciones de mecanografía útiles en la captura de manuscritos.

Algunos avances recientes en esta área, permiten el procesamiento de texto en forma multilingüe, permitiendo para esto el manejo alterno de teclados en varios idiomas tales como: español, ingles, árabe, japones, etc.

d) Aplicaciones de inteligencia artificial.

La inteligencia artificial pretende principalmente emular comportamientos inteligentes, asemejándose un poco al razonamiento humano. Dentro de ella se incluyen:

. Comprensión de lenguaje natural

. Visión por computadora
. Robótica
. Sistemas expertos
. Juegos(ajedrez y otros)

Anteriormente la inteligencia artificial era tratada en laboratorios de investigación, realizándose experimentos para modelar distintos comportamientos inteligentes. Actualmente, varios de estos proyectos se encuentran en práctica, teniéndose resultados positivos que se reflejan tanto en áreas de producción de automóviles, como en monitoreo de instrumentos complejos.

e) Aplicaciones de programación de sistemas.

La finalidad de estas aplicaciones, es desarrollar programas que hagan más amigable la interface entre la computadora(en el aspecto de hardware) y los usuarios(programadores y operadores). Estos programas incluyen:

. Sistemas operativos
. Compiladores
. Ensambladores
. Intérpretes
. Rutinas de entrada/salida
. Planificadores y evaluadores de la utilización de recursos de la computadora, etc.

Dentro de la programación de sistemas, se manejan sucesos impredecibles, tales como errores de entrada/salida, se coordinan las actividades de varios programas independientes(usuarios en línea), ya que por ejecutarse asíncronamente, puede llegar a ocurrir un conflicto en la utilización de los recursos de la computadora(unidades de disco y cinta, impresoras, etc.).

4).- Panorama general de los lenguajes de programación.

Los lenguajes de programación surgen por la necesidad de facilitar la comunicación del hombre con la computadora. El decir a la computadora como se deberá comunicar con el usuario, es conocido como **programación.**

Dentro de la programación de computadoras, existe una diversa gama de lenguajes enfocados a cada área de aplicación. Las generaciones de computadoras dieron pauta a la evolución de los lenguajes de programación, de tal manera que para la primera generación, la técnica de programación era el lenguaje máquina, mientras que la segunda generación marca el surgimiento de los lenguajes ensambladores y en la tercera generación, se emplean los lenguajes de alto nivel y para la cuarta y quinta generación los lenguajes de cuarta generación e inteligencia artificial. Para las dos últimas generaciones los lenguajes tienden a ser más sencillos de aplicar y mucho más poderosos. Cabe hacer notar que, el avance tecnológico en hardware (generaciones), no inhibe la utilización de técnicas anteriores en la generación en curso.

Una de las clasificaciones más comunmente utilizada en computación, es de acuerdo a su nivel. Dentro de esta, podemos enmarcar cuatro niveles de lenguajes de programación, como puede verse a continuación:

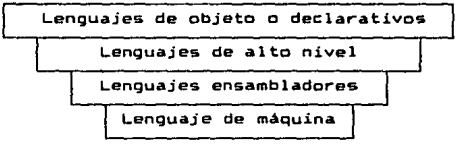


Figura 1.1 : Niveles en los lenguajes de programación.

a) Lenguaje máquina:

Es el lenguaje de más bajo nivel (que entiende la computadora) y es el más cercano a las caracteristicas de cada computadora(CPU). Cada instrucción de este lenguaje consistirá en una secuencia de "1"'s y "0"'s, la cual le indicará a la computadora la acción que deberá ejecutar. Esta secuencia de unos y ceros, anteriormente era proporcionada a la computadora mediante un arreglo de "n" interruptores los cuales al estar apagados

indicarian un "0" y al estar prendidos un "1", siendo "n" la longitud de la palabra o instrucción de cada máquina . Conforme se evolucionó en la forma de comunicarse con la computadora, esta secuencia de unos y ceros, podía proporcionarse mediante "teclados", en los cuales al presionarse alguna tecla, proporcionarían la secuencia de "0"'s y "1"'s asociada a ella, por lo cual se simplificó un poco la programación en lenguaje máquina, pues ahora se podrían registrar teclas (letras, números) para cada secuencia de unos y ceros, en lugar de indicar la secuencia manualmente.

b) Lenguaje ensamblador.

El lenguaje ensamblador simplemente es una representación simbólica del lenguaje máquina asociado a la computadora, pero resulta más sencillo programar en lenguaje ensamblador que en lenguaje máquina. A cada instrucción se le asignan de una a tres letras nemónicas, las cuales el programador las puede asociar fácilmente con la instrucción que ejecutará la computadora.

Cada instrucción puede tener cero, uno o dos operandos, los cuales serán utilizados ó afectados dependiendo de la naturaleza de la instrucción.

La programación en lenguaje ensamblador, no es entendible directamente para las computadoras, por lo tanto, deberá existir una interface con el lenguaje máquina para que el programa sea ejecutado. Esta interface es conocida comunmente como programa "ensamblador" y su finalidad será convertir de lenguaje ensamblador a lenguaje máquina como se muestra en la figura 1.2.

El ensamblador presenta ventajas con relación a otros porque genera poco código, son de ejecución muy rápida y permiten hacer uso de todas las posibilidades que el hardware ofrece, aunque resulta difícil de usar pues cada instrucción está representada por un número binario o hexadecimal.

Los lenguajes de máquina y los lenguajes ensambladores dependen directamente de la computadora que se esté utilizando, es decir, cada computadora (por ejemplo VAX de Digital) tiene su propio lenguaje máquina y lenguaje ensamblador, los cuales serán diferentes de los de otras computadoras.

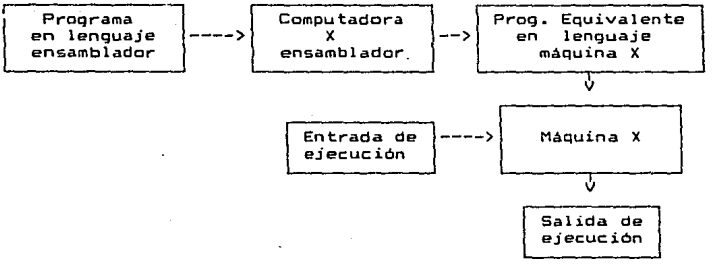


Figura 1.2 : Trayectoria de un programa en lenguaje ensamblador.

c) Lenguajes de alto nivel.

Tienen como antecedente los lenguajes de nivel medio, los cuales incorporan palabras y frases que se acercan mucho al lenguaje natural utilizado comunmente, aunque no se incorporan todas las instrucciones como en el lenguaje ensamblador. Como ejemplo de estos lenguajes de nivel medio tenemos al lenguaje "C".

Son los más versátiles, e incorporan palabras y frases para la totalidad de sus instrucciones, por lo cual son los más utilizados en la programación de aplicaciones. Permiten una programación más sencilla, ya que los algoritmos de solución de problemas son expresados en un nivel y estilo de escritura facilmente legible y comprensible por otros programadores.

Los lenguajes de alto nivel tienen normalmente la característica de "transportabilidad", lo cual es de gran importancia, puesto que, aplicaciones programadas en un lenguaje X de la máquina Y, podrá ser transferido casi por completo, sin ajustes ó revisiones sustanciales al mismo lenguaje X de la máquina Z. Como

puede verse, los lenguajes de alto nivel son "independientes" de la computadora que se utilice(máquina).

Se han desarrollado diversos lenguajes de alto nivel, cubriendo en su totalidad las áreas de aplicación de la computación, como vemos a continuación:

. Aplicaciones científicas
  Algol, Pascal, APL, Fortran, Basic

. Procesamiento de datos
  Cobol, RPG II, Basic

. Procesamiento de textos
  SNOBOL

. Programación de sistemas
  C, Ada

. Inteligencia artificial
  Lisp, PROLOG

Así como PL/1 el cual es de propósito general, es decir fue diseñado pensando en cubrir todas las áreas mencionadas(excepto inteligencia artificial).

Para los lenguajes de alto nivel, existen dos métodos de ejecución. El primero consiste en el "compilador", que es una interface que permite traducir de lenguaje de alto nivel a lenguaje máquina, conociéndose estos como programa fuente y programa objeto respectivamente. El segundo método consiste en un "intérprete", el cual traduce cada instrucción a lenguaje máquina y la ejecuta directamente.

La "compilación" es un proceso más eficiente que la "interpretación" para la mayoría de los tipos de computadoras. Los tiempos de ejecución comparativamente, son mucho menores para programas compilados que para los interpretados. Esto se debe principalmente a que en los intérpretes, las instrucciones dentro de un "bucle" (ciclo repetitivo de las mismas instrucciones) deben ser reinterpretados cada vez que se ejecutan por el intérprete. En cambio, con el compilador cada instrucción es interpretada y traducida a lenguaje máquina solo una vez.

Generalmente los intérpretes tienen una mayor calidad de diagnosis y depuración que la de los compiladores, puesto que los mensajes de error son indicados directamente a la instrucción que origina el error. Frecuentemente la interpretación es preferible a la compilación, por ejemplo; para aplicaciones educativas ó cuando las aplicaciones se encuentran en fase experimental o de prueba, los programadores se inclinan por los lenguajes interpretados, puesto que en cada ejecución pueden ocurrir errores que serían detectados más fácilmente y así podrían realizar los correcciones necesarias al programa para su buen funcionamiento.

La principal ventaja de eficiencia de ejecución, adjudicada a los compiladores frente a los intérpretes, puede pronto verse superada, debido a la evolución de las computadoras(quinta generación), en las cuales se pretende integrar un intérprete de lenguaje de alto nivel en el mismo equipo, como ejemplo de esto se tienen las nuevas máquinas LISP, diseñadas recientemente por Symbolics y Xerox Corporations, ademas de algunas computadoras personales que tienen integrado BASIC en una memoria de lectura.

Algunos lenguajes son principalmente interpretados, por ejemplo APL, PROLOG, LISP, BASIC y otros compilados como FORTRAN, PASCAL, ALGOL, COBOL, PL/1, SNOBOL, C, Ada y Modula-2.

En algunas ocasiones, se podrán utilizar ambos métodos, tal es el caso de BASIC, LISP, SNOBOL4.

Para ejemplificar las diferencias básicas entre los lenguajes de alto nivel, ensambladores y de máquina, veamos los siguientes 3 segmentos de programa que cumplen con la misma función:

PASCAL	Leng. Ensamblador	Leng. Máquina
Z:=W+X*Y	L 3,X M 3,Y A 3,W ST 3,Z	41 3 0C/A4 3A 3 0C/AB 1A 3 0C/A0 50 3 0C/A4

Figura 1.3 : Comparación de instrucciones de lenguajes.

Como puede verse, el lenguaje máquina es el menos comprensible para el usuario, en cambio la computadora solo puede trabajar con él. Asi mismo el lengüaje de alto nivel (PASCAL en este caso) es mucho más comprensible para el usuario y el lenguaje ensamblador en segundo orden. Es por esto que los compiladores e intérpretes son indispensables para una comunicación más facil entre el usuario y la computadora.

d) Lenguajes de objeto o declarativos.

Son muy recientes y nacieron a raíz de lo que se conoce como "inteligencia artificial". Se basan principalmente en la heuristica, es decir, en la toma de decisiones siguiendo ciertas reglas, y en algunos casos se basan en el conocimiento adquirido, lo cual se conoce como "autoaprendizaje". Estos lenguajes incorporan funciones que ejecutan procesos completos predefinidos, que en los lenguajes de procedimiento seria tedioso implementar.

Son fundamentalmente lenguajes de órdenes, denominados por sentencias que indican "lo que hay que hacer" en lugar de "como hacerlo". Estas sentencias, por lo general, son muy parecidas a palabras de algun idioma en especial (castellano, ingles, etc.), por lo que tienen una gran potencia expresiva y funcional.

Como ejemplo de estos lenguajes se tienen:

- Algunos lenguajes estadisticos como SAS y SPSS.
- Lenguajes de búsquedas en bases de datos, como NATURAL e IMS.
- PROLOG(Programación lógica).
- LISP(Procesador de listas).
- Asi como los lenguajes de cuarta generación que serán tratados en el siguiente capitulo.

Estos lenguajes fueron desarrollados con la finalidad de facilitar su aprendizaje y utilización, tratando de evitar con esto la necesidad de programadores y prácticas de programación.

5).- Evolución de los lenguajes de programación.

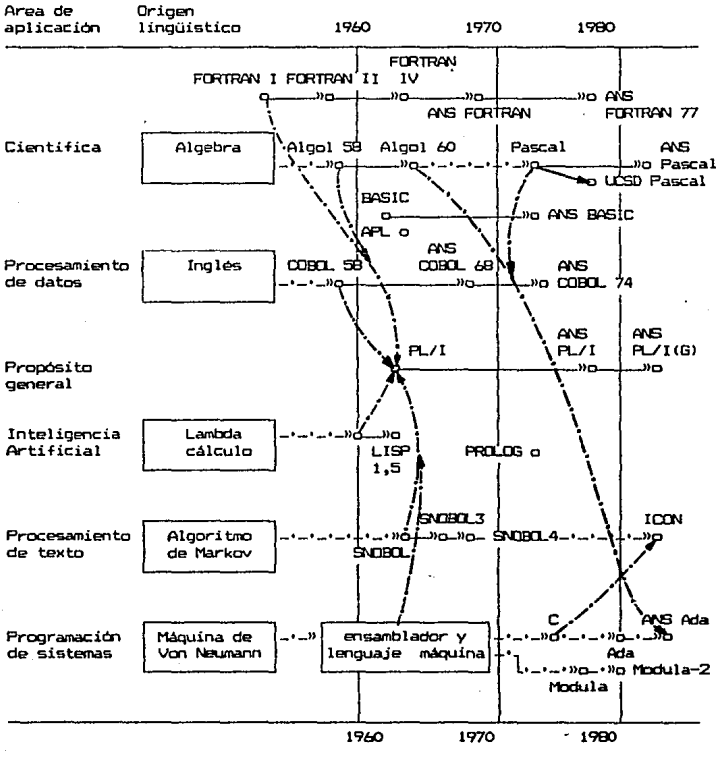
a) Generalidades.

Dentro de los lenguajes de programación, se maneja una serie de instrucciones comunes entre ellos, que han dado la pauta para estandarizar y mejorar los nuevos lenguajes de programación. Por ejemplo, en los lenguajes más recientes se prohibe (o bien no se permite) el uso de algunas instrucciones tradicionalmente usadas en los lenguajes más antiguos, esto se debe a que dichas instrucciones pueden ser muy problemáticas si no son utilizadas en forma adecuada. Este es el caso de la instrucción GOTO, la cual transfiere el control a otra instrucción del programa, al ser utilizada varias veces, llega un momento en que el programa tiene una secuencia de ejecución de instrucciones tan compleja, que es casi incomprensible hasta por el propio programador.

El principal progreso que se ha tenido en los lenguajes de programación, se lo debemos sin duda alguna, a la necesidad de estandarizar la programación. Esta necesidad surge, porque los programas eran diseñados en una forma tan personalizada, que resultaba casi imposible que otros programadores pudieran hacer adecuaciones a programas no desarrollados por ellos.

En los intentos de estandarización de la programación, surge la programación estructurada y la ingeniería del software, dejando atrás las técnicas tradicionales de programación, aunque estas actualmente sigan siendo utilizadas por algunos programadores obsoletos.

Es importante ver la evolución de los lenguajes de programación con el paso del tiempo y la influencia que ejerce las nuevas arquitecturas y aplicaciones de las computadoras en los nuevos progresos de los lenguajes de programación. En la figura 1.4 se muestran los avances más sobresalientes en los lenguajes de programación.



—·—·—» indica fuerte influencia
————» indica acendencia directa

Figura 1.4 : Evolución de los lenguajes de programación.

b) Programación estructurada.

Esta técnica de programación se basa principalmente en:

- Modularidad
- Programación de arriba-abajo (top-down)
- Cinco instrucciones básicas
  - IF THEN ELSE
  - FOR-NEXT
  - DO WHILE
  - REPEAT UNTIL
  - CASE OF

Todos los lenguajes que tengan estas facilidades, pueden ser considerados como lenguajes estructurados, podemos mencionar dentro de estos a:

- ALGOL
- PASCAL
- BUSSINES BASIC
- DBASE

En todos los lenguajes tradicionales pueden ser implementados los principios de la programación estructurada. Esto puede ser un poco complejo, debido a el arreglo de instrucciones que deberá hacerse, pero redituará un beneficio invalorable para la estandarización de los programas.

Otro punto muy importante que debe tener un programa que presume de cumplir con los estandares, es que deberá tener los comentarios necesarios (no excesivos) en el lugar adecuado, de manera que ayuden al mejor entendimiento de su funcionamineto.

c) Ingeniería del software.

Es una tecnología reciente, enfocada principalmente a la metodología que se debe utilizar para un buen desarrollo de sistemas, quedando fuera los detalles de los lenguajes de programación.

6).- Diseño de los lenguajes de programación.

El diseño de un lenguaje de programación requiere el conocimiento de muchos campos, especialmente la arquitectura de la computadora, lenguajes formales, teoría de autómatas y lingüística. Un buen diseñador de lenguajes debe tomar en cuenta los intereses del programador que utilizara el lenguaje, los del implementador de lenguaje diseñado y los intereses propios.

El diseño de lenguajes de programación se encuentra dividido en las siguientes partes:

. Sintaxis
. Semántica
. Pragmática

a) Sintaxis.

"La mayoría de los lenguajes de programación permiten una definición rigurosa o formal. Es decir puede usarse una breve expresión matemática para caracterizar la validez sintáctica de las sentencias o instrucciones. Esto facilita enormemente la tarea del implementador, cuyo compilador o interprete deberá analizar sintácticamente el texto del programa antes de que pueda ejecutarse. Los aspectos sintácticos incluyen "la descripción formal y abreviaciones, el conjunto de caracteres del lenguaje, la eliminación de ambigüedad en el lenguaje, reconocimiento, análisis lexicográfico de la sintaxis sobre el estilo de la programación y viceversa"(*2).

La sintaxis de un lenguaje de programación, "es un conjunto de reglas y criterios de escritura que permiten la formación de programas correctos en un lenguaje, considerando sólo el punto de vista de representación. La sintaxis no se refiere al significado o comportamiento en tiempo de ejecución de un programa"(*3).

(*2) Lenguajes de programación. Allan B. Tucker, 2a. Ed. 1987, pp 15.
(*3) Lenguajes de programación. Allan B. Tucker, 2a. Ed. 1987, pp 243.

b) Semántica.

Pretende dar significado a las distintas formas sintácticas de un programa y asigna una representación en el lenguaje máquina adecuada para ellas. "Algunos aspectos importantes de la semántica de un lenguaje son la vinculación, coerción, implementación de diferentes tipos y estructuras de datos, asignación dinámica de memoria, diagnóstico en tiempo de ejecución, manejo de excepciones, ámbito, procedimientos compilados por separado, entrada-salida y optimización de codigo"(*4).

Todo esto en forma agrupada seria(*5):

- Tipos, vinculación, operadores y coerción
- Asignación de memoria
- Estructuras de control
- Procedimientos y parámetros
- Entornos en tiempo de ejecución

c) Pragmática.

Se refiere a los aspectos adicionales de la implementación y estilo de programación que tiende a influir en las características de los lenguajes. Estos aspectos incluyen la "gestión de concurrencia, optimización, entorno de programación y diagnosis"(*6). Dentro del estilo de programación tenemos:

- Ingeniería del software
- Programación funcional
- Abstracción de datos
- Programación objeto
- Programación lógica

Los aspectos pragmáticos quizás son los más extensos y de mayor influencia para las futuras tendencias de los lenguajes de programación.

(*4) Lenguajes de programación. Allan B. Tucker, 2a. Ed. 1987, pp 15.
(*5) Lenguajes de programación. Allan B. Tucker, 2a. Ed. 1987, pp 371.
(*6) Lenguajes de programación. Allan B. Tucker, 2a. Ed. 1987, pp 16.

II.— CARACTERISTICAS DE LOS LENGUAJES DE CUARTA GENERACION

1).- Conceptos preliminares.

a) Tres generaciones del esfuerzo de programación.

La evolución de comunicaciones entre hombre y computadora, han tomado dos senderos distintos:

- Datos
- Programas

Los programas necesitan manejar datos y decir a la máquina como coleccionar, limpiar, almacenar, recuperar, computar, y presentar información. Los primeros 30 años de computación estuvierón caracterizados por un esfuerzo de programación fijo y tedioso. Sin embargo, los programadores tuvieron que empezar lentamente a ponerse lejos de la máquina física. Se tuvierón que mover hacia una máquina abstracta que viene a cerrar las necesidades humanas a su manera de pensar. Esto estuvo guiado a la introducción de abstracciones para datos y programas, incluyendo operaciones y aspectos de control.

Los comandos de computadora aspiran a dos áreas distintas pero relacionadas:

- Flujo del programa
- Control de datos

La evolución de lenguajes de programación, generalmente han seguido una trayectoria hacia una mayor abstracción del flujo del programa y control de datos. Evidentemente las metas que vemos hoy, no son las de los 50's, esto es porque resulta importante buscar desarrollos históricos, para ganar perspectivas.

Los lenguajes varian en el camino, ellos se direccionan a las necesidades de programación del proyecto: para un nivel bajo (que significa orientación a código máquina), a muy alto nivel, y el último conocido como cuarta generación. ¿Pero que es una generación en los lenguajes de programación?

La primera generación del esfuerzo de programación fue centrada en nivel lenguaje máquina. Frecuentemente llamado "bajo nivel", por la habilidad que se debe poseer de manipular códigos a nivel bits.

La segunda generación de lenguajes de programación, fue basada en códigos simbólicos, y usaron una rutina ensambladora para convertirlo a lenguaje máquina. Estos lenguajes simbólicos (ensambladores) son lenguajes de nivel intermedio. Como algunos avances importantes dentro de estos se encuentran:

- uso de subrutinas.
- esfuerzo hacia la generación de un lenguaje universal de computadora (el cual fallo).
- Se marca la división entre compilación e interpretación.

En pocas palabras, los interpretes permiten la detección y corrección inmediata de errores, haciendo posible el "rastreo", y no requieren trabajo adicional para optimización.

Los compiladores ofrecen ventajas como:

- Ejecución rápida.
- Requieren menor memoria para el programa.
- Recompilan fácilmente.

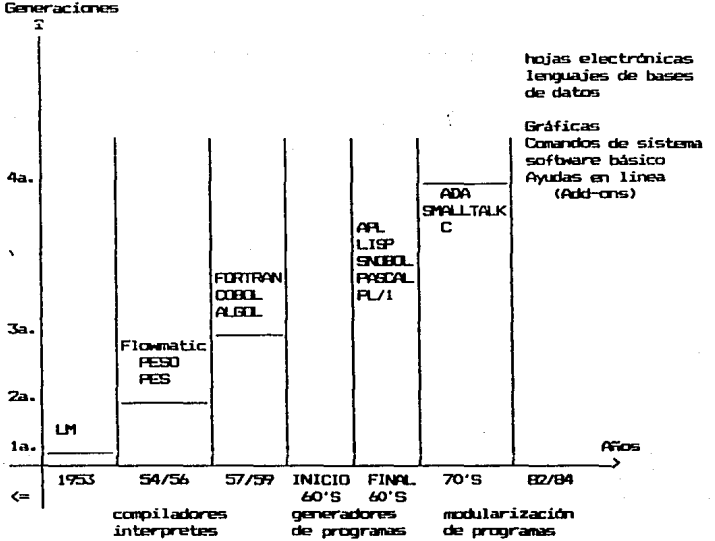
La tercera generación de los lenguajes de programación fue orientada hacia los procedimientos, y se dividió largamente entre negocios y cientificos. Estos fueron llamados lenguajes de alto nivel.

- Cientificos
  . Fortran (1957)
  . Algol (1959), Pascal (1968)
  . C (1970)
  . PL/1 (1967)

- Negocios
  . Cobol (1958)
  . PL/1 (1967)

b) Hacia la cuarta generación.

Los 70's fueron estériles, pues aparecieron las primeras miradas hacia ellos. Su principal contribución fue la modularización de programas a través de jerarquías de datos tipo abstracto como se ve en la figura 2.1.



LM - Lenguaje máquina
PESO - Programa en Ensamblador Simbólico Optimo
PES - Programa en Ensamblador Simbólico

Figura 2.1 : Evolución de los lenguajes de programación.

Un ejemplo de lenguaje orientado a objetos es SMALLTALK de Xerox. Los objetos son datos o procedimientos. "Un objeto llama un mensaje" es igual a "una función es aplicada a un objeto","un objeto manda un mensaje a otro objeto" significa "un procedimiento llama a otro procedimiento", "un objeto regresa un mensaje" corresponde a "un procedimiento regresa un valor".

Entre los criterios que son usados para evaluar que tan bien un lenguaje dado cumple con las necesidades de programación, distinguimos:

1.- Caracteristicas de computación.
- fácilidad con que se escribe un programa.
- simplicidad del lenguaje para propósitos de aprendizaje.
- habilidad de responder a requerimientos especificos en la escritura de programas.
- legibilidad de el código.
- presentación concisa del programa.
- documentación basada en la computadora.

2.- Aspectos de la base de datos.
- manejo de estructuras de datos.
- orientación a datos tipo abstractos.
- manejo de archivos.
- manipulación de entrada y salida.
- perspectiva abierta a comunicación de datos.
- caracteristicas de estructuras de control.

3.- Enfoques generales.
- eficiencia del código final.
- testabilidad.
- herramientas en línea para rastreo.
- habilidad para descomponerse.
- observación de estandares.
- disponibilidad de paquetes.
- habilidad de transportar de mainframes a micros.
- disponibilidad de entrenamiento a programadores.

Muchos de estos criterios se complementan unos con otros, asi como algunos se contradicen. "El lenguaje perfecto" todavia no ha sido creado, y los esfuerzos de arquitectura lingüistica frecuentemente trabajan contrariamente a la solución principal, por causa del medio ambiente ejecutivo. Las interfaces de usua-

rio, recursos compartidos, y sistemas distribuidos complican adicionalmente la tarea.

En efecto, uno de los problemas con los lenguajes de programación es separar las especificaciones, diseño, e implementación para cada uno. Otro problema clave es el desarrollo de lemguajes válidos para familias de sistemas de cómputo. Las tendencias de diseño hoy incluyen el desarrollo de lenguajes que puedan ser formalizados incrementalmente, la orientación de aplicaciones que permitan la participación del usuario, y características de lógica intensiva. Esto es fundamental para la construcción de sistemas de inteligencia artificial y robótica.

Algunos de los problemas clave en el diseño de lenguajes son; como verificar las especificaciones formales en orden de satisfacer los requerimientos originales, y como mostrar que los requerimientos no funcionales están satisfechos. Algunos atributos no funcionales pueden ser:

- Cuantificables:
  Ejemplos son 16 dígitos decimales significativos, y un significado de tiempo entre fallas de X años en una misión de N años.

- No cuantificables:
  También como atributos son la flexibilidad del lenguaje, entendibilidad, fácilidad y utilidad general.

Un lenguaje no debe envolver un solo sistema, sino varios tipos de sistemas como:

1. Enfoques de hojas de cálculo.
2. Sistemas manejadores de bases de datos (SMBD y Querys).
3. Lenguajes orientados a productividad.
4. Ayudas en línea (Add-ons) al software existente.

Una característica común de la nueva generación de de lenguajes es el énfasis que deben poner en la intensidad de "testabilidad", la cual puede reducirse a través de:

- Derivación formal para requerimientos.
- Desarrollar prototipos y reusar las partes examinadas.
- Empleo de un estilo válido de programación.
- Generación de casos examen.

2).- Principales características de los lenguajes de cuarta generación.

a) Antecedentes

La elección de un lenguaje de programación impacta grandemente en la forma que se desarrollarán las aplicaciones, calidad del software resultante, fácilidad, costos de mantenimiento, y lo más importante, en la productividad de los programadores. Esta elección deberá ser resultado de un estudio cuidadoso y bien documentado de las herramientas de software que en la actualidad pueden soportar las computadoras y medios de comunicación.

Por consiguiente, los lenguajes de cuarta generación son un tema muy importante en el marco de los objetivos estratégicos estabilizados por la gerencia. Su implementación es un problema urgente, y se necesitan decisiones para orientar a la compañía en futuros compromisos. Al mismo tiempo, los lenguajes de quinta generación (sistemas expertos) han abierto oportunidades separadas a cualquier método previo. Así, los profesionistas en computación y los usuarios finales, encuentran fácil la comunicación con máquinas inteligentes. En contraparte con esto, en hardware ya no se hacen proyectos con chips de pequeña o mediana escala de integración. Todo el esfuerzo se está dirigiendo a los chips de larga y muy larga escala de integración, dándose mayor énfasis a estos últimos.

Los lenguajes de cuarta generación no solo tienen fácilidades para usuarios finales, estan enfocados también a especialistas en computación, aunque a un grado diferente de sofisticación. Estan en constante evolución, por lo que pueden esperarse adiciones significativas en periodos semestrales.

b) Conceptos fundamentales.

Un lenguaje es considerado de cuarta generación, si por medio de el se pueden realizar tareas tradicionales de informática, bajo el concepto básico de solo decir "que hacer" sin necesidad de decir "como hacerlo". Esto no resulta tan cierto en algunos lenguajes de cuarta generación, ya que para algunas tareas especificas (de mayor grado de dificultad), resulta necesario describir en detalle la manera en que deberá de hacerse.

Una de las ventajas principales de los lenguajes de cuarta generación, es la fácilidad con la que los usuarios finales pueden aprenderlo y aplicarlo. Esto por consiguiente le ayuda a ser capaz de satisfacer sus requerimientos minúsculos por si mismo, eliminando asi las sobrecargas de trabajo que le pudieran surgir al departamento de informática.

Actualmente los usuarios son más capaces de identificar sus problemas reales, porque la computadora y los lenguajes fáciles de usar han llegado a sus manos. Originalmente, se tenia una estructura en la que el personal especializado se encargaba de administrar y resolver los problemas en la computadora. Muy pronto, será necesario descentralizar los departamentos de proceso de datos, y no tener intermediarios ni estructuras burocráticas, en vez de esto, debera darse al usuario el control del software y hardware y la operación de los mismos. No resulta necesario que los usuarios entiendan el diseño de microprocesadores, tan solo necesitan tener una buena apreciación de conceptos básicos y herramientas que pueden utilizar como:

- El manejo de sus datos.
- Sistemas contra software de aplicación.
- Protocolos de comunicación.
- Medios de entrada/salida.
- Menús, prompts y medios de ayuda.

La computación ha tenido una gran penetración en los negocios, al grado de que en sus inicios, los gerentes y profesionistas realizaban un 70% de su trabajo en forma directa, y el resto con ayuda de la computadora. Actualmente, se puede decir que los papeles se han invertido, la computadora realiza del 70% al 90% del trabajo, y el resto los gerentes y profesionistas, los cuales se esfuerzan en pensar como hacer que la computadora trabaje cada dia más por ellos.

Los especialistas en computación se esfuerzan cada dia más para fácilitar las tareas de los usuarios, permitiendo comunicación con bases de datos, computadoras centrales, estaciones de trabajo, procesadores de palabra, manejo de fórmulas, recuperación y manejo de documentos, y sistemas que brinden un mayor grado de eficiencia en el manejo de información.

Al mismo tiempo, como una experiencia de usuarios, un sentido de resultados personales y como obligación de los profesionistas en computación, surgen los lenguajes de cuarta generación. Por tal razón, la implementación de tecnologías modernas esta marcada por tres tendencias interrelacionadas:

1. Disponer de herramientas cada día más poderosas.
2. Mayor tiempo y espacio de separación entre planeación y ejecución.
3. La más rápida y firme fase de desarrollo.

En hardware lo nuevo es pensar en microelectrónica y arquitecturas de microprocesadores, esperando con esto, incrementar al menos mil veces el poder de las computadoras.

En software la orientación también cambiará, el software nuevo (y eventualmente hardware) enfocará sus componentes a la inteligencia artificial, particularmente a sistemas expertos. Esto originará máquinas con capacidades inteligentes casi humanas, en estas se incluye:

. Lenguaje natural
. Entendimiento
. Visión
. Dialecto
. Y varios tipos de razonamiento autómata

Por encima de esta base tecnológica, están siendo desarrolladas nuevas aplicaciones orientadas a usuarios finales, éstas serán con un estilo muy diferente al original. El poder de las computadoras y comunicaciones como una solución tecnológica al problema, se vuelve solo una parte de la respuesta.

La respuesta real es difundir ampliamente el conocimiento y habilidades.Los usuarios finales estan lejos de aceptar la proposición que solo unos pocos privilegiados quienes se entienden con la operación diaria de las máquinas, pueden ser intimamente envueltos con los procesos de información. Este ha sido el lado oscuro de las soluciones tecnológicas. Ahora los usuarios finales quieren participar.

La participación es precisamente la que profesionistas de computación pueden ofrecer a través de lenguajes de cuarta generación, los cuales ofrecen la posibilidad de tener al mismo

tiempo una herramienta, una misión, y un mensaje.

Una buena respuesta a un problema bien definido es:

1. identificar el problema.
2. escoger el análisis y lenguaje apropiado.

En su mayoria, los lenguajes de cuarta generación estan definidos con claridad y su sintaxis es especificada con un formalismo riguroso. Tan clara especificación es una condición necesaria (pero no suficiente) para una implementación realizable y efectiva.

b) Nuevas perspectivas para profesionistas de computación.

Cuando hablamos de sistemas de información como una industria de conocimiento, una de nuestras obligaciones clave es la educación. Hay una interdependencia natural y creciente entre el aprender de toda la vida y los resultados profesionistas. Siempre la fundación en que el entrenamiento en computadora y el resto de comunicaciones cientificas deben ser reestructuradas. Atrás en los 60's miles de especialistas en computación estaban trabajando en la industria, y fueron educados como matemáticos, físicos, e ingenieros. Ellos aprendieron la ciencia de la computación en sus trabajos, en efecto, por prácticas cotidianas.

Solo en 1964 fue el primer grado avanzado de ciencias de la computación. Sus inicios fueron concentrados 2 décadas atrás, lo cual no fue apreciado por la falta de conocimiento.

La primera generación de paquetes de PC's de 8 bits, tenian un solo fin, tal como:

- hoja de cálculo (VisiCalc)
- Procesador de palabra (Word star)
- Gráficas
- Bases de datos (Dbase II)

Todos estos paquetes son inteligentes, pero su uso no esta libre de problemas. El problema mayor es que las convenciones de edición y formato de datos no siempre son compatibles, esto confunde a los usuarios y los obliga a realizar conversiones con funciones propias que cumplan con tal fin.

La proliferación de estos y varios paquetes más, proporcionan una oportunidad a los profesionistas de computación, para ayudar a los usuarios en la tarea de aprendizaje de los mismos. Como ejemplo de ellos tenemos:

- Sorcim Supercalc
- SuperWriter
- Super file
- Micro prowordstar
- Lotus 123
- Symphony
- MDBS knoledge manager
- Framework

Una nueva oportunidad se pone ahora a la vista. La industria de computación y comunicaciones esta entrando a su segunda epoca, en la cual la computadora sirve como un comunicador. Los diseñadores de sistemas deben ser capaces de ligar tantas computadoras y bases de datos (que manejen grandes cantidades de información) como sea posible. Las bases de datos y estaciones de trabajo deberán poder ser accesadas por cualquiera que trabaje dentro de la red (network).

Esto hace sobresalir el crecimiento necesitado de ligas y compatibilidad de PC's a mainframes . Los usuarios finales de microcomputadoras no pueden ser enteramente independientes de las computadoras centrales y las bases de datos. Se necesitan soluciones técnicas para trabajar en redes que permitan accesar la información de centros de mainframes por usuarios finales. Los especialistas de computación y comunicaciones deben analizar los resultados y revisar las implicaciones de desarrollos de software recientes.

Por último incluir el crecimiento en número de paquetes para PCs y mainframes que permitan a los usuarios de las estaciones de trabajo:

- Extraer información de bases de datos de los mainframes.
- Ponerla dentro de un almacenamiento de la PC para un uso local del personal de cómputo.

Los especialistas en computación deben ser capaces de asegurar un análisis detallado de estos resultados, evaluar desarrollos en tecnología de LANs, e imponer las reglas de sistemas totalmente distribuidos. Ellos deben estabilizar definitivamente la estrategia de PCs para que siga la tendencia, la tecnología de microcomputadoras esta aquí, y sigue extendiéndose. Dentro de los intereses de los profesionistas de computación tenemos:

- Alentar una rápida adquisición de micros para automatizar el lugar de trabajo.
- Proteger el acceso a base de datos y facilitarlo para usuarios autorizados.
- Traer (import) y llevar (export) en una base dinámica.
- Evitar duplicidad de funciones centrales de proceso de datos, a menos que exista una razón para hacerlo.

Hay varias formas de implementar una LAN, pero debe observarse que:

- Se minimize el costo de líneas de comunicación.
- Dar mejor recuperación de información.
- Hacer fácil la delegación a los usuarios de la tarea de extraer información de la base de datos del mainframe.

Por guardar los commponentes pequeños haciéndolos inteligentes, nos ponemos en una posición fuerte. podemos formar, afinar y reestructurar el sistema como requerimientos y cargas emitidas.

El énfasis en las micros no significa que los mainframes esten fuera, la regla esta cambiando, pero la demanda por ellos crece. IBM dice que el poder invertido en instalaciones de mainframeses un impresionante 80 a 120 porciento por año por causa de las PC's.

c) El enfoque del usuario.

La tecnología continua avanzando a un paso acelerado, creando nuevas aplicaciones y oportunidades. También se vislumbra qûe adelante hay problemas que amenazan con crecer. Estos obstaculos pueden ser tecnológicos y humanos, y si no son librados con el direccionamiento adecuado, pueden bloquear la continuidad del progreso.

Lo que necesitamos en la nueva ciencia de la información es el proceso de "ingeniería en reversa (reverse engineering)" aplicado en la cima tecnológica en que ponemos a las aplicaciones. Aquí otra vez, los servicios que pueden ser ofrecidos por los lenguajes de cuarta y quinta generación serán instrumentales en puntear la brecha entre avances tecnológicos y la necesidad de renovar el conocimiento de profesionistas.

Veamos el proceso:

Para examinar la telaraña de líneas del circuito en un semiconductor, los diseñadores en fábricas competitivas pueden descubrir y replicar las ideas que pudieron haber llevado a la compañía original años y millones de pesos desarrollarla. La ingeniería en reversa es una práctica común entre fabricantes de semiconductores.

Similarmente, aplicando la ingeniería en reversa a los principios del desarrollo de sistemas, tenemos que examinar la calidad percibida por el usuario en sistemas interactivos. Los cinco problemas clásicos son:

1. Utilidad adecuada.
2. Control de usuario.
3. Autodescriptivo.
4. Fácil de aprender.
5. Tolerancia de errores.

Todos estos problemas tienen correspondencia directa con expectativas de usuarios. Una utilidad adecuada implica preparación independiente de tareas, la habilidad de usar valores default cambiables, salida ajustable a la eficiencia del usuario, y un diálogo enfocado al área de aplicación.

El control de usuario envuelve la manipulación del avance del diálogo (incluyendo velocidad), caminos definibles por el usuario para cambiar información, aprendizaje del sistema por experiencias pasadas y el punto actual del diálogo. También significa acciones reversibles y confirmación del usuario para pasos especiales y críticos. Un diálogo es controlable por el usuario, si en relidad el usuario es capaz de encaminarlo a respuestas que satisfagan sus necesidades.

Ser autodescriptivo comprende clarificación adecuada de la capacidad de las herramientas de software y hardware que se pone a disposición del usuario, así como las asunciones para su uso. En los requerimientos de usuario, el propósito y método debe ser explicado fácilmente durante el diálogo. La explicación debe ser dirigida al usuario e incluir la situación del diálogo y su contexto.

Fácil de aprender envuelve la capacidad de explicación paso a paso durante la ejecución del diálogo. Si se presentan términos técnicos, deberán de ajustarse al nivel de conocimiento del usuario:

- Pequeños ejemplos para principiantes.
- Ayuda para reconocimientos más complejos por comunicadores expertos.

En ambos casos, un diálogo es fácil de seguir si el usuario esta soportado, y puede adquirir conocimiento acerca de la aplicación y ejecutar la tarea de una manera autosuficiente.

La tolerancia de errores tiene varios aspectos. Básicamente, proclama seguridad y habilidad. Para puntos de vista de operación, requiere información acerca de procedimientos de corrección, la habilidad de encontrar todos los errores detectables, y la provisión de herramientas para su corrección. Solo mensajes comprensibles, relevantes y constructivos pueden ser admitidos. La tolerancia de errores es la capacidad de un sistema para procesar entradas de usuario en el camino correcto. Aunque algunas entradas erroneas pueden irse dentro del sistema. Sin embargo, el resultado deseado de la tarea puede ser archivado. De otra manera, el usuario sera provisto de información comprensible acerca del origen del error y los pasos a seguir para corregirlo.

Los profesionistas de computación y comunicaciones, se deben direccionar a satisfacer estos puntos y aplicar sus conocimientos para diseñar estrategias como:

- Configurar, esto es, adaptar un sistema de acuerdo a los parámetros dados.
- Adaptar software y hardware a los requerimientos y necesidades de aplicación del usuario.
- Proporcionar un diálogo inteligente , incluyendo control y

evaluación.

También las metas en la estrategia de diseño son controlar, registrar, medir, analizar y justificar el comportamiento del usuario y del sistema. Parte del reto es el establecimiento de criterios para mejorar el sistema.

Mientras que las aplicaciones de programadores pueden concentrarse en la implementación de tareas especificas, el autor del diálogo interactivo debe tener una herramienta para programar la interface usuario-máquina. La propia evaluación de un diálogo interactivo debe permitir la adaptación del sistema a diferentes demandas del usuario. Esta adaptación también debe ser implementada por el usuario.

La estrategia de diseño debe incluir también gran atención al tiempo de respuesta. Los gerentes y profesionistas no deben sentarse ociosamente frente al video:

. 1 segundo de tiempo de respuesta es optimo;
. 2 segundos es aceptable

En términos de implementación global, la flexibilidad es un premio. Cada usuario final descubrirá en su camino como el personal de computación esta yendo a trabajar. Justo como el cerebro en cada persona es único, así son las capacidades que soportará.

Hasta la disponibilidad de personal de cómputo, que empezó en el medio en 1975, los sistemas de información siempre tuvieron una operación centralizada que requería interfaces complejas entre el usuario y la máquina. Lentamente venimos a realzar que el valor real del personal de cómputo y los lenguajes amigables con el usuario, no son hacer las tareas de un pequeño sistema de negocios a un costo bajo, es dar asistencia a la gente para hacer su trabajo mejor y más rápido, siendo esto cierto para usuarios finales, como para profesionistas de computación.

3).- Inversión en software.

Los lenguajes de cuarta generación han demostrado dar un incremento de productividad del rango de 600% a 6000%. Esto excede remotamente lo que teníamos durante las tres primeras décadas del

uso de la computadora. En los 70's, la programación estructurada, generadores de reportes, y lenguajes Query fueron introducidos para favorecer la productividad del programador. Maliciosamente a estas ayudas, los programadores se sentian incapaces de responder a los requerimientos del usuario. Los mejoramientos de productividad fueron por lo tanto marginales.

Los lenguajes de cuarta generación pueden contribuir a una mayor eficiencia en el uso de la máquina, porque se requieren de pocas instrucciones como en los lenguajes Query estructurados (SQL), por lo tanto, son de 5 a 20 veces más rápidos que Cobol en ambientes IBM. En adición, no requieren la virtuosidad técnica en la parte del programador que requerian los lenguajes viejos.

Recientemente, los participantes del curso Unix/c/Ingres, incluyeron 2/3 partes de profesionistas de procesamiento de datos y 1/3 parte de usuarios finales. Despues del curso, tomo a los usuarios finales 4 semanas programar una aplicación de seguros de exportación en Ingres y C; la mitad del tiempo fue invertido en adquirir practica en el uso del lenguaje. Este proyecto estaba pensado por el departamento de proceso de datos para 3 años, y en lenguajes de cuarta generación seria reducido aproximadamente a 3 meses.

Otra contribución importante de los lenguajes de cuarta generación, es la nueva libertad en encontrar la portabilidad de aplicaciones programadas. La doble emergencia de sistemas operativos portables y compiladores de dos niveles, aislan programas de aplicación en detalles de hardware, y hace esto factible de mover sin mayores problemas de una pieza de hardware a otra. Esto cambia la base en que las decisiones de negocios seran hechas en el futuro.

Los cambios tecnológicos hacen la base para tomar decisiones, para su obtención el manejo debe estar:

. Más para la interface de software
. Mucho menos para el hardware

Los lenguajes de cuarta generación y procesadores genericos hacen el hardware inconsecuente con el usuario. Hay 3 pasos para la libertad del software y hardware:

1. Sistemas operativos comodos, tal como C/PM para 8 bits,

MS-DOS para 16 bits, u UNIX para 32 bits en micros.
2. Compiladores de 2 niveles (Front end and Back end).
3. Procesadores genericos (Corriendo simultaneamente sistemas operativos y compiladores).

Tambien se deben de estabilizar estandares aceptables en otras áreas claves como:

4. Gráficas.
5. Sistemas manejadores de bases de datos (SMBD).
6. Comunicaciones punto a punto.

4).- Elementos que mejoran la productividad.

Cuando la tecnologia avanza en una fase impresionante, la dificultad que sobra descansa en la interacción humana con las máquinas a su disposición. En esta conexión, mucho depende en motivación, y mucho también en entendimiento propio de las herramientas que tenemos disponibles. En experiencia humana podemos sumar cualquier cosa al inventario del conocimiento (si lo planeamos asi). El documentarse en computación es muy importante, y esto es aplicable tanto para usuarios finales como para profesionistas en computación.

Estamos hablando de 2 métodos diferentes de documentación en computación. Los usuarios finales deben estar trabajando extensivamente con los lenguajes de cuarta generación fáciles de utilizar; hojas de cálculo, paquetes gráficadores, dispositivos de correo electrónico, servicios de calendario (agenda), procesador de palabras, y software integrado. Para conocer como usar estas herramientas, el usuario final deberá a mediados de los 80's ser estandar con la literatura de cómputo , pero a finales de la década , la documentación de usuario será medida por la habilidad de interactuar con sistemas expertos de una manera inteligente.

Los avances de la ciencia y tecnologia, obligan a tomar nuevos puntos de vista para problemas viejos como calidad, privacia, seguridad, pirateria, y también productividad.

La <u>productividad del programador</u> es tradicionalmente definida como el número de lineas fuente diseñadas, codificadas, examinadas y documentadas por unidad de tiempo, tipicamente una hora,

en dia, o un mes. Esto comprende, entre otras cosas, calidad de resultados. Pero es necesario cuantificarlo, el algoritmo es:

$$\text{Productividad general} = \frac{\text{beneficio}}{\text{costo}}$$

Ahorrar en costos es una mala politica, sobre todo si se refleja en empleados con un nivel bajo en conocimientos. Esto originaria también un beneficio mucho menor y el radio de este seria una desventaja.

"Quien paga un cacahuate, atrae solo monos"(*7); pero aunque sea uno de los prerrequisitos, una buena paga no es una condición suficiente para alta productividad. La motivación, trabajo duro, imaginación, y una mente abierta son también prerrequisitos, y más particularmente, es la verdad del aprendizaje de toda la vida.

Tanto para usuarios finales como para profesionistas de computación y comunicaciones, debemos poner a su alcance alta tecnologia para trabajar. La figura 2.2 refleja los resultados obtenidos dentro de una fábrica de computadoras. Al disponer de alta tecnologia, se obtiene una considerable mejora en los beneficios obtenidos.

Para apreciar mejor cómo y porqué, los lenguajes de cuarta generación ayudan a los profesionistas en esta dirección, cambiemos a aspectos estrictamente técnicos. Clásicamente, la productividad es una medida de salida primariamente basada en unidades fisicas: Es producido en relación a un factor de entrada tal como horas de labor. Pero nosotros sabemos que, para ser significativa, una medida de productividad debe ser ligada al problema y al sistema de producción en turno.

(*7) Fourth and fifth generation programing languaje V.I. Chorafas Dimitris, pp 235.

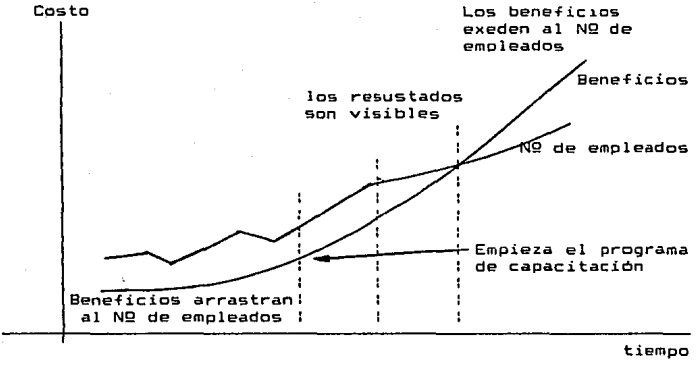


Figura 2.2 : Efecto positivo en la relación beneficio costo al disponer de alta tecnología para trabajar.

También apreciamos que hay diferentes enfoques de medición de productividad:

- Los economistas realzan la habilidad de un sistema de producción para buenas entregas y servicios para consumo.
- Los gerentes de negocios usualmente comparan la relación entrada/salida en departamentos o negocios similares.
- Y en fábricas, comparan la producción con horas de trabajo utilizadas.

La productividad de las organizaciones no puede ser siempre medida en términos de labor por hora. Debemos también explicar factores de clasificación para eficacia gerencial a satisfacción de los usuarios y derivado para la calidad del conocimiento poseído por el desarrollador del sistema. Los objetivos de productividad para desarrollo de sistemas pueden obligar a los nuevos 10 mandamientos:

1. Reducir costos.
2. Implementación más rápida.
3. Plan de trabajo seguro.
4. Muy pocas horas del trabajador.
5. Enfoques interactivos.
6. Calidad del sistema.
7. Caracteristicas de integración.
8. Control de usuario.
9. Mantenimiento muy sencillo.
10. Flexibilidad completa.

La posibilidad de efectuar estos 10 mandamientos vendrá más aparentemente cuando la herramientas de lenguajes de cuarta generación sean aplicadas más extensamente. La mejor prueba será la terminación de proyectos líderes. Con implementaciones prósperas bajo control, la firma tendrá ganada experiencia valuable en promover cambios. No solo será un monto significante de educación que ocurrió, si no la aplicación será propiamente el mejor demostrador de factibilidad.

También es una nota apreciable que incrementa la productividad intelectual de los trabajadores, no es meramente un problema si se proporciona una mejor herramienta. También tenemos que estar dispuestos a mejorar todos los aspectos del trabajo, pues el tiempo es precioso.

5).- Programación en cuarta generación.

a) Generalidades.

El desarrollo de software es tanto un arte como una ciencia, y un programa de computadora es un activo mercadeable como otros productos raramente diseñados para esto. En estas pocas palabras descansa el concepto de programación en cuarta generación.

Si el software es una comodidad mercadeable (como lo es), debemos ser entusiastas e industrializar esta producción, y utilizar dentro del proceso no solo el poder de la computadora, si no también cualquier cosa necesaria para la optimización del esfuerzo. Sobre todo, esto significa documentar propiamente políticas y procedimientos.

En un encuentro con la "Microelectrinics and computer technology Corporation of Austin, Texas"(*8), se dieron las siguientes estadisticas.

Alrededor del 80% de programas de computadora, usan el 2% del ciclo de máquina, pero su existencia es critica.

La productividad de programación es el problema de esta clase, no el consumo de mayor o menor potencia de la computadora.

Alrededor del 2% de los programas usan el 50% del ciclo de la máquina.

Ellos deben ser programados en "C" (en medio ambiente Unix) o cualquier otro lenguaje similar (PASCAL o BASIC). Y no en el gordo y rechinante COBOL, ni tampoco FORTRAN. Y como complemento, "el sobrante 18% usa el 48% del ciclo de la máquina".

Estos programas existentes y proyectados deben ser escogidos cuidadosamente , así como el lenguaje a utilizar. La oportunidad es que despues de un estudio, sea factible desarrollar la mitad en lenguajes de cuarta generación y el resto en C o en un lenguaje similar. Severamente, esto tenderá a disminuir la población de programas de computadora a ser hechos en una relación de 90 a 10 porciento.

El 90% del esfuerzo de programación debe ser hecho, a través de lenguajes de cuarta generación para la alta productividad en el desarrollo de software. Esto puede requerir más ciclos de máquina. ¿Y luego que?.

Primero, la labor es la comodidad más cara de hoy, esto es lo que debemos cuidar. El hardware es poco en comparación con esto, además los lenguajes de cuarta generación permiten el desarrollo de prototipos. En turno, este camino salva el poder de la computadora porque hace posible la detección y corrección de manchas débiles en el cercano estado de desarrollo. **El desarrollo de prototipos es la primera piedra de la programación en cuarta generación.**

(*8) Fourth and fifth generation programing languaje V.I. Chorafas Dimitris, pp 187.

Segundo, arriba del 75% del poder de la computadora será instalado al final de esta década bajo nuestro escritorio, algunos estiman que será cerca del 90% en los 90's.

b) La elección de un lenguaje de programación

Cuando hablamos de lenguajes de programación, nuestra primera, segunda y tercera elección debe ir a la eficiencia en desarrollo de software, planes de trabajo cortos, y asegurar calidad. Esto significa que ahora y en el futuro, solo los niveles muy altos de estructura merecerán la atención de los gerentes.

Para un alto nivel de software, necesitamos una super estructura, una estructura principal y unos cimientos. Dependiendo del avance de la tecnología, estos 3 conceptos serán útiles para software y hardware.

Actualmente y por algunos años más, tendremos a nuestra disposición lenguajes de cuarta generación clasificados en 5 grupos:

1. Programar extensiones al sistema operativo a través de intérpretes de comandos.
2. Sistemas manejadores de bases de datos y querys.
3. Lenguajes nuevos de programación tales como gráficas y comunicación de datos.
4. Herramientas orientadas a la productividad, a través de precompiladores.
5. Sistemas de hojas de cálculo.

Recordemos siempre que la evolución tecnológica cambia fuertemente la percepción, contenido, y funcionalidad de los sistemas a nuestra disposición. Esto es cierto tanto en software como en hardware, aunque la larga mayoría de los usuarios piensen que los cambios se realizan al lado del hardware.

Los sistemas manejadores de bases de datos ha tenido la misma evolución:

- Jerarquica (1ª generación).
- Networking (1½ generación).
- Capacidades orientadas a consulta (Querys 2ª generación).
- Relacional y con éstas, fácilidades de lenguajes de programación (3ª generación).

El software Integrado tiene sus generaciones:

- Hojas de cálculo (0ª generación)
- Lotus 1-2-3 (1ª generación)
- Symphony, del mismo fabricante de Lotus (2ª generación)

Los lenguajes de cuarta generación ofrecidos en las 5 clases mencionadas, son razonablemente eficientes y bien desarrollados; son tocados fundamentalmente y son orientados a manejar el flujo de datos. Con estos tipos de lenguajes, podemos escribir programas basados en estructuras de datos.

Los lenguajes que podemos escoger en ambiente UNIX son:

- Ingres y Oracle para sistemas grandes.
- Informix, Sequitar, Mistres, y Unify para sistemas pequeños.

Univac

- Mapper

IBM

- Application system (AS,IBM)
- Cross-System Product (CSP, IBM)
- ISQL
- QMF
- QBE
- DB2I
- NATURAL
- ORACLE

Hewlett-Packard

- . Powerhouse
- . Speedware
- . Today
- . Oracle

PC's compatibles con IBM

- . Micropowerhouse
- . Microspeedware
- . Today
- . Oracle
- . Dbase III plus y Clipper (3½ generación)
- . Dbase IV

Observese la gran compatibilidad y portabilidad que posee Oracle, pues se encuentra implementado en una gran variedad de equipos, tanto grandes como pequeños.

c) Prototipos.

Los lenguajes de cuarta generación hacen fácil la manipulación de datos y el cambio de la estructura de la base de datos. La clave es experimentación prospera a través de prototipos. Con los prototipos, el analista programador o usuario final pueden intentar diferentes caminos para hacer las cosas, incluyendo la forma de la estructura de la base de datos. Es importante con lenguajes de cuarta generación que las alternativas desarrolladas son autodocumentadas, el usuario participa, y la documentación no es osificada.

A través de prototipos, los usuarios finales y especialistas:

1. Se alejan de una escritura de especificaciones enormes.
2. Ayudados mejoran la calidad del producto con un poco de ayuda.
3. Dieron para la primera vez en este medio ambiente, realimentación de vida real en los resultados obtenidos.
4. Guardan su dinámica del sistema y apertura al cambio.

El último punto es el más importante con un medio ambiente distribuido a el nivel de estación de trabajo. En cualquier tiempo dado, una estación de trabajode usuario final puede comunicarse con un número fijo predefinido de aplicaciones anfitrionas. Alternativamente, la estación de trabajo de usuario pueden envolver en un agregado, y eventualmente constituir un nodo de alta eficacia de mensajeria. Después, serán requeridas nuevas facilidades para guardar el rastro de miles de destinos finales y para responder dinámicamente a información acerca de nuevos destinos y nuevas rutas.

Propiamente hechos, los prototipos de proyectos se reflejan incremento de eficacia , soportan planes de trabajo cortos, aseguran ahorro en costo y usuarios felices.

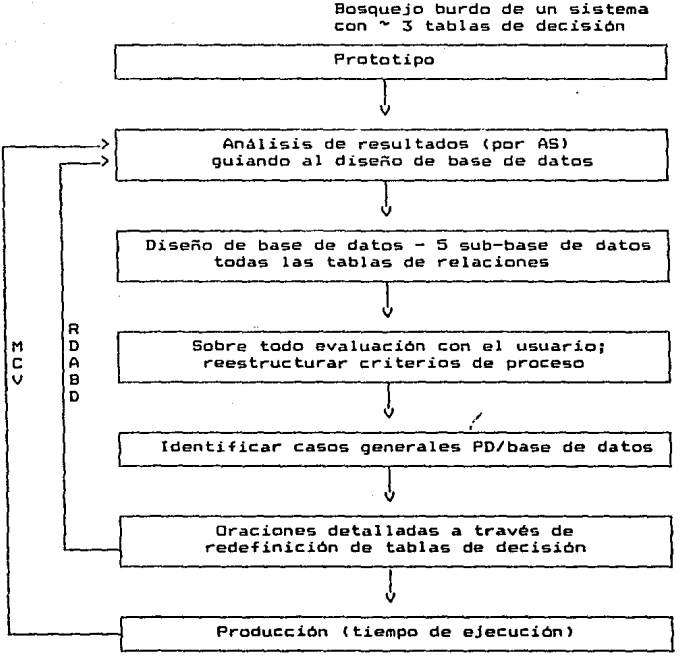
El análisis de sistemas clásico desaparecerá pronto, porque muchas tareas de descripción son ejecutadas automáticamente por la computadora; En la figura 2.3 se muestra el procedimiento completo.

Un medio ambiente ideal para prototipos de software tiene 3 componentes:

1. Un lenguaje de cuarta generación, para permitir desarrollo rápido de prototipos.
2. Recursos de datos manejados propiamente, para base de datos y comunicación de datos.
3. Buen conocimiento del empleo de lenguajes de cuarta generación y estructuración de datos, por parte de los usuarios finales y profesionistas.

El profesionista en computación es necesitado como constructor de prototipos. El mejor equipo incluirá un usuario final y un profesionista. El usuario es la persona que necesita que el trabajo se haga, y el profesionista en computación conoce los trucos del negocio.

Examinemos nociones avanzadas como lejanas. Los lenguajes de cuarta generación ofrecen la oportunidad de volverse libres de la rigidez de los enfoques tradicionales; ellos ofrecen no solo el más rápido desarrollo, sino también la oportunidad de dar mejores sistemas al usuario. La clave es la habilidad de adoptar un enfoque experimental para desarrollos.



AS - Análisis de sistemas
MCV - Mantenimiento del ciclo de vida
RDABD - Realimentación del desarrollo ajustando la base de datos
PD - Proceso de datos

Figura 2.3 : Procedimiento para usar lenguajes de cuarta generación con propósitos de prototipos.

Por desarrollo de funciones de prototipo en lugar de grandes especificaciones, el usuario es para el primer tiempo dado, realimentación en vida real de que el resultado final gustara. Podemos buscar en ellos, controlarlos, alterarlos, y actualizarlos sin creer que el proyecto con 6 meses atras de tarea, ocurrió con la intervención del viejo análisis de sistemas y programación.

Un prototipo tipico envolvera 5 tipos de módulos:

1. Identificación y descripción de todos los archivos y registros. Incluyendo identificación de campos, formato, rango de valores admisibles y otras cosas para cada campo.
2. Todas las salidas definidas. Y posiblemente alternativas para su representación (tipicamente en pantalla).
3. Listado de todos los programas y comandos. Expresados en el lenguajes de cuarta generación que se haya escogido.
4. Identificación de el tipo de dispositivos interactivos empleados (terminales tontas o preferiblemente inteligentes).
5. Ligamientos al mundo exterior. Particularmente rutinas y medios de comunicación.

Muchos lenguajes de cuarta generación trabajan a través de metáforas, esto es, caminos de descripción que esta haciendo la computadora en contraste con lo que hace la gente. Una metáfora ayuda en mencionar la acción, mandar un mensaje, o bien dar una interface con tecnologia comun.

Los prototipos pueden seguir también la descomposición funcional o preferiblemente, el enfoque de estructura de datos. Estas son las 2 escuelas del pensamiento desarrolladas desde mediados de los 70's con análisis estructurado. Lo último es más reciente y ayuda a resolver algunos problemas duros. Primero, es difícil encontrar el mejor camino para implementar una función descompuesta. Segundo, siempre es mas difícil reconciliar la estructura de una función con las estructuras de datos.

Aunque la automatización de descomposición funcional ha resultado en muchas herramientas nuevas (incluyendo generadores de aplicaciones y lenguajes de definición), los sucesos de estas herramientas palidecen cuando son comparados, con los resultados

que pueden ser obtenidos con lenguajes de cuarta generación a lo largo de la línea de estructuras de datos.

Las estrategias de descomposición parecen especialmente débiles en la fase de los cambios ocurridos en todo el concepto del manejo de información. Estos cambios son evidenciados por la amplia aceptación de los sistemas manejadores de base de datos y cómputos de usuarios finales. Al mismo tiempo, las fácilidades soportadas a través de lenguajes de cuarta generación tienen que ser instrumentales en el incremento de la calidad del software. Esto es particularmente valuable por los profesionistas, puesto que la programación de computadoras esta gobernado por la ley de Murphy: **"Cualquier sistema eventualmente se viene abajo".**

El examinar software nuevo es un proceso muy largo; frecuentemente se consumen más recursos que al programarlo, y es necesario envolver un amplio rango de usuarios. Tipicamente se hacen distinciones por muchos estados de examen. Es ventajoso si cada uno de estos estados se basan en la computadora.

Recapitulando, los prototipos pueden ser de gran valor en 2 casos especificos:

1. Cuando el usuario no esta seguro de sus necesidades. El enfoque de programación visual ayuda a clarificar estas necesidades, dando un marco de examen por la descripción de la implementación.

2. Cuando el profesionista de cómputo quiere experimentar para optimizar el diseño, archivos, y estructura del código.

La optimización puede ser hecha solo por especialistas que estan cultos en el uso de las herramientas desarrolladas, apreciativos acerca del uso de recursos, al tanto de la necesidad de tiempos de respuesta cortos, y capaces de entender los requerimientos de información de la organización.

El concepto del usuario más que diseño del manejador de tecnologia, debe ser el cimiento. Para mejorar la productividad de analistas programadores en un rango de 600 a 6000 porciento, son necesarios los lenguajes de cuarta generación. Para que sea factible, debemos gastar el tiempo necesario para el prototipo, en la tabla 2.4 se ilustra este comportamiento:

Tabla 2.4 : Productividad del programador bajo UNIX/Ingres contra GCOS III/Cobol.

Fase del proyecto	Unix/Ingres	GcosIII/Cobol	Relación
Preparación del formato de video para control y entrada de datos	2 hrs.	3 semanas (120 min.)	1:60
Diseñar una forma de salida			
Sofisticada	7 a 30 min.	2 días (1000 min.)	1:53
No sofisticada	100 min.	3 días (1500 min.)	1:15
Programa para recuperación de información	flexible 2 a 3 min.	inflexible 2 a 3 semanas (5000 a 7500 min.)	1:2500

Un prototipo puede incluir proceso de datos tradicional, procesador de palabras, manejo de documentos, correo electrónico, y otros. Puede ser creado a través de una hoja de cálculo, un lenguaje tipo base de datos, y debe ser específico para el trabajo, no una generalidad.

6).- Aplicación técnica.

a) Puntos clave con lenguajes de cuarta generación.

Para automatizar el desarrollo de software, es necesario un lenguaje de alto nivel (preferiblemente lenguajes de cuarta generación), un sistema manejador de base de datos avanzado, un editor inteligente, y una librería de software reusable. Este proceso envuelve:

- Oraciones no procedimentales (decir que hacer, no como hacerlo).

- Translación de descripciones de lenguajes de cuarta generación a un conjunto de especificaciones para módulos de programas.
- Recuperación de componentes apropiados para la librería reusable.
- Generación de nuevos módulos como se necesiten.
- Eliminación del manejo de archivos a través del sistema manejador de base de datos.
- Integración de los componentes de software dentro del programa ejecutable.

Este modo de operación permite un prototipeo rápido, y la generación automática de programas. El rango de herramientas es impresivo; hojas de cálculo para aplicaciones gerenciales, SQL, Ingres, y Mapper para aplicaciones de negocios, APL y CAD/CAM para aplicaciones cientificas e ingenieriles, C y Prolog para escritura de compiladores, y Unix/C-shell para construcción de sistemas.

Un lenguajes de cuarta generación próspero produce aplicaciones más rápidamente que los lenguajes de alto nivel (COBOL, FORTRAN, PL/1, etc.), con una relación de 60 a 1 para rutinas simples y/o fuertes aplicaciones orientadas a base de datos. Los lenguajes de cuarta generación emplean sintaxis que es tipicamente no procedimental, remueve y automatiza el detalle de trabajo repetitivo en consultas, reportes, y otros. Actua como lenguaje de control con comandos de sistema, e integra instrucciones escritas en lenguaje de alto nivel. Ellos son idealmente orientados a manejo de base de datos en una manera relacional.

Un punto a favor de los lenguajes de cuarta generación es que pueden ser aprendidos en 3 días o menos. Estos también pueden ser entendidos por usuarios finales y por especialistas en sistemas. Funciones adicionales (add-ons) incluyen soporte de pantallas, diccionario de datos, procesos transaccionales, mensajes, e intercambio de archivos. Además tienen caracteristicas como privacia, seguridad, comunicación de caracteristicas de base de datos, y recuperan medios ambientes. Todas estas fácilidades basadas en la computadora, revolucionalizan todo el proceso de análisis de sistemas.

Los analistas de sistemas, que pretendan ser realmente buenos, deben imaginar intensamente y comprensivamente. Deben

ponerse en el lugar del usuario y buscar una perspectiva en el desarrollo de sistemas. La imaginación debe ser puesta en acción, para también entrenarse. La persona que hace esto después de desarrollar su conocimiento en el curso debido, obtiene una imaginación fuerte.

b) Un enfoque conceptual (visual) y práctico.

Las capacidades de pensamiento visual realzan la imaginación del analista. El pensamiento visual guía a programación visual. Con programación visual, "lo que vemos es lo que conseguimos". El usuario define una pantalla y la asocia con una expresión sin tener que preocuparse por la estructura y manejo de archivos, posteriormente esto es realizado a través de una interface de alto nivel, utilizando "generadores" para hacer el proceso. En la tabla 2.5 podemos observar esto.

El análisis de sistemas y diseño de base de datos, son una fuerte pareja con lenguajes de cuarta generación. En bancos que tienen programado, el despacho automático de papel comercial y financiero a través de COBOL y PL/1, tipicamente dedicaron 2 años de trabajo por un grupo de 4 ó 5 personas. Invariablemente, el resultado es 10 años de trabajador invertidos.

Para esta aplicación se envuelven cuatro subsistemas principales:

1. La construcción de una base de datos de despacho automático (colección de elementos de información no esta incluido en el tiempo estimado).
2. Programas utiles para actualizar la base de datos.
3. El subsistema de despacho automático (orientado básicamente a procesos batch).
4. Programas interactivos para manejar excepciones y rechazarlas.

Los programas disponibles para este sistema tienen un promedio de 4,000 a 5,000 instrucciones Cobol, sin contar rutinas de manejo de archivos y otras utilerias.

Tabla 2.5 : Pensamiento visual.

Principios	Caracteristicas
Modelo explicito de usuario	Presentación de escritorio (Desktop) Iconos Ventanas
Viendo y apuntando	Ratón menús menús encimados
Comandos universales	Insertar eliminar buscar copiar cambiar mover ayuda mostrar propiedades salir escapar fin
El principio es: lo que vemos es lo que conseguimos	

INGRES fue utilizado como lenguajes de cuarta generación para realizar el mismo trabajo. El primer prototipo abarco el 80% del trabajo requerido, y fue creado por un analista programador experimentado en un lapso de 4 horas. El propósito fue demostrar la factibilidad. El prototipo completo al 100% del trabajo tomo una semana.

Para usar las fácilidades incluidas en Ingres y su subconjunto de consulta, el analista programador, necesita solo crear dos rutinas de lectura, un formato de pantalla, un par de rutinas de cálculo, y llamar a macros de la base de datos. Un 90% del trabajo estimado fue hecho en EQUEL, y el 10% en C. Lo más importante es que este enfoque de implementacion, va separado del

análisis clásico de sistemas. El nuevo análisis de sistemas bajo lenguajes de cuarta generación es mostrado en la tabla 2.6. Este concepto es también comparado con las tareas clásicas de análisis y programación.

El proceso mostrado se asemeja a una solución de paquete de herramientas. En un medio ambiente de aplicación bien escrita, estos serán muchos módulos reusables en cada aplicación. Esto es vital para un producto, cuando los usuarios desarrollan aplicaciones a través de paquete de herramientas. También las facilidades a ser incluidas en un enfoque de "paquete de herramientas" serán:

1. Administración de pantallas.
2. Manejo de formas.
3. Construcción de tablas.
4. Especificaciones de base de datos y comunicaciones.
5. Ventanas para revisar aplicaciones.
6. Menús encimados.
7. Iconos como representaciones gráficas de artefactos de estaciones de trabajo.
8. Estandares para ayudas y menús.
9. Comandos universales incluyendo ayudas, iniciar, mover, cancelar, deshacer, etc.
10. Interfaces en lenguaje natural, incluyendo capacidades de entrada/salida de voz.

En muchos sistemas manejados por usuarios, primero tenemos solo una vaga idea de "lo que podemos crear" con lenguajes de cuarta generación. La percepción de "que podemos hacer" crece con la experiencia, población de usuarios, y prueba y error. Los beneficios se incrementan cuando nuevos usuarios adoptan la implementación de programas.

Los usuarios y gerentes pueden inicialmente no agarrar todas las ramificaciones del sistema, pero se hacen mejoras dinámicamente cuando el desarrollo del código es automatizado.

Tales resultados no pueden ser archivados con el viejo tipo de análisis formal de sistemas por especificaciones, que inherentemente implican un bloqueo en requerimientos (de otra manera el proyecto podría nunca terminar), codificación manual, documentación manual, y actualización manual. No pueden ser alargados con software preempacado.

Tabla 2.6 : Cuadro conceptual para lenguajes de cuarta generación

Análisis de sistemas y programación clásicos	Lenguajes de cuarta generación
Las especificaciones formales de la aplicación deben ser creadas	Los usuarios no necesitan definir lo que necesitan hasta que tengan una versión de la aplicación para intentarlo (prototipo)
El desarrollo de la aplicación usualmente requiere meses y en ocasiones años	El tiempo de desarrollo de la aplicación toma días o a lo más semanas
Se requieren conocimientos profesionistas de proceso de datos	Los usuarios pueden crear sus propias aplicaciones
	Los analistas usan lenguajes de cuarta generación para implementaciones sofisticadas
	No se necesita personal de apoyo, clásico en programación
Con lenguajes de procedimiento, el programador debe especificar "como debe de ejecutarse" una tarea	El usuario especifica que debe de hacerse, por indicaciones de un lenguaje no procedural
	El generador crea la aplicación
Ambos, análisis de sistemas y programas son documentados formalmente a través de métodos manuales	Los lenguajes de cuarta generación son autodocumentables, a través de la computadora
El trabajo de mantenimiento es tedioso y consume tiempo; absorve una larga porción del personal de proceso de datos	El trabajo de mantenimiento clásico practicamente desaparece
	El nuevo enfoque es redefinir "que hacer" para los lenguajes de cuarta generación

Un nuevo tipo de profesionistas en computación y comunicaciones seran natos con el uso de lenguajes de cuarta y quinta generación. Estos nuevos profesionistas harán un trabajo mucho mejor si memorizan los siguientes 12 factores clave en las nuevas tecnologias:

1. Los costos de mantenimiento deben ser estimados en la etapa de planeación del sistema; las técnicas escogidas deben permitir su reducción al mínimo.
2. El requerimiento de portabilidad debe ser examinado y entendido a lo largo del ciclo de vida del proyecto. Debemos empezar en el final deseado, no en el principio.
3. Cerca del 90% del esfuerzo de programacion debe ser en lenguajes de cuarta generación.
4. El impacto de la tecnologia de base de datos debe ser completamente entendida.
5. La capacidad de comunicación de datos debe ser proyectada desde el principio.
6. Desarrollar prototipos debe ser una parte integral del proyecto.
7. Los usuarios deben producir al principio una descripción verbal, y mejorarla paulatinamente.
8. No aplicar soluciones caseras a base de datos y comunicaciones. Solo enfoques profesionistas son aceptables, y esto significa sistemas manejadores de base de datos comodos, particularmente de tipo relacional.
9. La representación gráfica debe ser estandar, y esto es verdad para texto fácil y entrada de datos.
10. Ayuda en linea, "prompts", y explicaciones deben ser soportados. El medio ambiente debe ser seguro pero indulgente.
11. Cualquier programa, procedimiento, subsistema o sistema, debe ser sujeto a un control de calidad riguroso y seguro, incluyendo la robustez (fuerza) del programa y documentación.
12. Para asegurar calidad, se deben incluir examenes amplios del sistema para recursos distribuidos, redes, y personal de cómputo.

7).- Escogiendo un lenguaje portable.

Como buenas ideas, los lenguajes de programación buenos son elegantes en su simplicidad. La simplicidad debe ser expresada en 2 sentidos:

1. La interface hacia el usuario final.
2. Las caracteristicas de diseño propias del lenguaje.

Un buen criterio para la selección de un paquete de software es determinar si el paquete es enseñado en una universidad; otro es el tamaño de la población que lo ocupa. No todas las aplicaciones serán cubiertas por paquetes, y la elección de un lenguaje es por tanto necesaria.

Los lenguajes de cuarta generación presentan otras 2 caracteristicas, que representan referencias clave para el futuro:

3. Siendo flexibles y accesibles(open-ended), los lenguajes de cuarta generación hacen posible avanzar con la evolución tecnológica.
4. La necesidad de portabilidad de programas.

Porque en la rápida fase de cambio, la portabilidad de programas es uno de los tópicos clave más interesantes. La portabilidad significa la evasión de boletines de software y es, sobre todo, una filosofía de usar los recursos disponibles.

En politicas y procedimientos que gobiernan el desarrollo de programas, es necesario archivar la portabilidad, pero los problemas técnicos también deben ser observados.

Para que un programa sea transportable:

1. El lenguaje utilizado debe estar expresado en dialectos diferentes.
2. Los compiladores utilizados deben ser homogeneos.
3. El sistema operativo debe ser el mismo(por ejemplo UNIX).
4. El microprocesador debe ser el mismo (por ejemplo el Motorola 68000 o Intel 80186).

Un ejemplo de portabilidad a nivel de sistemas es a través de redes de PC's. Si LAN son observados y las reglas procedurales disponibles (archivo, servidor, piping, buzones) son honrados, los programas tienen una buena oportinidad de ser portables. En la figura 2.7 se muestra el caso en que 2 programas no son portables.

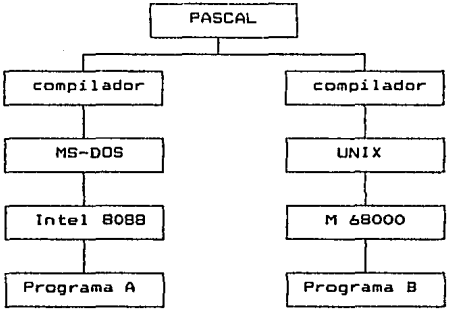


Figura 2.7 : No portabilidad de programas.

Notese que el lenguaje puede tener diferencias a nivel de diseño (diferentes dialectos), o por causa de los compiladores. Sin embargo, si es necesario transportar programas entre dos sistemas operativos y/o equipos diferentes, podemos adoptar los siguientes caminos:

1. Encontrar un camino común para accesar los 2 sistemas operativos antes de programar.
2. Escribir las rutinas de tránsito necesarias para la interface entre los sistemas operativos.
3. Correr un simulador de un sistema operativo en el otro. Esto reduce la eficiencia en una decima parte.

El mejor aviso que puede darse en vistas de la portabilidad de programas, es estar cerca de la industria estandar. La portabilidad de un programa esta intimamente ligada con la portabilidad de la máquina. Lo último es un prerequisito para lo anterior, este envuelve 9 caracteristicas; las más criticas son el microprocesador, el sistema operativo, y el lenguaje de programación.

1. El microprocesador (compatibilidad, estandar).
2. El uso de ROM (no se deben llamar a funciones en ROM).
3. La interface con la CRT (monitor).
4. El teclado.
5. El bus de Entrada/Salida.
6. Otras caracteristicas del hardware:
   - extensión de signo
   - orden en que los bits se ponen en la palabra
   - número de registros variables que se pueden manejar
   - el manejo de campos, palabras y caracteres
7. El sistema operativo y compatibilidad en interface genérica.
8. Compatibilidad de archivos de datos.
   - drives de 5¼", 48 TPI (tracks por pulgada)
9. Lenguaje de programación (no dialectos).

En conclusión, vemos que la compatibilidad y portabilidad de programas es un problema muy complejo que depende de varios hechos que no pasa por nuestra mente.

Las caracteristicas 1,6, y 8 son opciones de comodidad, pero las otras pueden ser patentadas. Un equipo que es exactamente igual a otro, en este aspecto, será una copia.

8).- Lenguajes para desarrollo eficiente de programas y potabilidad.

Como cualquier otro tema de computación, los lenguajes de programación tienen que someterse a desarrollos significativos, que son expresados en generaciones. A diferencia de otros temas, sin embargo, gran parte de desarrollos fueron concentrados en la primera decada de la era de la computación. No obstante, hoy hablamos de los lenguajes de cuarta generación, frecuentemente fallamos al realizar lo que las otras 3 generaciones hicieron en

los 50's. Esto ha tenido impacto en la sofisticación de programas y en el costo de la programación.

Los desarrollos en lenguajes de programación son necesarios para ahorrar tiempo en instruir a la máquina sobre las funciones que se deben realizar. Esto es igualmente cierto en construcción y mantenimiento de base de datos accesibles para el lenguaje de la computadora. En principio, las máquinas no deben ser programadas por el camino usado anteriormente en los lenguajes de alto nivel (Cobol, Fortran, etc.). Los paquetes son una buena solución desde mediados de los 70's, ahora la mejor respuesta esta dada por los lenguajes de cuarta generación.

Si un esfuerzo integrado es hecho con la iniciativa correcta, programando en lenguajes de cuarta generación obtendremos ganancias significantes en productividad. Las mejoras en la productividad del software pueden tener un largo y adecuado pago , provisto por acción sostenida y compromiso. En la larga carrera, la mayor ganancia en productividad vendra para un entendimiento más fundamental de la comunicación hombre-máquina. Tenemos una comprensión incompleta en este problema, pero estadísticas disponibles indican que, con lenguajes de cuarta generación, la productividad de programadores puede mejorar entre un 600 y 6000 porciento. Esto es válido para todos los tipos de computadoras, aunque este impacto sera sentido principalmente en estaciones de trabajo. Todavia, algunas herramientas solas pueden ser de valor limitado si otros factores no contribuyen a la meta de productividad, que tiene 4 componentes:

1. Motivación.
2. Aprendizaje efectivo de la nueva metodologia.
3. Herramientas basadas en la computadora.
4. Soportar estaciones de trabajo en cada escritorio de programador.

Es relativamente fácil para los programadores aprender a usar las nuevas herramientas de programación (si ellos lo quieren hacer). Por usar las caracteristicas de lenguajes de cuarta generación, los desarrolladores pueden producir miles de lineas de código en poco tiempo. Con mejores herramientas de desarrollo tambin se mejora la motivación. El objeto de un lenguaje de cuarta generación es dar un sistema de programación rápido, versátil, y organizado inteligentemente capaz de ahorrarle tiempo

al programador, eliminando la necesidad de codificación en bajo nivel, dando fácilidades de examinar, asegurar la documentación del programa, y actualizar sta cuando el programa sea modificado.

Esta observación es la más importante que puede aparecer a primera vista; En el lenguaje usamos condiciones mentales, esto es cierto tanto en un lenguaje de programación como en un lenguaje natural.

III.- ANALISIS DEL SISTEMA GENERADOR DE SISTEMAS.

1).- Antecedentes.

Varios han sido los intentos por desarrollar un sistema generador de sistemas, este es uno más entre tantos, que pretenden facilitar las tareas de programación de sistemas.

El mejor intento que se puede mencionar, son los lenguajes de cuarta generación que con solo decirles (en algunos casos) "que hacer", no hay necesidad de "programar" algun algoritmo que realize la tarea deseada.

Dentro del área de microcomputadoras, existen también otros intentos como:

- View-Gen.
- Flashcode.
- Foxbase.
- Quickcode, Quickreport.
- Genifer.

Estos pretenden que en base a algunos atributos y características que el usuario le indica al sistema, se generen programas en código DBASE que cumplan con la función deseada. Estos tiene una gran desventaja, pues para un programa de mantenimiento a un archivo de dos datos, genera un código DBASE tan extenso que reduciria nuestro potencial de almacenamiento hasta en 10 veces.

El sistema generador de sistemas pretende conjuntar parte de todos estos intentos, tomando ideas de ellos y tratando de optimizar el código general. Tambien pretende facilitar la tarea de programación, solicitando para ello algunas especificaciones de funcionamiento y descripción de pantalla, generando en base a esto el programa que efectue la función correspondiente.

2).- Caracteristicas y alcances del Sistema Generador de Sistemas.

Generalmente, una aplicación de este tipo no puede incluir todas las facilidades y opciones para el desarrollo de sistemas, es decir, no podemos cubrir en un 100% el desarrollo de un sistema. Normalmente, con Lenguajes de cuarta generación podemos desarrollar el 90% de una aplicación; el 10% restante, lo podemos desarrollar en nuestro lenguaje favorito. Esto se ha vuelto una regla, por lo cual el Sistema Generador de Sistemas deberá preveer esta situación, permitiendo el uso de rutinas en otro lenguaje.

El Sistema Generador de Sistemas, esta direccionado al campo de las microcomputadoras, razón por la cual, no debemos pretender cubrir todas las alternativas de proceso, ni tampoco un buen potencial de desarrollo.

El Sistema Generador de Sistemas debe permitir el desarrollo de sistemas en una forma amigable con el usuario, fácil de comprender, sin demasiadas restricciones, y debe generar los programas de tamaño compacto.

Se debe permitir el manejo del Sistema Generador de Sistemas en equipos que tengan configuración minima de memoria(256 Kb) y dos unidades de disco, pues es la configuración más común para los usuarios pequeños. Con menor frecuencia, existen usuarios con computadoras que solo tienen una unidad de discos, por lo cual, debemos preveer ambas situaciones.

Por el enfoque hacia las microcomputadoras que se le dio al Sistema Generador de Sistemas, no es conveniente considerar el manejo de varias compañias, ni sistemas. Solo debemos permitir el manejo de los datos generales de una compañia, asi como tambien los datos del sistema a desarrollar.

Una caracteristica que daria buena imagen al Sistema Generador de Sistemas, seria el manejo de menús encimados(pop-up menús) pues se estaria utilizando tecnologia más actual de desarrollo de software, además de la flexibilidad que se le da al usuario, puesto que, este manejo de menús, muestran toda la trayectoria de opciones escogidas por el usuario para llegar a la opción actual. Otra ventaja importante de este manejo de menús, es la sencillez

con la que el usuario puede elegir la opción a ejecutar, pues solo requiere utilizar las teclas de movimiento(flechas), o bien, alguna letra capital que identifique a la opción deseada.

En cuanto a la organización de archivos, es necesario "no inventar el hilo negro". Ya existen manejadores de archivos o bases de datos eficientes, el Sistema Generador de Sistemas deberá ser compatible con estas tecnologías, o bien utilizar alguna de ellas. Sería engorroso desarrollar un manejador de archivos con una filosofía nueva, pues además de su difícil programación, podría surgir alguna acción de rechazo hacia el. Es por esto que, resulta conveniente seleccionar bien el manejador de archivos que utilizaremos para el Sistema Generador de Sistemas, pues esto afecta directamente en la eficiencia y adecuación a estandares de software en microcomputadoras.

El usuario podrá indicar al Sistema Generador de Sistemas, los menús, archivos, funciones, reportes, y otros atributos que deberá tener el sistema deseado. Posterior a esto, se deberá generar el código de los programas, así como todo lo necesario para su funcionamiento. Una vez hecho esto, el usuario realizará pruebas del sistema, es natural que alguno de los programas, no realize la función adecuada, razón por la cual, el usuario tendrá la necesidad de corregir esta función y no requerirá generar nuevamente el sistema en su totalidad. Lo anterior nos obliga a contemplar la generación de parcialidades del sistema, queriendo decir con esto, menús, archivos, reportes, funciones, etc..

Algo muy importante que hay que considerar, es el manejo de seguridades. Se debe contemplar que usuarios sin acceso o desconocidos, solo puedan efectuar las funciones permitidas para ellos.

Todas las pantallas, ya sean menús o de procesos, deberán contemplar datos importantes como:

- Nombre de la compañía.
- Nombre del sistema.
- Fecha.
- Hora.
- Nombre del menú.
- nombre de la función, etc.

Para los archivos, deben permitir el almacenamiento de datos de tipo:

- Numéricos(reales y enteros).
- Caracteres.
- Lógicos.
- Fechas.

El tipo de acceso conveniente para los archivos es:

- Directo
- Secuencial
- Indexado por llave única(Maestro)
- Indexado por llave repetida(Detalle)

Las funciones son algo muy importante en el sistema, por lo cual debemos permitir todo tipo de libertades de proceso, tales como:

- Manejo de más de un archivo.
- Manejo de pantallas.
  - Datos generales
  - Datos repetitivos
- Validaciones de datos.
  - Opcionales y requeridos
  - LLaves
  - Chequeo de existencia en otros archivos
  - Rango de valores
  - Asignacion de valores
  - Despliegue
- Mensajes de error.
- Corrección de datos.

En cuanto a reportes, estos deben permitir el uso de:

- Encabezados y Pies.
  - Al inicio del reporte
  - De página
  - A un corte de control(sort)
  - Al final del reporte.
- Formato de etiquetas.
- Funciones estandar como fecha y hora.

3).- Elección del lenguaje de desarrollo.

El Sistema Generador de Sistemas por su naturaleza, debe incluir un gran número de opciones que faciliten la tarea del desarrollo de sistemas, por este motivo, la elección del lenguaje de desarrollo es de trascendental importancia para ello. A continuación se presenta el análisis que se llevo a cabo para la elección del lenguaje mas adecuado para este desarrollo:

Lenguajes considerados:

a) Lenguajes tradicionales(3ª generación):
   - TURBO PASCAL
   - TURBO BASIC
   - COBOL

b) Lenguajes mnemonicos:
   - C
   - Ensamblador

c) Lenguajes de inteligencia artificial:
   - PROLOG
   - LISP

d) Lenguajes de cuarta generación(3 ½ª).
   - DBASE, CLIPPER

Desventajas:

En las tres primeras alternativas, el desarrollo del Sistema Generador de Sistemas resultaría difícil, pues se requiere de un conocimiento avanzado del lenguaje que se eligiera. Esto nos obligaría a un mayor esfuerzo, pues primeramente necesitaríamos aprender a programar a un buen nivel en el lenguaje elegido, y por otro lado, sería como un compromiso lograr una implementación buena y eficiente, para que nuestro esfuerzo fuera adecuadamente valorado, obligandonos con esto a:

- implementar manejo de Bases de datos o un buen manejo de archivos.
- Dar herramientas de ayuda a los usuarios.
- Documentar muy ampliamente los programas.
- Permitir un manejo muy versátil de ventanas.

En DBASE, la principal desventaja sería que, no se cuenta con la flexibilidad necesaria para implementar todas las opciones que debe contemplar el Sistema Generador de Sistemas. La forma de

atacar esto es utilizando rutinas en otro lenguaje que cumplan con dichas funciones.

Ventajas:

Para las tres primeras opciones, las principales ventajas son:

- El tiempo de ejecución es muy reducido, puede decirse que el Sistema Generador de Sistemas sería muy veloz.
- Manejo eficiente de memoria, pues tenemos un mayor control en la administración de esta.
- El espacio de almacenamiento es pequeño.

En cuanto que DBASE cuenta con las siguientes ventajas:

- Compatibilidad:
  Tendremos código y archivos totalmente compatibles con la mayoría de sistemas de PC's, gracias a que DBASE es el sistema de bases de datos universal en microcomputadoras.
- Implementación sencilla:
  Resulta mucho mas fácil y rápido programar en DBASE que en cualquiera de los lenguajes mencionados. Ademas se cuentan con librerías de usuario, utilizables en DBASE que facilitan un gran número de tareas.
- Tiempo de ejecución:
  En tiempo de ejecución, DBASE esta perdido respecto a los lenguajes de las opciones anteriores. Una alternativa para subsanar esto es compilar los programas con otro paquete, tambien muy popular conocido como CLIPPER. Este es más poderoso que DBASE, y nos genera una versión compilada de toda nuestra aplicación, dejandola como código ejecutable, por lo cual nuestros tiempos de ejecución pueden verse reducidos hasta en un 50%. En forma alterna, tenemos otro paquete muy similar a DBASE, pero más rápido, que nos brinda algunas funciones similares a CLIPPER, pero sin generar una versión ejecutable de nuestra aplicación, este es conocido como FOXBASE.

Del análisis anterior se desprende que el lenguaje más adecuado para el desarrollo del Sistema Generador de Sistemas es DBASE, mejorando el tiempo de ejecición en una versión compilada con CLIPPER, o bien utilizar FOXBASE como segunda alternativa.

4).- Introducción a DBASE III plus.

Es un sistema administrador de bases de datos que incorpora un lenguaje de programación de computadora, y con el, es posible realizar tanto una consulta de datos interactiva como un diseño e implementación de sistemas de información.

Es considerado como un lenguaje de muy alto nivel, debido a que integra instrucciones de lenguaje común en programación, que efectuan funciones poderosas que en otros lenguajes representarian largas secuencias de instrucciones.

Los commandos y funciones de DBASE pueden ser ejecutados en forma interactiva, donde en cada linea se representa un comando o función y se conoce como "linea de comando" con sus respectivos parámetros. Estas mismas líneas de comando pueden ser incluidas dentro de un archivo, el cual se conoce como programa o "archivo de comandos", el cual será ejecutado secuencialmente.

Las estructuras de programación con las que cuenta DBASE son:
- Secuenciales o de serie.
- Condicionales.
- Iterativas o de repetición.
- Procedimientos o estructuras.

Secuencial:

Todo programa en DBASE se ejecutara secuencialmente a menos que se encuentre alguna instrucción que le indique lo contrario, como las condicionales, las iteractivas y de procedimientos.

Condicionales:

Son utilizadas para ejecutar u omitir la ejecución de instrucciones de DBASE, en base a alguna condición o estado en que se encuentre el programa. Estas instrucciones son:

```
IF Condición THEN
   instrucciones cuando la
   condición es verdadera
ELSE
   instrucciones cuando la
   condición es falsa
ENDIF
```

```
DO CASE
CASE Condición 1
  instrucciones
CASE Condición 2
  istrucciones
    .
OTHERWISE
  instrucciones
ENDCASE
```

Iteractivas o de repetición:

Estan representadas por el DO-WHILE, el cual nos permite repetir una serie de instrucciones, cuando se cumple una condición, hasta que esta sea falsa.

Procedimientos o estructuras:

Permiten pasar el control del programa a otro archivo de comandos, o a un procedimiento.

DBASE permite almacenar información de los siguientes tipos:

- Estructurados.
  - . Caracteres
  - . Números
  - . Fechas
  - . Valores lógicos(.F.,.T.)
- No estructurados.
  - . Memo:

    Puede almacenar hasta 5000 caracteres por registro; realmente lo que sucede es que se crea un archivo nuevo, el cual contendrá esta información, y en DBASE únicamente se le hace referencia.

Se permite la ejecución de subrutinas externas en lenguaje ensamblador mediante los comandos LOAD y CALL.

En DBASE Programer's Utilities se incluye una serie de utilerias desarrolladas en ensamblador para facilitar al programador avanzado el desarrollo de sus aplicaciones, como por ejemplo:

- Control de puertos.
- Salvar pantallas para recuperación posterior.
- Atributos de pantallas.
- Generador de menús de barras.
- Protección de bases de datos.
- Otros.

Permite la interface con otros lenguajes como:

- Herramientas DBASE para "C".
- PASCAL.
- Libreria de gráficas(pays,barras).

Todas estas herramientas proporcionan al programador la posibilidad de explotar la información de DBASE gráficamente. Las gráficas clásicas de negocios que están incluidas en el paquete son:

- Pastel
- Barras
- Líneas
- Puntos
- Barras acumuladas
- etc.

También se pueden usar otras funciones para dibujar líneas, marcos, círculos, puntos y arcos.

IV.- DESARROLLO DEL SISTEMA GENERADOR DE SISTEMAS.

Las actividades realizadas para el desarrollo del Sistema Generador de Sistemas, fueron enfocadas al concepto de desarrollo de sistemas de cuarta generación. La aplicación de está tecnologia puede verse a lo largo de este capitulo, el cual se organiza de la siguiente manera:

1).- Proceso de instalación.
2).- Consideraciones de operación.
3).- Utilerías.
4).- Diagrama de funciones(figura 4.1).
5).- Diagrama de archivos(figura 4.2).
6).- Elementos de los archivos.
7).- Listados de los programas.
   . Pantalla
   . Validaciones
   . Programa

Posteriormente, se presenta un ejemplo de aplicación que consiste en un pequeño sistema para el control de pedidos, del cual se presenta:

8).- Implantación del sistema.
9).- Diagrama de funciones(figura 4.3).
10).- Diagrama de archivos(figura 4.4).
11).- Reportes generados.
12).- Listados de programas.
   . Pantalla
   . Validaciones
   . Programa

Nótese que en el ejemplo de aplicación no se presentan consideraciones de operación ni utilerías, esto es debido a que pueden ser las mismas enunciadas en los puntos 2 y 3 de la presentación del Sistema Generador de Sistemas.

1).- Proceso de instalación.

El Sistema Generador de Sistemas está diseñado para trabajar en dos configuraciones:

1. En PC's con dos unidades de disco.
2. En PC's con disco duro.

Para el primer caso es recomendable hacer una copia del diskette original del sistema.

Para la instalación en disco duro, basta con bajar todos los archivos contenidos en el diskette original al directorio SGS.

En ambos casos es necesario disponer de DBASE, puesto que el código proporcionado no se encuentra en versión compilada.

2).- Consideraciones de operación.

Una vez instalado el Sistema Generador de Sistemas, para su ejecución debemos efectuar los siguientes pasos:

- Correr DBASE.
- Dentro de DBASE, dar el comando "DO GSP".
- Proporcionar la clave de usuario "SGS ".
- Proporcionar el password del usuario "SGSO".

Después de estos pasos, será presentado el menú principal del sistema y se podrá iniciar el registro de nuestra aplicación.

En el Sistema Generador de Sistemas se implemento el manejo de menús encimados, en los cuales la opción actual se encuentra sobremarcada. La selección de opciones en estos menús es mediante las siguientes teclas:

-> Para cambiar a la opción primaria a la derecha de la opción actual.

<- Para cambiar a la opción primaria a la izquierda de la opción actual.

↑ Para resaltar la opción que esté arriba de la opción actual, en el caso de que sea la primera, se resaltará la última opción.

↓ Para resaltar la opción que esté abajo de la opción actual, en el caso de que sea la última, se resaltará la primera opción.

Cuando se desee ejecutar la opción que se encuentre resaltada, se deberá presionar la tecla return <↵, o bien la letra que esté remarcada en la opción deseada.

En el caso de que la opción escogida sea otro menú, este será encimado sobre la pantalla, permitiendo la visualización de la trayectoria seguida para llegar hasta esta opción.

Si la opción seleccionada corresponde a algún programa, este será ejecutado respetando el encabezado del sistema. Cuando se termina de ejecutar el programa, automáticamente se restauran los menús seleccionados hasta antes de la ejecución.

Para el caso de funciones, se implementó un manejo que permite efectuar todas las acciones usuales(alta, baja, modificación y consulta), sin necesidad de cambiar de pantalla. Aquí también se resalta la función a ejecutar, debiendo seleccionarse de la misma manera que en los menús.

Todos los campos de captura tienen un manejo similar al de DBASE, podemos movernos entre los campos con las siguientes teclas:

-> Para movernos un campo a la derecha.

<- Para regresarnos un campo a la izquierda.

Home o ** Nos mueve al inicio del campo en curso. En caso de ya estar en el inicio, se reinicia la acción que se este ejecutando.

End o * Nos mueve al final del campo en curso. En caso de ya estar en el final, se permite seleccionar nuevamente la acción a efectuar en la pantalla.

La tecla return es usada para aceptar el dato del campo <— en curso.

Por último, el Sistema Generador de Sistemas muestra mensajes que orientan al usuario para una mejor operación del sistema.

3).- Utilerias.

Acompañando al Sistema Generador de Sistemas se encuentran las siguientes utilerias:

1. Proceso de inicialización del sistema, el cual comprende el blanqueo de archivos y la creación de archivos de índice. Este proceso lo ejecuta el programa "GSINSTAL".

2. Programa para reformatear programas, el cual transforma las instrucciones más usadas en DBASE a mayúsculas, dejando nombres de variables en minúsculas. Este proceso lo ejecuta el programa "GSMAYUSC".

Adicionalmente, dentro del sistema se cuenta con un menú de utilerias que pueden ser de gran utilidad en el diseño de su aplicación.

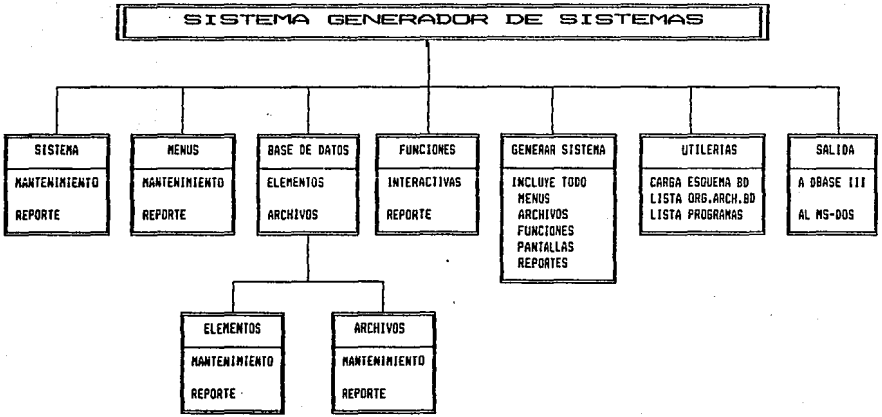


Figura 4.1 : Diagrama del Sistema Generador de Sistemas.

DIAGRAMA DE ARCHIVOS

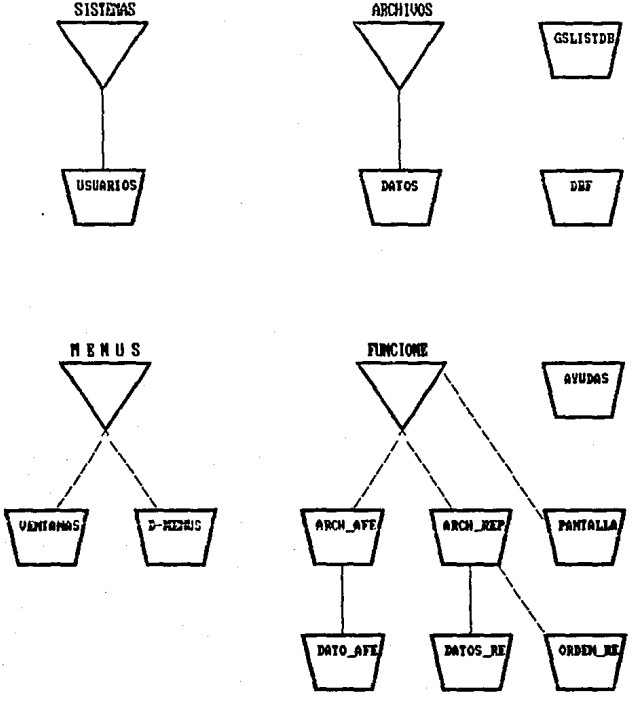


Figura 4.2 : Diagrama de archivos del Sistema Generador de Sistemas.

ELEMENTO TIPO LONG. DECIMALES MASCARA Y VALORES

ARCHIVO: ARCH_AFE

	ACCES_KEY	C	30	0
	ACCES_NDX	C	8	0
	ARCH_AFE	C	8	0
	CVE_FUNC	C	8	0
	LINK_KEY	C	30	0
	LINK_NDX	C	8	0
	TIPO_AFE	C	2	0
	* TOTAL DE ELEMENTOS: 7			

ARCHIVO: AYUDAS

	CVE_AYUDA	C	10	0
	TEXTO	M	10	0
	TIPO	C	1	0
	* TOTAL DE ELEMENTOS: 3			

ARCHIVO: DATOS

	FIELD_DEC	N	2	0
	FIELD_LEN	N	2	0
	FIELD_NAME	C	10	0
	FIELD_TYPE	C	1	0
	FILENAME	C	8	0
	* TOTAL DE ELEMENTOS: 5			

ARCHIVO: DATOS_RE

	ARCH_AFE	C	8	0
	COLUMNA	N	2	0
	CVE_FUNC	C	8	0
	DATO_AFE	C	10	0
	FORMULA	C	30	0
	GRUPO	C	2	0
	MASCARA	C	30	0
	NIV_RESET	N	1	0
	NIV_TOTAL	N	1	0
	RENGLON	N	2	0
	TIPO_AFE	C	1	0
	* TOTAL DE ELEMENTOS:11			

ARCHIVO: DATO_AFE

	AFECTACION	C	1	0
	ARCH_AFE	C	8	0
	CAPTURA	C	1	0
	COLUMNA	N	2	0
	CVE_FUNC	C	8	0
	DATO_AFE	C	10	0
	ETIQUETA	C	20	0
	FILTRO	C	8	0
	LONG_VAR	N	2	0

ELEMENTO	TIPO	LONG.	DECIMALES	MASCARA Y VALORES
ARCHIVO: DATO_AFE				
LOOKUP_ON	C	1	0	
LOOK_FILE	C	8	0	
MASCARA	C	25	0	
ORDEN	C	2	0	
PROCESO	C	1	0	
RENGLON	N	2	0	
TIPO_VAR	C	1	0	
VALOR	C	30	0	
* TOTAL DE ELEMENTOS: 8				
ARCHIVO: DBF				
FIELD_DEC	N	2	0	
FIELD_LEN	N	2	0	
FIELD_TYPE	C	1	0	
FILENAME	C	10	0	
* TOTAL DE ELEMENTOS: 4				
ARCHIVO: D_MENUS				
CVE_FUNC	C	8	0	
CVE_MENU	C	8	0	
* TOTAL DE ELEMENTOS: 2				
ARCHIVO: FUNCIONE				
CVE_FUNC	C	8	0	
SEGURIDAD	N	2	0	
TIPO_FUNC	C	2	0	
* TOTAL DE ELEMENTOS: 3				
ARCHIVO: MENUS				
COLUMNA	N	2	0	
CVE_FUNC	C	8	0	
CVE_MENU	C	8	0	
NOMB_FUNC	C	30	0	
RENGLON	N	2	0	
RET_CODE	C	1	0	
TIPO_FUNC	C	1	0	
* TOTAL DE ELEMENTOS: 7				
ARCHIVO: ORDEN_RE				
CVE_FUNC	C	8	0	
NIV_CORTE	C	2	0	
VARIABLE	C	30	0	
* TOTAL DE ELEMENTOS: 3				
ARCHIVO: PANTALLA				
ATRIBUTO	C	2	0	
COLUMNA	N	2	0	

ELEMENTO	TIPO	LONG.	DECIMALES	MASCARA Y VALORES
ARCHIVO: PANTALLA				
CVE_FUNC	C	8	0	
RENGLON	N	2	0	
TEXTO	C	80	0	
* TOTAL DE ELEMENTOS: 3				
ARCHIVO: SISTEMAS				
ACCESOS	N	10	0	
CARATULA	C	8	0	
CVE_MENU	C	8	0	
CVE_SIST	C	2	0	
FECHA_FIN	D	8	0	
FECHA_INI	D	8	0	
FECHA_INST	D	8	0	
NOMB_CIA	C	60	0	
NOMB_SIST	C	50	0	
SUBCARATUL	C	8	0	
TRAYECTO	C	20	0	
* TOTAL DE ELEMENTOS:11				
ARCHIVO: USUARIOS				
CVE_SIST	C	2	0	
NIV_ACCESO	N	2	0	
NOMB_USUA	C	30	0	
PASWORD	C	4	0	
USUARIO	C	4	0	
* TOTAL DE ELEMENTOS: 5				
ARCHIVO: VENTANAS				
COLUMNA1	N	2	0	
COLUMNA2	N	2	0	
CVE_FUNC	C	8	0	
RENGLON1	N	2	0	
RENGLON2	N	2	0	
TIPO_VENT	C	1	0	
* TOTAL DE ELEMENTOS: 6				

7).- Listados de programas.

```
*********************************************************
*            Sistema Generador de Sistemas              *
*            ejecutador del menu principal              *
* programa: gsp.prg                  autor: SGS/ENE-88 *
*********************************************************
SET echo off
SET talk off
SET print off
SET device TO screen
CLEAR ALL
CLOSE ALL
@ 0,0 CLEAR
SET PROCEDURE TO gsr
PUBLIC teclado,msg,rb,cb,wbox,max_func,wf,niv_gs,gs_rb
PUBLIC opcion,menu,sub_menu,prog_exe,wcolor,gs_device
STORE SPACE(16) TO prog_exe
STORE SPACE(40) TO menu,sub_menu
msg=SPACE(80)
niv_gs=SPACE(10)
gs_usuario="12  "
gs_sist="  "
gs_cia="  "
gsniv_seg=0
DO usuarios WITH gs_usuario,gs_sist,gs_cia,gsniv_seg
niv_gs=""
gs_rb=7
gs_device="PROW()"
gs_siglas="GS"
gs_ncia="UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO"
gs_nsuc="FACULTAD DE INGENIERIA"
gs_nsist="SISTEMA GENERADOR DE SISTEMAS"
* definicion de ventanas de menus
*  1,2 renglon1       3,2 columna1
*  5,2 renglon2       7,2 columna2
*  9,2 # funciones   11,1 borde
box ="03000579071S  W+/W"
box1="07001316021D  W+/W"
box2="07051321021D  W+/W"
box3="07181330021D  W+/W"
box4="07301345021D  W+/W"
 box41="10351660051D  W+/W"
 box42="12351760041D  W+/W"
```

```
box5="07441459061D  W+/W"
 box51="10501867031D  W+/W"
box6="07551275041D  W+/W"
box7="07671379021D  W+/W"
boxw="00000000031D14W+/W"
boxx="00000000031D15W+/W"

* definicion de funciones de menus
f1 ="0102MS         SISTEMA"
f2 ="0111MM         MENUS"
f3 ="0118MB         BASE-DE-DATOS"
f4 ="0133MF         FUNCIONES"
f5 ="0144MG         GENERAR-SISTEMA"
f6 ="0161MU         UTILERIAS"
f7 ="0172MA         SALIDA"

f11="0202FM         MANTENIMIENTO"
f12="0402FR         REPORTE"

f21="0202FM         MANTENIMIENTO"
f22="0402FR         REPORTE"

f31="0202WE         ELEMENTOS"
f32="0402WA         ARCHIVOS"

f41="0202FI         INTERACTIVAS"
f42="0402FR         REPORTES"

f51="0102FI         INCLUYE TODO"
f52="0204FM         MENUS"
f53="0304FA         ARCHIVOS"
f54="0404FF         FUNCIONES"
f55="0504FP         PANTALLAS"
f56="0604FR         REPORTES"

 f511="0202FD         DBASE III PLUS"
 f512="0402FC         CLIPPER"
 f513="0602RF         FIN"


f61="0102FC         CARGA ESQUEMA BD"
f62="0202FO         LISTA ORG.ARCH.BD"
f63="0302FP         LISTA PROGRAMAS"
f64="0402RF         FIN"
```

```
f71="0202sD         DBASE III"
c71="'-1' TO niv_gs"
f72="0402SA         AL MS-DOS"
c72="'-2' TO niv_gs"

fw1="0102FM         MANTENIMIENTO"
fw2="0202FR         REPORTES"
fw3="0302RF         FIN"

fx1="0102FA         ALTAS"
fx2="0202FB         BAJAS"
fx3="0302FM         MODIFICACIONES"
fx4="0402FC         CONSULTAS"
fx5="0502FR         REPORTES"
fx6="0602RF         FIN"

*falta fy

*inician instrucciones del programa principal
wniv_gs=SPACE(10)
wopcion=0
STORE SPACE(80) TO ret_code,ret_code0
teclado=" 19 4 5 24 13 28"
tecla=0
opcion0=1
max_func0=val(SUBS(box,9,2))
opcion=1
load savescr
@0,0 CLEAR
DO caratula
DO menus WITH ret_code0
SGS=" SISTEMA GENERADOR DE SISTEMAS "
@3,24 GET SGS
CLEAR GETS
opcion=0
niv_gs="1"
DO WHILE niv_gs>="0"
  IF opcion=0
    DO menus WITH ret_code
  ENDIF
  DO espera WITH teclado,tecla,ret_code,ret_code0
  DO accion WITH niv_gs,tecla,opcion0,opcion,boxx,
                 ret_code,ret_code0
```

```
* Ejecuta programas permitiendo reasignacion de PROCEDURES
  IF prog_exe<>SPACE(16)
   call savescr WITH "SO"
   SET safety off
   save TO gsvariab
   SET safety on
   RELEASE ALL except prog_exe,nomb,gs_rb,gs*,gs_device
   DO &prog_exe
   CLEAR MEMORY
   CLOSE DATABASE
   PUBLIC teclado,msg,rb,cb,wbox,max_func,wf,niv_gs,gs_rb
   PUBLIC opcion,menu,sub_menu,prog_exe,wcolor,gs_device
   reSTORE FROM gsvariab additive
   @ 0,0 CLEAR
   call savescr WITH "RO"
   STORE SPACE(16) TO prog_exe
  ENDIF
ENDDO
CLOSE DATABASE
@ 0,0 CLEAR
IF niv_gs="-2"
 QUIT
ENDIF
CLEAR ALL
SET talk on
SET echo on
* fin del programa principal
```


ESTA TESIS NO DEBE
SALIR DE LA BIBLIOTECA

```
*********************************************************
*            Sistema Generador de Sistemas              *
*          procedimientos generales del SGS             *
* programa: gsr.prg                  autor: SGS/ENE-88 *
*********************************************************
PROCEDURE usuarios
PARAMETERS gs_usuario,gs_sistema,gs_cia,gsniv_seg
STORE "     " TO gs_password,gs_usuario
gs_sistema="  "
gs_cia="  "
intentos=1
USE usuarios INDEX usuarios
DO WHILE .T.
 @14,5 SAY "USUARIO : " GET gs_usuario
 READ
 SEEK gs_usuario
IF FOUND()
  @16,5 SAY "PASSWORD: " GET gs_password
  CLEAR GETS
  c=1
  DO WHILE .T.
   t=INKEY()
   IF t<>0
    IF .NOT.(T=13 .OR. t=6 .or. c=4)
     gs_password=STUFF(gs_password,c,1,CHR(t))
    ELSE
     EXIT
    ENDIF
    c=c+1
   ENDIF
  ENDDO
  IF pasword=gs_password
   gsniv_seg=niv_acceso
   @0,0 CLEAR
   EXIT
  ELSE
   SET message TO "Password incorrecto, intente de nuevo"
  ENDIF
 ELSE
  SET message TO "Clave de usuario invalida, intente nuevamente"
 ENDIF
 IF intentos>3
  CLOSE DATABASE
  @0,0 CLEAR
```

```
  @0,0 SAY "ACCESO NO AUTORIZADO"
  CANCEL
 ENDIF
 intentos=intentos+1
ENDDO
USE
RETURN

PROCEDURE caratula
  @ 1,6 SAY dATE()
  @ 1,(80-LEN(TRIM(gs_ncia)))/2 SAY trim(gs_ncia)
  @ 1,65 SAY time()
  @ 2,(80-LEN(TRIM(gs_nsuc)))/2 SAY trim(gs_nsuc)
RETURN

PROCEDURE menus
PARAMETERS ret_code
  ret_code=""
  vbox="BOX"+RTRIM(niv_gs)
  wbox=&vbox
  wCOLOR=SUBS(wbox,15)
  DO ventana WITH wbox
  cont_func=0
  max_func=val(SUBSTR(wbox,9,2))
  DO WHILE cont_func<max_func
    cont_func=cont_func+1
    vf="F"+niv_gs+STR(cont_func,1)
    wvf=&vf
    r=rb+val(SUBSTR(wvf,1,2))
    c=cb+val(SUBSTR(wvf,3,2))
    @ r,c SAY RTRIM(SUBSTR(wvf,15))
    ret_code=RTRIM(ret_code)+" "+STR(asc(SUBSTR(wvf,6,1)),3)
    SET COLOR TO &wcolor
    c1=AT(SUBS(wvf,6,1),subs(wvf,15))-1
    @ r,c+c1 SAY SUBS(wvf,c1+15,1)
    SET COLOR TO
 ENDDO
 opcion=IIF(opcion=0,val(SUBS(wbox,11,1)),opcion)
 vf="F"+niv_gs+STR(opcion,1)
 DO marca_fu WITH &vf
 msg="00Seleccione opcion con letra nemonica o "+CHR(24)+" "+
     chr(25)+" "+chr(26)+" "+chr(27)+" y pulse "+chr(17)+
     chr(196)+chr(217)
RETURN
```

```
PROCEDURE ventana
PARAMETERS wbox
  rb=val(SUBSTR(wbox,1,2))
  cb=val(SUBSTR(wbox,3,2))
  r2=val(SUBSTR(wbox,5,2))
  c2=val(SUBSTR(wbox,7,2))
  doble=IIF(SUBS(wbox,12,1)="D","double","")
  SET COLOR TO &wcolor
  @ rb,cb CLEAR TO r2,c2
  @ rb,cb TO r2,c2 &doble
  SET COLOR TO
RETURN

PROCEDURE borra_ve
PARAMETERS wbox
  r1=val(SUBSTR(wbox,1,2))
  c1=val(SUBSTR(wbox,3,2))
  r2=val(SUBSTR(wbox,5,2))
  c2=val(SUBSTR(wbox,7,2))
  @ r1,c1 CLEAR TO r2,c2
RETURN

PROCEDURE marca_fu
PARAMETERS fx
  r=rb + val(SUBSTR(fx,1,2))
  c=cb + val(SUBSTR(fx,3,2))
  SET COLOR TO &wcolor
  @ r,c SAY RTRIM(SUBSTR(fx,15))
  SET COLOR TO
RETURN

PROCEDURE limpia_f
PARAMETERS fx
  r=rb + val(SUBSTR(fx,1,2))
  c=cb + val(SUBSTR(fx,3,2))
  SET COLOR TO
  @ r,c SAY RTRIM(SUBSTR(fx,15))
  SET COLOR TO &wcolor
  c1=AT(SUBS(fx,6,1),SUBS(fx,15))-1
  @ r,c+c1 SAY SUBS(fx,c1+15,1)
  SET COLOR TO
RETURN
```

```
PROCEDURE espera
PARAMETERS teclas,t,r_c,r_c0
 t=0
 codigos=teclas+r_c+r_c0
 SET message TO SUBS(msg,3)
 DO WHILE AT(" "+LTRIM(STR(t,3)),codigos)=0
  t=0
  SET escape off
  DO WHILE t=0
   @ 1,65 SAY time()
   t=INKEY()
  ENDDO
  SET escape on
 ENDDO
 SET message TO " "
RETURN

PROCEDURE accion
PARAMETER niv_gs,t,opcion0,opcion,boxx,r_c,r_c0
  vf0="F"+STR(opcion0,1)
  wf0=&vf0
  vf="F"+niv_gs+STR(opcion,1)
  wf=&vf
  DO CASE
   CASE t=19
     opcion0=IIF(opcion0=1,max_func0,opcion0 - 1)
     opcion=0
     @ val(SUBS(box,5,2))+1,0 CLEAR
     niv_gs=""
   CASE t=4
     opcion0=IIF(opcion0=max_func0,1,opcion0 + 1)
     opcion=0
     @ val(SUBS(box,5,2))+1,0 CLEAR
     niv_gs=""
   CASE t=5
     opcion=IIF(opcion=1,max_func,opcion - 1)
   CASE t=24
     opcion=IIF(opcion=max_func,1,opcion+1)
   CASE t=13
    DO accion13 WITH niv_gs,wniv_gs,opcion,r_c,boxx,teclado
    RETURN
   CASE t=28
*    DO help
   OTHERWISE
```

```
    IF AT(" "+STR(t,3),r_c)>0
     opcion=int((AT(" "+STR(t,3),r_c)-1)/4) + 1
     DO accion13 WITH niv_gs,wniv_gs,opcion,r_c,boxx,teclado
     RETURN
    ELSE
     IF AT(" "+STR(t,3),r_c0)>0
      opcion0=int((AT(" "+STR(t,3),r_c0)-1)/4) + 1
      opcion=0
      @ val(SUBS(box,5,2))+1,0 CLEAR
      niv_gs=""
     ENDIF
    ENDIF
   ENDCASE
   IF opcion>0
     DO limpia_f WITH wf
     vf="F"+niv_gs+STR(opcion,1)
     wf=&vf
     DO marca_fu WITH wf
   ELSE
    IF .NOT. niv_gs$"WXYZ"
     rb=val(SUBSTR(box,1,2))
     cb=val(SUBSTR(box,3,2))
     DO limpia_f WITH wf0
     vf0="F"+STR(opcion0,1)
     wf0=&vf0
     niv_gs=STR(opcion0,1)
     DO marca_fu WITH wf0
    ENDIF
   ENDIF
RETURN

PROCEDURE accion13
PARAM niv_gs,wniv_gs,opcion,r_c,boxx,teclado
IF t<>13
 IF opcion>0
  DO limpia_f WITH wf
  vf="F"+niv_gs+STR(opcion,1)
  wf=&vf
  DO marca_fu WITH wf
 ELSE
  IF .NOT. niv_gs$"WXYZ"
   rb=val(SUBSTR(box,1,2))
   cb=val(SUBSTR(box,3,2))
   DO limpia_f WITH wf0
```

```
   vf0="F"+STR(opcion0,1)
   wf0=&vf0
   niv_gs=STR(opcion0,1)
   DO marca_fu WITH wf0
  ENDIF
 ENDIF
ENDIF
tipo_f=SUBSTR(wf,5,1)
DO CASE
 CASE tipo_f="R"
  IF niv_gs$"WXYZ"
   niv_gs=wniv_gs
  ENDIF
  opcion=val(SUBS(niv_gs,LEN(niv_gs),1))
  niv_gs=SUBS(niv_gs,1,LEN(niv_gs)-1)
  teclado=" 19 4 5 24 13 28"
  DO borra_ve WITH wbox
  vbox="BOX"+RTRIM(niv_gs)
  wbox=&vbox
  DO menus WITH r_c
 CASE tipo_f="s"
  STORE '-1' TO niv_gs
 CASE tipo_f="S"
  STORE '-2' TO niv_gs
 CASE tipo_f="C"
  niv_gs=niv_gs+STR(opcion,1)
  comando=c&niv_gs
  niv_gs=SUBS(niv_gs,1,LEN(niv_gs)-1)
 CASE tipo_f$"WXYZ"
  r1=val(SUBSTR(wbox,1,2))+val(subs(wf,1,2))+1
  c=val(SUBSTR(wbox,3,2))+5
  c1=IIF(c>62,62,c)
  c=val(SUBS(box&tipo_f,9,2))+1
  cc=val(SUBS(box&tipo_f,13,2))+2
  STORE IIF(r1<10,"0"+STR(r1,1),str(r1,2))+iif(c1<10,"0"+
        str(c1,1),str(c1,2))+str(r1+c,2)+str(c1+cc,2)+
        SUBSTR(box&tipo_f,9) TO box&tipo_f
  wniv_gs=niv_gs+STR(opcion,1)
  niv_gs=tipo_f
  opcion=0
 CASE tipo_f="F"
  DO ejecuta WITH niv_gs,opcion
 OTHERWISE
  niv_gs=niv_gs+STR(opcion,1)
```

```
  opcion=0
ENDCASE
RETURN

PROCEDURE ejecuta
 PARAMETERS niv_gs,opcion
 prog_exe=IIF(niv_gs$"WXYZ",wniv_gs,niv_gs)+STR(opcion,1)
 prog_exe=gs_siglas+prog_exe+".PRG"
 IF file(prog_exe)
  IF gsniv_seg < val(SUBS(wf,7,2))
   msg="El usuario no tiene acceso a esta funcion"
   STORE SPACE(16) TO prog_exe
  ENDIF
 ELSE
  msg="Funcion no disponible en disco("+prog_exe+")"
  STORE SPACE(16) TO prog_exe
 ENDIF
RETURN
*********************************************************
*              PROCEDIMIENTOS funcionales               *
*********************************************************
*declaracion de variables PUBLICAS
PROCEDURE PUBLICAS
 cd=2
 wpub="W01"+SUBS(d01,AT(">",d01)+1,7)+SPACE(230)
 DO WHILE cd<=datos_tot
  cds=IIF(cd>9,STR(cd,2),"0"+str(cd,1))
  vd="D"+cds
  wpub=TRIM(wpub)+",W"+IIF(cd>9,STR(cd,2),"0"+
       STR(cd,1))+SUBS(&vd,AT(">",&vd)+1,7)
  cd=cd+1
 ENDDO
 PUBLIC &wpub
RETURN

PROCEDURE lee_datos
 PARAMETER dato_i,dato_f,rc
 cd=dato_i
 DO WHILE cd <= dato_f
  cds=IIF(cd>9,STR(cd,2),"0"+str(cd,1))
  vd="D"+cds
  DO captura WITH &vd
  IF .NOT. ok
   EXIT
```

```
  ENDIF
  cd=cd+1
 ENDDO
RETURN

PROCEDURE acepta_de
rc=IIF(rc<ri,ri-1,rc)
DO WHILE .T.
 IF .NOT. repite_det  .AND. rc=rf
  msg="Ojo Ya se llego al limite de registros a dar de alta"
  EXIT
 ENDIF
 rc=IIF(rc=rf,ri,rc+1)
 DO lee_datos WITH datos_gen+1,datos_tot,rc
 IF ok
  IF accion="A"
   DO alta WITH arch_det
   DO asigna WITH datos_gen+1,datos_tot,.F.
   IF LEN(audit_det)<>0
    DO alta WITH audit_det
    DO asigna WITH datos_gen+1,datos_audd,.F.
   ENDIF
  ENDIF
 ELSE
  EXIT
 ENDIF
ENDDO
RETURN

PROCEDURE captura
PARAMETERS td
long=val(SUBS(td,31,2))
pp=AT("-",td)+2
decl=SUBS(td,23,1)
capt=SUBS(td,24,1)
tipo=SUBS(td,25,1)
look=SUBS(td,26,1)
vvar=TRIM(SUBS(td,3,20))
wvar="W"+cds+TRIM(SUBS(td,pp,7))
ok=.T.
DO CASE
CASE tipo="D"
 dvar=&vvar
 @ rc+val(SUBS(td,27,2)),val(subs(td,29,2))
```

```
      GET dvar PICTURE TRIM(subs(td,33,25))
 STORE &vvar TO &wvar
 CLEAR GETS
 RETURN
CASE UPPER(tipo)="C"
 vllave=SUBS(td,58,30)
 STORE &vllave TO &wvar
 IF tipo="c"
   dvar=&wvar
   @ rc+val(SUBS(td,27,2)),val(subs(td,29,2))
        GET dvar PICTURE TRIM(subs(td,33,25))
   CLEAR GETS
 ENDIF
CASE tipo="K" .AND. accion="m" .and. cds>"01"
 STORE &vvar TO &wvar
 RETURN
ENDCASE
IF look="V"
 vv="VALUES"+cds
 wvalues=&vv
 vrango=IIF(","$wvalues,"RANGE "+TRIM(wvalues),"")
 msg="00Los valores aceptables son:"+values&cds
ELSE
 vrango=""
ENDIF
var=SPACE(long)
vard=ctod('  /  /  ')
IF accion="m" .AND. cd>1 .and. SUBS(td,3,pp-3) <> "SGSWORK"
   .OR. capt+tipo$'OCRCOcRc'
 DO CASE
 CASE decl="N"
  STORE trans(&vvar,SUBS(td,33,25)) TO var
 CASE decl="D"
   STORE &vvar TO vard
 CASE decl="M"
 OTHERWISE
  STORE &vvar TO var
 ENDCASE
ENDIF
ok=.F.
DO WHILE .NOT. ok
 SET message TO SUBS(msg,3)
 rk=-231
 IF capt$'OR'
```

```
 IF LEN(vrango)>0
  IF accion="A"
   IF tipo$"Cc"
    vn=&wvar
   ELSE
    vn=0
   ENDIF
  ELSE
   vn=&vvar
  ENDIF
  @ rc+val(SUBS(td,27,2)),val(subs(td,29,2))
    GET vn PICTURE TRIM(subs(td,33,25)) &vrango
  READ
  rk=READKEY()
  var=transf(vn,TRIM(SUBS(td,33,25)))
 ELSE
  IF dec1='D'
   @ rc+val(SUBS(td,27,2)),val(subs(td,29,2))
       GET vard PICTURE TRIM(subs(td,33,25))
   READ
   rk=READKEY()
   STORE dtoc(vard) TO var
  ELSE
   @ rc+val(SUBS(td,27,2)),val(subs(td,29,2))
        GET var PICTURE TRIM(subs(td,33,25))
   READ
   rk=READKEY()
  ENDIF
 ENDIF
ENDIF
SET message TO " "
IF "**"$var  .OR. rk=258 .or. rk=2
 ok=.F.
 RETURN
ENDIF
IF "*"$var  .OR. rk=259 .or. rk=3
 ok=.F.
 accion="SF"
 RETURN
ENDIF
IF "<"$var .OR. rk=256  .or. rk=0
 cd=val(SUBS(td,1,2))
 cd=IIF(cd>dato_i,cD-1,dato_i)
 IF cd>1
```

```
  cds=IIF(cd>9,STR(cd,2),'0'+str(cd,1))
  STORE val(SUBS(d&cds,1,2))-1 TO cd
 ENDIF
 ok=.T.
 EXIT
ENDIF
arch=SUBS(td,3,pp-5)
IF capt$"OR"
 DO CASE
 CASE dec1="N"
  STORE val(var) TO &wvar
 CASE dec1="D"
   STORE vard TO &wvar
 OTHERWISE
  STORE var TO &wvar
  IF var="SGSWORK "
   ok=.T.
   msg="00Dar datos requeridos o SELECCIONE opcion"
   EXIT
  ENDIF
 ENDCASE
ENDIF
msg="00Dar datos o SELECCIONAR opcion"
IF SUBS(td,24,1)="R" .AND. var$SPACE(long)
  msg="01Es un dato requerido, debe dar algun valor"
ELSE
 IF look="V" .AND. LEN(vrango)=0
  IF .NOT. " "+TRIM(var)$wvalues
   msg="05Los valores aceptables son:"+values&cds+
       ". corriga por favor"
  ENDIF
 ENDIF
 IF SUBS(msg,1,2)="00"
  DO CASE
  CASE tipo="K"
   vllave=TRIM(SUBS(td,58,30))
   IF arch="SGSWORK"
    arch=SUBS(td,88,8)
   ENDIF
   SELECT &arch
   IF LEN(NDX(1))>0
    SEEK &vllave
   ELSE
    LOCATE for &vllave
```

```
   ENDIF
   IF FOUND()
    IF accion="A"
     msg="02La clave ya existe en el archivo"
    ENDIF
   ELSE
    IF accion<>"A"
     msg="03No existe la clave, intente otra vez"
    ENDIF
   ENDIF
  CASE tipo="L" .AND. capt$"rn"    .and. .NOT. accion+capt$"mn"
   vllave=TRIM(SUBS(td,58,30))
   arch=TRIM(SUBS(td,88,8))
   SELECT &arch
   wr1=RECNO()
   IF LEN(NDX(1)) > 0
    SEEK &vllave
   ELSE
    LOCATE for &vllave
   ENDIF
   IF FOUND()
    IF look="N"
     IF .NOT. (cd=1 .AND. accion<>"A")
      msg="04Ya existe en el archivo de busqueda"
     ENDIF
    ENDIF
   ELSE
    IF look="O"
     msg="05No existe en el archivo de busqueda"
    ENDIF
   ENDIF
   IF accion="M" .AND. wr1 <= reccount()
    GO wr1
   ENDIF
   IF SUBS(msg,1,2)<>"00" .AND. type('MSG&CDS')="C"
    msg=msg&cds
   ENDIF
  CASE tipo="C"
   vllave=TRIM(SUBS(td,58,30))
   STORE &vllave TO &wvar
  ENDCASE
 ENDIF
ENDIF
IF SUBS(msg,1,2)="00"
```

```
  ok=.T.
  msg="OODar datos requeridos o SELECCIONE opcion"
 ELSE
  ok=.F.
  IF capt$"N" .AND. tipo$"lk"
   SET mess TO SUBS(msg,3)
   cd=val(SUBS(td,1,2)) -1
   ok=.T.
  ENDIF
 ENDIF
ENDDO
RETURN

PROCEDURE despliega
PARAMETER dato_i,dato_f,rc
IF accion="M"
 DO asigna WITH dato_i,dato_f,.T.
ENDIF
cd=dato_i
DO WHILE cd <=dato_f
 cds=IIF(cd>9,STR(cd,2),"0"+str(cd,1))
 wd=d&cds
 tipo=SUBS(wd,25,1)
 IF tipo<>"C"
  vvar=TRIM(SUBS(wd,3,20))
  IF .NOT. "SGSWORK"$vvar
   @ rc+val(SUBS(wd,27,2)),val(subs(wd,29,2))
     GET &vvar PICTURE   TRIM(subs(wd,33,25))
  ENDIF
 ENDIF
 cd=cd+1
ENDDO
CLEAR GETS
RETURN

PROCEDURE despga_de
rc=ri
ok=.F.
DO WHILE .NOT.OK
 DO despliega WITH datos_gen+1,datos_tot,rc
 IF .NOT. EOF()
  SKIP
 ELSE
  rc=rC-1
```

```
 ENDIF
 IF rc=rf .OR. EOF()
  IF accion="B"
   ok=.T.
  ELSE
   DO opciones
  ENDIF
 ENDIF
 rc=IIF(rc=rf,ri,rc+1)
ENDDO
RETURN

PROCEDURE cambios
n_dato=1
msg="OO "
DO WHILE n_dato>0
 SET message TO SUBS(msg,3)
 @ 23,1 SAY "Dar el numero de dato a corregir(0 para
                finalizar): " GET n_dato PICTURE "99"
READ
 SET message TO " "
 IF n_dato>0
  IF n_dato<=id_tot
   IF n_dato<=id_gen
    msg="OODar el valor deseado"
    widi=val(SUBS(idgen,n_dato*5-3,2))
    widf=val(SUBS(idgen,n_dato*5-1,2))
    rc=rb
    IF type('PRIMARIO')="c"
     SELECT &primario
    ELSE
     SELECT 1
    ENDIF
    GO val(SUBS(wrec,1,7))
    DO lee_datos WITH widi,widf,rb
    IF .NOT. ok
     EXIT
    ELSE
     DO asigna WITH widi,widf,.F.
    ENDIF
   ELSE
    rc=n_dato - id_gen + ri -1
    ir=(rC-ri+1)*7
    IF SUBS(wrec,ir+1,7)<>"       "
```

```
      IF LEN(arch_det)>0
       arch=SUBS(arch_det,1,9)
       SELECT &arch
      ELSE
       SELECT 2
      ENDIF
      GO val(SUBS(wrec,ir+1,7))
      DO lee_datos WITH datos_gen+1,datos_tot,rc
      IF .NOT. ok
       EXIT
      ELSE
       DO asigna WITH datos_gen+1,datos_tot,.F.
      ENDIF
     ELSE
      msg="06Numero de dato fuera de rango o registro inexistente"
     ENDIF
    ENDIF
   ELSE
    msg="06Numero de dato fuera de rango"
   ENDIF
  ENDIF
ENDDO
accion="M"
RETURN

PROCEDURE asigna
PARAMETERS dato_i,dato_f,lect
cd=dato_i
wrep=SPACE(254)
wrep=""
ir=IIF(rc<ri,0,(rC-ri+1)*7)
IF lect
 wrec=STUFF(wrec,1+ir,7,STR(RECNO(),7))
ENDIF
DO WHILE cd<=dato_f
 cds=IIF(cd>9,STR(cd,2),"0"+str(cd,1))
 wd=d&cds
  decl=SUBS(wd,23,1)
  IF decl<>"M"
   vvar=TRIM(SUBS(wd,3,20))
   archivo1=SUBS(vvar,1,AT("-",vvar)-1)
   wvar="W"+cds+SUBS(vvar,AT("-",vvar)+2,7)
   IF archivo1<>"SGSWORK"
    IF lect .OR.(SUBS(WD,25,1)="K" .AND. accion="m")
```

```
      STORE &vvar TO &wvar
     ELSE
      wrep=wrep+vvar+" WITH "+wvar+","
      IF mod(cD-dato_i,6)=0 .OR. cd=dato_f
       wrep=SUBS(wrep,1,LEN(wrep)-1)
       REPL &wrep
       wrep=""
      ENDIF
     ENDIF
    ELSE
     IF  cd=dATO_F .AND. .NOT. LECT .AND. LEN(wrep)>7
      wrep=SUBS(wrep,1,LEN(wrep)-1)
      REPL &wrep
      wrep=""
     ENDIF
    ENDIF                                          \
   ENDIF
  cd=cd+1
ENDDO
RETURN
PROC alta
PARAM archivos
1=1
DO WHILE LEN(archivos)>=1*9
 arch=SUBS(archivos,1*9-8,9)
 SELE &arch
 APPE BLANK
 1=1+1
ENDDO
RETURN
PROCEDURE baja
PARAM primario
DO baja_det WITH arch_det,0
1=LEN(arch_gen)/9
SELE &primario
GO val(SUBS(wrec,1,7))
DO WHILE 1>0
 arch=SUBS(arch_gen,1*9-8,9)
 SELE &arch
 DELETE
 PACK
 1=L-1
ENDDO
RETURN
```

```
PROC baja_det
PARAM archivos,reg
arch=SUBS(archivos,1,9)
SELE &arch
IF reg=0
 GO TOP
ELSE
 GO reg
ENDIF
DO WHILE .NOT. EOF()
 l=LEN(archivos)/9
 DO WHILE l>1
  arch1=SUBS(archivos,l*9-8,9)
  SELE &arch1
  DELETE
  l=L-1
 ENDDO
 SELE &arch
 DELETE
 IF reg=0
  SKIP
 ELSE
  EXIT
 ENDIF
ENDDO
DO empaca WITH archivos
RETURN

PROC empaca
PARAM archivos
l=1
DO WHILE LEN(archivos)>=l*9
 arch=SUBS(archivos,l*9-8,9)
 SELE &arch
 PACK
 l=l+1
ENDDO
RETURN

PROC opc_abmc
PARAM opc
opciones="Alta  Baja  Modifica  Consulta  Fin"
tab="A01B07M13C23F33"
```

```
acciones="ABMCF"
accion=" "
wrec=SPACE((rf-ri+2)*7)
rc=rb
DO opc_abmch WITH rb,tab,opciones,opc,acciones,accion
RETURN

PROC opc_abmch
PARAM rb,tabs,opciones,opc,acciones,accion
tec="  19   4  13"
t=1
max_fun=LEN(tabs)/3
rc1=""
DO WHILE t<LEN(tabs)
 rc1=rc1+STR(asc(SUBS(tabs,t,1)),4)
 t=t+3
ENDDO
t=0
msg="00Seleccione opcion con letra nemonica o "+CHR(24)+
     " "+chr(25)+" "+chr(26)+" "+chr(27)+" y pulse "+
     chr(17)+chr(196)+chr(217)
en=0
@ rb-1,0 SAY SPACE(80)
DO WHILE .T.
 @ rb-1,0 SAY opciones
 wtab=val(SUBS(tabs,opc*3-1,3))
 SET COLOR TO w+/w
 @ rb-1,wtab-1 SAY SUBS(opciones,wtab,AT("  ",
                        subs(opciones,wtab))-1)
 SET COLOR TO
 IF en>0
  accion=SUBS(acciones,opc,1)
  EXIT
 ENDIF
 DO espera WITH tec,t,rc1,""
 en=AT(" "+STR(t,3),rc1)
 IF t=19
  opc=IIF(opc=1,max_fun,opC-1)
 ELSE
  IF t=4
   opc=IIF(opc=max_fun,1,opc+1)
  ELSE
   IF en>0
    opc=int((eN-1)/4)+1
```

```
    ELSE
     accion=SUBS(acciones,opc,1)
     EXIT
    ENDIF
   ENDIF
  ENDIF
ENDDO
RETURN

PROC opciones
op=" "
DO CASE
CASE accion="B"
 opt="DESEA EFECTUAR LA BAJA(S/N)?:"
 opv="SN"
CASE accion="M"
 IF rc=rf
  opt="DAR OPCION DESEADA (Otros,Alta,Baja,Modificacion,Fin):"
  opv="OABMF"
 ELSE
  opt="DAR OPCION DESEADA (Alta,Baja,Modificacion,Fin):"
  opv="ABMF"
 ENDIF
CASE accion="C"
 IF rc=rf
  opt="DAR OPCION DESEADA (Otros,Fin):"
  opv="OF"
 ELSE
  opt="dar 'F' para continuar:"
  opv="F"
 ENDIF
ENDCASE
wrc=rc
msg="OODar opcion deseada"
DO WHILE  op<>"F"
 @ 23,0 CLEAR
 SET message TO SUBS(msg,3)
 @ 23,0 SAY opt GET op
 READ
 SET message TO " "
 IF .NOT. op$opv
  msg="1ODebe dar un valor aceptable de opcion"
 ELSE
  ok=.T.
```

```
DO CASE
CASE op="S"
 EXIT
CASE op="N"
 ok=.F.
 EXIT
CASE op="O"
 IF EOF()
  GO TOP
 ENDIF
 rc=ri
 wrc=ri
 ok=.F.
 EXIT
CASE op="A"
 accion="A"
 IF type('FILTRO_DET')='c'
  SET FILTER TO
  DO acepta_de
  SET FILTER TO &filtro_det
 ELSE
  DO acepta_de
 ENDIF
 accion="M"
CASE op="B"
 id=id_gen+1
 msg="00 "
 DO WHILE id>0
  @23,0 CLEAR
  SET message TO SUBS(msg,3)
  @23,0 SAY "DAR IDENTIFICACION DEL REGISTRO A DAR DE BAJA:"
        GET id PICTURE "99"
  READ
  SET message TO " "
  IF id>id_gen .AND. id<=id_tot
   ir=(id-id_gen)*7
   IF SUBS(wrec,ir+1,7)<>SPACE(7)
    DO baja_det WITH arch_det,val(SUBS(wrec,ir+1,7))
    wrec=STUFF(wrec,1+ir,7,SPACE(7))
    @ ri+ir/7-1,2 SAY SPACE(78)
    msg="00BAJA DEL REGISTRO "+STR(id,2)+" efectuada"
   ELSE
    msg="11EL REGISTRO YA ESTA DADO DE BAJA"
   ENDIF
```

```
    ELSE
     IF id<>0
      msg="12Identificacion fuera de rango"
     ENDIF
    ENDIF
   ENDDO
  CASE op="M"
   DO cambios
  CASE op="F"
   EXIT
  ENDCASE
 ENDIF
ENDDO
rc=IIF(rc>wrc,rc,wrc)
RETURN


*********************************************************
*            Sistema Generador de Sistemas              *
*          procedimientos de reportes del SGS           *
* programa: gsrr.prg                 autor: SGS/ENE-88  *
*********************************************************
PROCEDURE PUBLICAR
*declara variables publicas para reportes
 cd=2
 wpub="ST01"+SPACE(230) &&+TRIM(SUBS(s01,10,7))+space(230)
 DO WHILE cd<=num_subt
  cds=IIF(cd>9,STR(cd,2),"0"+str(cd,1))
  vd="S"+cds
  wpub=TRIM(wpub)+",ST"+cds &&+SUBS(&vd,AT(">",&vd)+1,7)
  cd=cd+1
 ENDDO
PUBLIC &wpub
RETURN

PROCEDURE reporta
PARAMETERS grupo,num_imp
 cd=1
 DO WHILE cd <= num_imp
  vgrupo=grupo+IIF(cd>9,STR(cd,2),"0"+str(cd,1))
  wgrupo=&vgrupo
  tipo=SUBS(wgrupo,1,1)
  AT=SUBS(wgrupo,2,1)
  r=&gs_device+val(SUBS(wgrupo,3,2))
```

```
  c=val(SUBS(wgrupo,5,2))
  DO CASE
   CASE tipo="T" &&titulos y enunciados
    @ r,c SAY SUBS(wgrupo,7)
   CASE tipo="D"
    vvar=SUBS(wgrupo,7,20)
    vpic=SUBS(wgrupo,27)
    @ r,c SAY &vvar PICTURE vpic
   CASE tipo="F"
    vvar=SUBS(wgrupo,7)
    @ r,c SAY &vvar
  ENDCASE
  cd=cd+1
 ENDDO
 gs_cl=&gs_device
RETURN
PROCEDURE acumula
 cd=1
 DO WHILE cd<=num_subt
  cds=IIF(cd>9,STR(cd,2),"0"+str(cd,1))
  ws=s&cds
  vsubt='ST'+cds &&+SUBS(ws,AT(">",ws)+1,7)
  wsubt=&vsubt
  niv=SUBS(ws,1,2)
  vvar=SUBS(ws,5)+'+'+vsubt
  STORE &vvar TO &vsubt
  cd=cd+1
 ENDDO
RETURN

PROCEDURE cortes_i
PARAMETERS cortes
 cc=1
 DO WHILE cc<=cc_max
  ccs=STR(cc,1)
  vcc=cc&ccs
  wcc&ccs=&vcc
  DO reporta WITH 'E'+ccs,cont_e&ccs
  cc=cc+1
 ENDDO
 cortes="111111111"
 DO reSET WITH cortes
RETURN
```

```
PROCEDURE cortes
PARAMETERS cortes
 cc=1
 DO WHILE cc<=cc_max
  ccs=STR(cc,1)
  vcc=cc&ccs
  IF wcc&ccs<>&vcc
   cortes=STUFF(cortes,cc,1,"1")
   DO reporta WITH 'P'+ccs,cont_p&ccs
   wcc&ccs=&vcc
   DO reporta WITH 'E'+ccs,cont_e&ccs
  ENDIF
  cc=cc+1
 ENDDO
 DO reSET WITH cortes
RETURN
PROCEDURE cortes_f
PARAMETERS cortes
 cc=1
 DO WHILE cc<=cc_max
  ccs=STR(cc,1)
  vcc='P'+ccs
  DO reporta WITH vcc,cont_p&ccs
  cc=cc+1
 ENDDO
RETURN

PROCEDURE reset
PARAMETERS cortes
 cd=1
 DO WHILE cd<=num_subt
  cds=IIF(cd>9,STR(cd,2),"0"+str(cd,1))
  ws=s&cds
  vsubt='ST'+cds &&+SUBS(ws,AT(">",ws)+1,7)
  wsubt=&vsubt
  niv=SUBS(ws,1,2)
  gs_reSET=VAL(SUBS(WS,3,2))
  IF SUBS(cortes,gs_reSET,1) = "1"
   STORE 0 TO &vsubt
  ENDIF
  cd=cd+1
 ENDDO
 cortes="000000000"
RETURN
```

```
PROCEDURE pausa
 @23,0 SAY "Pulse cualquier tecla para continuar..."
 DO WHILE INKEY()=0
 ENDDO
 @0,0 CLEAR
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*      forma para manto. de datos del sistema           *
* programa: gf11.prg                 autor: SGS/SEP-88 *
*********************************************************
@ 8,0 CLEAR
@ 8-1,0 SAY ""
*        1         2         3         4         5         6         7
*12345678901234567890123456789012345678901234567890123456789012345678901234567
TEXT
   SIGLAS DEL SISTEMA:
01 COMPANIA  :
02 FECHA DE INICIO:                          03 FECHA DE VENCIMIENTO:
04 TRAYECTO DE INSTALACION:

            ************* U S U A R I O S ***************
   CLAVE              NOMBRE                PASWORD      NIVEL DE ACCESO
05                                                          Consulta
06                                                           0 a 50
07
08                                                          Mantenimiento
09                                                           51 a 99
10
ENDTEXT
```

```
*********************************************************
*           Sistema Generador de Sistemas               *
*        mantenimiento de datos del sistema             *
* programa: gs11.prg                  autor: SGS/MAR-88 *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
*datos generales
opc=1
repite_det=.T.
datos_tot=11
datos_gen=7
datos_audd=11
datos_audg=7
id_tot=10
id_gen=4
rb=gs_rb
ri=14
rf=18
rc=rb
ok=.F.
primario="SISTEMAS "
arch_gen=""
audit_gen=""
arch_det="USUARIOS "
audit_det=""
*tablas de datos
idgen=" 0303 0404 0506 0707"  &&1,2 #dato inicial 3,2 #dato final
idx= "0710"
*
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           C-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista             N-no exista
*        V-checar valores
* 27,2  desplazamiento del renglon base(rb)
```

```
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*          1         2         3         4         5         6
*    12345678901234567890123456789012345678901234567890123456789
d01="01SGSWORK->CVE_SIST    CNcN002202@?
     SISTEMAS->CVE_SIST          "
msg01="01No existe en el archivo del sistema en desarrollo"
d02="02SISTEMAS->NOMB_SIST CRNN002550@S50                          "
d03="03SISTEMAS->NOMB_CIA  CRNN011560@S60?                         "
d04="04SISTEMAS->FECHA_INI DRNN02190B@E                            "
d05="05SISTEMAS->FECHA_FIN DRNN02670B@E                            "
d06="06SISTEMAS->FECHA_INSTDNCN00000B@S2
     DATE()                      "
d07="07SISTEMAS->TRAYECTO  CRNN032720@S20                          "

*detalle
d08="07USUARIOS->USUARIO   CRKN000404@S4?
     W08USUARIO                  "
d09="08USUARIOS->NOMB_USUA CRNN001130@S30?                         "
d10="09USUARIOS->PASWORD   CRNN004404@S4?                          "
d11="10USUARIOS->NIV_ACCESONRNV006002@S2 99                        "
values11=" 1,98"

wrec=SPACE((rf-ri+2)*7)
SELECT 2
USE usuarios INDEX usuarios
SELECT 1
USE sistemas INDEX sistemas
*declaracion de variables PUBLICAS
DO PUBLICAS
w01cve_sist=cve_sist
accion=" "
DO gf11
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="00Dar datos requeridos"
  SELE sistemas
  GO 1
  DO lee_datos WITH 1,1,rb
  IF ok
```

```
    IF accion$"BCM"
     IF accion="B"
      wrec=STR(RECNO(),7)
     ENDIF
     DO despliega WITH 2,datos_gen,rb
     SELECT usuarios
     GO TOP
     DO despga_de
    ENDIF
    IF ok
     DO CASE
     CASE accion="A"
      DO lee_datos WITH 2,datos_gen,rb
      IF ok
       DO alta WITH arch_gen
       DO asigna WITH 1,datos_gen,.F.
       DO acepta_de
      ENDIF
     CASE accion="B"
      DO opciones
      IF ok
       DO baja WITH "SISTEMAS "
      ENDIF
     ENDCASE
    ENDIF
    DO gf11
    EXIT
   ENDIF
  ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
```

```
*********************************************************
*           Sistema Generador de Sistemas               *
*           reporte de datos del sistema                *
* programa: gs12.prg                 autor: SGS/FEB-89 *
*********************************************************
PUBLIC primer_cc,wcc1,gs_cl
SET PROCEDURE TO gsrr
*datos generales
num_subt=1
cc_max=0
cont_e1=0
cont_p1=0
primer_cc=.T.
gs_hoja=0
gs_linea=5
gs_cortes="0000000000"
*tablas de datos
* ei     encabezado inicial
* ep,pp  encabezado y pie de pagina
* e#,p#  encabezado y pie al corte de control #
* pf     pie final
* ld     lineas de detalle del reporte

* 1,1    tipo de impresion/PROCESO
* 2,1    nivel al que se imprime(corte de control)
* 3,2    desplazamiento del renglon actual
* 5,2    columna de impresion
* 7,n    enunciado o formula      o     7,20 variable     27,n PICTURE

*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789012345678901
ep01="F00001DATE()"
ep02="T00026SISTEMA GENERADOR DE SISTEMAS"
ep03="F00101TIME()"
ep04="T00025REPORTE DE ENUNCIADOS DE MENUS"
ep05="T00070HOJA: "
ep06="f00076gs_hoja PICTURE '9999'"
ep07="T00103SIGLAS DEL SISTEMA:"
ep08="F00022SISTEMAS->CVE_SIST"
ep09="F00025SISTEMAS->NOMB_SIST"
ep10="T00103COMPANIA   :"
ep11="F00015SISTEMAS->NOMB_CIA"
ep12="T00103FECHA DE INICIO:"
ep13="F00019SISTEMAS->FECHA_INI"
```

```
ep14="T00046FECHA DE VENCIMIENTO:"
ep15="F00067SISTEMAS->FECHA_FIN"
ep16="T00103TRAYECTO DE INSTALACION:"
ep17="F00032SISTEMAS->TRAYECTO"
ep18="T00201USUARIO            N O M B R E"
ep19="T00042PASSWORD    NIV. ACCESO"

ld01="F00104USUARIOS->USUARIO"
ld02="F00011USUARIOS->NOMB_USUA"
ld03="F00044USUARIOS->PASWORD"
ld04="F00060USUARIOS->NIV_ACCESO"

pf01="T00201TOTAL DE USUARIOS"
pf02="F00020ST01"
* variables para subtotales
* 1,2 nivel       3,2 reSET at          5,n  variable o formula
s01="0001 1 "
* cortes de control
cc1="USUARIOS->USUARIO"
SELECT 1
USE sistemas INDEX sistemas
SELECT 2
USE usuarios INDEX usuarios

*declaracion de variables PUBLICAS
DO PUBLICAR

SELECT usuarios
IF type('&CC1')="n"
 wcc1=-1.23
ELSE
 wcc1="-bsa"
ENDIF
IF gs_device="PROW()"
 SET device TO print
 SET print on
 pausa=.F.
 gs_tope=60
ELSE
 pausa=.T.
 gs_tope=20
ENDIF
gs_cl = gs_tope + 1
accion=" "
```

```
GO TOP
@ 0,0 CLEAR
DO WHILE accion<>"F"
  IF gs_cl > gs_tope
   IF .NOT. gs_hoja = 0
    IF pausa
     DO pausa
    ELSE
     eject
    ENDIF
   ENDIF
   gs_hoja=gs_hoja+1
   DO reporta WITH 'EP',19
   DO cortes_i WITH gs_cortes
  ENDIF
* checa si hay cortes de control y reporta lo adecuado ,e y p
  DO cortes WITH gs_cortes
  DO reporta WITH 'LD',4
  DO acumula
  SKIP
  IF EOF()
   EXIT
  ENDIF
ENDDO
DO cortes_f WITH gs_cortes
DO reporta WITH 'PF',2
IF pausa
 DO pausa
ENDIF
IF gs_device="PROW()"
 eject
 SET print off
 SET device TO screen
ENDIF
CLOSE DATABASES
RELEASE ALL LIKE w*
SET PROCEDURE TO gsr
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*          forma para manto. de enunc. menus            *
* programa: gf21.prg                 autor: SGS/SEP-88 *
*********************************************************
@ 8,0 CLEAR
@ 8-1,0 SAY ""
#1        2         3         4         5         6         7
#12345678901234567890123456789012345678901234567890123456789012345678
TEXT
                                                CODIGO   MARCO DEL MENU Simple
    FUNCION              N O M B R E       TIPO RET.    R1 C1   R2 C2  Doble
                                              01          ,  a  ,
                                            DESPLAZAMIENTO(R,C)
02
03
04
05
06
07
08
09
10
11
12
ENDTEXT
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*         mantenimiento de enunciados de menus          *
* programa: gs21.prg                  autor: SGS/MAR-88 *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
*datos generales
opc=1
repite_det=.T.
datos_tot=19
datos_gen=11
datos_audd=19
datos_audg=11
id_tot=12
id_gen=1
rb=gs_rb
ri=11
rf=21
rc=rb
ok=.F.
arch_gen="VENTANAS "
audit_gen=""
arch_det="D_MENUS   MENUS     "
audit_det=""
*tablas de datos
idgen=" 0711"  && 1,2 #dato inicial     3,2 #dato final
idx= "1217"
*
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           C-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista             N-no exista
*        V-checar valores
* 27,2  desplazamiento del renglon base(gs_rb)
* 29,2  columna para el dato
```

```
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789012345678901
d01="01MENUS->CVE_FUNC      CRLO02030B@?
     W01CVE_FUN                    MENUS    "
msg01="01No existe en el archivo de menus pENDIENTES de alta"
d02="01MENUS->CVE_MENU      CNCNO20008@S8?
     W01CVE_FUN                    "
d03="01MENUS->NOMB_FUNC     CNDNO21330@S30?                          "
d04="01MENUS->TIPO_FUNC     CNDNO24501@?                             "
d05="01MENUS->RET_CODE      CNDNO25002@?                             "
d06="01VENTANAS->CVE_FUNC   CNCNO2000B@S8?
     W01CVE_FUN                    "
d07="07VENTANAS->RENGLON1   NRNV02570Z@S2 99                         "
values07=" 0,23"
d08="08VENTANAS->COLUMNA1   NRNV02600Z@S2 99                         "
values08=" 0,79"
d09="09VENTANAS->RENGLON2   NRNV02650Z@S2 99                         "
values09=" 0,23"
d10="10VENTANAS->COLUMNA2   NRNV02680Z@S2 99                         "
values10=" 0,79"
d11="11VENTANAS->TIPO_VENT  CRNV02750I@?                             "
values11=" D S C"

*detalle
d12="12D_MENUS->CVE_MENU    CNCN000008@S8?
     W01CVE_FUN                    "
d13="13D_MENUS->CVE_FUNC    CRKN000308@S8?
     W01CVE_FUN+W13CVE_FUN         "
d14="14MENUS->CVE_FUNC      CNCN000000@S8?
     W13CVE_FUN                    "
d15="15MENUS->NOMB_FUNC     CRNN001330@S30?                          "
d16="16MENUS->TIPO_FUNC     CRNV004501@?                             "
values16=" M F R C W X s S "
d17="17MENUS->RET_CODE      CRNN005001@?                             "
d18="18MENUS->RENGLON       NRcV005702@S2 99                         1"
values18="1,15"
d19="19MENUS->COLUMNA       NRcV006002@S2 99                         2"
values19="1,70"

wrec=SPACE((rf-ri+2)*7)
```

```
SELECT 1
USE menus INDEX menus
SELECT 2
USE ventanas INDEX ventanas
SELECT 3
USE d_menus INDEX d_menus
SELECT 1
SET relation TO cve_func into ventanas

*declaracion de variables PUBLICAS
DO PUBLICAS
w01cve_fun=SPACE(8)
SELECT d_menus
SET FILTER TO cve_menu=w01cve_fun
SET relation TO cve_func into menus
accion=" "
DO gf21
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  SELE menus
  IF accion="A"
   SET FILTER TO cve_menu=SPACE(8)
  ELSE
   SET FILTER TO
  ENDIF
  DO lee_datos WITH 1,1,rb
  IF ok
   SELECT menus
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    SET FILTER TO cve_menu=w01cve_fun
    GO TOP
    DO despliega WITH 2,datos_gen,rb
    SELECT d_menus
    GO TOP
    DO despga_de
   ELSE
    SET FILTER TO
   ENDIF
   IF ok
```

```
    DO CASE
    CASE accion="A"
     DO lee_datos WITH 2,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
      DO acepta_de
     ENDIF
    CASE accion="B"
     DO accion
     IF ok
      DO baja WITH "MENUS     "
     ENDIF
    ENDCASE
   ENDIF
   DO gf21
  ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*          reporte de enunciados de menus               *
* programa: gs22.prg                autor: SGS/NOV-88 *
*********************************************************
PUBLIC primer_cc,wcc1,gs_c1
SET PROCEDURE TO gsrr
*datos generales
num_subt=1
cc_max=1
cont_e1=7
cont_p1=2
primer_cc=.T.
gs_hoja=0
gs_linea=5
gs_cortes="0000000000"
*tablas de datos
* ei     encabezado inicial
* ep,pp encabezado y pie de pagina
* e#,p# encabezado y pie al corte de control #
* pf     pie final
* ld     lineas de detalle del reporte

* 1,1    tipo de impresion/PROCESO
* 2,1    nivel al que se imprime(corte de control)
* 3,2    desplazamiento del renglon actual
* 5,2    columna de impresion
* 7,n    enunciado o formula      o    7,20 variable    27,n PICTURE
*        1         2         3         4         5         6
*    12345678901234567890123456789012345678901234567890123456789012345678901
ep01="F00001DATE()"
ep02="T00026SISTEMA GENERADOR DE SISTEMAS"
ep03="F00101TIME()"
ep04="T00025REPORTE DE ENUNCIADOS DE MENUS"
ep05="T00070HOJA: "
ep06="f00076gs_hoja PICTURE '9999'"

e101="F00201MENUS->CVE_MENU      "
e102="F00010MENUS->NOMB_FUNC     "
e103="T00040VENTANA: "
e104="f00049ventanas->renglon1 PICTURE '99,'"
e105="f00052ventanas->columna1 PICTURE '99'"
e106="f00055ventanas->renglon2 PICTURE ' A 99,'"
e107="f00061ventanas->columna2 PICTURE '99'"
```

```
ld01="F00107MENUS->TIPO_FUNC"
ld02="F00010D_MENUS->CVE_FUNC"
ld03="F00020MENUS->NOMB_FUNC"
ld04="f00053menus->renglon PICTURE '99,'"
ld05="f00056menus->columna PICTURE '99'"
ld06="f00060ventanas->renglon1 PICTURE '99,'"
ld07="f00063ventanas->columna1 PICTURE '99'"
ld08="f00066ventanas->renglon2 PICTURE ' A 99,'"
ld09="f00072ventanas->columna2 PICTURE '99'"

p101="T00101TOTAL DE MENUS"
p102="F00016ST01"
* variables para subtotales
* 1,2 nivel       3,2 reSET at          5,n  variable o formula
s01="0001 1 "
* cortes de control
cc1="D_MENUS->CVE_MENU"
SELECT 1
USE menus INDEX menus
SELECT 2
USE ventanas INDEX ventanas
SELECT 3
USE d_menus INDEX d_menus
SELECT 1
SET relation TO cve_func into ventanas

*declaracion de variables PUBLICAS
DO PUBLICAR
SELECT d_menus
IF type('&CC1')="n"
 wcc1=-1.23
ELSE
 wcc1="-bsa"
ENDIF
SET relation TO cve_func into menus
IF gs_device="PROW()"
 SET device TO print
 SET print on
 pausa=.F.
 gs_tope=60
ELSE
 pausa=.T.
 gs_tope=20
```

```
ENDIF
gs_cl = gs_tope + 1
accion=" "
GO TOP
@ 0,0 CLEAR
DO WHILE accion<>"F"
  SET relation TO cve_menu into menus
  IF gs_cl > gs_tope
   IF .NOT. gs_hoja = 0
    IF pausa
     DO pausa
    ELSE
     eject
    ENDIF
    DO reporta WITH 'PP',0
   ENDIF
   gs_hoja=gs_hoja+1
   DO reporta WITH 'EP',6
   DO cortes_i WITH gs_cortes
  ENDIF
* checa si hay cortes de control y reporta lo adecuado ,e y p
  DO cortes WITH gs_cortes
  SET relation TO cve_func into menus
  DO reporta WITH 'LD',9
  DO acumula
  SKIP
  IF EOF()
   EXIT
  ENDIF
ENDDO
DO cortes_f WITH gs_cortes
IF pausa
 DO pausa
ENDIF
IF gs_device="PROW()"
 eject
 SET print off
 SET device TO screen
ENDIF
CLOSE DATABASES
RELEASE ALL LIKE w*
SET PROCEDURE TO gsr
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*   forma para manto. de bases de datos(elementos)      *
* programa: gf311.prg                 autor: SGS/SEP-88 *
*********************************************************
@ RB,0 CLEAR
@ RB-1,0 SAY ""
*         1         2         3         4         5         6         7.
*1234567890123456789012345678901234567890123456789012345678901234567890123456
TEXT
              ************** E L E M E N T O S ***************
     NOMBRE  TIPO LEN DEC      MASCARA                VALORES
01
02
03
04
05
06
07
08
09
10
11
12
13
ENDTEXT
```

```
*********************************************************
*            sistema generador de datos                 *
*      mantenimiento de bases de dATOS(elementos)       *
* programa: gs311.prg                 autor: SGS/MAR-88 *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
*datos generales
opc=1
repite_det=.T.
datos_tot=8
datos_gen=1
datos_audd=08
datos_audg=1
id_tot=13
id_gen=0
rb=gs_rb
ri=09
rf=21
rc=rb
ok=.F.
primario="ARCHIVOS "
arch_gen=""
audit_gen=""
arch_det="DATOS     "
audit_det=""
*tablas de datos
idgen="" && 1,2 #dato inicial    3,2 #dato final
idx= "0208"
*
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave             L-look up(checa con arch.)
*        D-desplegar         N-inguna
*        C-calcular          c-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista            N-no exista
*        V-checar valores
* 27,2  desplazamiento del renglon base(rb)
```

```
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1         2         3         4         5         6
*    12345678901234567890123456789012345678901234567890123456789012345678901234567890
d01="01archivos->filename  cnco002208@?
     'SGSWORK '                    archivos"

* detalle
d02="02   datos->filename  cncn000308@s8
     'SGSWORK '                    "
d03="03   DATOS->FIELD_NAMECRKN000310@S10?
     W01FILENAM+W03FIELD_N         "
d04="04   DATOS->FIELD_TYPECRNV001501@                                    "
values04=" C N D L M "
d05="05   DATOS->FIELD_LEN NRNV00190299
values05=" 0,30"
d06="06   DATOS->FIELD_DEC NONV00230299
values06=" 0,10"
d07="07   DATOS->PICTURE   CONN002720@S20
d08="08   DATOS->VALUES    CONN004820@S20?
*d09="09  AYUDAS->CVE_AYUDA CNCN006910@S10
       W03FIELD_N                   "

wrec=SPACE((rf-ri+2)*7)
*SELECT 3
*use ayudas INDEX ayudas
SELECT 2
USE datos INDEX datos
SET FILTER TO filename='SGSWORK '
*SET relation TO 'F'+field_name into ayudas
SELECT 1
USE archivos INDEX archivos
SET relation TO filename into datos

*declaracion de variables PUBLICAS
DO PUBLICAS
accion=" "
DO gf311
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
```

```
msg="OODar datos requeridos"
SELE archivos
DO lee_datos WITH 1,1,rb
IF ok
 IF accion$"BCM"
  IF accion="B"
   wrec=STR(RECNO(),7)
  ENDIF
  DO despliega WITH 2,datos_gen,rb
  SELECT datos
  GO TOP
  DO despga_de
 ENDIF
 IF ok
  DO CASE
  CASE accion="A"
   DO lee_datos WITH 2,datos_gen,rb
   IF ok
    DO alta WITH arch_gen
    DO asigna WITH 1,datos_gen,.F.
    DO acepta_de
   ENDIF
  CASE accion="B"
   DO opciones
   IF ok
    DO baja WITH "DATOS     "
   ENDIF
  ENDCASE
 ENDIF
 DO gf311
 accion="S"
 ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w$
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*       reporte de elementos de base de datos           *
* programa: gs312.prg                autor: SGS/FEB-88 *
*********************************************************
PUBLIC primer_cc,wcc1,gs_cl
SET PROCEDURE TO gsrr
*datos generales
num_subt=0
cc_max=0
primer_cc=.T.
gs_hoja=0
gs_linea=5
gs_cortes="0000000000"
*tablas de datos
* ei     encabezado inicial
* ep,pp  encabezado y pie de pagina
* e#,p#  encabezado y pie al corte de control #
* pf     pie final
* ld     lineas de detalle del reporte

* 1,1    tipo de impresion/PROCESO
* 2,1    nivel al que se imprime(corte de control)
* 3,2    desplazamiento del renglon actual
* 5,2    columna de impresion
* 7,n    enunciado o formula      o    7,20 variable    27,n PICTURE
*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789012345678
ep01="F00001DATE()"
ep02="T00026SISTEMA GENERADOR DE SISTEMAS"
ep03="F00101TIME()"
ep04="T00021REPORTE DE ELEMENTOS DE BASES DE DATOS"
ep05="T00070HOJA: "
ep06="f00076gs_hoja PICTURE '9999'"
ep07="T00101 ELEMENTO  TIPO LONGITUD DECIMALES"
ep08="T00040MASCARA              VALORES"
ep09="f00100REPL('-',79)"
ep10="T00100 "

ld01="F00101DATOS->FIELD_NAME"
ld02="F00015DATOS->FIELD_TYPE"
ld03="F00020DATOS->FIELD_LEN"
ld04="F00025DATOS->FIELD_DEC"
ld05="F00035DATOS->PICTURE"
```

```
ld06="f00065datos->values PICTURE '@S14'"
* variables para subtotales
* 1,2 nivel        3,2 reSET at          5,n  variable o formula

SELECT 1
USE datos INDEX datos
SET FILTER TO filename='SGSWORK '

*declaracion de variables PUBLICAS
DO PUBLICAR
IF type('&CC1')="n"
 wcc1=-1.23
ELSE
 wcc1="-bsa"
ENDIF
IF gs_device="PROW()"
 SET device TO print
 SET print on
 pausa=.F.
 gs_tope=60
ELSE
 pausa=.T.
 gs_tope=20
ENDIF
gs_c1 = gs_tope + 1
*do reporta WITH 'EI',0
accion=" "
GO TOP
@ 0,0 CLEAR
DO WHILE accion<>"F"
  IF gs_c1 > gs_tope
   IF .NOT. gs_hoja = 0
    IF pausa
     DO pausa
    ELSE
     eject
    ENDIF
    DO reporta WITH 'PP',0
   ENDIF
   gs_hoja=gs_hoja+1
   DO reporta WITH 'EP',10
  ENDIF
  DO reporta WITH 'LD',6
  SKIP
```

```
   IF EOF()
    EXIT
   ENDIF
 ENDDO
 IF pausa
  DO pausa
 ENDIF
 IF gs_device="PROW()"
  eject
  SET print off
  SET device TO screen
 ENDIF
 CLOSE DATABASES
 RELEASE ALL LIKE w*
 SET PROCEDURE TO gsr
 RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*   forma para manto. de bases de datos(estructura)     *
* PROGRAMA: gf32.prg                AUTOR: SGS/SEP-88   *
*********************************************************
@ RB,0 CLEAR
@ RB-1,0 SAY ""
*         1         2         3         4         5         6         7
*12345678901234567890123456789012345678901234567890123456789012345678901234567
TEXT
   ARCHIVO:                01 INDEX1:                LLAVE:
                           02 INDEX2:                LLAVE:

            ************** E L E M E N T O S ***************
     NOMBRE  TIPO LEN DEC       MASCARA                VALORES
03
04
05
06
07
08
09
10
11
12
ENDTEXT
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*     mantenimiento de bases de datos(archivos)         *
* programa: gs321.prg              autor: SGS/MAR-88 *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
*datos generales
opc=1
repite_det=.T.
datos_tot=14
datos_gen=6
datos_audd=14
datos_audg=6
id_tot=12
id_gen=2
rb=gs_rb
ri=12
rf=21
rc=rb
ok=.F.
primario="ARCHIVOS "
arch_gen="ARCHIVOS "
audit_gen=""
arch_det="DATOS     "
audit_det=""
filtro_det="WO1FILENAM=FILENAME"
*tablas de datos
idgen=" 0203 0406" && 1,2 #dato inicial    3,2 #dato final
idx= "0713"
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura(N-o capturar, R-equerido, O-pcional)
* 25,1  validacion
*        K-llave             L-look up(checa con arch.)
*        D-desplegar         N-inguna
*        C-calcular          c-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista            N-no exista
*        V-checar valores
* 27,2  desplazamiento del renglon base(rb)
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
```

```
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789012345678890
d01="01ARCHIVOS->FILENAME  CRK0001108@?
     W01FILENAM                    ARCHIVOS"
d02="02ARCHIVOS->INDEX1    CRNNO0330B@?                              "
d03="03ARCHIVOS->LLAVE1    CRNNO0493O@?                              "
d04="04ARCHIVOS->INDEX2    CONNO1330B@?                              "
d05="05ARCHIVOS->LLAVE2    CONNO1493O@?                              "
d06="05SGSWORK->CVE_SIST   CNCNO000002@?
     SISTEMAS->CVE_SIST            "

* detalle
d07="07    DATOS->FILENAME  CNCN000308@S8
     W01FILENAM                    "
d08="08    DATOS->FIELD_NAMECRKD000310@S10?
     W01FILENAM+W08FIELD_N         DATOS    "
d09="08SGSWORK->FIELD_NAME cnlo000310@s10?
      'SGSWORK '+w08field_n        datos    "
msg09="01No existe en el archivo de elementos del sistema"
d10="10    DATOS->FIELD_TYPECNcV001501@
     DATOS->FIELD_TYPE             "
values10=" C N F L M "
d11="11    DATOS->FIELD_LEN NNcV00190299
     DATOS->FIELD_LEN              "
values11=" 0,20"
d12="12    DATOS->FIELD_DEC NNcV00230299
     DATOS->FIELD_DEC              "
values12=" 0,10"
d13="13    DATOS->PICTURE    COcNO02720@S20
     DATOS->PICTURE                "
d14="14    DATOS->VALUES     COcNO04B20@S20?
     DATOS->VALUES                 "
wrec=SPACE((rf-ri+2)*7)
SELECT 4
USE sistemas INDEX sistemas
GO 1
SELECT 1
USE archivos INDEX archivos
SELECT 2
USE datos INDEX datos
*declaracion de variables PUBLICAS
DO PUBLICAS
```

```
accion=" "
DO gf321
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  SELE archivos
  DO lee_datos WITH 1,1,rb
  IF ok
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    DO despliega WITH 2,datos_gen,rb
    SELECT datos
    SET FILTER TO filename=w01filenam
    GO TOP
    DO despga_de
    SELE datos
    SET FILTER TO
   ENDIF
   IF ok
    DO CASE
    CASE accion="A"
     DO lee_datos WITH 2,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
      DO acepta_de
     ENDIF
    CASE accion="B"
     DO accion
     IF ok
      DO baja WITH "DATOS     "
     ENDIF
    ENDCASE
   ENDIF
   DO gf321
  ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
RETURN
```

```
*********************************************************
*           Sistema Generador de Sistemas               *
*        reporte de archivos de base de datos           *
* programa: gs322.prg                autor: SGS/FEB-88 *
*********************************************************
PUBLIC primer_cc,wcc1,gs_c1
SET PROCEDURE TO gsrr
*datos generales
num_subt=1
cc_max=1
cont_e1=2
cont_p1=2
primer_cc=.T.
gs_hoja=0
gs_linea=5
gs_cortes="0000000000"
*tablas de datos
* ei     encabezado inicial
* ep,pp encabezado y pie de pagina
* e#,p# encabezado y pie al corte de control #
* pf     pie final
* ld     lineas de detalle del reporte

* 1,1    tipo de impresion/PROCESO
* 2,1    nivel al que se imprime(corte de control)
* 3,2    desplazamiento del renglon actual
* 5,2    columna de impresion
* 7,n    enunciado o formula        7,20 variable     27,n PICTURE
*              1         2          3         4         5
*     12345678901234567890123456789012345678901234567890123456789012345
ep01="F00001DATE()"
ep02="T00026SISTEMA GENERADOR DE SISTEMAS"
ep03="F00101TIME()"
ep04="T00021REPORTE DE ARCHIVOS DE BASE DE DATOS"
ep05="T00070HOJA: "
ep06="f00076gs_hoja PICTURE '9999'"
ep07="T00101 ELEMENTO  TIPO LONGITUD DECIMALES"
ep08="T00040MASCARA             VALORES"
ep09="f00100REPL('-',79)"
ep10="T00100 "

e101="T00200ARCHIVO: "
e102="F00012DATOS->FILENAME "
```

```
ld01="F00104DATOS->FIELD_NAME"
ld02="F00020DATOS->FIELD_TYPE"
ld03="F00022DATOS->FIELD_LEN"
ld04="F00027DATOS->FIELD_DEC"
ld05="F00035DATOS->PICTURE"
ld06="f00065datos->values PICTURE '@S14'"

p101="T00100TOTAL DE ELEMENTOS:"
p102="f00020st01 PICTURE '9999'"

* variables para subtotales
* 1,2 nivel       3,2 reSET at          5,n  variable o formula
s01="0001 1 "
cc1="DATOS->FILENAME"

SELECT 1
USE datos INDEX datos
SET FILTER TO filename<>"SGSWORK "
SELECT 2
USE archivos INDEX archivos
SELECT 1
SET relation TO filename into archivos

*declaracion de variables PUBLICAS
DO PUBLICAR
IF gs_device="PROW()"
 SET device TO print
 SET print on
 pausa=.F.
 gs_tope=60
ELSE
 gs_tope=20
 pausa=.T.
ENDIF
gs_c1 = gs_tope + 1
SELECT datos
IF type('&CC1')="n"
 wcc1=-1.23
ELSE
 wcc1="-bsa"
ENDIF
*do reporta WITH 'EI',0
accion=" "
GO TOP
```

```
@ 0,0 CLEAR
DO WHILE accion<>"F"
  IF gs_cl > gs_tope
   IF .NOT. gs_hoja = 0
    IF pausa
     DO pausa
    ELSE
     eject
    ENDIF
    DO reporta WITH 'PP',0
   ENDIF
   gs_hoja=gs_hoja+1
   DO reporta WITH 'EP',10
   DO cortes_i WITH gs_cortes
  ENDIF
* checa si hay cortes de control y reporta lo adecuado, e y p
  DO cortes WITH gs_cortes
  DO reporta WITH 'LD',6
  DO acumula
  SKIP
  IF EOF()
   EXIT
  ENDIF
ENDDO
DO cortes_f WITH gs_cortes
IF pausa
 DO pausa
ENDIF
IF gs_device="PROW()"
 eject
 SET print off
 SET device TO screen
ENDIF
CLOSE DATABASES
RELEASE ALL LIKE w*
SET PROCEDURE TO gsr
RETURN
```

```
*********************************************************
*              Sistema Generador de Sistemas            *
*       forma para manto. de arch. afe. de funciones    *
* programa: gf41.prg                  autor: SGS/SEP-88 *
*********************************************************
@ RB,0 CLEAR
@ RB-1,0 SAY ""
*        1         2         3         4         5         6         7
*1234567890123456789012345678901234567890123456789012345678901234567890123456 7
TEXT
                                          NIVEL      MARCO DEL MENU Simple
    FUNCION          NOMBRE          TIPO SEG.       R1 C1   R2 C2  Doble
                                          01         02    ,    a    ,

                 ARCHIVO TIPO INDEX     RELACIONADO AL
                                         ARCHIVO  DATO

            03
            04
            05
            06
            07
ENDTEXT
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*      mantenimiento de arch. afe. de funciones         *
* programa: gs41.prg                 autor: SGS/MAR-88 *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
*datos generales
opc=1
repite_det=.F.
datos_tot=20
datos_gen=13
datos_audd=20
datos_audg=13
id_tot=7
id_gen=2
rb=gs_rb
ri=14
rf=18
rc=ri
ok=.F.
primario="FUNCIONE "
arch_gen="FUNCIONE VENTANAS "
audit_gen=""
arch_det="ARCH_AFE "
audit_det=""
*tablas de datos
idgen=" 0507 0813"  && 1,2 #dato inicial    3,2 #dato final
idx= "1417"
*
*  1,2  numero de dato maestro
*  3,20 archv0->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           c-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista             N-no exista
*        V-checar valores
* 27,2  desplazamiento del renglon base(rb)
```

```
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789012345678901
d01="01FUNCIONE->CVE_FUNC  CRK0020308@?
     W01CVE_FUN                    "
d02="01SGSWORK->CVE_FUNC   CNL0020308@?
     W01CVE_FUN                    MENUS    "
msg02="01No existe en el archivo de funciones pENDIENTES de alta"
d03="03MENUS->CVE_MENU     CNCN020008@S8?
     W01CVE_FUN                    "
d04="04MENUS->NOMB_FUNC    CNDN021230@S30?
     MENUS->NOMB_FUNC              "
d05="05MENUS->TIPO_FUNC    CNDN024401@?
     MENUS->TIPO_FUNC              "
d06="06FUNCIONE->SEGURIDAD NRNV025002@S2 99                              "
values06=" 0,99"
d07="07FUNCIONE->TIPO_FUNC CNCN020002@S2
     W05TIPO_FU                    "
d08="08VENTANAS->CVE_FUNC  CNCN020008@S8?
     W01CVE_FUN                    "
d09="09VENTANAS->RENGLON1  NRNV025702@S2 99                              "
values09=" 0,23"
d10="10VENTANAS->COLUMNA1  NRNV026002@S2 99                              "
values10=" 0,79"
d11="11VENTANAS->RENGLON2  NRNV026502@S2 99                              "
values11=" 0,23"
d12="12VENTANAS->COLUMNA2  NRNV026802@S2 99                              "
values12=" 0,79"
d13="13VENTANAS->TIPO_VENT CRNV027501@?                                  "
values13=" D S C"

*detalle
d14="14ARCH_AFE->CVE_FUNC  CNCN000008@S8?
     W01CVE_FUN                    "
d15="15ARCH_AFE->ARCH_AFE  CRKN001408@S8?
     W14CVE_FUN+W15ARCH_AF         ARCH_AFE"
msg15="01EL archivo ya se registro para la funcion"
d16="15SGSWORK->ARCH_AFE   CNL0001408@S8?
     W15ARCH_AF                    ARCHIVOS"
msg16="01EL archivo no existe en la base de datos del sistema"
```

```
d17="17ARCH_AFE->TIPO_AFE  CRNV002302@S2?                    "
values17=" GP GE DP DE RE AG AD"
d18="18ARCH_AFE->ACCES_NDX CONN002608@S8?                    "
d19="19ARCH_AFE->LINK_NDX  CONN003508@S8                     "
d20="20ARCH_AFE->LINK_KEY  CONN004410@S10                    "

wrec=SPACE((rf-ri+2)*7)
SELECT 1
USE funcione INDEX funcione
SET FILTER TO tipo_func="F"
SELECT 2
USE ventanas INDEX ventanas
SELECT 3
USE arch_afe INDEX arch_afe
SELECT 4
USE menus INDEX menus
SELECT 5
USE archivos INDEX archivos
SELECT 1
SET relation TO cve_func into ventanas

*declaracion de variables PUBLICAS
DO PUBLICAS
SELECT menus
w01cve_fun=SPACE(8)
SET FILTER TO tipo_func="F" .AND. cve_func=w01cve_fun
GO TOP
SELECT arch_afe
SET FILTER TO cve_func=w01cve_fun

opciones="Alta  Baja  Modifica  Consulta  Pantalla
          Datos afectados  Fin"
tab="A01B07M13C23P33D43F60"
acciones="ABMCPDF"
accion=" "
DO gf41
DO WHILE accion<>"F"
 DO opc_abmch WITH rb,tab,opciones,opc,acciones,accion
 IF accion$"PD"
  ok=.T.
  IF type('W01CVE_FUN')='1'
   ok=.F.
  ELSE
   IF LEN(TRIM(w01cve_fun))=0
```

```
   ok=.F.
  ENDIF
 ENDIF
 IF .NOT. ok
  msg="OODar datos requeridos"
  SELE funcione
  DO lee_datos WITH 1,2,rb
 ENDIF
 IF ok
  IF accion="P"
   DO gs411 WITH w01cve_fun
  ELSE
   DO gs412 WITH w01cve_fun
  ENDIF
 ENDIF
 loop
ENDIF
DO WHILE .NOT. accion$"SF"
 msg="OODar datos requeridos"
 SELE funcione
 DO lee_datos WITH 1,2,rb
 IF ok
  SELE menus
  GO TOP
  SELECT funcione
  IF accion$"BCM"
   IF accion="B"
    wrec=STR(RECNO(),7)
   ENDIF
   GO TOP
   DO despliega WITH 3,datos_gen,rb
   SELECT arch_afe
   GO TOP
   DO despga_de
  ENDIF
  IF ok
   DO CASE
   CASE accion="A"
    DO lee_datos WITH 3,datos_gen,rb
    IF ok
     DO alta WITH arch_gen
     DO asigna WITH 1,datos_gen,.F.
     DO acepta_de
    ENDIF
```

```
     CASE accion="B"
      DO opciones
      IF ok
       DO baja WITH primario
      ENDIF
     ENDCASE
    ENDIF
    DO gf41
   ENDIF
  ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*                edicion de pantallas                   *
* programa: gs411.prg                 autor: SGS/JUL-88 *
*********************************************************
PARAMETER wcve_func
SET talk off
SET echo off
SELE 2
USE ventanas INDEX ventanas
SELE 1
USE pantalla INDEX pantalla
SET FILTER TO cve_func=wcve_func
SET relation TO wcve_func into ventanas
GO TOP
wri_pant=ventanas->renglon1
@0,0 CLEAR
IF EOF()
 r=wri_pant
 DO WHILE r<22
  rs=IIF(r>9,STR(r,2),'0'+str(r,1))
  STORE SPACE(80) TO wr&rs
  r=r+1
 ENDDO
ELSE
 DO WHILE .NOT. EOF()
  rs=IIF(renglon>9,STR(renglon,2),'0'+str(renglon,1))
  STORE texto TO wr&rs
  @ renglon,columna SAY texto
  SKIP
 ENDDO
ENDIF
r=ROW()
c=COL()
DO WHILE .T.
 @23,0 SAY "F          F          F ELIMINA F INSERTA "
 @24,0 SAY "1          2          3 RENGLON 4 RENGLON "
 @23,40 SAY "F ADICION F COPIA    F GUARDA  F ABANDONA"
 @24,40 SAY "5  CAMPO  6 RENGLON 7  FORMA   8  FORMA  "
 @wri_pant,0 SAY ''
 rs=IIF(r>9,STR(r,2),'0'+str(r,1))
 @ r,c SAY ''
 t=0
 DO WHILE t=0
```

```
 t=INKEY()
ENDDO
DO CASE
CASE t=-2
 wr=r
 DO WHILE wr<21
  rsa=IIF(wr>9,STR(wr,2),'0'+str(wr,1))
  wr=wr+1
  rs=IIF(wr>9,STR(wr,2),'0'+str(wr,1))
  STORE wr&rs TO wr&rsa
  @wR-1,0say wr&rsa
 ENDDO
 STORE SPACE(80) TO wr21
 @21,0 SAY SPACE(80)
 @r,c SAY ''
CASE t=-3
 wr=21
 @r,0 SAY SPACE(80)
 DO WHILE wr>r
  rsa=IIF(wr>9,STR(wr,2),'0'+str(wr,1))
  wr=wR-1
  rs=IIF(wr>9,STR(wr,2),'0'+str(wr,1))
  STORE wr&rs TO wr&rsa
  @wr,0say wr&rsa
 ENDDO
 rs=IIF(r>9,STR(r,2),'0'+str(r,1))
 STORE SPACE(80) TO wr&rs
 @r,c SAY ''
CASE t=-4
 wlong=0
 @ 22,0 SAY "De cuantas posiciones es el campo?:" GET wlong
 READ
 wlong=IIF(wlong+c>79,79-c,wlong)
 STORE STUFF(wr&rs,c+1,wlong,REPL(CHR(177),WLONG)) TO wr&rs
 c=c+wlong
 @22,0 SAY SPACE(80)
 @r,0 SAY wr&rs
CASE t=-5
 wreng=0
 @ 22,0 SAY "Que renglon desea copiar?:" GET wreng
 READ
 wrengs=IIF(wreng>9,STR(wreng,2),"0"+str(wreng,1))
 STORE wr&wrengs TO wr&rs
 @22,0 SAY SPACE(80)
```

```
   @r,0 SAY wr&rs
  CASE t=-6
   EXIT
  CASE t=-7
   RETURN
  CASE (t>31 .AND. t<127) .OR. (t>127 .and. t<256)
   IF c<79
    STORE STUFF(wr&rs,COL()+1,1,CHR(t)) TO wr&rs
    @ ROW(),COL() SAY CHR(t)
    c=c+1
   ENDIF
  OTHERWISE
   DO CASE
   CASE t=127
    IF c>0
     c=C-1
     STORE STUFF(wr&rs,COL()+1,1,' ') TO wr&rs
     @ r,c SAY ' '
    ENDIF
   CASE t=5
    r=IIF(r>wri_pant,R-1,r)
   CASE t=24
    r=IIF(r<21,r+1,r)
   CASE t=4
    r=IIF(r<21 .AND. c=79,r+1,r)
    c=IIF(c=79,0,c+1)
   CASE t=19
    r=IIF(r>wri_pant .AND. c=0,R-1,r)
    c=IIF(c=1,79,C-1)
   CASE t=13
    r=IIF(r<21,r+1,r)
    c=0
   ENDCASE
  ENDCASE
  ENDDO
GO TOP
IF EOF()
 r=wri_pant
 DO WHILE r<22
  rs=IIF(r>9,STR(r,2),'0'+str(r,1))
  APPEND BLANK
  REPLACE cve_func WITH wcve_func,renglon with r
  REPLACE columna WITH 0,texto with wr&rs
  r=r+1
```

```
 ENDDO
ELSE
 DO WHILE .NOT. EOF()
  rs=IIF(renglon>9,STR(renglon,2),'0'+str(renglon,1))
  REPLACE texto WITH wr&rs
  SKIP
 ENDDO
ENDIF
RETURN
```

```
********************************************************
*              Sistema Generador de Sistemas           *
*       forma para manto. de arch. afe. de funciones   *
* programa: gf412.prg                 autor: SGS/SEP-88 *
********************************************************
@ RB,0 CLEAR
@ RB-1,0 SAY ""
*        1         2         3         4         5         6         7
*1234567890123456789012345678901234567890123456789012345678901234567890123456
TEXT
   TIPO DE AFECTACION: ░ (General, Detalle)
  SEQ A  R C   ARCHIVO->DATO      CAPT  MASCARA   EN ARCHIVO    VALORES/FORMULA
01 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
02 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
03 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
04 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
05 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
06 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
07 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
08 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
09 ░░ ░ ░░,░░ ░░░░░░░░ ░░░░░░░░░░ ░ ░░░░░░░░░░ ░ ░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░
ENDTEXT
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*      mantenimiento de datos afe. de funciones         *
* programa: gs412.prg                autor: SGS/MAR-88 *
*********************************************************
PARAMETERS pcve_func
*PUBLIC ok,rc,wrec,cds,accion
*datos generales
opc=1
repite_det=.T.
datos_tot=16
datos_gen=1
datos_audd=16
datos_audg=1
id_tot=9
id_gen=0
rb=gs_rb
ri=10
rf=17
rc=rb
ok=.F.
primario="ARCH_AFE "
arch_gen=""
audit_gen=""
arch_det="DATO_AFE "
audit_det=""
*tablas de datos
idgen=" "  && 1,2 #dato inicial    3,2 #dato final
idx= "0215"
*
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           c-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista             N-no exista
*        V-checar valores
```

```
* 27,2  desplazamiento del renglon base(rb)
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789012345678890
d01="01SGSWORK->TIPO_AFE    CRNV002301@?                                    "
values01=" G D "

* detalle
d02="02DATO_AFE->CVE_FUNC  CNCN000008@S8
     PCVE_FUNC                    "
d03="03DATO_AFE->ORDEN     CRNN000302@S2
     W02CVE_FUN+W03ORDEN          "
d04="04DATO_AFE->PROCESO   CRNV000601@?                                    "
values04=" D C c K L N "
d05="05DATO_AFE->RENGLON   NRNN000802@S2 99                                "
d06="06DATO_AFE->COLUMNA   NRNN001102@S2 99                                "
d07="07DATO_AFE->ARCH_AFE  CRLO001408@S8
     W02CVE_FUN+W07ARCH_AF        ARCH_AFE"
d08="08DATO_AFE->DATO_AFE  CRLO002310@S10
     W07ARCH_AF+W08DATO_AF        DATOS    "
msg08="01El dato no pertenece a el archivo"
d09="09DATO_AFE->CAPTURA   CRNV003401@S1                                   "
values09=" R O N "
d10="09DATO_AFE->TIPO_VAR  CNCN000001@S1
     DATOS->FIELD_TYPE            "
d11="09DATO_AFE->LONG_VAR  NNCN000002@S2
     DATOS->FIELD_LEN             "
d12="12DATO_AFE->MASCARA   COcN003625@S10
     DATOS->PICTURE               "
d13="13DATO_AFE->LOOKUP_ON LRNV004701@S1
values13=" O N V "
d14="14DATO_AFE->LOOK_FILE COLO004908@S8
     W02CVE_FUN+W13LOOK_FI        ARCH_AFE"
d15="15DATO_AFE->VALOR     COcN005830@S20
     DATOS->VALUES                "
d16="16DATO_AFE->AFECTACIONCNCN000001@S1
     W01TIPO_AFE                  "
wrec=SPACE((rf-ri+2)*7)
SELECT 2
USE dato_afe INDEX dato_afe
```

```
SELECT 3
USE arch_afe INDEX arch_afe
SELECT 5
USE datos INDEX datos

*declaracion de variables PUBLICAS
DO PUBLICAS
w01tipo_afe=" "
w07arch_af=SPACE(8)
SELECT dato_afe
SET FILTER TO afectacion=w01tipo_afe .AND. cve_func=pcve_func
accion=" "
DO gf412
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  SELE arch_afe
  DO lee_datos WITH 1,1,rb
  IF ok
   SELE dato_afe
   GO TOP
   SELECT arch_afe
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    GO TOP
    DO despliega WITH 2,datos_gen,rb
    SELECT dato_afe
    GO TOP
    DO despga_de
   ENDIF
   IF ok
    DO CASE
    CASE accion="A"
     DO lee_datos WITH 2,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
      DO acepta_de
     ENDIF
    CASE accion="B"
     DO opciones
```

```
      IF ok
       DO baja WITH primario
      ENDIF
     ENDCASE
    ENDIF
    DO gf412
   ENDIF
  ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*      forma para manto. de arch. afe. de reportes      *
* programa: gf42.prg                 autor: SGS/SEP-88 *
*********************************************************
@ RB,O CLEAR
@ RB-1,O SAY ""
*        1         2         3         4         5         6         7
*12345678901234567890123456789012345678901234567890123456789012345678901234567
TEXT
                                                    NIVEL
    REPORTE              NOMBRE                TIPO SEG.
   [          ] [                              ]  [ 01 ]

                ARCHIVO TIPO INDEX    RELACIONADO AL
                                       ARCHIVO  DATO

            03 [          ]  [ ]  [          ]  [          ]
            04 [          ]  [ ]  [          ]  [          ]
            05 [          ]  [ ]  [          ]  [          ]
            06 [          ]  [ ]  [          ]  [          ]
            07 [          ]  [ ]  [          ]  [          ]
ENDTEXT
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*      mantenimiento de arch. afe. de reportes          *
* programa: gs42.prg              autor: SGS/MAR-88      *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
*datos generales
opc=1
repite_det=.F.
datos_tot=14
datos_gen=07
datos_audd=14
datos_audg=07
id_tot=7
id_gen=2
rb=gs_rb
ri=14
rf=18
rc=ri
ok=.F.
primario="FUNCIONE "
arch_gen="FUNCIONE "
audit_gen=""
arch_det="ARCH_AFE "
audit_det=""
*tablas de datos
idgen=" 0607"  && 1,2 #dato inicial    3,2 #dato final
idx= "0814"
*
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           c-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista             N-no exista
*        V-checar valores
* 27,2  desplazamiento del renglon base(rb)
```

```
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789012345678901
d01="01FUNCIONE->CVE_FUNC  CRK002030B@?
     WO1CVE_FUN                    "
d02="01SGSWORK->CVE_FUNC   CNL0020308@?
     WO1CVE_FUN                    MENUS    "
msg02="01No existe en el archivo de funciones pENDIENTES de alta"
d03="03MENUS->CVE_MENU     CNCN020008@S8?
     WO1CVE_FUN                    "
d04="04MENUS->NOMB_FUNC    CNDN021230@S30?                                  "
d05="05MENUS->TIPO_FUNC    CNDN024401@?                                     "
d06="06FUNCIONE->SEGURIDAD NRNV025002@S2 99                                 "
values06=" 0,99"
d07="06FUNCIONE->TIPO_FUNC CNCN020002@S2
     WO5TIPO_FU                    "

*detalle
d08="0BARCH_AFE->CVE_FUNC  CNCN000008@S8?
     WO1CVE_FUN                    "
d09="09ARCH_AFE->ARCH_AFE  CRKN001408@S8?
     WOBCVE_FUN+WO9ARCH_AF         ARCH_AFE"
msg09="01EL archivo ya se registro para el reporte"
d10="09SGSWORK->ARCH_AFE   CNL0001408@S8?
     WO9ARCH_AF                    ARCHIVOS"
msg10="01EL archivo no existe en la base de datos del sistema"
d11="11ARCH_AFE->TIPO_AFE  CRNV002302@S2?                                   "
values11=" GP GE DP DE RE AG AD"
d12="12ARCH_AFE->ACCES_NDX CONN002608@S8?                                   "
d13="13ARCH_AFE->LINK_NDX  CONN003508@S8                                    "
d14="14ARCH_AFE->LINK_KEY  CONN004410@S10                                   "

wrec=SPACE((rf-ri+2)*7)
SELECT 1
USE funcione INDEX funcione
SELECT 2
USE ventanas INDEX ventanas
SELECT 3
USE arch_afe INDEX arch_afe
SET FILTER TO cve_func=funcione->cve_func
```

```
SELECT 4
USE menus INDEX menus
SELECT 5
USE archivos INDEX archivos

*declaracion de variables PUBLICAS
DO PUBLICAS
w01cve_fun=SPACE(8)
SELECT menus
SET FILTER TO tipo_func="R" .AND. cve_func=w01cve_fun
opciones="Alta  Baja  Modifica  Consulta  Datos reportados  Fin"
tab="A01B07M13C23D33F51"
acciones="ABMCDF"
accion=" "
DO gf42
DO WHILE accion<>"F"
 DO opc_abmch WITH rb,tab,opciones,opc,acciones,accion
 IF accion="D"
  ok=.T.
  IF type('W01CVE_FUN')='l'
   ok=.F.
  ELSE
   IF LEN(TRIM(w01cve_fun))=0
    ok=.F.
   ENDIF
  ENDIF
  IF .NOT. ok
   msg="OODar datos requeridos"
   SELE menus
   SET FILTER TO
   DO lee_datos WITH 1,2,rb
  ENDIF
  IF ok
   DO gs421 WITH w01cve_fun
  ENDIF
  loop
 ENDIF
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  DO lee_datos WITH 1,2,rb
  IF ok
   SELECT menus
   GO TOP
   SELECT funcione
```

```
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    DO despliega WITH 3,datos_gen,rb
    SELECT arch_afe
    GO TOP
    DO despga_de
   ENDIF
   IF ok
    DO CASE
    CASE accion="A"
     DO lee_datos WITH 3,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
      DO acepta_de
     ENDIF
    CASE accion="B"
     DO opciones
     IF ok
      DO baja WITH "MENUS     "
     ENDIF
    ENDCASE
   ENDIF
   DO gf42
  ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
RETURN
```

```
*************************************************************
*              Sistema Generador de Sistemas                *
*      forma para manto. de arch. afe. de funciones         *
* programa: gf421.prg                  autor: SGS/SEP-88    *
*************************************************************
@ RB,0 CLEAR
@ RB-1,0 SAY ""
*         1         2         3         4         5         6         7
*12345678901234567890123456789012345678901234567890123456789012345678901234567
TEXT
   GRUPO DE REPORTE:  (EI EP E# D# P# PP PF)
                                                          * NIVEL DE *
 TIPO R C   ARCHIVO    DATO          MASCARA          FORMULA     TOTAL RESET
01
02
03
04
05
06
07
08
09
10
ENDTEXT
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*     mantenimiento de datos afe. de reportes           *
* programa: gs421                    autor: SGS/MAR-88  *
*********************************************************
PARAMETERS pcve_func
*PUBLIC ok,rc,wrec,cds,accion
*datos generales
opc=1
repite_det=.T.
datos_tot=12
datos_gen=1
datos_audd=12
datos_audg=1
id_tot=6
id_gen=0
rb=gs_rb
ri=10
rf=17
rc=rb
ok=.F.
primario="ARCH_AFE "
arch_gen=""
audit_gen=""
arch_det="DATOS_RE "
audit_det=""
*tablas de datos
idgen=" "  && 1,2 #dato inicial    3,2 #dato final
idx= "0212"
*
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           c-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista             N-no exista
*        V-checar valores
```

```
* 27,2  desplazamiento del renglon base(rb)
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1         2         3         4         5         6
*    12345678901234567890123456789012345678901234567890123456789012345678901
d01="01SGSWORK->GRUPO_REP  CRNV0021O2@?                                  "
values01=" EI EP E1 E2 E3 D1 D2 D3 P3 P2 P1 PP PF "

* detalle
d02="02DATOS_RE->CVE_FUNC  CNCN000008@S8
     PCVE_FUNC                  "
d03="03DATOS_RE->GRUPO     CNCN000302@S2
     W01GRUPO_R                 "
d04="04DATOS_RE->TIPO_AFE  CRNV000301@?                                  "
values04=" T F D "
d05="05DATOS_RE->RENGLON   NRNN000502@S2 99                              "
d06="06DATOS_RE->COLUMNA   NRNN000802@S2 99                              "
d07="07DATOS_RE->ARCH_AFE  COLO001108@S8
     W02CVE_FUN+W07ARCH_AF         ARCH_AFE"
d08="08DATOS_RE->DATO_AFE  COLO002010@S10
     W07ARCH_AF+W08DATO_AF         DATOS    "
msg08="01El dato no pertenece a el archivo"
d09="10DATOS_RE->MASCARA   CONN003130@S17                                "
d10="11DATOS_RE->FORMULA   CONN004913@S20                                "
d11="12DATOS_RE->NIV_TOTAL NONV0070019                                   "
values11=" 1,3 "
d12="13DATOS_RE->NIV_RESET NONN0076019                                   "
values12=" 1,3 "

wrec=SPACE((rf-ri+2)*7)
SELECT 2
USE datos_re INDEX datos_re
SELECT 3
USE arch_afe INDEX arch_afe
SELECT 5
USE datos INDEX datos

*declaracion de variables PUBLICAS
DO PUBLICAS
w01cve_fun=SPACE(8)
accion=" "
```

```
DO gf421
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  SELE arch_afe
  DO lee_datos WITH 1,1,rb
  IF ok
   SELECT arch_afe
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    GO TOP
    DO despliega WITH 2,datos_gen,rb
    SELECT datos_re
    GO TOP
    DO despga_de
   ENDIF
   IF ok
    DO CASE
    CASE accion="A"
     DO lee_datos WITH 2,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
      DO acepta_de
     ENDIF
    CASE accion="B"
     DO accion
     IF ok
      DO baja WITH primario
     ENDIF
    ENDCASE
   ENDIF
   DO gf421
  ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
RETURN
```

```
*************************************************************
*            Sistema Generador de Sistemas                  *
*                generador de codigo                        *
* programa: gsp51.prg                autor: SGS/JUN-88      *
*************************************************************
SET talk off
SET echo off
wgs_todo=.T.
SELECT 1
USE sistemas
wsiglas=cve_sist
wsistema=SPACE(25-LEN(TRIM(nomb_sist))/2)+trim(nomb_sist)
wcia=nomb_cia
wtrayecto=TRIM(trayecto)+"\"
wrutinas=wtrayecto+wsiglas+"R.PRG"
wrutinasr=wtrayecto+wsiglas+"RR.PRG"
wsavescr=wtrayecto+"SAVESCR.BIN"
@ 7,0 CLEAR
* 8,0 " Generacion de Menus en PROCESO"
DO gs52
* 10,0 "Generacion de Bases de Datos en PROCESO"
DO gs53
* 14,0 "Generacion de Funciones de mantenimiento en PROCESO"
DO gs54
* 16,0 "Generacion de pantallas del sistema en PROCESO"
DO gs55
* 18,0 "Generacion de Funciones de reporte en PROCESO"
DO gs56
* 20,0 "Generacion del sistema concluida                    "
```

```
*********************************************************
*            Sistema Generador de Sistemas .            *
*                generador de menus                     *
* programa: gsp52.prg                autor: SGS/JUN-88  *
*********************************************************
IF type('WGS_TODO')="u"
 SET talk off
 SET echo off
 SELECT 1
 USE sistemas
 wsiglas=cve_sist
 wsistema=SPACE(25-LEN(TRIM(nomb_sist))/2)+trim(nomb_sist)
 wcia=nomb_cia
 wtrayecto=TRIM(trayecto)+"\"
 wrutinas=wtrayecto+wsiglas+"R.PRG"
 wrutinasr=wtrayecto+wsiglas+"RR.PRG"
 wsavescr=wtrayecto+"SAVESCR.BIN"
 @ 7,0 CLEAR
ENDIF
@ 08,0 SAY "Generacion de Menus en PROCESO"
SELECT 1
USE funcione INDEX funcione
SELECT 2
USE menus INDEX menus
SET FILTER TO cve_menu=cve_func
SET relation TO cve_func into funcione
SELECT 3
USE d_menus INDEX d_menus
SET relation TO cve_func into menus
SELECT 4
USE ventanas INDEX ventanas
SET relation TO cve_func into menus
prog_princ=wsiglas+"P.PRG"
niv=""
wmenu=wsiglas+"M      "

SET delimiter on
SET delimiter TO '[]'
SET console off
SET alternate TO &wtrayecto&prog_princ

SET alternate on
STORE dtoc(dATE()) TO wfecha
??[*********************************************************]
```

```
?[*            Sistema Generador de Sistemas              *]
?[*            ejecutador del menu principal              *]
?[* programa: ]+prog_princ+[              autor: SGS/]+WFECHA+[ *]
?[************************************************************]
?[SET echo off]
?[SET talk off]
?[CLEAR ALL]
?[CLOSE ALL]
?[@ 0,0 CLEAR]
?[SET PROCEDURE TO ]+wsiglas+[r]
?
?[PUBLIC teclado,msg,rb,cb,wbox,max_func,wf,niv_gs]
?[PUBLIC opcion,menu,sub_menu,prog_exe,wcolor,gs_rb]
?[STORE SPACE(16) TO prog_exe]
?[STORE SPACE(40) TO menu,sub_menu]
?[msg=SPACE(80)]
?[niv_gs=SPACE(10)]
?[gs_usuario="     "]
?[gs_sist="  "]
?[gs_cia="  "]
?[gsniv_seg=0]
?[do usuarios WITH gs_usuario,gs_sist,gs_cia,gsniv_seg]
?[niv_gs=""]
?[sysname="]+TRIM(WSISTEMA)+["]
?[gs_siglas="]+WSIGLAS+["]
?[gs_ncia="]+WCIA+["]
?[gs_nsist=sysname]
?[gs_nsuc=""]
?[gs_rb=7]
?
?[* definicion de ventanas de menus]
?[*  1,2 renglon1       3,2 columna1]
?[*  5,2 renglon2       7,2 columna2]
?[*  9,2 # funciones   11,1 opcion inicial]
?[* 12,1 borde         13,2 en detalle long. max. de enunciados]
?[* 15,x COLORES de la ventana]

extrac=.F.
DO WHILE .NOT. extrac
 SELECT d_menus
 SEEK wmenu
 IF .NOT. FOUND()
  ?
  ? [* error]
```

```
 ? [* falta definir el menu: ]+wmenu+[ en d_menus]
 ? [*]
 cm&niv=0
 mm&niv=""
ELSE
 LOCATE for cve_menu$wmenu
 cf=0
 wcm=0
 wmm=""
 DO WHILE FOUND()
  cf=cf+1
  IF menus->tipo_func="M"
   wcm=wcm+1
   wmm=wmm+cve_func+STR(cf,1)
  ENDIF
  vf="F"+STR(cf,1)
  r1=STR(menus->renglon,2)
  c1=STR(menus->columna,2)
  &vf=SPACE(LEN(niv))+"f"+niv+STR(cf,1)+'="'+r1+c1+
      SUBS(menus->tipo_func,1,1)
  &vf=&vf+menus->ret_code+STR(funcione->seguridad,2)+
      SPACE(6)+TRIM(menus->nomb_func)+'"'
  IF menus->tipo_func$"FR"
   REPLACE menus->cve_menu WITH wsiglas+niv+STR(cf,1)
  ENDIF
  continue
 ENDDO
 mm&niv=wmm
 cm&niv=0
 SELECT ventanas
 SEEK wmenu
 IF .NOT. FOUND()
  ?
  ? [* error]
  ? [* falta adicionar el menu en ventanas]
  ? [*]+wmenu+[ ]+menus->nomb_func
 ELSE
  r1=STR(renglon1,2)
  c1=STR(columna1,2)
  r2=STR(renglon2,2)
  c2=STR(columna2,2)
  wbox=SPACE(LEN(niv))+"box"+niv+'="'+r1+c1+r2+c2+
       STR(cf,2)+"1"+ventanas->tipo_vent+'  w+/w"'
  ?
```

```
    ? [* ventana ]+TRIM(menus->nomb_func)
    ? wbox
    mm&niv=wmm
    wcm=1
    DO WHILE wcm<=cf
     wcfs=STR(wcm,1)
     ? f&wcfs
     wcm=wcm+1
    ENDDO
   ENDIF
  ENDIF
  DO WHILE .T.
   wcm=cm&niv+1
   IF wcm>LEN(mm&niv)/8
    IF LEN(niv)=0
     extrac=.T.
     EXIT
    ELSE
     niv=SUBS(niv,1,LEN(niv)-1)
    ENDIF
   ELSE
    cm&niv=wcm
    wmenu=SUBS(mm&niv,wcm*9-8,8)
    niv=niv+SUBS(mm&niv,wcm*9,1)
    EXIT
   ENDIF
  ENDDO
ENDDO
?
?[* ventanas estandar de menus detalle]
?
?[boxw="000000000031D14W+/W"]
?[fw1="0102FM          MANTENIMIENTO"]
?[fw2="0202FR          REPORTE"]
?[fw3="0302RF          FIN"]
?
?[boxx="000000000061D15W+/W"]
?[fx1="0102FA          ALTAS"]
?[fx2="0202FB          BAJAS"]
?[fx3="0302FM          MODIFICACIONES"]
?[fx4="0402FC          CONSULTAS"]
?[fx5="0502FR          REPORTES"]
?[fx6="0602RF          FIN"]
?
```

```
?[* falta fy]
?
?[wniv_gs=SPACE(10)]
?[wopcion=0]
?[STORE SPACE(80) TO ret_code,ret_code0]
?
?[*inician instrucciones del programa principal]
?[teclado=" 19 4 5 24 13 28"]
?[tecla=0]
?[opcion0=1]
?[max_func0=val(SUBS(box,9,2))]
?[opcion=1]
?[load savescr]
?[do caratula]
?[do menus WITH ret_code0]
?[opcion=0]
?[niv_gs="1"]
?[DO WHILE niv_gs>="0"]
?[ IF opcion=0]
?[  DO menus WITH ret_code]
?[ ENDIF]
?[ DO espera WITH teclado,tecla,ret_code,ret_code0]
?[ DO accion WITH niv_gs,tecla,opcion0,opcion,boxx,
                 ret_code,ret_code0]
?
?[*  ejecuta programas permitiendo reasignacion de PROCEDURES]
?[  IF prog_exe<>SPACE(16)]
?[   call savescr WITH "S0"]
?[   SET safety off]
?[   save TO gsvariab]
?[   SET safety on]
?[   RELEASE ALL except prog_exe,nomb,gs*]
?[   DO ]+CHR(38)+[prog_exe]
?[   CLEAR MEMORY]
?[   CLOSE DATABASE]
?[   PUBLIC teclado,msg,rb,cb,wbox,max_func,wf,niv_gs]
?[   PUBLIC opcion,menu,sub_menu,prog_exe,wcolor,gs_rb]
?[   reSTORE FROM gsvariab additive]
?[   @ 0,0 CLEAR]
?[   call savescr WITH "R0"]
?[   STORE SPACE(16) TO prog_exe]
?[  ENDIF]
?[ENDDO]
?[CLOSE ALL]
```

```
?[@ 0,0 CLEAR]
?[if niv_gs="-2"]
?[ QUIT]
?[ENDIF]
?[CLEAR ALL]
?[SET talk on]
?[SET echo on]
?[* fin del programa principal]
SET console on
SET alternate off
SET alternate TO
SET delimiter TO ""
SET delimiter off
@ 12,0 SAY "Generacion de Rutinas del Sistema en PROCESO"
SET PROCEDURE TO
SET console off
!copy gsr.prg &wrutinas
@12,0 SAY ""
!copy gsrr.prg &wrutinasr
@12,0 SAY ""
!copy savescr.bin &wsavescr
SET console on
SET PROCEDURE TO gsr
@ 20,0 SAY "Generacion de menus concluida"
*fin del programa
```

```
*********************************************************
*             Sitema Generador de Sistemas              *
*             generador de bases de datos               *
* programa: gsp53.prg                autor: SGS/ENE-89  *
*********************************************************
SET safety off
IF type('WGS_TODO')="u"
 SELECT 1
 USE sistemas
 wsiglas=cve_sist
 wsistema=SPACE(25-LEN(TRIM(nomb_sist))/2)+trim(nomb_sist)
 wtrayecto=TRIM(trayecto)+"\"
 @ 7,0 CLEAR
ENDIF
@ 10,0 SAY "Generacion de Base de Datos en PROCESO "
SELECT 2
USE archivos INDEX archivos
SELECT 3
USE datos INDEX datos
SET FILTER TO filename<>"SGSWORK"
              .AND. filename=archivos->filename
SELECT archivos
GO TOP
DO WHILE .NOT. EOF()
 wfilename=filename
 windex1=index1
 windex2=index2
 wllave1=llave1
 wllave2=llave2
 SELECT 4
 USE dbf
 zap
 PACK
 SELECT datos
 GO TOP
 DO WHILE .NOT. EOF()
  SELECT dbf
  APPEND BLANK
  REPLACE field_name WITH datos->field_name,
          field_type with datos->field_type
  REPLACE field_len  WITH datos->field_len,
          field_dec with datos->field_dec
  SELECT datos
  SKIP
```

```
 ENDDO
 SELECT dbf
 USE
 create &wtrayecto&wfilename FROM dbf
 IF LEN(TRIM(windex1))>0
  INDEX on &wllave1 TO &wtrayecto&windex1
  IF LEN(TRIM(windex2))>0
   INDEX on &wllave2 TO &wtrayecto&windex2
  ENDIF
 ENDIF
 SELECT archivos
 SKIP
ENDDO
wtrayec=SUBS(wtrayecto,1,LEN(wtrayecto)-1)
SET console off
!copy usuarios.dbf &wtrayec
!copy sistemas.dbf &wtrayec
!copy usuarios.ndx &wtrayec
!copy sistemas.ndx &wtrayec
SET console on
@ 20,0 SAY "Generacion de base de datos concluida"
SET safety on
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*       generador de funciones de mantenimiento         *
* programa: gs54.prg                 autor: SGS/MAR-88  *
*********************************************************
IF type('WGS_TODO')="u"
 SELECT 1
 USE sistemas
 wsiglas=cve_sist
 wsistema=SPACE(25-LEN(TRIM(nomb_sist))/2)+trim(nomb_sist)
 wtrayecto=TRIM(trayecto)+"\"
 @ 7,0 CLEAR
ENDIF
@ 14,0 SAY "Generacion de Funciones de mantenimiento en PROCESO"
SELECT 4
USE ventanas INDEX ventanas
SELECT 1
USE funcione INDEX funcione
SET FILTER TO tipo_func="F"
SET relation TO cve_func into ventanas
SELECT 2
USE menus INDEX menus
SET FILTER TO tipo_func="F"
SET relation TO cve_func into funcione
SELECT 3
USE arch_afe INDEX arch_afe
SET FILTER TO cve_func=menus->cve_func
SELECT 5
USE dato_afe INDEX dato_afe
SET FILTER TO cve_func=menus->cve_func
STORE CHR(38)+chr(38) TO com

* PROCESO
SET delimiter on
SET delimiter TO '[]'
SET console off
SELE menus
GO TOP
DO WHILE .NOT. EOF()
 wfuncion=SPACE(25-LEN(TRIM(nomb_func))/2)+trim(nomb_func)
 STORE 0 TO cd,cgen,cdet
 prog_princ=TRIM(cve_menu)+".PRG"
 forma=STUFF(prog_princ,2,1,"F")
 SET alternate TO &wtrayecto&prog_princ
```

```
SET alternate on
STORE dtoc(dATE()) TO wfecha
??[*************************************************************]
?[*]+wsistema+[*]
?[*]+wfuncion+[*]
?[* programa: ]+prog_princ+[            autor: SGS/]+WFECHA+[ *]
?[*************************************************************]
?[PUBLIC ok,rc,wrec,cds,accion]
?[* tabla de datos]
?[*  1,2  numero de dato maestro]
?[*  3,20 archivO->variable]
?[* 23,1  tipo variable(C,N,D,L)]
?[* 24,1  tipo captura]
?[*       N-o capturar]
?[*       R-equerido]
?[*       O-pcional]
?[* 25,1  validacion]
?[*       K-llave              L-look up(checa con arch.)]
?[*       D-desplegar          N-inguna]
?[*       C-calcular           c-alcula y despliega]
?[* 26,1  look up(checa con archivo)]
?[*       O-exista             N-no exista]
?[*       V-checar valores]
?[* 27,2  desplazamiento del renglon base(rb)]
?[* 29,2  columna para el dato]
?[* 31,2  tamano del dato]
?[* 33,25 PICTURE]
?[* 58,30 formula de calculo o llave de acceso]
?[* 88,8  archivo para look up]
?[*        1         2         3         4         5         6]
?[*12345678901234567890123456789012345678901234567890123456789]
SELECT dato_afe
GO TOP
DO WHILE .NOT. EOF()
 cd=cd+1
 cds=IIF(cd>9,STR(cd,2),'0'+str(cd,1))
 var=TRIM(arch_afe)+'->'+trim(dato_afe)
 ?
 ??[d]+cds+[="]+orden+var+SPACE(20-LEN(var))+tipo_var+
           captura+PROCESO+LOOKUP_ON
 ??STR(renglon,2)+str(COLUMNA,2)+STR(long_var,2)+mascara
 IF PROCESO$"LCCK"
  ?? valor
 ENDIF
```

```
 IF LEN(TRIM(look_file))>0
  ??look_file
 ENDIF
 ?? [ ]
 IF lookup_on="V"
  ?[values]+cds+[=" ]+VALOR+[ ]
 ENDIF
 IF afectacion='D'
  cdet=cdet+1
 ELSE
  cgen=cgen+1
 ENDIF
 SKIP
ENDDO
cid_det=cdet+cgeN-1
cid_gen=cgeN-1
?[ ]
?[*datos generales]
?[opc=1]
?[repite_det=.T.]
?[dATOS_TOT=]+STR(cdet+cgen,2)
?[dATOS_GEN=]+STR(cgen,2)
?[dATOS_AUDD=]+STR(cdet+cgen,2)
?[dATOS_AUDG=]+STR(cgen,2)
?[id_tot=]+STR(cid_det,2)
?[id_gen=]+STR(cid_gen,2)
?[rb=gs_rb]
?[ri=]+STR(ventanas->renglon1,2)
?[rf=]+STR(ventanas->renglon2,2)
?[rc=rb]
?[ok=.F.]
?[wrec=SPACE((rf-ri+2)*7)]
?[* 1,2 #dato inicial    3,2 #dato final]
?[idgen="]+IIF(CGEN >1," 2"," 0")+
            IIF(cgen>1,STR(cgen,2)," 0")+[ ]
?[idx= "]+STR(CGEN+1,2)+STR(CGEN+CDET,2)+[ ]
?
?[* declaracion de variable PUBLICAS]
?[  DO PUBLICAS]
?
* SELECCION de archivos
 SELECT arch_afe
 GO TOP
 warea=1
```

```
STORE "" TO  arch_gen,arch_det,arch_audd,
             arch_audg,primario,secundario
DO WHILE .NOT. EOF()
 DO CASE
 CASE tipo_afe="PR"
  primario=arch_afe+" "
 CASE tipo_afe$"GP"
  primario=arch_afe+" "
  arch_gen=arch_afe+" "
 CASE tipo_afe="G"
  arch_gen=arch_gen+arch_afe+" "
 CASE tipo_afe="DP"
  STORE arch_afe+" " TO secundario,arch_det
 CASE tipo_afe="DE"
  arch_det=arch_det+" "+arch_afe
 CASE tipo_afe="AG"
  arch_audg=arch_audg+" "+arch_afe
 CASE tipo_afe="AD"
  arch_audd=arch_audd+" "+arch_afe
 ENDCASE
 ?[SELECT ]+STR(warea,2)
 ?
 ??[use ]+arch_afe
 IF LEN(TRIM(acces_ndx))>0
  ??[ INDEX ]+acces_ndx
 ENDIF
 IF LEN(TRIM(link_ndx))>0
  ?[SET relATION TO ]+TRIM(link_key)+[ into ]+link_ndx
 ENDIF
 ?
 warea=warea+1
 SKIP
ENDDO
?[arch_gen="]+ARCH_GEN+["]
?[audit_gen="]+ARCH_AUDG+["]
?[arch_det="]+ARCH_DET+["]
?[audit_det="]+ARCH_AUDD+["]
IF LEN(TRIM(primario))=0
 primario=secundario
ENDIF
?[SELECT ]+primario
?[accion=" "]
?[do ]+forma
?[DO WHILE accion<>"F"]
```

```
?[ DO opc_abmc WITH opc]
?[ DO WHILE .NOT. accion$"SF"]
?[  msg="OODar datos requeridos"]
?[  SELE ]+primario
?[  DO lee_datos WITH 1,1,rb]
?[  IF ok]
?[   SELECT ]+primario
?[   IF accion$"BCM"]
?[    IF accion="B"]
?[     wrec=STR(RECNO(),7)]
?[    ENDIF]
?[    GO TOP]
?[    DO despliega WITH 2,datos_gen,rb]
IF LEN(TRIM(secundario))>0
 ?[    SELECT ]+secundario
 ?[    GO TOP]
 ?[    DO despga_de]
ENDIF
?[   ENDIF]
?[   IF ok]
?[    DO CASE]
?[    CASE accion="A"]
?[     DO lee_datos WITH 2,datos_gen,rb]
?[     IF ok]
?[      DO alta WITH arch_gen]
?[      DO asigna WITH 1,datos_gen,.F.]
IF LEN(TRIM(secundario))>0
 ?[       DO acepta_de]
ENDIF
?[     ENDIF]
?[    CASE accion="B"]
?[     DO opciones]
?[     IF ok]
?[      DO baja WITH primario]
?[     ENDIF]
?[    ENDCASE]
?[   ENDIF]
?[   DO ]+forma
?[  ENDIF]
?[ ENDDO]
?[ENDDO]
?[CLOSE DATABASES]
?[RELEASE ALL LIKE w$]
?[RETURN]
```

```
 SELE menus
 SKIP
ENDDO
SET console on
SET alternate off
SET alternate TO
SET delimiter TO ""
SET delimiter off
@ 20,0 SAY "Generacion de funciones de mantenimiento concluida"
*fin del programa
```

```
*********************************************************
*          Sistema Generador de Sistemas                *
*       generador de pantallas de mantenimiento         *
* programa: gs55.prg                 autor: SGS/MAR-88 *
*********************************************************
IF type('WGS_TODO')="u"
 SELECT 1
 USE sistemas
 wsiglas=cve_sist
 wsistema=SPACE(25-LEN(TRIM(nomb_sist))/2)+trim(nomb_sist)
 wtrayecto=TRIM(trayecto)+"\"
 @7,0 CLEAR
ENDIF
@ 16,0 SAY "Generacion de pantallas del sistema en PROCESO"
SELECT 1
USE funcione INDEX funcione
SET FILTER TO tipo_func="F"
SELECT 2
USE pantalla INDEX pantalla
SET FILTER TO cve_func=funcione->cve_func
SELE 3
USE menus INDEX menus
SET FILTER TO tipo_func='F'
SET relation TO cve_func into funcione
STORE CHR(38)+chr(38) TO com

* PROCESO
SET delimiter on
SET delimiter TO '[]'
SET console off
SELE menus
GO TOP
DO WHILE .NOT. EOF()
 wfuncion=SPACE(25-LEN(TRIM(nomb_func))/2)+trim(nomb_func)
 STORE 0 TO cd,cgen,cdet
 prog_princ=TRIM(cve_menu)+".PRG"
 forma=STUFF(prog_princ,2,1,"F")
 SET alternate TO &wtrayecto&forma
 SET alternate on
 STORE dtoc(dATE()) TO wfecha
 ??[*********************************************************]
 ?[*]+wsistema+[*]
 ?[*]+wfuncion+[*]
 ?[* forma: ]+forma+[              autor: SGS/]+WFECHA+[ *]
```

```
 ?[************************************************************]
 SELE pantalla
 GO TOP
 ?[@rb,0 CLEAR]
 ?[@rb-1,0 SAY '']
 ?[*         1         2         3         4         5         6]
 ?[*123456789012345678901234567890123456789012345678901234567890]
 ?[text]
 DO WHILE .NOT. EOF()
  ?SUBS(texto,1,79)
  SKIP
 ENDDO
 ?[endtext]
 SELE menus
 SKIP
ENDDO
SET console on
SET alternate off
SET alternate TO
SET delimiter TO ""
SET delimiter off
@ 20,0 SAY "Generacion del pantallas del sistema concluida
CLOSE DATABASE
*fin del programa
```

```
*************************************************************
*            Sistema Generador de Sistemas                  *
*          generador de funciones de reporte                *
* programa: gs56.prg                  autor: SGS/ENE-89     *
*************************************************************
IF type('WGS_TODO')="u"
 SELECT 1
 USE sistemas
 wsiglas=cve_sist
 wsistema=SPACE(25-LEN(TRIM(nomb_sist))/2)+trim(nomb_sist)
 wtrayecto=TRIM(trayecto)+"\"
 @ 7,0 CLEAR
ENDIF
@ 18,0 SAY "Generacion de Funciones de reporte en PROCESO"
SELECT 1
USE funcione INDEX funcione
SET FILTER TO tipo_func="R"
SELECT 2
USE menus INDEX menus
SET FILTER TO tipo_func="R"
SET relation TO cve_func into funcione
SELECT 3
USE orden_re INDEX orden_re
SET FILTER TO cve_func=menus->cve_func
SELECT 4
USE arch_afe INDEX arch_afe
SET FILTER TO cve_func=menus->cve_func
SELECT 5
USE datos_re INDEX datos_re
*SET relation TO arch_afe+dato_afe into datos
STORE CHR(38)+chr(38) TO com

* PROCESO
SET delimiter on
SET delimiter TO '[]'
SET console off
SELE menus
GO TOP
DO WHILE .NOT. EOF()
 wfuncion=SPACE(25-LEN(TRIM(nomb_func))/2)+trim(nomb_func)
 STORE 0 TO cd,ct,cs
 STORE "" TO wtab_grupos
 prog_princ=TRIM(cve_menu)+".PRG"
 forma=STUFF(prog_princ,2,1,"F")
```

```
wdevice="ROW()"
IF wdevice="PROW()"
 wpausa=".F."
 wtope_imp="66"
ELSE
 wpausa=".T."
 wtope_imp="20"
ENDIF
SET alternate TO &wtrayecto&prog_princ
SET alternate on
STORE dtoc(dATE()) TO wfecha
??[*************************************************]
?[*]+wsistema+[*]
?[*]+wfuncion+[*]
?[* programa: ]+prog_princ+[          autor: SGS/]+WFECHA+[ *]
?[*************************************************]
?[PUBLIC primer_cc,wcc1,gs_c1]
?[*tablas de datos a imprimir]
?[*   ei     encabezado inicial]
?[*   ep,pp  encabezado y pie de pagina]
?[*   e#,p#  encabezado y pie al corte de control #]
?[*   pf     pie final]
?[*   ld     lineas de detalle del reporte]
?
?[*   1,1    tipo de impresion/PROCESO]
?[*   2,1    nivel al que se imprime(corte de control)]
?[*   3,2    desplazamiento del renglon actual]
?[*   5,2    columna de impresion]
?[*   7,n    enunciado o formula  7,20 variable    27,n PICTURE]
?
?[* variables para subtotales]
?[* 1,2 nivel       3,2 reSET at       5,n  variable o formula]
?
?[*         1         2         3         4         5         6]
?[*12345678901234567890123456789012345678901234567890123456789]
?
SELECT datos_re
wgrupo=grupo
DO WHILE .NOT. EOF()
 IF grupo<>wgrupo
  IF cd>0
   STORE cd TO cont_&wgrupo
   STORE wtab_grupos+" "+wgrupo TO wtab_grupos
   ??[cont_]+wgrupo+[=]+cds
```

```
  ?
  ?
  cd=0
 ENDIF
 wgrupo=grupo
ENDIF
cd=cd+1
cds=IIF(cd>9,STR(cd,2),'0'+str(cd,1))
var=TRIM(arch_afe)+IIF(LEN(trim(arch_afe))=0,
                       '','->')+trim(dato_afe)
var=var+SPACE(20-LEN(var))
??grupo+cds+[="]+tipo_afe+STR(niv_total,1)+
               str(renglon,2)+str(columna,2)
DO CASE
CASE tipo_afe="D"
 IF var="SGSWORK->GS_TOTAL"
  ct=ct+1
  cts=IIF(ct>9,STR(ct,2),"0"+str(ct,1))
  ??[ st]+cts
  IF LEN(TRIM(mascara))>0
   ??TRIM(mascara)
  ENDIF
  ??["]
  ?[s]+cts+[="]+STR(NIV_TOTAL,2)+STR(NIV_RESET,2)+
                 TRIM(FORMULA)+["]
  ?
 ELSE
  ??var
  IF LEN(TRIM(mascara))>0
   ??TRIM(mascara)
  ENDIF
  ??["]
 ENDIF
CASE tipo_afe="F"
 ??TRIM(formula)+["]
CASE tipo_afe="T"
 ??TRIM(formula)+["]
ENDCASE
?
SKIP
ENDDO
IF cd>0
 STORE cd TO cont_&wgrupo
 STORE wtab_grupos+" "+wgrupo TO wtab_grupos
```

```
 ??[cont_]+wgrupo+[=]+cds
 ?
 ?
ENDIF
?[* datos para direccionar la impresion]
?[gs_device="]+WDEVICE+["]    && ROW()  o prow()
?[pausa=]+wpausa              && .T. o .F.
?[gs_tope=]+wtope_imp         && lineas maximas de pagina
?
?[*datos generales fijos]
?[primer_cc=.T.]
?[gs_hoja=0]
?[gs_linea=5]
?[gs_cl=gs_tope+1]
?[gs_cortes="0000000000"]

?[* datos generales variables]
?[num_subt=]+STR(ct,2)
?[cc_max=]+STR(cs,2)
?[ ]
?[* declaracion de variable PUBLICAS]
?[  DO PUBLICAR]
?
?[* cortes de control, orden del sort]
SELECT orden_re
GO TOP
DO WHILE .NOT. EOF()
 cs=cs+1
 ?[cc]+niv_corte+[="]+VARIABLE+["]
 SKIP
ENDDO
* SELECCION de archivos
SELECT arch_afe
GO TOP
warea=1
STORE "" TO  arch_gen,arch_det,arch_audd,
             arch_audg,primario,secundario
DO WHILE .NOT. EOF()
 DO CASE
 CASE tipo_afe$"GP"
  primario=arch_afe+" "
  arch_gen=arch_afe+" "
 CASE tipo_afe="G"
  arch_gen=arch_gen+arch_afe+" "
```

```
CASE tipo_afe="DP"
 STORE arch_afe+" " TO secundario,arch_det
CASE tipo_afe="DE"
 arch_det=arch_det+" "+arch_afe
CASE tipo_afe="AG"
 arch_audg=arch_audg+" "+arch_afe
CASE tipo_afe="AD"
 arch_audd=arch_audd+" "+arch_afe
ENDCASE
?[SELECT ]+STR(warea,2)
?
??[use ]+arch_afe
IF LEN(TRIM(acces_ndx))>0
 ??[ INDEX ]+acces_ndx
ENDIF
IF LEN(TRIM(link_ndx))>0
 ?[SET relATION TO ]+TRIM(link_key)+[ into ]+link_ndx
ENDIF
?
warea=warea+1
SKIP
ENDDO
IF LEN(TRIM(primario))>0
 ?[SELECT ]+primario
ELSE
 ?[***** error *****]
 ?[   falta especificar archivo general primario]
 ?
ENDIF
IF "EI"$wtab_grupos
 ?[do reporta WITH 'EI',cont_ei]
ENDIF
?[accion=" "]
?
?[go TOP]
?[@ 0,0 CLEAR]
?[DO WHILE accion<>"F"]
?[  IF gs_cl > gs_tope]
?[   IF .NOT. gs_hoja = 0]
?[    IF pausa]
?[     DO pausa]
?[    ENDIF]
IF "PP"$wtab_grupos
 ?[    DO reporta WITH 'PP',cont_pp]
```

```
ENDIF
?[   ENDIF]
?[   gs_hoja=gs_hoja+1]
IF "EP"$wtab_grupos
 ?[   DO reporta WITH 'EP',cont_ep]
ENDIF
?[   DO cortes_i WITH gs_cortes]
?[  ENDIF]
?
?[* checa si hay cortes de control y reporta lo adecuado,e y p]
?
?[  DO cortes WITH gs_cortes]
?
IF "LD"$wtab_grupos
 ?[  DO reporta WITH 'LD',cont_ld]
ENDIF
?
?[  DO acumula]
?
?[  SKIP]
?[  IF EOF()]
?[   EXIT]
?[  ENDIF]
?[ENDDO]
?[do cortes_f WITH gs_cortes]
?
?[CLOSE DATABASES]
?[RELEASE ALL LIKE w*]
?[return]
?[*fin del programa]
SELE menus
SKIP
ENDDO
SET console on
SET alternate off
SET alternate TO
SET delimiter TO ""
SET delimiter off
@ 20,0 SAY "Generacion de funciones de reporte concluida        "
*fin del programa
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*      carga de elementos y archivos existentes         *
*          en la base de datos del sistema              *
* programa: gs61.prg                autor: SGS/ENE-89 *
*********************************************************
@ 7,0 CLEAR
SET safety off
STORE SPACE(8) TO iniciales
STORE SPACE(20) TO wtrayecto
@ 8,1 SAY "DAR EL TRAYECTO DONDE SE ENCUENTRAN LOS ARCHIVOS:"
      GET wtrayecto
@10,1 SAY "DAR INICIALES DE PROGRAMAS A LISTAR(DEBEN SER .DBF):"
      GET iniciales
READ
wtrayecto=TRIM(wtrayecto)
wdir=wtrayecto+TRIM(iniciales)+".DBF"
!del gslistdb.dir
!dir &wdir > gslistdb.dir
SELECT 5
USE datos INDEX datos
SELECT 6
USE archivos INDEX archivos
SELECT 1
USE gslistdb INDEX gslistdb
zap
APPEND FROM gslistdb.dir sdf
DELETE for archivo=' ' .OR. archivo="gslistdb"
PACK
GO TOP
DO WHILE .NOT. EOF()
  warch=wtrayecto+archivo
  SELECT 2
  USE &warch
  copy structure extended TO dbf
  USE dbf
  GO TOP
  DO WHILE .NOT. EOF()
   SELECT datos
   APPEND BLANK
   REPLACE filename WITH 'SGSWORK',
           field_name with dbf->field_name
   REPLACE field_type WITH dbf->field_type
   REPLACE field_len WITH dbf->field_len,
```

```
            field_dec with dbf->field_dec
    APPEND BLANK
    REPLACE filename WITH gslistdb->archivo,
            field_name with dbf->field_name
    REPLACE field_type WITH dbf->field_type
    REPLACE field_len WITH dbf->field_len,
            field_dec with dbf->field_dec
    SELECT 2
    SKIP
   ENDDO
   SELECT 6
   APPEND BLANK
   REPLACE filename WITH gslistdb->archivo
   SELECT 1
   SKIP
ENDDO
RETURN
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*      utileria para listar estructura de archivos      *
* programa: gs62.prg                 autor: SGS/MAR-88  *
*********************************************************
@ 7,0 CLEAR
SET safety off
STORE SPACE(8) TO iniciales
STORE SPACE(20) TO wtrayecto
@ 8,1 SAY "DAR EL TRAYECTO DONDE SE ENCUENTRAN LOS ARCHIVOS:"
      GET wtrayecto
@10,1 SAY "DAR INICIALES DE PROGRAMAS A LISTAR(DEBEN SER .DBF):"
      GET iniciales
READ
wtrayecto=TRIM(wtrayecto)
wdir=wtrayecto+TRIM(iniciales)+".DBF"
!del gslistdb.dir
!dir &wdir > gslistdb.dir
SELECT 1
USE gslistdb INDEX gslistdb
zap
APPEND FROM gslistdb.dir sdf
DELETE for archivo=' ' .OR. archivo="gslistdb"
PACK
GO TOP
disp stru TO print
DO WHILE .NOT. EOF()
   warch=wtrayecto+archivo
   SELECT 2
   USE &warch
   display structure TO print
   SELECT 1
   SKIP
ENDDO
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*     utileria para listar estructura de archivos       *
* programa: gs62.prg                  autor: SGS/MAR-88 *
*********************************************************
@ 7,0 CLEAR
SET safety off
STORE SPACE(8) TO iniciales
STORE SPACE(20) TO wtrayecto
@ 8,1 SAY "DAR EL TRAYECTO DONDE SE ENCUENTRAN LOS ARCHIVOS:"
      GET wtrayecto
@10,1 SAY "DAR INICIALES DE PROGRAMAS A LISTAR(DEBEN SER .DBF):"
      GET iniciales
READ
wtrayecto=TRIM(wtrayecto)
wdir=wtrayecto+TRIM(iniciales)+".DBF"
!del gslistdb.dir
!dir &wdir > gslistdb.dir
SELECT 1
USE gslistdb INDEX gslistdb
zap
APPEND FROM gslistdb.dir sdf
DELETE for archivo=' ' .OR. archivo="gslistdb"
PACK
GO TOP
disp stru TO print
DO WHILE .NOT. EOF()
   warch=wtrayecto+archivo
   SELECT 2
   USE &warch
   display structure TO print
   SELECT 1
   SKIP
ENDDO
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*            utileria para listar programas             *
* programa: gs63.prg                autor: SGS/MAR-88 *
*********************************************************
@ 7,0 CLEAR
SET safety off
STORE SPACE(8) TO iniciales
STORE SPACE(20) TO wtrayecto
@ 8,1 SAY "DAR EL TRAYECTO DONDE SE ENCUENTRAN LOS ARCHIVOS:"
      GET wtrayecto
@10,1 SAY "DAR INICIALES DE PROGRAMAS A LISTAR(DEBEN SER .PRG):"
      GET iniciales
READ
wtrayecto=TRIM(wtrayecto)
wdir=wtrayecto+TRIM(iniciales)+".PRG"
!del gslistdb.dir
!dir &wdir > gslistdb.dir
SELECT 1
USE gslistdb INDEX gslistdb
zap
APPEND FROM gslistdb.dir sdf
DELETE for archivo=' '
PACK
SET device TO print
SET print on
SET console off
GO TOP
DO WHILE .NOT. EOF()
  warch=wtrayecto+TRIM(archivo)+'.'+extension
  type &warch TO print
  eject
  SKIP
ENDDO
SET console on
SET device TO scre
SET print off
```

```
*********************************************************
*            Sistema Generador de Sistemas              *
*                rutina de instalacion                  *
* programa: gsinstal.prg          autor: SGS/OCT-88      *
*********************************************************
@ 0,0 CLEAR
!del *.ndx
SET safety off
gs_siglas="  "
SELECT 2
USE sistemas
   zap
   APPEND BLANK
   @ 0,0 SAY "SIGLAS DEL SISTEMA A INSTALAR:"
         GET gs_siglas PICTURE "@!"
   READ
   REPLACE cve_sist WITH gs_siglas
   INDEX on cve_sist TO sistemas
SELECT 1
USE usuarios
   zap
   APPEND BLANK
   REPLACE usuario WITH 'SGS',pasword with 'SGSO',
           cve_sist with gs_siglas
   REPLACE nomb_usua WITH "SISTEMA GENERADOR DE SISTEMAS",
           niv_acceso with 99
   INDEX on usuario TO usuarios

USE funcione
   zap
   INDEX on cve_func TO funcione

USE pantalla
   zap
   INDEX on cve_func+STR(renglon,2)+str(columna,2) TO pantalla

USE arch_afe
   zap
   rc=0
   INDEX on cve_func+STR(rc)+arch_afe+tipo_afe TO arch_afe

USE archivos
   zap
   INDEX on filename TO archivos
```

```
USE datos
   zap
   INDEX on filename+field_name TO datos

USE ventanas
   zap
   INDEX on cve_func TO ventanas

USE dato_afe
   zap
   INDEX on cve_func+orden+arch_afe+dato_afe TO dato_afe

USE datos_re
   zap
   INDEX on cve_func+grupo TO datos_re

USE orden_re
   zap
   INDEX on cve_func+niv_corte TO orden_re

USE d_menus
   zap
   INDEX on cve_menu+cve_func TO d_menus

USE menus
   zap
   APPEND BLANK
   REPLACE cve_func WITH sistemas->cve_sist+"M",
           nomb_func with "menu principal",
           tipo_func with "M"
   INDEX on cve_func TO menus

USE gslistdb
   zap
   INDEX on archivo+extension TO gslistdb

? "INSTALACION DE ARCHIVOS INDICE EFECTUADA"
SET safety on
@ 0,0 CLEAR
```

8).- Implantación del sistema.

Los sistemas generados por el Sistema Generador de Sistemas, tienen un proceso de instalación muy sencillo. Consiste en indicar dentro de la opción "SISTEMA" del Sistema Generador de Sistemas, el trayecto de instalación a utilizar para nuestra aplicación. El trayecto de instalación es similar al que maneja MS-DOS y se indica de la siguiente manera:

B:\EMPRESA\APLICACION

Para el ejemplo de aplicación el trayecto de instalación es:

C:\DB_TESIS\SISTEMA

Todos los programas necesarios para el funcionamiento de nuestra aplicación(programas, archivos de datos e indices), son generados dentro del trayecto de instalación especificado, por lo tanto, el sistema quedara instalado al momento de generarse.

Los pasos para ejecutar algun sistema generado son los siguientes:

- Ejecutar DBASE.
- Dar el comando "DO PEP".
- Proporcionar la clave de usuario.
- Proporcionar el pasword del usuario.

El nombre del programa(PEP), se obtiene de las siglas que se registraron en el Sistema Generador de Sistemas, seguidas de la letra "P".

La operación del sistema generado se realiza de manera similar a la descrita anteriormente en los puntos 1 y 2.

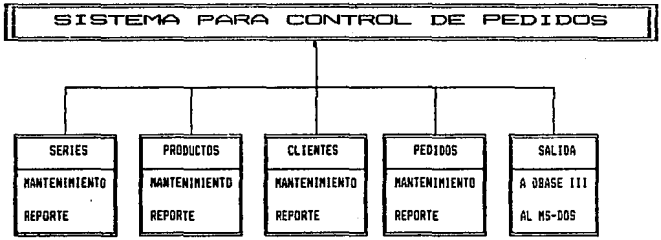


Figura 4.3 : Diagrama del ejemplo de aplicación.

DIAGRAMA DE ARCHIVOS

(EJEMPLO DE APLICACION: SISTEMA DE PEDIDOS)

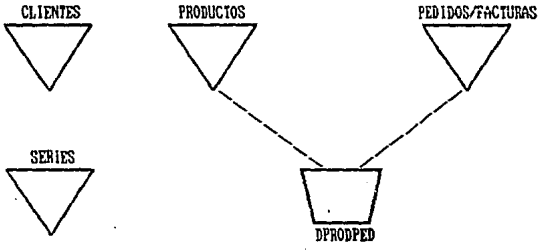


Figura 4.4 : Diagrama de archivos del ejemplo de aplicacion.

ELEMENTO	TIPO	LONG.	DECIMALES	MASCARA Y VALORES
ARCHIVO: CLIENTES				
CVE_CLIENT	C	4	0	
DIRECCION1	C	30	0	@S30
DIRECCION2	C	30	0	@S30
EXT	N	5	0	
NOMBRE	C	30	0	
TEL	N	12	0	
* TOTAL DE ELEMENTOS: 6				
ARCHIVO: DPRODPED				
CANTIDAD	N	6	0	
CVE_PEDIDO	C	4	0	
CVE_PROD	C	4	0	
PRECIO	N	6	0	
USUARIO	C	4	0	
* TOTAL DE ELEMENTOS: 5				
ARCHIVO: PEDIDOS				
CVE_CLIENT	C	4	0	
CVE_PEDIDO	C	4	0	
FECHA_PED	D	8	0	
FECHA_SIST	D	8	0	
STATUS	C	1	0	
USUARIO	C	4	0	
* TOTAL DE ELEMENTOS: 6				
ARCHIVO: PRODUCTO				
CVE_PROD	C	4	0	
EXISTENCIA	N	6	0	
NOMBRE	C	30	0	
PRECIO_UNI	N	6	0	
REORDEN	N	6	0	
* TOTAL DE ELEMENTOS: 5				
ARCHIVO: SERIES				
CONSECUTIV	N	4	0	
CVE_SERIE	C	2	0	
NOMBRE	C	30	0	
STATUS	C	1	0	
* TOTAL DE ELEMENTOS: 4				

04/10/89 SISTEMA PARA EL CONTROL DE PEDIDOS
08:38:50 REPORTE DE SERIES HOJA: 1

CVE	N O M B R E	CONSECUTIVO
DE	DESPENSA INTEGRADA	4
PE	PEDIDOS DOMESTICOS	2
PV	PRODUCTOS VARIOS	0
SD	SERVICIOS A DOMICILIO	3

04/10/89 SISTEMA PARA EL CONTROL DE PEDIDOS
08:40:11 REPORTE DE PRODUCTOS HOJA: 1

CVE	D E S C R I P C I O N	PRECIO UNITARIO
PR01	LECHE EN POLVO NIDO	1,500.00
PR02	CORN-FLAKES	2,455.00
PR03	CARNE DE RES/CORTES FINOS	25,000.00
PR04	PAN INTEGRAL BIMBO	1,234.00

CLIENTE: CL01 SRA. LEONOR REYES GARCIA
DIRECCION:
CERRADA DEL CEDRO No. 25
COL. MOCTEZUMA, D.F.
TELEFONO: 211-34-35 EXTENSION: 234

CLIENTE: CL02 ROSARIO DEL VIVAR
DIRECCION:
AV. DEL TALLER No. 25
COL. NUEVA ANZURES
TELEFONO: 550-12-35 EXTENSION: 23332

04/10/89 SISTEMA PARA EL CONTROL DE PEDIDOS
08:41:24 REPORTE DE PEDIDOS HOJA: 1

CLIENTE: CL01 SRA. LEONOR REYES GARCIA
DIRECCION: CERRADA DEL CEDRO No. 25
COL. MOCTEZUMA, D.F.
TELEFONO: 211-34-35 EXTENSION: 234

PRODUCTO	DESCRIPCION	CANTIDAD	PRECIO	T O T A L
PR01	LECHE EN POLVO NIDO	10	1,455.00	14,550.00
PR02	CORN-FLAKES	123	1,400.00	172,200.00
PR03	CARNE DE RES/CORTES FINOS	8	25,000.00	200,000.00
	TOTAL DEL PEDIDO:			386,750.00

* * * GRACIAS POR SU COMPRA * * *

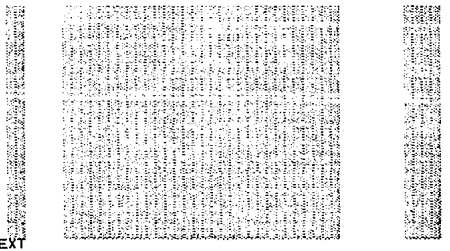
12).- Listados de programas.

```
*********************************************************
*          Sistema para el Control de Pedidos           *
*            ejecutador del menu principal              *
* programa: pep.prg              autor: SGS/04/10/89 *
*********************************************************
SET echo off
SET talk off
CLEAR ALL
CLOSE ALL
@ 0,0 CLEAR
SET PROCEDURE TO per
PUBLIC teclado,msg,rb,cb,wbox,max_func,wf,niv_gs
PUBLIC opcion,menu,sub_menu,prog_exe,wcolor,gs_rb
STORE SPACE(16) TO prog_exe
STORE SPACE(40) TO menu,sub_menu
msg=SPACE(80)
niv_gs=SPACE(10)
gs_usuario="    "
gs_sist="  "
gs_cia="  "
gsniv_seg=0
do usuarios WITH gs_usuario,gs_sist,gs_cia,gsniv_seg
niv_gs=""
sysname="        Sistema para el Control de Pedidos"
gs_siglas="PE"
gs_ncia="PROVEEDORA COMPUTACIONAL, S.A. DE C.V."
gs_nsist=sysname
gs_nsuc=""
gs_rb=7
* definicion de ventanas de menus
*  1,2 renglon1       3,2 columna1
*  5,2 renglon2       7,2 columna2
*  9,2 # funciones   11,1 opcion inicial
* 12,1 borde         13,2 en detalle long. max. de enunciados
* 15,x COLORES de la ventana
* ventana MENU PRINCIPAL
box=" 3 0 579 51D  w+/w"
f1=" 1 5MS 0       SERIES"
f2=" 120MP 0       PRODUCTOS"
f3=" 137MC 0       CLIENTES"
f4=" 154ME 0       PEDIDOS"
f5=" 168MT 0       TERMINAR"
```

```
* ventana SERIES
 box1=" 7 01216 21D  w+/w"
 f11=" 1 2FM53       MANTENIMIENTO"
 f12=" 3 2RR54       REPORTE"

* ventana PRODUCTOS
 box2=" 7171133 21D  w+/w"
 f21=" 1 2FM25       MANTENIMIENTO"
 f22=" 3 2RR 0       REPORTE"

* ventana CLIENTES
 box3=" 7441250 21D  w+/w"
 f31=" 1 2FM78       MANTENIMIENTO"
 f32=" 3 2RR 0       REPORTE"

* ventana PEDIDOS
 box4=" 7511268 21D  w+/w"
 f41=" 1 2FM67       MANTENIMIENTO"
 f42=" 3 2RR 0       REPORTE"

* ventana TERMINAR
 box5=" 7631277 21D  w+/w"
 f51=" 1 2sR 0       A DBASE III"
 f52=" 3 2SR 0       AL MS-DOS"

* ventanas estandar de menus detalle
boxw="0000000031D14W+/W"
fw1="0102FM         MANTENIMIENTO"
fw2="0202FR         REPORTE"
fw3="0302RF         FIN"

boxx="0000000061D15W+/W"
fx1="0102FA         ALTAS"
fx2="0202FB         BAJAS"
fx3="0302FM         MODIFICACIONES"
fx4="0402FC         CONSULTAS"
fx5="0502FR         REPORTES"
fx6="0602RF         FIN"

wniv_gs=SPACE(10)
wopcion=0
STORE SPACE(80) TO ret_code,ret_code0
*inician instrucciones del programa principal
```

```
teclado=" 19 4 5 24 13 28"
opcion0=1
max_func0=val(SUBS(box,9,2))
opcion=1
LOAD savescr
do caratula
DO menus WITH ret_code0
opcion=0
niv_gs="1"
DO WHILE niv_gs>="0"
 IF opcion=0
  DO menus WITH ret_code
 ENDIF
 DO espera WITH teclado,tecla,ret_code,ret_code0
 DO accion WITH niv_gs,tecla,opcion0,opcion,boxx,
               ret_code,ret_code0
*  llama al programa a ejecutarse reasignando de PROCEDURES
  IF prog_exe<>SPACE(16)
   CALL savescr WITH "S0"
   SET safety off
   SAVE TO gsvariab
   SET safety on
   RELEASE ALL except prog_exe,nomb,gs*
   DO &prog_exe
   CLEAR MEMORY
   CLOSE DATABASE
   PUBLIC teclado,msg,rb,cb,wbox,max_func,wf,niv_gs
   PUBLIC opcion,menu,sub_menu,prog_exe,wcolor,gs_rb
   RESTORE FROM gsvariab additive
   @ 0,0 CLEAR
   call savescr WITH "R0"
   STORE SPACE(16) TO prog_exe
  ENDIF
ENDDO
CLOSE ALL
@ 0,0 CLEAR
IF niv_gs="-2"
 QUIT
ENDIF
CLEAR ALL
SET talk on
SET echo on
* fin del programa principal
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*               MANTENIMIENTO DE SERIES                 *
* forma: PF11.PRG                  autor: SGS/04/09/89  *
*********************************************************
@rb,0 CLEAR
@rb-1,0 SAY ''
*        1         2         3         4         5         6         7
*12345678901234567890123456789012345678901234567890123456789012345678901234567
TEXT
    SERIE              N O M B R E                NUMERO INICIAL
01
02
03
04
05
06
07
08
09
10
11
12
13
14
ENDTEXT
```



```
*********************************************************
*         Sistema para el Control de Pedidos            *
*               MANTENIMIENTO DE SERIES                 *
* programa: PE11.PRG               autor: SGS/04/09/89  *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
* tabla de datos
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           c-alcula y despliega
* 26,1  look up(checa con archivo)
```

```
*         O-exista            N-no exista
*         V-checar valores
* 27,2  desplazamiento del renglon base(rb)
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1          2          3          4          5          6
*    12345678901234567890123456789012345678901234567890123456789012345678901234567890
d01="01SGSWORK->CVE_SERIE  CNCO 0 0 2
     '  '                      "
d02="02SERIES->CVE_SERIE   CRKO 0 5 2
     WO2CVE_SER                 "
d03="04SERIES->NOMBRE      CRDO 01130                                    "
d04="05SERIES->STATUS      CNCO 060 1
     'A'                     "
d05="06SERIES->CONSECUTIV  NRNO 048 49999                                "
*datos generales
opc=1
repite_det=.T.
dATOS_TOT= 5
dATOS_GEN= 1
dATOS_AUDD= 5
dATOS_AUDG= 1
id_tot= 4
id_gen= 0
rb=gs_rb
ri= 8
rf=21
rc=rb
ok=.F.
wrec=SPACE((rf-ri+2)*7)
* 1,2 #dato inicial    3,2 #dato final
idgen=" 0 0"
idx= " 2 5"
* declaracion de variable PUBLICAS
  DO PUBLICAS
SELECT  1
use SERIES   INDEX SERIES
arch_gen=""
audit_gen=""
arch_det="SERIES   "
audit_det=""
```

```
SELECT SERIES
accion=" "
do PF11.PRG
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  SELE SERIES
  DO lee_datos WITH 1,1,rb
  IF ok
   SELECT SERIES
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    GO TOP
    DO despliega WITH 2,datos_gen,rb
    SELECT SERIES
    GO TOP
    DO despga_de
   ENDIF
   IF ok
    DO CASE
    CASE accion="A"
     DO lee_datos WITH 2,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
      DO acepta_de
     ENDIF
    CASE accion="B"
     DO opciones
     IF ok
      DO baja WITH primario
     ENDIF
    ENDCASE
   ENDIF
   DO PF11.PRG
  ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
RETURN
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*                 REPORTE DE SERIES                     *
* programa: PE12.PRG              autor: SGS/04/09/89   *
*********************************************************
PUBLIC primer_cc,wcc1,gs_cl
*tablas de datos a imprimir
*   ei    encabezado inicial
*   ep,pp encabezado y pie de pagina
*   e#,p# encabezado y pie al corte de control #
*   pf    pie final
*   ld    lineas de detalle del reporte
*   1,1   tipo de impresion/PROCESO
*   2,1   nivel al que se imprime(corte de control)
*   3,2   desplazamiento del renglon actual
*   5,2   columna de impresion
*   7,n   enunciado o formula o 7,20 variable    27,n PICTURE
* variables para subtotales
* 1,2 nivel       3,2 reSET at          5,n  variable o formula
*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789
EP01="F0 4 9DATE()"
EP02="T0 024Sistema para el Control de Pedidos"
EP03="F0 1 9TIME()"
EP04="T0 033REPORTE DE SERIES"
EP05="F0 061GS_HOJA PICTURE 'HOJA: 99'"
EP06="T0 114CVE          N O M B R E                  CONSECUTIVO"
EP07="F0 109REPL('_',60)"
cont_EP=07
LD01="F0 114SERIES->CVE_SERIE"
LD02="F0 019SERIES->NOMBRE "
LD03="F0 055SERIES->CONSECUTIV  "
cont_LD=03
* datos para direccionar la impresion
gs_device="PROW()"
IF gs_device="PROW()"
 SET device TO print
 SET print on
 pausa=.F.
 gs_tope=55
ELSE
 pausa=.T.
 gs_tope=20
ENDIF
```

```
*datos generales fijos
primer_cc=.T.
gs_hoja=0
gs_linea=5
gs_cl=gs_tope+1
gs_cortes="0000000000"
* datos generales variables
num_subt= 0
cc_max= 0
* declaracion de variable PUBLICAS
  DO PUBLICAR
SELECT  1
use SERIES   INDEX SERIES
accion=" "
go TOP
@ 0,0 CLEAR
DO WHILE accion<>"F"
  IF gs_cl > gs_tope
   IF .NOT. gs_hoja = 0
    IF pausa
     DO pausa
    ENDIF
   ENDIF
   gs_hoja=gs_hoja+1
   DO reporta WITH 'EP',cont_ep
   DO cortes_i WITH gs_cortes
  ENDIF
* checa si hay cortes de control y reporta ,e y p
  DO cortes WITH gs_cortes
  DO reporta WITH 'LD',cont_ld
  DO acumula
  SKIP
  IF EOF()
   EXIT
  ENDIF
ENDDO
do cortes_f WITH gs_cortes
IF gs_device="PROW()"
 EJECT
 SET device TO SCREEN
 SET print OFF
ENDIF
CLOSE DATABASES
RELEASE ALL LIKE w*
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*              MANTENIMIENTO DE PRODUCTOS               *
* forma: PF21.PRG                  autor: SGS/04/09/89 *
*********************************************************
@rb,0 CLEAR
@rb-1,0 SAY ''
*       1         2         3         4         5         6         7
*12345678901234567890123456789012345678901234567890123456789012345678901234567
TEXT
 PRODUCTO          NOMBRE                    PRECIO   EXISTENCIA
01
02
03
04
05
06
07
08
09
10
11
12
13
14
ENDTEXT
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*              MANTENIMIENTO DE PRODUCTOS               *
* programa: PE21.PRG               autor: SGS/04/09/89 *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
* tabla de datos
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           c-alcula y despliega
* 26,1  look up(checa con archivo)
```

```
*         O-exista            N-no exista
*         V-checar valores
* 27,2  desplazamiento del renglon base(rb)
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*              1         2         3         4         5         6
*    12345678901234567890123456789012345678901234567890123456789012345678901
d01="00SGSWORK->CVE_PROD    CNCO 0 0 4
     .    .                    "
d02="01PRODUCTO->CVE_PROD  CRKN 0 3 4@S4
     W02CVE_PRO                 "
d03="02PRODUCTO->NOMBRE     CRNN 0 830@S30                    "
d04="03PRODUCTO->PRECID_UNINRNN 039 6999,999.99               "
d05="04PRODUCTO->EXISTENCIANRNV 052 699,999                   "
values05=" 0,99999                  .      "
*datos generales
opc=1
repite_det=.T.
dATOS_TOT= 5
dATOS_GEN= 1
dATOS_AUDD= 5
dATOS_AUDG= 1
id_tot= 4
id_gen= 0
rb=gs_rb
ri= 8
rf=21
rc=rb
ok=.F.
wrec=SPACE((rf-ri+2)*7)
* 1,2 #dato inicial    3,2 #dato final
idgen=" 0 0"
idx= " 2 5"
* declaracion de variable PUBLICAS
  DO PUBLICAS
SELECT  1
use PRODUCTO INDEX PRODUCTO
arch_gen=""
audit_gen=""
arch_det="PRODUCTO "
audit_det=""
```

```
SELECT PRODUCTO
accion=" "
do PF21.PRG
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  SELE PRODUCTO
  DO lee_datos WITH 1,1,rb
  IF ok
   SELECT PRODUCTO
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    GO TOP
    DO despliega WITH 2,datos_gen,rb
    SELECT PRODUCTO
    GO TOP
    DO despga_de
   ENDIF
   IF ok
    DO CASE
    CASE accion="A"
     DO lee_datos WITH 2,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
      DO acepta_de
     ENDIF
    CASE accion="B"
     DO opciones
     IF ok
      DO baja WITH primario
     ENDIF
    ENDCASE
   ENDIF
   DO PF21.PRG
  ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
return
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*                 REPORTE DE PRODUCTOS                  *
* programa: PE22.PRG            autor: SGS/04/09/89     *
*********************************************************
PUBLIC primer_cc,wcc1,gs_cl
*tablas de datos a imprimir
*   ei    encabezado inicial
*   ep,pp encabezado y pie de pagina
*   e#,p# encabezado y pie al corte de control #
*   pf    pie final
*   ld    lineas de detalle del reporte
*   1,1   tipo de impresion/PROCESO
*   2,1   nivel al que se imprime(corte de control)
*   3,2   desplazamiento del renglon actual
*   5,2   columna de impresion
*   7,n   enunciado o formula  o  7,20 variable    27,n PICTURE
* variables para subtotales
* 1,2 nivel       3,2 reSET at         5,n  variable o formula
*          1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789
EP01="FO 4 9DATE()"
EP02="TO 024Sistema para el Control de Pedidos"
EP03="FO 1 9TIME()"
EP04="TO 030REPORTE DE PRODUCTOS"
EP05="FO 060GS_HOJA PICTURE 'HOJA: 99'"
EP06="TO 109CVE          D E S C R I P C I O N          PRECIO
      UNITARIO"
EP07="FO 109REPL('_',60)"
cont_EP=07
LD01="FO 1 9PRODUCTO->CVE_PROD"
LD02="FO 014PRODUCTO->NOMBRE "
LD03="FO 055PRODUCTO->PRECIO_UNI PICTURE '999,999.99'"
cont_LD=03
gs_device="PROW()"
IF gs_device="PROW()"
 SET device TO print
 SET print on
 pausa=.F.
 gs_tope=55
ELSE
 pausa=.T.
 gs_tope=20
ENDIF
```

```
*datos generales fijos
primer_cc=.T.
gs_hoja=0
gs_linea=5
gs_cl=gs_tope+1
gs_cortes="0000000000"
* datos generales variables
num_subt= 0
cc_max= 0
* declaracion de variable PUBLICAS
  DO PUBLICAR
SELECT  1
use PRODUCTO    INDEX PRODUCTO
accion=" "
go TOP
@ 0,0 CLEAR
DO WHILE accion<>"F"
  IF gs_cl > gs_tope
   IF .NOT. gs_hoja = 0
    IF pausa
     DO pausa
    ENDIF
   ENDIF
   gs_hoja=gs_hoja+1
   DO reporta WITH 'EP',cont_ep
   DO cortes_i WITH gs_cortes
  ENDIF
* checa si hay cortes de control y reporta ,e y p
  DO cortes WITH gs_cortes
  DO reporta WITH 'LD',cont_ld
  DO acumula
  SKIP
  IF EOF()
   EXIT
  ENDIF
ENDDO
do cortes_f WITH gs_cortes
IF gs_device="PROW()"
 EJECT
 SET device TO SCREEN
 SET print OFF
ENDIF
CLOSE DATABASES
RELEASE ALL LIKE w*
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*              MANTENIMIENTO DE CLIENTES                *
* forma: PF31.PRG                  autor: SGS/04/09/89 *
*********************************************************
@rb,0 CLEAR
@rb-1,0 SAY ''
*        1         2         3         4         5         6         7
*1234567890123456789012345678901234567890123456789012345678901234567890123456789
TEXT

01 CLIENTE: [redacted]

02 DIRECCION: [redacted]

03 TELEFONO:  [redacted]  EXTENSION: [redacted]1
ENDTEXT


*********************************************************
*         Sistema para el Control de Pedidos            *
*              MANTENIMIENTO DE CLIENTES                *
* programa: PE31.PRG               autor: SGS/04/09/89 *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
* tabla de datos
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           c-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista             N-no exista
*        V-checar valores
* 27,2  desplazamiento del renglon base(rb)
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
```

```
* 88,8  archivo para look up
*              1          2          3          4          5          6
*     12345678901234567890123456789012345678901234567890123456789012345678901
d01="01CLIENTES->CVE_CLIENTCRKO 111 4@S4
     W01CVE_CLI                  "
d02="02CLIENTES->NOMBRE     CRNO 11630@S30                              "
d03="03CLIENTES->DIRECCION1CRNO 31630@S30                               "
d04="04CLIENTES->DIRECCION2CONO 41630@S30                               "
d05="05CLIENTES->TEL        NRNO 61612999-99-99                         "
d06="06CLIENTES->EXT        NONO 641 599999                             "
*datos generales
opc=1
repite_det=.T.
dATOS_TOT= 6
dATOS_GEN= 6
dATOS_AUDD= 6
dATOS_AUDG= 6
id_tot= 5
id_gen= 5
rb=gs_rb
ri= 8
rf=21
rc=rb
ok=.F.
wrec=SPACE((rf-ri+2)*7)
* 1,2 #dato inicial    3,2 #dato final
idgen=" 2 6"
idx= " 7 6"
* declaracion de variable PUBLICAS
  DO PUBLICAS
SELECT  1
use CLIENTES INDEX CLIENTES
arch_gen="CLIENTES "
audit_gen=""
arch_det=""
audit_det=""
SELECT CLIENTES
accion=" "
do PF31.PRG
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  SELE CLIENTES
```

```
  DO lee_datos WITH 1,1,rb
  IF ok
   SELECT CLIENTES
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    GO TOP
    DO despliega WITH 2,datos_gen,rb
   ENDIF
   IF ok
    DO CASE
    CASE accion="A"
     DO lee_datos WITH 2,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
     ENDIF
    CASE accion="B"
     DO opciones
     IF ok
      DO baja WITH primario
     ENDIF
    ENDCASE
   ENDIF
   DO PF31.PRG
  ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
return
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*                 REPORTE DE CLIENTES                   *
* programa: PE32.PRG              autor: SGS/04/09/89   *
*********************************************************
PUBLIC primer_cc,wcc1,gs_c1
*tablas de datos a imprimir
*   ei     encabezado inicial
*   ep,pp encabezado y pie de pagina
*   e#,p# encabezado y pie al corte de control #
*   pf     pie final
*   ld     lineas de detalle del reporte
*   1,1    tipo de impresion/PROCESO
*   2,1    nivel al que se imprime(corte de control)
*   3,2    desplazamiento del renglon actual
*   5,2    columna de impresion
*   7,n    enunciado o formula      o     7,20 variable     27,n PICTURE
* variables para subtotales
* 1,2 nivel        3,2 reSET at          5,n  variable o formula

*              1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789
EP01="FO 4 9DATE()"
EP02="TO 024Sistema para el Control de Pedidos"
EP03="FO 1 9TIME()"
EP04="TO 033REPORTE DE CLIENTES"
EP05="FO 060GS_HOJA PICTURE 'HOJA: 99'"
EP06="FO 109REPL('_',60)"
cont_EP=06
LD01="TO 314CLIENTE: "
LD02="FO 023CLIENTES->CVE_CLIENT"
LD03="FO 028CLIENTES->NOMBRE"
LD04="TO 114DIRECCION: "
LD05="FO 128CLIENTES->DIRECCION1"
LD06="FO 128CLIENTES->DIRECCION2"
LD07="FO 114CLIENTES->TEL PICTURE 'TELEFONO:  999-99-99'"
LD08="FO 043CLIENTES->EXT PICTURE 'EXTENSION: 99999'"
cont_LD=08
* datos para direccionar la impresion
gs_device="PROW()"
IF gs_device="PROW()"
 SET device TO print
 SET print on
 pausa=.F.
```

```
 gs_tope=55
ELSE
 pausa=.T.
 gs_tope=20
ENDIF
*datos generales fijos
primer_cc=.T.
gs_hoja=0
gs_linea=5
gs_cl=gs_tope+1
gs_cortes="0000000000"
* datos generales variables
num_subt= 0
cc_max= 0
* declaracion de variable PUBLICAS
  DO PUBLICAR
* cortes de control, orden del sort
SELECT  1
use CLIENTES    INDEX CLIENTES
SELECT CLIENTES
accion=" "
go TOP
@ 0,0 CLEAR
DO WHILE accion<>"F"
  IF gs_cl > gs_tope
   IF .NOT. gs_hoja = 0
    IF pausa
     DO pausa
    ENDIF
   ENDIF
   gs_hoja=gs_hoja+1
   DO reporta WITH 'EP',cont_ep
   DO cortes_i WITH gs_cortes
  ENDIF
* checa si hay cortes de control y reporta ,e y p
  DO cortes WITH gs_cortes
  DO reporta WITH 'LD',cont_ld
  DO acumula
  SKIP
  IF EOF()
   EXIT
  ENDIF
ENDDO
do cortes_f WITH gs_cortes
```

```
IF gs_device="PROW()"
 EJECT
 SET device TO SCREEN
 SET print OFF
ENDIF
CLOSE DATABASES
RELEASE ALL LIKE w*
return
*fin del programa
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*              MANTENIMIENTO DE  PEDIDOS                *
* forma: PF41.PRG                    autor: SGS/04/09/89 *
*********************************************************
@rb,0 CLEAR
@rb-1,0 SAY ''
*         1         2         3         4         5         6
*123456789012345678901234567890123456789012345678901234567890123456
TEXT
   PEDIDO:                      01 FECHA:
02 CLIENTE:

PRODUCTO          NOMBRE                    CANTIDAD  PRECIO      IMPORTE
03
04
05
06
07
08
09
10
11
12
13
ENDTEXT
```

```
*********************************************************
*        Sistema para el Control de Pedidos             *
*             MANTENIMIENTO DE PEDIDOS                  *
* programa: PE41.PRG                autor: SGS/04/09/89 *
*********************************************************
PUBLIC ok,rc,wrec,cds,accion
* tabla de datos
*  1,2  numero de dato maestro
*  3,20 archivO->variable
* 23,1  tipo variable(C,N,D,L)
* 24,1  tipo captura
*        N-o capturar
*        R-equerido
*        O-pcional
* 25,1  validacion
*        K-llave              L-look up(checa con arch.)
*        D-desplegar          N-inguna
*        C-calcular           c-alcula y despliega
* 26,1  look up(checa con archivo)
*        O-exista             N-no exista
*        V-checar valores
* 27,2  desplazamiento del renglon base(rb)
* 29,2  columna para el dato
* 31,2  tamano del dato
* 33,25 PICTURE
* 58,30 formula de calculo o llave de acceso
* 88,8  archivo para look up
*             1         2         3         4         5         6
*    12345678901234567890123456789012345678901234567890123456789012345678
d01="00SGSWORK->CVE_SERIE  CRL0 010 2@S2
     W01CVE_SER                    SERIES  "
d02="01SGSWORK->CONSECUTIV NRLN 013 4@S4 9999
     W01CVE_SER+STR(W02CONSECU,4)  PEDIDOS "
d03="02PEDIDOS->CVE_PEDIDO CNC0 010 4
     W01CVE_SER+STR(W02CONSECU,4)  "
d04="03PEDIDOS->CVE_CLIENT CRL0 113 4
     W04CVE_CLI                    CLIENTES"
d05="04CLIENTES->NOMBRE    CND0 1 1830                              "
d06="06DPRODPED->CVE_PROD  CRK0 0  3 4
     W03CVE_PED+W06CVE_PRO          "
d07="07SGSWORK->CVE_PROD   CNL0 0  0 4
     W06CVE_PRO                    PRODUCTO"
d08="08PRODUCTO->NOMBRE    CND0 0  830@S30                          "
d09="09DPRODPED->CANTIDAD  NRN0 039 6999,999                        "
```

```
d10="10DPRODPED->CVE_PEDIDOCNCO 0 0 4
     W03CVE_PED                  "
d11="11DPRODPED->PRECIO     NRcO 046 69999,999                    "
d12="12SGSWORK->TOTAL       NNcO 0571299,999,999,999              "

*datos generales
opc=1
repite_det=.T.
dATOS_TOT=12
dATOS_GEN= 5
dATOS_AUDD=12
dATOS_AUDG= 5
id_tot=11
id_gen= 4
rb=gs_rb
ri=10
rf=21
rc=rb
ok=.F.
wrec=SPACE((rf-ri+2)*7)
* 1,2 #dato inicial     3,2 #dato final
idgen=" 2 5"
idx= " 612"

* declaracion de variable PUBLICAS
  DO PUBLICAS

SELECT  1
use CLIENTES INDEX CLIENTES
SELECT  2
use DPRODPED INDEX DPRODPED
SET relATION TO CVE_PROD into PRODUCTO
SELECT  3
use PEDIDOS  INDEX PEDIDOS
SELECT  4
use PRODUCTO INDEX PRODUCTO
SELECT  5
use SERIES   INDEX SERIES
arch_gen="PEDIDOS  "
audit_gen=""
arch_det="DPRODPED "
audit_det=""
SELECT PEDIDOS
accion=" "
```

```
do PF41.PRG
DO WHILE accion<>"F"
 DO opc_abmc WITH opc
 DO WHILE .NOT. accion$"SF"
  msg="OODar datos requeridos"
  SELE PEDIDOS
  DO lee_datos WITH 1,1,rb
  IF ok
   SELECT PEDIDOS
   IF accion$"BCM"
    IF accion="B"
     wrec=STR(RECNO(),7)
    ENDIF
    GO TOP
    DO despliega WITH 2,datos_gen,rb
    SELECT DPRODPED
    GO TOP
    DO despga_de
   ENDIF
   IF ok
    DO CASE
    CASE accion="A" _
     DO lee_datos WITH 2,datos_gen,rb
     IF ok
      DO alta WITH arch_gen
      DO asigna WITH 1,datos_gen,.F.
      DO acepta_de
     ENDIF
    CASE accion="B"
     DO opciones
     IF ok
      DO baja WITH primario
     ENDIF
    ENDCASE
   ENDIF
   DO PF41.PRG
  ENDIF
 ENDDO
ENDDO
CLOSE DATABASES
RELEASE ALL LIKE w*
return
```

```
*********************************************************
*         Sistema para el Control de Pedidos*           *
*                  REPORTE DE CLIENTES                  *
* programa: PE32.PRG              autor: SGS/04/09/89   *
*********************************************************
PUBLIC primer_cc,wcc1,gs_cl
*tablas de datos a imprimir
*   ei     encabezado inicial
*   ep,pp  encabezado y pie de pagina
*   e#,p#  encabezado y pie al corte de control #
*   pf     pie final
*   ld     lineas de detalle del reporte
*   1,1    tipo de impresion/PROCESO
*   2,1    nivel al que se imprime(corte de control)
*   3,2    desplazamiento del renglon actual
*   5,2    columna de impresion
*   7,n    enunciado o formula      o    7,20 variable    27,n PICTURE
*             1         2         3         4         5         6
*     12345678901234567890123456789012345678901234567890123456789
EP01="FO 4 9DATE()"
EP02="TO 024Sistema para el Control de Pedidos"
EP03="FO 1 9TIME()"
EP04="TO 030REPORTE DE PEDIDOS"
EP05="FO 060GS_HOJA PICTURE 'HOJA: 99'"
EP06="FO 109REPL('_',60)"
EP07="TO 214CLIENTE: "
EP08="FO 023CLIENTES->CVE_CLIENT"
EP09="FO 028CLIENTES->NOMBRE"
EP10="TO 114DIRECCION: "
EP11="FO 028CLIENTES->DIRECCION1"
EP12="FO 128CLIENTES->DIRECCION2"
EP13="FO 114CLIENTES->TEL PICTURE 'TELEFONO:  999-99-99'"
EP14="FO 043CLIENTES->EXT PICTURE 'EXTENSION: 99999'"
EP15="TO 209PRODUCTO      DESCRIPCION       CANTIDAD"
EP16="TO 049PRECIO     T O T A L"
EP17="FO 109REPL('=',60)"
cont_EP=17
LD01="FO 109PRODUCTO->CVE_PROD"
LD02="FO 014PRODUCTO->NOMBRE "
LD03="FO 040DPRODPED->CANTIDAD PICTURE '99,999'"
LD04="FO 046DPRODPED->PRECIO PICTURE '999,999.99'"
LD05="FO 057DPRODPED->PRECIO*DPRODPED->CANTIDAD
      PICTURE '99999,999.99'"
cont_LD=05
```

```
P101="TO 157------------"
P102="TO 133TOTAL DEL PEDIDO: "
P103="FO 057ST03 PICTURE '99999,999.99'"
CONT_P1=03
PF01="TO 320* * *  GRACIAS POR SU COMPRA  * * *"
CONT_PF=1
* datos para direccionar la impresion
gs_device="PROW()"
IF gs_device="PROW()"
 SET device TO print
 SET print on
 pausa=.F.
 gs_tope=55
ELSE
 pausa=.T.
 gs_tope=20
ENDIF
*datos generales fijos
primer_cc=.T.
gs_hoja=0
gs_linea=8
gs_c1=gs_tope+1
gs_cortes="0000000000"
* datos generales variables
num_subt= 3
cc_max= 1
cont_e1=0
* declaracion de variable PUBLICAS
  DO PUBLICAR
* cortes de control, orden del sort
* variables para subtotales
* 1,2 nivel       3,2 reSET at          5,n  variable o formula
S01="0001DPRODPED->CANTIDAD"
S02="0001DPRODPED->PRECIO"
S03="0001DPRODPED->PRECIO*DPRODPED->CANTIDAD"
CC1="DPRODPED->CVE_PEDIDO"
SELECT  1
use CLIENTES    INDEX CLIENTES
SELECT 2
USE PRODUCTO INDEX PRODUCTO
SELECT 3
USE DPRODPED INDEX DPRODPED
SET RELATION TO CVE_PROD INTO PRODUCTO
SELECT 4
```

```
USE PEDIDOS INDEX PEDIDOS
SET RELATION TO CVE_CLIENT INTO CLIENTES
SELECT DPRODPED
accion=" "
go TOP
@ 0,0 CLEAR
DO WHILE accion<>"F"
  IF gs_ci > gs_tope
   IF .NOT. gs_hoja = 0
    IF pausa
     DO pausa
    ENDIF
   ENDIF
   gs_hoja=gs_hoja+1
   DO reporta WITH 'EP',cont_ep
   DO cortes_i WITH gs_cortes
  ENDIF
* checa si hay cortes de control y reporta ,e y p
  DO cortes WITH gs_cortes
  DO reporta WITH 'LD',cont_ld
  DO acumula
  SKIP
  IF EOF()
   EXIT
  ENDIF
ENDDO
do cortes_f WITH gs_cortes
DO reporta WITH 'PF',cont_pf
IF gs_device="PROW()"
 EJECT
 SET device TO SCREEN
 SET print OFF
ENDIF
CLOSE DATABASES
RELEASE ALL LIKE w*
return
*fin del programa
```

```
*********************************************************
*         Sistema para el Control de Pedidos            *
*                  rutina de instalacion                *
* programa: PEINSTAL.PRG          autor: SGS/04/09/89   *
*********************************************************
@ 0,0 CLEAR
!del *.ndx
SET safety off
SELECT 1
USE clientes
   zap
   INDEX on cve_client TO clientes

USE dprodped
   zap
   INDEX on cve_pedido+cve_prod TO dpedprod
   INDEX on cve_prod+cve_pedido TO dprodped

USE pedidos
   zap
   INDEX on cve_pedido TO pedidos

USE producto
   zap
   INDEX on cve_prod TO productos

USE series
   zap
   INDEX on cve_serie TO series

? "INSTALACION DE ARCHIVOS INDICE EFECTUADA"
SET safety on
@ 0,0 CLEAR
```

V.- APLICACIONES Y PERSPECTIVAS.

1).- El progreso tecnológico.

a) Tecnología de la información como una arma competitiva.

La tecnología de la información es vista en muchas industrias y sectores industriales como una arma competitiva. Es una parte integral de un plan estratégico. No hay sistemas de información puramente estratégicos, ni estrategias puramente financieras.

La estrategia está basada en una visión clara y elecciones directivas no ambiguas. cuando la administración (dirección) no puede plantearse metas, productos, sistemas, posición competitiva y un plan concreto, esta no puede progresar.

El continuo avance tecnológico lleva consigo una aceleración de los ciclos de vida. Por eso no debemos proyectar sistemas con más de 5 años de vida. Esto implica por consiguiente:

1. Alta productividad en desarrollo de software.
2. Tiempo de desarrollo menor a 4 meses, extendiéndose máximo a medio año.
3. Bajo costo de componentes (Software/hardware) y política abierta de ventas.
4. Soporte para nuevas funciones, anteriormente ofrecidas en forma manual o no del todo.
5. Las funciones ya basadas en la computadora deben ser reestructuradas, modernizadas, personalizadas e integradas.

Repitiendo el aviso arriba mencionado; no diseñar un sistema nuevo, a menos que se tenga una noción del próximo.

La información es generalmente la entrada y salida del conocimiento de los trabajadores. Como el mundo de negocios viene más complejo, éste usa y genera más información, esto es:

- Causa cuellos de botella en la oficina.
- Desvía a la dirección de acciones importantes.
- Impide productividad.

Para poder adoptar un sistema de información, deberán resolverse las siguientes 6 preguntas:

1.- ¿Donde estamos?
recursos actuales, fuerza y debilidades.

2.- ¿Porque cambiar?
para explotar nuevas oportunidades, identificar perspectivas, evolución del mercado y desarrollo tecnológico. Para sobrevivir, debemos posicionar nuestra compañía con las fuerzas del futuro.

3.- ¿Qué podemos hacer?
resultados estratégicos, obstáculos, riscos, reservas, oportunidades y costo-beneficio son parte de la respuesta a darse para esta pregunta.

4.- ¿Que podríamos hacer?
debemos hacer elecciones básicas, evaluar componentes, analizar políticas de toma de decisiones, proceder con integración del sistema y elaborar posibles estrategias de implementación.

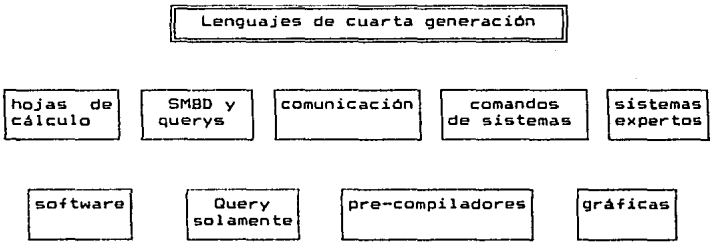
5.- ¿Como llegamos ahí?
Para responder debemos considerar relaciones con clientes, evolución del producto, distribución de recursos, desarrollo de sistemas de negocios, proyectos gerenciales de sistemas de información, prioridades y entrenamiento.

6.- ¿Ya llegamos?
Esta es la retroalimentación. Necesitamos ser capaces de aplicar una acción correctiva para llegar a la meta. Debemos tener métricas, mediciones y valuación.

b) Tecnología de computadoras; hoy y mañana en lenguajes de cuarta generación.

En ambiente de los mainframes, necesitamos personal especializado para programar y operar la máquina. En una época de personal de cómputo, cualquiera de nosotros podía ser el operador y programador de una máquina. Este es el propósito que tienen los lenguajes de cuarta generación. Ellos abren el hasta ahora exclusivo análisis de sistemas y ambiente de programación para cualquier usuario.

Por tanto, los cercanos ofrecimientos de lenguajes de cuarta generación que pensamos son solo para usuarios finales. En realidad, son mucho más que esto. ¿Pero hay una valuación realista de lenguajes de cuarta generación?¿Cual es el rango de ofrecimientos?¿Qué puede venir próximamente por el camino de la evolución?, esto puede verse en la siguiente figura:



SMBD - Sistemas Manejadores de Bases de datos.

Figura 5-1 : Areas de aplicación de los lenguajes de cuarta generación.

Dentro de las hojas de cálculo, podemos distinguir alrededor de dos docenas, de las cuales sobresalen:

- Multiplan
- Supercalc 2
- Visicalc IV
- Lotus 123

Visicalc fue la versión original que abrió amplias perspectivas en el enfoque de los usuarios.

Las hojas de cálculo son productos cómodos para ser muy usados en refinamiento, examinar, analizar y presentar

información. Su meta es asistir a la persona que toma decisiones a través de paquetes de soporte modulares, bien dirigidos y amigables que puedan ser obtenidos en el mercado, a un precio muy razonable.

Dentro del Software integrado se deben de tener valores adicionales, como ejemplos de estos tenemos a Lotus 123, Symphony, Supercalc 3, Intuit y Dicision Manager. Aqui también hay generaciones; Symphony (1984) es la segunda generación de Lotus 123 (1983).

SMBD y QUERY's incluyen
- Ingres
- Oracle
- Informix
- DB 2
- SQL
- Mapper
- Natural
- ADS
- Powerhouse
- Speedware
- Mating en nivel super-micro a mainframe

PFS:file, MDBS y Dbase III (plus) son también los sistemas personales que ofrecen ayuda a los usuarios de PC's.

Ejemplos de lenguajes Query solamente son:
- ADR5/ADI
- SAS

Ejemplos de lenguajes de comunicación de datos:
- PLPS (con videotex)
- Idel
- Visianswer
- NIL, un nuevo lenguaje orientado a redes de IBM

Los precompiladores vienen bajo diferentes nombres y son desarrollados en diferentes máquinas:

ADF, Focus y Ramis II para mainframes, IBM's application system (AS) es un producto de primera clase en los Sistemas manejadores de bases de datos relacionales, trabajando esto en mainframes y PC's.

The Information Facility (TIF) de IBM, corre bajo MVS/TSO y VM/CMS pero también en la PC XT/370.

En Comandos de sistemas, algunos de los mejores se encuentran en el ambiente de UNIX, pero otros enfoques se encuentran en desarrollo.

En gráficas, tenemos herramientas CAD/CAM que también estan soportadas en software integrado.

La programación puede ser hecha a través de una capacidad propia y sencilla de selección de menús. Esto puede ser presentado a los usuarios en plan ingles, o a través de iconos, en ambos casos se habla de programación en lenguaje natural, el cual está a la mitad del camino, entre lenguajes de cuarta y quinta generación. Para decidir en qué camino deberemos clasificarlo, debemos conocer cuanto abarca la inteligencia artificial en esta construcción. El mejor ejemplo de inteligencia artificial en acción son los sistemas expertos, el rango de aplicaciones en este campo es ya impresivo, particularmente en medicina e ingeniería.

Algunos lenguajes para desarrollo de sistemas expertos son EMYCIN y Prolog. Para cada uno de estos lenguajes de cuarta y quinta generación, hay un nicho en el medio ambiente de programación de hoy y mañana.

c) Lenguajes de cuarta generación y usuarios finales de computadoras.

En sistemas de información como en otros campos, para sobrevivir en un mercado competitivo, debemos estabilizar metas y objetivos. Debemos estudiar el flujo de trabajo, simplificar, consolidar, estabilizar radios de productividad e identificar áreas prioritarias que pueden ser racionalizadas, a través de computadoras y comunicaciones.

Las prioridades son situacionales y deben ser definidas por la Gerencia después de un estudio documentado y realista. En el caso general, las oportunidades se verán como estas:

1.- Procesamiento de palabras y datos integrado.

2.- Nuevos enfoques de captura y reporte de textos y datos:
   - captura en el punto de origen.
   - reporte en el punto destino.
3.- Correo electrónico y conferencias por computadora. El correo electrónico, es la primera función capaz de ayudar en comunicaciones entre gerentes y profesionistas. Las conferencias por computadora, es lo más moderno y un poco más complejo. La necesidad de esto es evidenciado por el hecho de que, más gente quiere comunicarse con la computadora que usarla para cómputo.
4.- Gráficas y capacidad de color. Si las hojas de cálculo son las primeras y las mejores herramientas para la productividad ejecutiva en el nivel personal de cómputo, las gráficas y fácilidades de color reportan y deciden sistemas de soporte.

Las estaciones de trabajo para comunicación también, ofrecen a sus usuarios valores adicionales como: servicios de calendario (agendas), comunicaciones de micro a mainframes y acceso a base de datos públicas.

En todas estas áreas, habilitación básica, diálogo gerencial, e integración tecnológica son la pavimentación del camino de la productividad. No solo necesitamos identificar propiamente los mejores sectores para automatización completa, debemos también levantar e implementar medidas que aseguren la calidad.

Necesitamos arquitecturas de sistemas que nos den una estabilidad básica para planeación, dar una dirección común para distribuir desarrollos y hacer posible la integracion de software y hardware para muchos fabricantes. Esta arquitectura debe asegurar que la tecnología será hecha para trabajar como la gente lo hace.

La gerencia debe interesarse en estabilizar la dirección correcta, porque los sistemas de información centralizada pueden ser inflexibles y también complicados. El personal clásico de procesamiento de datos, dedica su tiempo al desarrollo de sistemas y la mayor parte al mantenimiento de los mismos, ellos no se direccionan propiamente a las actividades vitales para los 80's:

1.- Integrar componentes de software y hardware

2.- Automatizar el lugar de trabajo
3.- Proveer comunicaciones eficientes
4.- Asegurar base de datos interactivas
5.- Reducir costos de labor y cargas de papel

El argumento no es escoger entre PC's y mainframes, este es un concepto desordenado; la pregunta crítica está fuera de lugar. El punto de partida deben ser ambas, PC's y mainframes. Luego la pregunta crítica viene:¿Como podemos integrarlas?, intentar hacer cualquier cosa en el mainframe es irracional; el más pequeño problema, causará un largo trabajo. Más del 90% del esfuerzo va dentro de las conexiones ambientales y menos del 10% en el problema.

Solo los proyectos muy largos deben ser puestos en los mainframes. ¿Pero deben ser verdaderamente muy largos?, los lenguajes de cuarta generación nos dan la fácilidad de abordarlos, haciéndolos razonablemente manejables.

En 1985 el autor(*9) contribuyó a la colección de papel comercial y un proyecto de despacho automático, que en otros bancos tenian ya dedicado 10 años de esfuerzo-trabajador en el clásico significado; Cobol y programación estructurada. Al usar Ingres en la supermicro, la cual fue elegida como la máquina despachadora automática y la PC como una estación personal de trabajo (con interface en Basic). Este proyecto fue completado en dos meses trabajador.

El mismo proyecto, a dos especialistas trabajando juntos (desarrollando prototipos a través de Ingres) les tomaria una semana desarrollarlo. Subsecuentemente, solo un especialista fue envuelto. El diseño y programación de la base de datos propia le tomó 3 semanas y programando para mejorar el tiempo de respuesta 1 semana mas. Se utilizarón otras 4 semanas para la programación en Basic de la interface para la PC.

Los mainframes deben ser reservados para 2 actividades en que ni la PC , ni la supermicro resultan apropiadas. La primera es almacenar y manejar las bases de datos que necesitaremos en el futuro. La segunda es actuar como interruptores gigantes de

(*9) Dimitris N. Chorafas, Forth and Fifth Generation Languaje V.II.

comunicación , para las estaciones de trabajo inteligentes que serán instaladas en todos los escritorios y LANs que se interconectarán. Cada tipo de máquina en computadoras y comunicaciones, necesitará un nivel de lenguaje de programación correspondiente a sus capacidades. Esto es por lo que se dice que los lenguajes de cuarta generación son mucho más que herramientas orientadas a usuarios finales.

Tres tendencias tienen que venir evidentemente en la profesión del procesamiento de datos, son debidas a la nueva tecnologia del software:

1.- Incrementar conocimientos de la aplicación y la habilidad de diseñar sistemas completos.
2.- El procesamiento de datos y sistemas para manejo de información gerencial, vienen más estrechamente integradas por completo en los equipos expertos en manejo de sus organizaciones y desistiendo de ser un técnico extraño.
3.- Reduciendo costos de desarrollo de aplicaciones con un menor orden de magnitud.

Esta reducción de costos no se aplica solo a nuevos desarrollos, con tan drástico mejoramiento de productividad, esto viene a remplazar económicamente los existentes programas de aplicación en lenguajes de alto nivel (Cobol, Fortran, PL/1, etc.). Y así, mejorar procedimientos y envolver directamente a los usuarios finales durante la fase de desarrollo. En este aspecto, los programas de aplicación es mas probable que cumplan los requerimientos del usuario. Los usuarios serán capaces de controlar el software de aplicación y también contribuir con el conocimiento de sus necesidades para hacerlo.

2).- Análisis del mercado y aplicaciones.

Una de las ventajas del Sistema Generador de Sistemas(SGS), es el mercado potencial de aplicación que puede alcanzar, puesto que esta versión fué implementada en el área de las microcomputadoras. Dentro de este medio, existe una gran mayoria de usuarios que aún no aprenden a manejar su "micro", por lo cual resultaria atractivo para ellos utilizar el Sistema Generador de Sistemas como herramienta para desarrollo de sistemas.

Sabemos que el sistema generador de sistemas, fue enfocado al ambiente de microcomputadoras, est lo hace caer dentro de un mercado inmensamente grande. En la actualidad, en México existen aproximadamente 500,000 computadoras personales, esta cifra la obtenemos de considerar que el 5% de la población posee una PC's. Con un mercado de esta escala, podemos pensar que el SGS podria introducirse paulatinamente, dependiendo de la aceptabilidad que tenga con los usuarios. Supongamos que el SGS sea aceptado por el 1% de los usuarios de PC's, esto representaria 5,000 copias instaladas que a un costo de $500 dolares seria 2,500,000 dolares que a la paridad de $3,300 pesos seria $8,250,000,000 pesos.

Como aplicaciones del Sistema Generador de Sistemas podemos tener basicamente el desarrollo de sistemas comerciales como:

- Nómina
- Contabilidad
- Facturación
- Pedidos
- Cuentas por Cobrar
- Seguros
- Otros

Como experiencia de aplicaciones del Sistema Generador de Sistemas, se tiene:

a) Sistema para el control de cotizaciones de pólizas de seguros para la compañía Segmebe, S.A.

En este sistema, se aplicó en una pequeña parte, pues el proyecto ya tenia un gran avance, por lo cual resultaba incosteable reemplazar los programas existentes, además de que el grado de complejidad y el tipo de funciones a realizar, no eran muy adecuados para el Sistema Generador

de Sistemas.

b) Se aplica también el manejador de menus, pues la filosofia que se tiene de menus por ventanas sobrepuestas y selección de opciones mediante las teclas del movimiento, daban mayor versatilidad y presentación al sistema.

Se ven futuras aplicaciones, puesto que hay gran demanda de desarrollo de sistemas para PC's en Dbase.

Dentro del hardware se tiene una visión muy amplia de futuros avances, incluyendo LANS. Las supercomputadoras deben considerar:

- LANS
  - Acceso a la base de datos central.
  - Facilidad de comunicación.

El objetivo de los proyectos de 5a. generación es diseñar y producir hardware y software para ingenieria del conocimiento en un amplio rango de aplicaciones:

- Sistemas expertos
- Entendimiento de lenguaje natural por la computadora
- Robótica

Para todas estas aplicaciones necesitamos más que un incremento dramático en las capacidades de las computadoras; también necesitamos innovaciones en la tecnología existente, que permitan a las nuevas generaciones de computadoras:

- Soportar bancos de conocimiento muy largos
- Permitir recuperaciones asociadas muy rápidas
- Ejecutar operaciones de inferencia lógica, tan rápido como las computadoras actuales realizan las operaciones aritméticas.
- Utilizar paralelismo para mejorar la velocidad de ejecución los programas.
- Desarrollar interfaces hombre-máquina que permitan un uso significante de imágenes y reconocimiento natural de voz(Speech).

3.- La necesidad de entrenar los recursos humanos.

Las herramientas están aquí, pero ¿Estamos listos para usarlas?. La gran pregunta es: ¿Pueden absorber los usuarios y los profesionistas de computación todos los nuevos desarrollos y descubrimientos(breakthrougs)?. Cuando algunas corporaciones hacen grandes esfuerzos por mantener sus sistemas de información tan avanzados como sea posible, otros se hechan para atrás en forma deliberada. Estos son aplastados por las necesidades masivas de reentrenar y renovar sus recursos humanos.

El entrenamiento en nuevas tecnologías es entendido como una tarea debastadora. Todavía hay plenitud de evidencia que sin entrenamiento masivo y subsecuentes implementaciones manuales (hand'on implementations), la ganancia para los lenguajes de cuarta generación estará limitada.

- Debemos de conceptualizar no solo el costo/beneficio de proyectos de racionalización.
- También se debe asignar un grado mayor al personal.

Porque la tecnología se mueve muy rápido y la orientación del mercado es muy cambiante, debemos de estar siempre alertas. Si tenemos un buen sistema y no lo actualizamos, sorpresivamente se volverá ineficiente y dejará la labor de las tareas intensivas, en pocas palabras se volverá obsoleto. Esto mismo es aplicable al personal. A corto plazo, debemos organizar un plan para la administración del conocimiento, tomando en cuenta las divergencias que se pueden dar por:

- El poder basado en la posición.
- El poder basado en el conocimiento.

El personal de cómputo tiene conocimientos que pueden volverse rápidamente obsoletos. La oportunidad de adquirir nuevos conocimientos y aplicarlos, es una de las mayores razones de cambio, incluyendo por supuesto, la rotación del personal.

Se debe capacitar a todo el personal con nivel gerencial, para ello, en la tabla 5.2 se muestra el temario que nos puede ayudar a cumplir este objetivo, en este curso se pretende dar una capacitación general sobre computación al nivel adecuado.

Tabla 5.2 : Curso de experiencias manuales para gerentes.

Primer dia	Segundo dia
1. Programa de estaciones de trabajo. . Acceso autorizado a BD . Comunicaciones . Tiempo de administración 2. Concepto de automatización de la oficina . Software, Hardware . Documentación . Floppy con datos . Usando la computadora 3. Experiencia en hojas de cálculo . Planeación de salarios . Moldeo de la admimistración . Transacciones contables . Reformas fiscales . Caja de ahorro del personal . Inversión contable . Presupuesto y control . Desarrollo de pronósticos	1. Gráficas . Introducción al manejo de gráficas y herramientas de decisión . Ejercicios de tabulación y graficación 2. Procesador de palabras . Correo electrónico . Servicios de agendas
	Tercer dia
	1. Redes . Protocolos de comunicación . LANS . Aplicación de correo electrónico . Comunicación con clientes 2. Acceso a bases de datos . Autenticidad-autorización . Acceso a BD propias . Acceso a BD públicas 3. Reacciones, deseos y recomendaciones

VI.- CONCLUSIONES.

El objetivo principal del Sistema Generador de Sistemas, es cubrir la mayor parte de los requerimientos del desarrollo de sistemas.

Por principio de cuentas, no se cambia el análisis y diseño clásico de sistemas. Las técnicas anteriormente utilizadas para estos puntos en general, son muy buenas, aunque para realizar un buen análisis y diseño de un sistema, no basta con conocerlas, sino que la persona que realice estas actividades, deberá ser muy diestro en estos menesteres, o bien deberá ser supervisado por alguien experimentado.

Algo nuevo y de gran utilidad que introducen los lenguajes de cuarta generación, es el concepto de **prototipos**. El sistema generador de sistemas también permite el uso de este concepto, por lo cual posee gran versatilidad para hacer adecuaciones a las aplicaciones que se desarrollen en el.

Entiéndase por un prototipo , la implementación parcial de un sistema en algún lenguaje de cuarta generación. El prototipo no pretende cubrir todos los requerimientos de proceso del sistema(aunque sería lo óptimo), sin embargo, se acerca bastante a la implementación real. La idea de los prototipos es presentar a los usuarios una imagen cercana al sistema real, para que estos sugieran cambios y hagan observaciones que ayuden a mejorar el prototipo y lo acerquen más a la implementación real. Por medio de esta retroalimentacion entre el usuario y los profesionistas de cómputo, formamos un equipo de los mejores para el desarrollo de sistemas, ya que uno sabe lo que quiere y otro sabe como implementarlo. Al formar equipos con estos integrantes, estamos ahorrando al especialista de cómputo, tiempo de aprendizaje de las funciones que debe cubrir el sistema.

Dentro de mis experiencias con lenguajes de cuarta generación, he comprobado que el desarrollo de prototipos no es tan sencillo. Durante la implementación del prototipo inicial, generalmente se presentan problemas por limitaciones del lenguaje y del programador. El programador sigue pensando en **como hacer las cosas como en un lenguaje de tercera generación**, pretende dar

un manejo muy similar a como lo hacia anteriormente. Este problema debe arrancarse de raiz, se debe cambiar la filosofia de programación pensando ya en cuarta generación, esto quiere decir que debemos dejar que el lenguaje haga lo que sabe hacer, sin obligarlo a hacer lo que queremos que haga. Ya no debemos preocuparnos por detalles de contadores, totalizadores, funciones de mantenimiento, reportes, etc., pues la mayoria de esto ya es contemplado por el lenguaje.

En base a una de mis experiencias con lenguajes de cuarta generación, les puedo mencionar que no es tan cierto que mejoran la productividad de sistemas. En el proyecto de un sistema para la cotización de pólizas de seguros, para la compañia Brockman y Schuh, S.A. de C.V., se utiliza para su desarrollo el lenguaje Speedware. Se desarrolla el prototipo del sistema, el cual al ser presentado a los usuarios sufre varios cambios, el tratar de implementar estos cambios, resultó una tárea muy dificil y en ocasiones imposible si se respetaba el diseño original del sistema. De esto se desprende que no tan solo con tener un lenguaje de cuarta generación mejoraremos nuestra productividad, hay que saber como utilizarlo más eficientemente y sobre todo, el momento adecuado en que debemos recurrir a otro lenguaje. Si esta aplicación se hubiera desarrollado en Cobol, el avance del proyecto seria un poco mejor, pues no hubieramos tenido que adecuar el diseño del sistema al lenguaje de programación, reflejándose esto indudablemente en tiempos de desarrollo.

En cambio, en otros lenguajes de cuarta generación tenemos mayor versatilidad, por lo cual hacer adecuaciones a un prototipo es una tárea un poco sencilla. También en base a mi experiencia, enmarco dentro de estos a Powerhouse, el cual tiene bases muy firmes para cumplir con los requerimientos y caracteristicas de un lenguaje de cuarta generación.

Existen también otros lenguajes de cuarta generación que sobresalen por otras caracteristicas, dentro de ellos tenemos a Oracle, el cual ha ganado popularidad en base a la compatibilidad de su diseño con diferentes equipos de cómputo. Esta compatibilidad, hace que cualquier aplicación desarrollada en Oracle en algun equipo, sea totalmente transportable a otro equipo en el que corra Oracle.

En cuanto a Dbase, lo considero como un lenguaje de 3ª y ½a. generación, ya que incorpora instrucciones que permiten una

programación al estilo de la tercera generación y algunas otras con bastante poderío que, en algunos casos supera el de los lenguajes de cuarta generación. Los fabricantes de Dbase siguen dando muestras de su interés en mejorar el producto, por esta razón, recientemente liberaron la versión Dbase IV, la cual incorpora al lenguaje SQL, además de mayores facilidades para los usuarios.

La mayoría de los fabricantes de lenguajes, se preocupan por mejorar su producto. El grito de la moda en computadoras personales, son los lenguajes en versión TURBO, los cuales tienen mayor velocidad de ejecución e incorporan un mayor número de instrucciones, dentro de estos tenemos:

- Turbo C
- Turbo Pascal
- Turbo Basic
- Turbo Prolog

Con estas mejoras, los lenguajes de cuarta generación no son tan fácilmente aceptados en el área de PC's, ya que existen programadores expertos en los lenguajes mencionados que confían ciegamente en ellos. La popularidad de que goza Dbase y el tiempo de respuesta de un lenguaje Turbo, difícilmente permitirán que ingresen los lenguajes de cuarta generación al ambiente de PC's.

El costo-beneficio de adquirir un lenguaje de cuarta generación, se puede definir tomando en cuenta lo siguiente:

COSTOS:

- Mayor consumo de recursos (memoria, espacio en disco).
- El tiempo de ejecución de los programas es mayor.
- En general, el rendimiento del equipo de cómputo (performance) se ve afectado.
- En los casos de implementaciones complejas, es necesario recurrir a otros lenguajes para realizar ciertas tareas.

BENEFICIOS:

- El tiempo de desarrollo de aplicaciones se reduce hasta en un 80%, dependiendo de la complejidad.

- Se proporcionan herramientas fáciles de usar, que ayudan en sus tareas a los usuarios y programadores inexpertos.
- Los usuarios pueden participar más estrechamente en el desarrollo de los prototipos, por consecuencia, las aplicaciones se adecuarán a sus necesidades.
- Se permite la interface con otros lenguajes, para que en caso de tener aplicaciones en ellos, se puedan integrar al prototipo. Al utilizar otros lenguajes como complemento de los lenguajes de cuarta generación, se pueden cubrir deficiencias de los mismos, optimizando así su comportamiento en puntos críticos. Se dice que una aplicación generalmente se desarrolla en un 98% mediante los lenguajes de cuarta generación y el 2% restante utilizando lenguajes de tercera generación. Esto es debido a las "pocas cosas que un lenguaje de cuarta generación no puede hacer".

- En una aplicación desarrollada en lenguajes de cuarta generación, se poseen ayudas en línea sin mayor esfuerzo de los programadores. Este es un punto muy importante puesto que los usuarios generalmente no leen los manuales del sistema, quieren aprender dentro de el. Adicionalmente, estas ayudas en línea pueden ser impresas para formar los manuales de usuario de la aplicación, tal es el caso de Speedware que genera manuales de documentación de usuario en varios idiomas.

- Otro punto importante es el manejo relacional de información que permiten algunos lenguajes de cuarta generación.

El Sistema Generador de Sistemas, trata de brindar la mayoría de las facilidades de los lenguajes de cuarta generación, orientando al usuario sobre la manera en que debe implementarse metódicamente un sistema:

1. Primero debe de tener una idea general del sistema, para que pueda definir el organigrama del mismo, registrandose en el Sistema Generador de Sistemas como menus.

2. Definir los datos y archivos que intervendrán en la aplicación.

3. Describir las pantallas y funciones deseadas en su

aplicación, las cuales debieron ser previstas dentro de los menus.

4. Se generan todos los programas y archivos necesarios para el funcionamiento de la aplicación.

El Sistema Generador de Sistemas, pretende brindar facilidades adicionales a los lenguajes de cuarta generación, tomando el concepto de la generación de programas. Esta generación se hace en base a lo especificado por el usuario al sistema.

Para el caso de esta versión, los programas son generados en lenguaje Dbase, puesto que en ambiente de PC's es el lenguaje más popular, por lo cual podria ser el de mayor aceptación, aunque no el mejor, contando con el manejo relacional de información que permite Dbase.

La principal desventaja de esta versión, es el tiempo de ejecución, puesto que Dbase tiene un tiempo de respuesta lento, por utilizar rutinas comunes programadas en Dbase. En cuanto a este punto, se pueden hacer mejoras reprogramando estas rutinas en un lenguaje de más bajo nivel, que incrementará al máximo esta velocidad de ejecución.

Como se mencionó anteriormente, los programas generados se encuentran en lenguaje Dbase, razón por la cual al ejecutarlos el tiempo de respuesta será igual o peor que la de Dbase. Pero recordemos que para mejorar el tiempo de ejecución de programas en Dbase, se tienen Foxbase y Clipper que nos ayudan a amortiguar esta desventaja.

La implementación del Sistema Generador de Sistemas mejorara en gran medida, cuando sea programado en un lenguaje de ejecución rápida y además los programas se generen en el mismo lenguaje, o bien en un pseudo lenguaje natural que al traducirse nos diera el programa equivalente en el lenguaje elegido. Esto es, integrar un lenguaje de cuarta generación al Sistema Generador de Sistemas.

Para que el Sistema Generador de Sistemas llegue a ser popular, primeramente se debe pensar en una versión comercial que prevea un mayor número de facilidades a los usuarios. Debe verse rodeado de un ambiente amigable, que permita la ejecución de diversas actividades para el diseño del sistema, orientando a los

usuarios en el transcurso de las mismas. Tomando en cuenta lo anterior y haciendo una elección adecuada del lenguaje de programación que se debe utilizar en la implementación, de tal manera que permita una ejecución rápida, sea de facil programación, sin olvidar la portabilidad del lenguaje y la popularidad del mismo, el Sistema Generador de Sistemas podría llegar a ser una herramienta muy buena para el desarrollo de sistemas.

Pero no hay que olvidar que los lenguajes de cuarta generación no son capaces de mejorar a los programadores. Ni la inteligencia artificial ni otras herramientas pueden sustituir la inteligencia de un programador. Cualquier empresa que desee incrementar la productividad de sus programadores, primeramente deberá proporcionarles las mejores herramientas de programación, e inmediatamente después darles la capacitación necesaria para una eficiente utilización de las mismas. Si se descuida este último punto, los programadores que desconozcan los conceptos básicos de estas herramientas, no serán capaces de utilizarlas de manera efectiva, originándose asi el concepto de que las herramientas de programación no son adecuadas para la aplicación en el mejor de los casos, o bien que no sirven. Existen varios lenguajes de cuarta generación que aún no demuestran, su superioridad ante los lenguajes de tercera generación. Dentro de éstos podemos mencionar ADA, PL/1 de IBM y Speedware como la habiamos mencionado.

Por otra parte, un programador muy eficiente que resuelva sus problemas sin importar la herramienta utilizada, con lenguajes de cuarta generación será mucho más eficiente. Al observar su comportamiento, le serán asignadas tareas que requieran mayores aptitudes y desarrolle sus capacidades, por lo cual las tareas de programación serán asignadas a personas menos capaces o incapaces de utilizar las herramientas de programación de una manera adecuada. Si los programadores no se sienten capaces de utilizar estas herramientas, ya sea por falta de capacitación, o bien porque la herramienta no es buena, se tiende a almacenar dichas herramientas, pues por presiones de trabajo que tiene el programador, no puede detenerse a experimentar con las "mejores herramientas de programación", pues el desarrollo de la aplicación no se puede retrasar.

BIBLIOGRAFIA.

1).- Procesamiento automático de datos(principios y procedimientos),
Elias M. Awad,
Ed. Diana,
México, 1976.

2).- Apuntes de computadoras y programación,
Facultad de Ingenieria U.N.A.M.

3).- Lenguajes de Programación,
Allen B. Tucker,
Ed. McGraw-Hill 2ª edición,
México 1987.

4).- Fourth and Fifth Generation Programing Languaje,
Dimitris N. Chorafas,
Ed. McGraw-Hill 1986,
V.I y V.II.

5).- Dbase III Plus, Herramientas Poderosas,
Rob Krumm,
Ed. Mcgraw-Hill 1988.