UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMAS DE APOYO PARA LA DOCUMENTACION Y MANTENIMIENTO DE SISTEMAS UTILIZANDO METODOLOGIAS FORMALES DE DESARROLLO

T E S I S

Que para obtener el Título de:

INGENIERO EN COMPUTACION

P r e s e n t a :

MAURICIO HORACIO VILLANUEVA CASTELLANOS

DIRECTOR: ING. ARTURO E. ROJAS FLORES

CO-DIRECTOR: ING. ALEJANDRO RAMIREZ LOZADA

MEXICO, D. F. 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

I.- I N T R O D U C C I O N

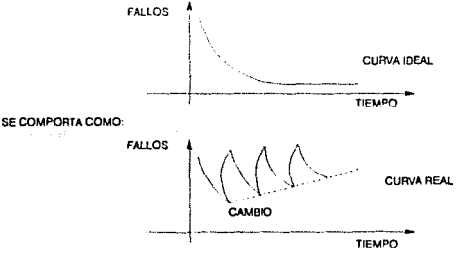
En las inicios de la computación, el software no jugaba un papel muy importante. El software que se desarrollaba era a la medida, es decir, no se tenía ninguna distribución.

Generalmente la persona que desarrollaba el software, se encargaba de depurarlo y realizar las corecciones si eran necesarias, por lo que no se contaba con una documentación formal.

Conforme el tiempo fue avanzando, surgieron mejoras en los componentes de hardware y en consecuencia los de software; se comenzó a desarrollar software con miras a una amplia distribución de miles de usuarios, pero con éste crecimiento del software se generó un problema, y era que los programas desarrollados tenían que ser corregidos cuando se encontraban fallas o se tuviera un requerimiento de ajuste por parte de los clientes, por lo que la actividad de mantenimiento se incremento.

El costo del mantenimiento del software comenzó a crecer y lo que antes se consideraba de poca importancia, se volvió alarmante. Para darse una idea, en los años 70's el costo de mantenimiento era entre el 35 y 40% del costo de desarrollo, en los 80's entre el 40 y 60% y en los 90's el porcentaje llega hasta el 80%; por ello es de gran importancia el prestar una mayor atención en el desarrollo de un producto para poder reducir éstos costos.

Una de las causas por las que se presentan estos costos tan elevados, es que a pesar de que un software idealmente debería comportarse de la manera:



Es debido a que durante la vida útil del software, sufre cambios y conforme se corrigen éstos cambios se pueden introducir nuevos defectos originando que se produzcan más fallas.

Cada falla en el software indica un error en el diseño o en el proceso de traslación a un lenguaje de máquina. Este comportamiento del software produce un incremento en los costos de mantenimiento.

En todas las organizaciones existe una premisa en cuanto a los sistemas que se encuentran funcionando; Se debe de efectuar el mantenimiento a éstos buscando reducir los costos con un menor tiempo de respuesta, es decir, obtener información oportuna al menor costo.

Es por ello que es evidente la necesidad de utilizar metodologías que faciliten el análisis, diseño y generación de documentación de sistemas que permitan de

Existen algunos puntos por lo que no se sigue una metodología:

- La gente se niega a cambiar su forma de desarrollar la cual ha mantenido durante años
- Se usan estándares y procedimientos para construir el software que no reflejan las prácticas adecuadas de desarrollo, además de que muchas veces el que desarrolla no conoce su existencia.
- Se piensa que se tienen las herramientas más modernas (hablando de hardware), sin saber que no sólo se necesita el mejor hardware para desarrollar.
- Se piensa que si se falla en la planeación, se puede solucionar agregando más gente al equipo de trabajo.
- Se piensa que una declaracción general de los objetivos, sin proporcionar los detalles inicialmente, es suficiente para comenzar a programar.
- Algunos programadores tienen la muy mala costumbre de irse a una terminal y empezar a programar.
- Se piensa que lo único que se debe entregar al final de un proyecto es el programa funcionando, es decir, se tiene ninguna o poca información de referencia, una falta de documentación que apoye al diseño realizado.
- Se piensa que una vez entregado el programa, el trabajo ha terminado.
- O simplemente el que desarrolla no conoce ninguna metodología formal para hacerlo.

Al igual que en las metodologías de desarrollo, también existen algunas razones del porqué no se elabora una documentación:

- El desarrollador no tiene tiempo para dedicarle a esa tarea "menor" que es la documentación.
- El desarrollador no tiene el menor interés de hacer del conocimiento de los demás, los mecanismos que le dan el PODER que significa controlar la información.
- El desarrollador no tiene la menor intención de generar una documentación, con la finalidad de volverse indeispensable dentro de una organización.

En éste momento nos podríamos hacer una pregunta: como reducir los costos de mantenimiento, la cual se podría responder con una sola palabra: CALIDAD.

La calidad de los sistemas a desarrollar es un elemento importante para la gente de desarrollo de sistemas, así como para el usuario de éstos. Las características más importantes de calidad en el desarrollo de sistemas son:
Utilidad, claridad, confiabilidad, eficiencia y economía.

Para poder alcanzar el factor de calidad en el desarrollo de sistemas, es necesario el uso de métodos, herramientas y procedimientos.

Los métodos incluyen tareas como: planificación, análisis de requerimientos del sistema y del software, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento, es decir, los métodos nos indican "como" construir el software.

Las herramientas proporcionan un soporte a los métodos.

Los procedimientos se encargan de unir a los métodos y herramientas, las cuales definen la secuencia en la que se aplican los métodos, que son los controles que ayudan a segurar la calidad y coordinar los cambios.

El uso de métodos, herramientas y procedimientos trae consigo la generación de la documentación de un sistema.

Pero, ¿Porqué es necesario obtener la documentación de un sistema?, las razones son:

- La documentación es la interfaz de entendimiento entre el elemento humano y la computadora.
- Un proyecto de software tiene una vida útil más larga cuando existe una documentación bien generada que facilite su mantenimiento.
- Los altos funcionarios de un organismo, pueden no tener mucho conocimiento del cuidado exhaustivo que se tiene para desarrollar un sistema.

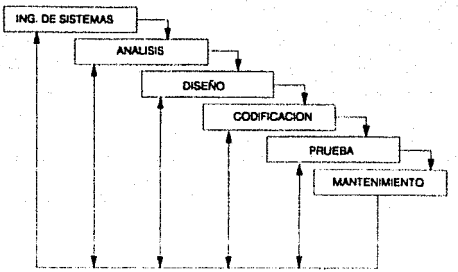
Ellos juzgan un proyecto únicamente por la documentación del sistema que se presenta.

- Existe un mayor entendimiento de las necesidades del usuario por parte personal de sistemas.
- El usuario no depende indefinidamente del personal de sistemas, al contar con manuales que faciliten la operación del sistema.

Existen tres paradigmas, para el desarrollo del software, que abarcan los métodos, herramientas y procedimientos:

Ciclo de vida clásico

- Ingeniería de sistema.
  Se encarga de establecer un conjunto de requerimientos de todos los elementos del sistema. Esta visión es importante cuando el software tiene que interrelacionarse con otros elementos del sistema como son hardware, personas y bases de datos.
- Análisis.
  Obtiene los requerimientos para desarrollar el software.
- Diseño
  El proceso de diseño traduce los requerimientos en una representación del software que pueda ser establecida de forma que se obtenga la calidad requerida antes de que comience la fase de codificación.
- Codificación.
  La codificación traduce el diseño en una forma legible para la computadora.
  Cuando se tiene un diseño detallado, entonces el paso de la codificación es mucho más sencillo.
- Prueba.
  La prueba se enfoca sobre la lógica interna del software, para asegurar que una entrada definida producirá los resultados que realmente se desean.
- Mantenimiento.
  El mantenimiento del software se aplica a cada uno de los pasos precedentes del ciclo de vida de un programa existente en vez de alguno nuevo.
  Esquemáticamente:



Prototipos.

El uso de prototipos es muy común cuando se desea identificar requerimientos detallados de entrada que no son muy claros, cuando se desea probar la eficiencia de algún algorítmo, para establecer la interacción hombre-máquina, etc.

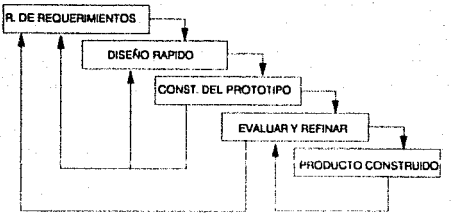
Los prototipos pueden tomar las siguientes formas:

a) Prototipo en papel

b) Un prototipo que implemente subconjuntos de la función requerida, o toda la función.

El diseño utilizando prototipos abarca las actividades:

- Recolección de requerimientos.
  Se definen objetivos globales y se identifican los requerimientos conocidos.
- Diseño rápido.
  Este diseño se enfoca principalmente a los aspectos visibles del usuario (Por ejemplo entrada y formatos de salida)
- Construcción del prototipo.
  El prototipo se construye y es evaluado para refinar los requerimientos del software.
- Evaluar y refinar requerimientos.
  Se produce un proceso iterativo en el que el prototipo se afina para que satisfaga las necesidades del cliente.
- Producto construido.
  Se utiliza el prototipo refinado para construir un producto final.
  Esquematicamente:

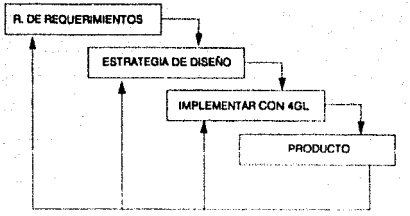


Técnicas de cuarta generación

Estan técnicas facilitan al desarrollador especificar algunas características del software a alto nivel. Un entorno para el desarrollo del software que soporta éste paradigma, incluye algunas de las siguientes herramientas: lenguajes no procedurales para consulta a base de datos, generación de informes (reportes), manipulación de datos, interacción y definición de pantallas y generación de código.

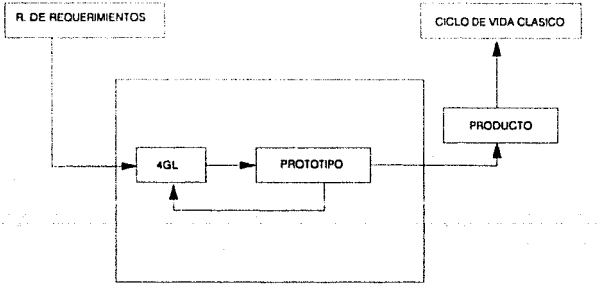
Este paradigma cuenta con las siguientes actividades.

- Recolección de requerimientos.
  El cliente debe describir los requerimientos y éstos deben traducirse directamente en un prototipo operacional.
- Estrategia de diseño e implementación usando 4GL.
  Cuando se tienen aplicaciones pequeñas, es posible ir directamente desde el paso de establecimiento de requerimientos a la implementación usando un lenguaje de cuarta generación no procedural.
  Cuando el proyecto es muy grande, aún es necesario definir una estrategia de diseño, a pesar de que se está usando un lenguaje de cuarta generación.
  El utilizar lenguajes de cuarta generación tan sólo facilita el desarrollo del sistema.
- Producto.
  Para transformar una implementación de cuarta generación en un producto, se deben efectuar pruebas.
- Gráficamente:



Combinación de los tres anteriores.

Este paradigma puede ser el más poderoso de los tres anteriores, debido a que cuando se realiza una combinación de los anteriores, se podría tener un software de muy buena calidad.



Para elegir el paradigma ha utilizar, es necesario establecer la naturaleza de la aplicación para determinar el más adecuado.

En muchos proyectos, la combinación de los paradigmas podrían darnos una solución más satisfactoria que si se elige alguno de ellos.

Independientemente del paradigma de software utilizado existen tres fases genéricas que se encuentran en todos los modelos de desarrollado de software, los cuales son: definición, desarrollo y mantenimiento.

Fase de definición.

La etapa de definición producirá 3 pasos específicos:

Planificación.

El objetivo de la planificación de un sistema a desarrollar, es el de proporcionar un área que permita al gestor realizar estimaciones de recursos, costos y métodos. Las tareas a definir en la planificación son:

- Alcance del software.- Se debe establecer un alcance del proyecto que no sea ambigüo ni incompresible a nivel de directivos y técnicos. La declaración del alcance del software debe estar delimitada.
- Recursos.- En este punto se realizará una estimación de los recursos requeridos para determinar el esfuerzo en el desarrollo del software.
  Existen tres tipos de recursos:
  a) Recursos humanos: Se refiere al número de personas requeridas para la realización de un proyecto, las cuales deberán tener cierto grado de especialidad.
  b) Recursos de software: se refiere al conjunto de herramientas que se utilizarán para elaborar el proyecto que se desea.
  c) Recursos de hardware: Se refiere a la computadora y periféricos asociados que se utilizarán durante la fase de desarrollo de software (sistema de desarrollo), a la máquina donde se ejecutará el software (máquina objetivo) y a los otros elementos de hardware adicionales.
- Estimación de costos y esfuerzos. La estimación de los costos del software nunca será una ciencia exacta.
  Existen demasiadas variables, tales como humanas, técnicas, de medio ambiente y políticas, que puedan afectar el costo final del software y al esfuerzo aplicado para desarrollarlo. A pesar de lo anterior, la estimación puede transformarse en pasos sistemáticos. La estimación se realiza utilizando una de varias técnicas que dependen de los datos históricos de productividad en el desarrollo del software.
- Métodos de planificación.
  Las herramientas y técnicas generalizadas de la planificación de un proyecto pueden utilizarse en el desarrollo del software.

La técnica de revisión y evaluación de proyectos (PERT) y el método de ruta crítica, son dos métodos de planificación de proyectos que pueden aplicarse al desarrollo del software. Ambas técnicas desarrollan una descripción de la red de tareas del proyecto.
Estas técnicas permiten al desarrollo establecer un camino que determina la duración del proyecto y calcular tiempos límite.

Análisis de requerimientos.

El análisis es un proceso de descubrimiento y refinamiento de los requerimientos del sistema.

El análisis de requerimientos facilita al desarrollador especificar la función y el comportamiento de los programas, indicar la interfase con otros elementos del sistema y establecer las ligaduras de diseño que debe cumplir el programa.

Existen varios métodos de análisis y especificación del software. Cada método tiene una única notación y punto de vista, pero a pesar de ello los métodos se relacionan por un conjunto de principios fundamentales.

a) El dominio de la información.

Este tiene 3 visiones diferentes de los datos que procesan los programas de computadoras:

- Flujo de información.
  Representa la manera en que los datos cambian conforme pasan a través de un sistema.
- Contenido de la información.
  El contenido de la información representa los elementos de datos individuales que componen a otros elementos mayores de información (Registros)
- Estructura de la información.
  Representa la organización lógica de los distintos elementos de datos.

b) Partición.

Normalmente los problemas son demasiado grandes y complejos para ser comprendidos como un todo, por ello es necesario dividir el problema en partes que puedan ser comprendidas.

c) Visiones lógicas y físicas.

La visión lógica de los requerimientos del software presenta las funciones que han de realizarse y la información que ha de procesarse independiente de los detalles de implementación.

La visión física presenta una manifestación de las funciones de

procesamiento y las estructuras de información.

En el análisis de requerimientos se debe enfocar en lo que el software debe realizar, en vez de como se implementará el procesamiento. Sin embargo, la visión física no debe ser interpretada necesariamente como una representación del como, en vez de ello el modelo físico representa el modo real de operación, esto es, la asignación existente o propuesta para todos los elementos del sistema.

Como resultado del análisis se desarrolla un documento denominado "la especificación estructurada", el cual debera ser revisado por el cliente para asegurar que se tiene la misma percepción acerca del sistema a desarrollar.

Fase de desarrollo.

La fase de desarrollo enfoca sobre el como.

Durante el desarrollo se intentan descubrir como han de diseñarse las estructuras de datos y la arquitectura del software, como han de implementarse los detalles procedimentales y como ha de realizarse la prueba

Una vez que se han establecido los requerimientos del software, la fase de desarrollo estará conformada por:

Diseño, generación de código y prueba.

Diseño.

El objetivo del diseño es el de producir una entidad o representación de un modelo que será construido más adelante.

El diseño de software puede ser visto desde dos puntos de vista:

a) Gestión.- El diseño va del diseño preliminar al diseño de detalle.

b) Técnica.- Diseño de datos, diseño arquitectónico y diseño procedimental.

- Diseño de datos.- Se enfoca sobre la definición de la estructura de datos.
  El diseño de datos es la primera de las tres actividades a efectuar durante el diseño del software.
  Independientemente de las técnicas de diseño utilizadas, los datos bien diseñados pueden crear una buena estructura del programa, modularidad y reducción de la complejidad procedimental.
  El diseño de datos se encuentra frecuentemente como parte de la tarea de análisis de requerimientos.
- Diseño arquitectónico.- El objetivo principal de éste diseño es el de desarrollar una estructura de programa modular y la de representar las relaciones de control entre los modulos.

El diseño arquitectónico mezcla la estructura del programa y la estructura de los datos y define las interfaces que facilitan el flujo de datos a lo largo del programa.

- Diseño procedimental.- El diseño procedimental se realiza después de que se ha establecido la estructura del programa y de los datos.
  La especificación procedimental, idealmente, requiere definir los detalles algorítmicos que deben establecerse en un lenguaje natural, pero el diseño procedimental debe especificar los detalles de los procedimientos sin ambigüedad y eso no se puede lograr usando un lenguaje natural.

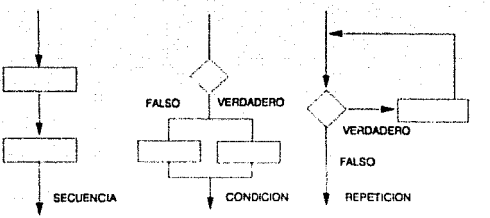
Existen algunas formas restringidas que son utilizadas para representar los detalles procedimentales que son:

1.- Programación estructurada.

Está formada por un conjunto de construcciones lógicas con las que podría formar cualquier programa, cada uno de estas construcciones tiene una lógica predecible, además de que facilita al usuario seguir el flujo procedimental. Estas construcciones estructuradas son: la secuencia, condición y repetición. El uso de costrucciones estructuradas reduce la complejidad del programa y facilita la legibilidad, prueba y mantenimiento.

2.- Herramientas gráficas de diseño.

El diagrama de flujo es la representación gráfica más ampliamente usada para el diseño procedimental.
Este diagrama es un gráfico muy sencillo que está formado por una caja que indica un paso de procedimiento, un rombo que representa una condición lógica y las flechas las cuales muestran el flujo de control.
Las tres condiciones básicas de la programación estructurada pueden ser representadas mediante un diagrama de flujo:



3.- Herramientas tabulares de diseño

En muchas aplicaciones puede ser necesario un módulo que evalue una condición o combinación de condiciones complejas y seleccionar las acciones apropiadas en base a condiciones. En éste grupo podemos encontrar las tablas de decisión y los árboles de decisión, en los que se establece una relación entre las condiciones, valores y acciones a tomar.

4.- Lenguaje de diseño de programas

Como lenguaje de diseño de programas se puede utilizar el español estructurado o pseudocódigo

Hay algunos conceptos fundamentales en el diseño de software que se deben tener en mente.

a) Modularidad.- El software se divide en elementos con nombres y direcciones separadas llamadas módulos, que se integran para satisfacer las necesidades del software. Cuando se divide el software el esfuerzo requerido para desarrollar será más pequeño.

b) Ocultamiento de la información.

Este punto indica que los módulos deberán diseñarse y especificarse de forma que la información contenida dentro de un módulo sea inaccesible a otros módulos que necesitan tal información.

c)Independencia funcional.

La independencia funcional se obtiene desarrollando módulos con una clara función y evitando un exceso de interacción con otros módulos. Esta independencia se mide usando dos medidas cualitativas del software que son: criterios de cohesión y acoplamiento.

Prueba.

La fase de prueba es un elemento crítico para la garantía de calidad del software, la cual representa la última fase en el desarrollo de un sistema.

La característica que deberá tener la prueba, es que está tiene que ser "destructiva", es decir, encontar el mayor número de errores que existan en el software.

Se dice que tiene que ser destructiva porque en la mayor parte del tiempo se prueba el software con casos sencillos, y esto es debido a que el que codifica no le agrada que le digan o darse cuenta por él mismo, que el programa que desarrolló presenta errores.

Existen dos formas en que puede ser probado el software:

a) Cuando se conoce una función específica, se puede llevar a cabo pruebas que demuestren que cada función es completamente operativa (caja negra) y,

b) Conociendo el funcionamiento de un producto se pueden realizar casos de prueba que aseguren que todas las piezas esten trabajando correctamente , es decir, que se usa la estructura de control de diseño procedimental para efectuar los casos de prueba (caja blanca).

Para probar un software completo se deberán de efectuar las siguientes pruebas:

a) Prueba de unidad.- Aquí se centra la atención a la unidad de diseño más pequeña, que es el módulo. Esta prueba esta orientada a la caja blanca.

b) Prueba de integración.- Esta prueba nos permite verificar las interfaces entre los módulos, obtener los módulos probados en unidad y construir una estructura del programa de acuerdo al diseño.

En esta parte se utiliza el concepto de resguardos (integración descendente) y conductores (integración ascendente).

c) Prueba de validación.- Aquí se validan los requerimientos establecidos como parte del análisis de requerimientos, comparandolos con el sistema que se ha construido. Se realizan pruebas en cajas negras.

d) Prueba del sistema.

Esta prueba cae fuera del alcance del proceso de desarrollo y se encarga de validar la interacción con otros elementos del sistema y de que estos funcionen adecuadamente.

Fase de mantenimiento.

Esta fase se enfoca a los cambios requeridos cuando se tienen fallas o cuando se modifican los requerimientos del sistema.

Durante la fase de mantenimiento se encuentran tres tipos de cambios:

a) Corección.

Este mantenimiento se realiza cuando el cliente descubre defectos en el software.

b) Adaptación.

El mantenimiento adaptativo ocurre cuando se modifica el software para ajustarlo a los cambios de su entorno externo.

c) Aumento.

Este mantenimiento ocurre cuando el cliente solicita modificaciones adicionales que pudiera ser beneficioso añadirlos, a este tipo de mantenimiento se le llama perfectivo.

El software se ha convertido en el elemento clave de la evolución de sistemas. Durante el desarrollo de un nuevo producto, independientemente del paradigma de ingeniería utilizado, se le debe de prestar mayor interés a la fase de mantenimiento por ser la que mayor tiempo consume y en consecuencia la que mayor costo genera.

Por esto es necesario el uso de un conjunto de métodos, herramientas y procedimientos para que se cree un ambiente que facilite el desarrollo de sistemas y que además ayude a generar una documentación del mismo paralelamente.

Se considera que un sistema es tan bueno como su documentación, por lo que es importante prestarle la atención debida a éste punto.

Debido a la necesidad de obtener la documentación de un sistema, hoy en día más que hacer documentación para software se debe generar software para documentación.

La herramienta presentada en éste trabajo tiene la finalidad de cumplir lo anterior, proporcionando la documentación que forma parte del manual de referencia técnica de sistemas desarrollados en pascal.

II.- O B J E T I V O

- Desarrollar un documentador que ayude a la elaboración del manual de referencia técnica y al mantenimiento de sistemas desarrollados en pascal, de una manera rápida, confiable y eficiente, aplicando metodologías de desarrollo adecuadas.

Analizando el objetivo se pueden encontrar varios puntos fundamentales que se buscan en el desarrollo de éste trabajo:

- Conocer y aplicar formalmente una metodología de desarrollo de sistemas.

- Definir la documentación que se debe producir de forma asociada con el sistema, así como su contenido.

- Desarrollar un documentador que proporcione, en línea, la información básica para la elaboración del manual técnico de cualquier sistema desarrollado en pascal.

- Facilitar el mantenimiento con ayuda de la documentación producida por el documentador.

La información que proporcionará el documentador, permitira obtener una visión genérica del flujo de contol y de las relaciones existentes entre los módulos, además de que permitirá una generación y actualización , de manera inmediata, del manual de referencia técnica.

La construcción del documentador se efectuará utilizando el paradigma del ciclo de vida clásico, las metodologías orientadas al flujo de datos (Análisis estructurado de Tom Demarco y Diseño estructurado de Yourdon-Constantine), así como conceptos básicos de programación estructurada, teoría de compiladores y estructuras de datos.

III.- PRESENTACION DE UNA METODOLOGIA DE DESARROLLO.

El propósito del capítulo es el presentar la metodología de desarrollo que se utilizará en las etapas de análisis y diseño del documentador de pascal, así como algunos conceptos utilizados en el desarrollo de la mismo.

El capítulo, además de proporcionar una guía para que cualquier persona, que no conozca las técnicas utilizadas tenga las conocimientos necesarios para poder involucrarse en el desarrollo de la herramienta, también sirva como un texto de consulta para los alumnos que deseen aprender las técnicas descritas.

El capítulo se dividirá en las secciones:

a) Análisis estructurado
b) Diseño estructurado
c) Conceptos básicos para la construcción de un analizador léxico y un analizador sintáctico.

3.1.- ANALISIS ESTRUCTURADO

La Técnica que se muestra a continuación fué desarrollada por Tom Demarco, la cual se explica en su libro " Structured Analysis and System Specification ".

Una definición de análisis estructurado:

" Es el uso de herramientas de diagramas de flujos de datos, diccionario de datos, español estructurado, tablas y árboles de decisión, para construir una clase de documento destino llamado 'La especificación estructurada' . Este documento se puede accesar de una manera selectiva sin tener la necesidad de leerlo de principio a fin para poder entenderlo".

Cada una de éstas herramientas contienen características propias para su manejo.

Características de las herramientas de análisis estructurado:

3.1.1.- DIAGRAMA DE FLUJO DE DATOS

Un diagrama de flujo de datos (que en lo consecuente se llamará DFD) es una representación en forma de red de un sistema.

El DFD muestra al sistema en términos de sus piezas componentes, con todas las interfaces entre componentes indicadas.

Elementos de un DFD:

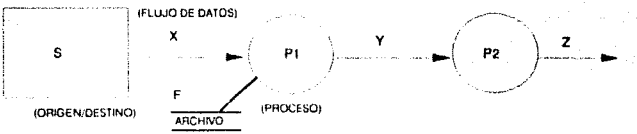
Los DFD's estan formados por 4 elementos básicos:

a) Flujos de datos (representados por vectores con una etiqueta)
b) Procesos (representados por círculos o burbujas)

c) Archivos (representados por líneas rectas)

d) Orígenes y destinos (representados por cajas)

En la figura mostrada a continuación, se indican todos los elementos involucrados en un DFD.



Interpretación del diagrama:

"X" llega del origen S y es transformado en "Y" por el proceso "P1", el cual requiere accesar el archivo "F" para poder realizar está operación; La "Y" es posteriormente transformada en "Z" por el proceso "P2".

3.1.1.1.- Flujo de datos

Un flujo de datos muestra la interfase que existe entre los componentes de un DFD. El flujo de datos no sólo se efectúa hacia/desde los archivos y a/desde las cajas de origen y destino, sino también entre procesos.

El flujo de datos se representa mediante un vector etiquetado.

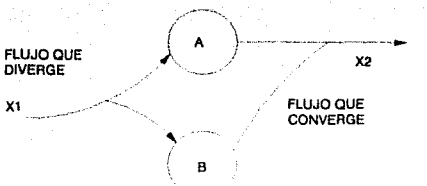
Formalmente:

Un flujo de datos, es un "conducto" a través del cual paquetes de información de conocida composición fluyen

Nunca hay que forzar que diferentes tipos de información conformen un mismo paquete, debido a que esto podría oscurecer la interfase (al decir diferente tipo de información se refiere a Información que no tiene nada en común).

Tom Demarco ofrece un conjunto de convenciones en la notación de Flujos de Datos. El menciona que este conjunto de convenciones no son totalmente universales, pero sin embargo, pueden considerarse útiles:

- Los nombres de flujos de datos pueden utilizar guiones, siempre y cuando sean utilizados títulos en mayúsculas.
- Dos flujos de datos no pueden tener el mismo nombre.
- Los nombres deben ser escogidos para representar los datos que se mueven a través del conducto (flujo).
- Se pueden tener flujos que convergen y divergen por ejemplo:



- Los flujos de datos que viajan desde y hacia un archivo sencillo NO REQUIEREN NOMBRE (el nombre del archivo será suficiente para describir el conducto (flujo)). Todos los flujos restantes deberán ser nombrados.

- Un flujo de datos NO es una representación de un flujo de control.

Por ejemplo:

Traer el siguiente empleado.

- Un flujo de datos NO es un activador de proceso.

NOTA: Un flujo de control no puede ser representado en un DFD, únicamente los flujos de datos.

3.1.1.2.- Proceso

Un proceso es una transformación de flujos de datos que llegan hacia él en flujos de datos que salen del mismo.

La notación utilizada para representar procesos en el DFD, es a través de círculos (burbujas).

Existen algunas características que los procesos deben cumplir:

- Cada burbuja necesita un nombre descriptivo. A pesar de que el proceso se describirá en mucho más detalle (por ejemplo español estructurado), el nombre del proceso debe dar al usuario una idea general de su actividad.
- El proceso debe ser etiquetado en término de sus entradas y sus salidas.
- En un DFD, a cada proceso se le deberá de dar un número único.

3.1.1.3.- Archivo

Para el análisis estructurado un archivo es un depósito temporal de datos, como podría ser una cinta, un área de disco, un archivo indexado, un pequeño libro o hasta un cesto de basura.

La convención utilizada para representar un archivo es por medio de una línea con el nombre del archivo lo más cercanamente posible.

Al escoger un nombre de archivo se tiene que cuidar que éste sea claro y entendible. Hay que evitar el uso de nombres codificados, debido a que esto acarrea ambigüedad al mostrarselo al usuario.

La dirección de las flechas de un archivo es significativa. Unicamente los flujos netos de y hacia un archivo deben ser mostrados. Por ejemplo, cuando se realiza una actualización a un archivo primero se tiene que leer para poder actualizarlo, en el DFD, sólo se tiene que mostrar la acción que se realiza por traer un archivo, es decir, únicamente se deben tomar en cuenta los "flujos netos".

Para mostrar dos caminos de acceso (lectura y escritura) se utilizan flujos con doble flecha.

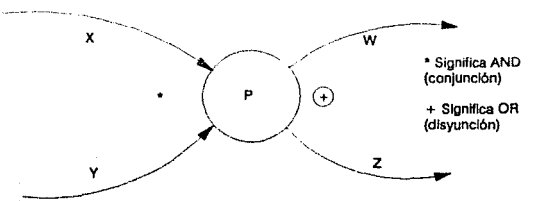
3.1.1.4.- Orígenes y destinos

El origen o destino es una persona u organización que se encuentra fuera del contexto del sistema, los cuales son generadores o receptores de datos del mismo.

Por convención, orígenes y destinos son representados en un DFD por cajas etiquetadas.

Debido a que los orígenes y destinos se encuentran fuera del área de mayor interés (en el desarrollo del sistema), ellos existen únicamente para proporcionar comentarios acerca de las conexiones del sistema hacia el "mundo exterior".

Notaciones procedurales



Algunos diseñadores cuando se encuentran que dos flujos de datos tienen que estar presentes al mismo tiempo, o sólo algunos de ellos pero no ambos, utilizan la notación mostrada en la figura anterior.

Tom Demarco no recomienda el uso de este tipo de notaciones porque la gente se acostumbra a ellas y con el tiempo se generan nuevos caminos para realizar más anotaciones y en consecuencia interdependencias más complicadas.

El camino para solucionar lo anterior es el uso de miniespecificaciones (español estructurado, tablas de decisión o árboles de decisión) para cada uno de los procesos, las cuales indicarán las relaciones que aplican entre los flujos de datos.

3.1.1.5.- Generación de diagramas de flujos de datos (DFD)

Guías para la generacion de DFD's.

1.- Identificar todos los flujos de datos netos de entrada y salida.

Lo primero que se deberá determinar es el contexto de estudio.

En la selección del contexto de estudio hay que mantener en mente que cada cosa que se deje fuera será ignorada más adelante. Al dejar algo fuera del contexto, se está decidiendo que no se dedicarán recursos de análisis en el estudio de ello. Habiendo determinado los límites, se deben identificar los flujos de datos que cruzan estos, es decir, las entradas y salidas netas.

2.- Llenar el cuerpo del DFD.

- Concentrarse primero en los flujos de datos
- Identificar el conjunto de información que el usuario trata como una unidad (por ejemplo, llegan juntos, se procesan juntos o el usuario identifica ésto como un todo).
- Introducir los flujos anteriores dentro del DFD y tratar de conectar estos con los flujos de datos de la periféria.
- Colocar "burbujas" donde un trabajo sea requerido para transformar un flujo de datos en un conjunto de flujos de datos. No preocuparse, por el momento, de nombrar estos procesos, se deben dejar en blanco.
- Buscar dentro de los procesos en blanco que estan formando el DFD, si existen algunos flujos internos que podrían ser usados dentro del proceso, (imaginados por el diseñador y corroborados por el usuario) y reemplazar el proceso simple por dos (o tres o cuatro, es decir, los que sean necesarios) y poner los flujos identificados entre ellos.
- Para cada flujo de datos efectuar las siguientes preguntas:

a) ¿Qué se necesita para construir ese flujo?
b) ¿De donde vienen esos componentes?
c) ¿ Puede cualquiera de los flujos de entrada ser transformados para formar cualquiera de los flujos de salida?
b) ¿Qué procesos serán requeridos para efectuar la transformación.?

- Introducir archivos en el DFD para representar los depósitos de datos que el usuario nos indique. Estar seguro de conocer el contenido de cada uno de los archivos para determinar los flujos que entran y salen de ellos.
- Estar preparado para regresar y modificar los límites del contexto.Por ejemplo: Se pudo haber olvidado una entrada que es requerida como punto clave (componente) de uno de los flujos de datos.

3.- Etiquetar los flujos de datos.

- Los nombres que se seleccionen deben ser legibles.
- Estar seguro de dar un nombre a cada uno de los flujos de datos.
- Estar seguro de dar un nombre "honesto". Esto aplica a un flujo de datos completo y no únicamente a su componente principal.
- Evitar nombres confusos como "datos e información".
- Tener cuidado de no agrupar "items" separados dentro de un flujo de datos, cuando ellos no tienen nada que ver con el flujo completo.

4.- Etiquetar procesos.

- Estar seguro de que el nombre sea "honesto", es decir, no hay que seleccionar un nombre que indica una operación en particular sabiendo que este proceso realiza otras operaciones adicionales.
- Tratar que los nombres consistan de un verbo simple que indique una acción fuerte y un objeto singular. Si se tienen dos verbos probablemente se tendrán que *realizar particiones futuras*.
- Cuidarse de palabras engañosas tales como "proceso", "manejar", debido a que estas palabras no significan alguna acción en particular. Si el mejor nombre que se puede tener es el anterior, entonces se ha conseguido un proceso sin nombre.
- Realizar particiones para evitar los procesos sin nombre. Despedazar éstos en dos o tres partes o agruparlos con otros de tal forma que se pueden nombrar facilmente.

5.- Ignorar inicialización y terminación

- No hay que preocuparse, por el momento, de como el sistema se inicia y como se termina.

6.- Omitir detalles en el manejo de errores triviales.

Existe una regla para determinar que errores deben ser permitidos durante ésta primera fase de análisis. Si el error no requiere reparación de procesos pasados, ignorar éste por el momento. Si el error requiere volver a previas actualizaciones o regresar un archivo o archivos a un estado previo, entonces no hay que ignorar éste.

7.- No mostar flujos de control o información de control

Una prueba para eliminar flujo de información que es utilizada con propósitos de control, es el preguntarse para que utiliza el proceso esta información. Si ésta es modificada y puesta como un flujo de datos de salida o parte de uno, entonces se trata de un legitimo flujo de datos.

Si éste sólamente sirve para indicar al proceso el inicio de su trabajo, entonces se trata de un flujo de control

8.- Estar preparado para comenzar de nuevo.

La mente humana es un procesador iteractivo, debido a que nunca hace algo exactamente correcto la primera vez. Es bueno tener una implementación imperfecta, porque eso significa que se puede repetir el trabajo una y otra vez, teniendose en consecuencia mejores resultados cada vez.

No debe importar cuantas veces se realiza un DFD, porque aunque esta actividad puede resultar trivial, se está pagando un pequeño precio por un resultado significativo mucho mayor.

3.1.1.6 Diagramas de flujos de datos nivelados.

Cuando se trabaja en el análisis de sistemas extremadamente largos, es necesario dividir estos en subsistemas. Si al realizar la división, los sistemas continúan siendo muy grandes, entonces se tiene que dividir a estos en subsistemas y así sucesivamente. La partición se detendrá cuando se tengan funciones primitivas del DFD.

3.1.3.1 Elementos de un diagrama de flujos de datos nivelado.

a) Diagrama de contexto (DC)

El DC, es aquel que documenta el dominio de estudio, mostrando el conjunto de flujos de datos que cruzan hacia afuera y hacia adentro del mismo, esto es, aquellos flujos de datos que definen los límites (flujo neto de datos).

El DC sirve con un único propósito, pero no menos importante, el de delinear (limitar) el dominio de estudio.

El nombre que usualmente se le da a la burbuja del diagrama de contexto indica el área que esta siendo estudiada, pero en muchos casos el nombre será sólo una aproximación del dominio verdadero, porque el diagrama de contexto no muestra una descomposición en detalle.

El diagrama de contexto tiene que pasar algunas pruebas de coherencia. Para aplicar esta prueba sólo hay que tomar en cuenta que el diagrama de contexto es una transformación, un proceso que transforma los flujos de datos de entrada, en flujos de datos de salida. Para realizar la prueba se deben tener en cuenta las siguientes preguntas:

- ¿ Es tal transformación posible?
- ¿ El conjunto de flujos de entrada son suficientes para construir los flujos de salida?
- ¿ El conjunto de salidas mantienen una relación lógica con el conjunto de entradas?

b) Primitivas funcionales

Son aquellas burbujas que no tienen futuras descomposiciones en redes de más bajo nivel.

Se definen las primitivas funcionales, cuando estas ya no pueden ser descompuestas en más niveles o cuando se decide que no hay descomposición futura porque los requerimientos han sido satisfechos.

c) Niveles intermedios.

Son aquellos niveles que muestran la partición de alguna o todas las áreas de una red, en componentes de una red que pueden sufrir una descomposición futura. En general se podrán varios niveles Intermedios, algunas veces tanto como ocho o nueve, aunque en tener algunos casos éstos niveles intermedios no existirán (sistemas muy triviales).

3.1.1.6.2.- Convenciones para las nivelaciones.

En el diagrama de contexto, se puede observar que este es el "padre" del primer nivel de partición de diagramas.

Ese diagrama es el "padre" de sus diagramas "hijo", los cuales constituyen el nivel dos. Puede haber tantos diagramas de nivel dos como burbujas hay en el "padre". Cada diagrama de nivel dos puede ser "padre" de algun número de diagramas de nivel tres y así en adelante.

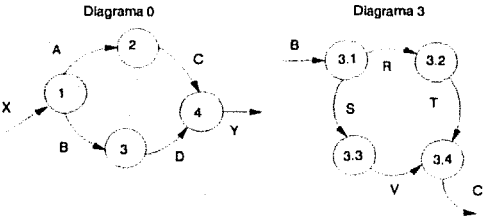
a) Relación Padre -Hijo.

En un diagrama en el que se tienen 5 burbujas se puede tener hasta 5 diagramas "hijo",o mucho menos que ese valor si en ese diagrama existen ya primitivas funcionales.

Para poder establecer la relación que existe entre un padre y un hijo, es

necesario poner en el diagrama hijo el número que identifica la burbuja de la que es hijo como el número de diagrama. Cada una de las burbujas en el diagrama hijo tendrá el mismo número, adicionandole un punto "." y un número de identificación a cada burbuja.

Por ejemplo:



b) Balance

Los flujos de datos que entran y salen de una burbuja de un diagrama padre, son equivalentes a las entradas y salidas netas de un diagrama hijo. Esta equivalencia es llamada "balance".

La regla para establecer el balance es la siguiente:

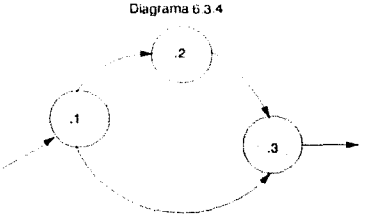
"Todos los flujos de datos que entran en un diagrama hijo deben ser representados en el padre por el mismo flujo de datos en la burbuja asociada. Las salidas del diagrama hijo deben ser las mismas salidas de la burbuja asociada en el padre con una única excepción: Rechazos triviales (Rechazos que no requieren revisión del estado de la información), los cuales no necesitan ser balanceados entre el padre y el hijo".

Nota: Aunque no se vean exactamente los mismos flujos entre la burbuja padre y el diagrama hijo, pueden existir flujos netos equivalentes. A esta descomposición se le llama "Descomposición paralela de datos o funciones". Para poder verificar este tipo de balance es necesario el uso del diccionario de datos.

c) Convenciones numéricas

- Cada diagrama recibe el número de la burbuja relacionada con el padre.
- El número de la burbuja está formada por la concatenación del número de diagrama, un punto decimal y un número local único.
- El primer nivel de diagrama es numerado como 0

- Las burbujas del diagrama 0 son numerados como 1,2,3 en vez de 0.1,0.2,0.3
- Cuando se tienen muchos niveles en un sistema se puede volver tedioso el manejar las convenciones anteriores. Para ello, existe una convención adicional:
  Poner en las burbujas únicamente la parte final de la concatenación (punto decimal y número local único). Por ejemplo:



d) Archivos locales

En ocasiones se pueden tener archivos en los diagramas hijos que no se encuentran en el diagrama padre y no por ello hay que pensar que el diagrama está desbalanceado.

Estos archivos no son mostrados mas que el diagrama hijo debido a que en el padre no tienen significado de interfase.

La regla para cuando el archivo es mostrado es:

- Un archivo es mostrado en un DFD al primer nivel donde es usado como una interfase entre dos procesos.

Una consecuencia lógica de utilizar ésta convención es:

- al primer nivel donde un archivo aparece, todas las referencias a éste son mostradas.

e) Información origen y destino

Cuando se desee saber hacia donde se dirige un flujo neto de salida/entrada en un diagrama, se debe revisar el diagrama padre y NO colocar una referencia en el diagrama hijo.

f) Extensión de la partición

Cuando se tienen demasiadas burbujas en un DFD, existe una gran dificultad para trabajar con ellas. No existe un número apropiado para trabajar adecuadamente; algunos autores, como Tom Demarco, opinan que realizar una partición de siete piezas es la ideal y parten de la idea de que el cerebro humano trabaja eficientemente con un conjunto de siete o un poco más piezas y menos eficiente con conjuntos mucho mayores (El número mágico 7 + -2).

A pesar de comentar lo anterior, De Marco piensa que el resultado de tener un DFD legible (Entendible) está a juicio del desarrollador, no sin dejar de resaltar los siguientes puntos:

- A cualquier nivel, particionar tanto como se pueda sin perder claridad. Lo mucho que se particione, serán los diagramas que se tendrán que usar y los que se tendrán que mantener en la especificación estructurada.
- Si se necesita un límite artificial de partición utilizar el número 7.
- Los diagramas que son claros son mucho mejor que los que no lo son.
- Ser indiferente a reglas absolutas de partición.
- Escuchar a los datos. Si estos indican que se deben distribuir en algún modo, hay que permitir que esto afecte a la partición. Una partición que es en este sentido natural, es más fácil de entender que aquella que se tiene dentro de cualquier límite artificial de partición.

3.1.1.6.3.- Consideraciones de nivel inferior.

He aquí tres diferentes criterios para decidir cuando el esfuerzo de partición ha ido lo suficientemente lejos:

- Detenerse cuando se crea que el interior de una burbuja de más bajo nivel puede ser descrita completamente en una miniespecificación de alrededor de una página.
- Detenerse cuando la burbuja tiene un simple flujo de entrada y un simple flujo de datos de salida.
- Detenerse cuando se tengan relaciones numéricas entre las entradas y las salidas.Tener relaaciones uno a uno o muchos a uno implica que la partición ha ido lo suficientemente lejos.
  Si no existe tal relación, por ejemplo dos entradas y dos salidas, entonces particiones futuras son requeridas.

Cada una de los tres diferentes criterios mostrados anteriormente deben ser aplicados uno a la vez, en un tiempo u otro de la creación del DFD.

3.1.1.6.4.- Documentación de procesos (mini especificaciones)

Cuando se tienen burbujas primitivas, se debe de escribir una descripción de ese proceso (español estructurado, tablas de decisión o árboles de decisión), está narrativa recibe el nombre de "miniespecificaciones".

Existe una regla para seleccionar las miniespecificaciones del DFD:

- Debe de existir una miniespecificación por cada burbuja en el DFD que no sufrirá una descomposición futura (primitiva funcional).

Las miniespecificaciones deben ser marcadas con el número de burbuja, de la burbuja relacionada.

En consecuencia a lo anterior, se tiene una segunda regla:

- El número de miniespecificaciones, en una especificación completa, será exactamente igual a el número de burbujas primitivas (primitivas funcionales) en el conjunto nivelado de DFD's.

3.1.2.- Diccionario de datos.

El diccionario de datos es una parte integral de la especificación estructurada, sin este, el diagrama de flujo de datos es únicamente un diagrama que proporciona una IDEA INCORRECTA de lo que está sucediendo en el sistema.

El diccionario de datos está principalmente compuesto de definiciones de:

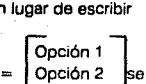
- Flujos de datos
- Componentes de los flujos de datos
- Archivos
- Procesos

Una definición del diccionario de datos es una descripción que consiste de un genero y una diferencia. El genero establece alguna clase que contiene la palabra que está siendo definida, y el conjunto de diferencia distingue a éste de los otros miembros de la clase.

Definiciones (convenciones)

- = Significa "Es equivalente a"
- + Significa "Y" (AND)
- [] Significa "uno o el otro"(EITHER-OR)
- {} Significa "Iteraciones de" (ITERATIONS OR)
- () Significa "Opcional" (OPTIONAL)
- Las iteraciones pueden tener límites, las cuales se expresan. {X} o {X} Cuando no se colocan límites ,se toma desde 0 hasta infinito.
- Al utilizar la clausula "uno o el otro" se pueden tener muchas

opciones y en lugar de escribir

Cualquier cosa =  se utiliza:

Cualquier cosa = [Opción1/Opción2/Opción3]
donde [/] = ó (disyunción)

- Cuando se encuentre que se tiene una definición extremadamente anidada y esto acarrea problemas para enterder la definición, se debe de considerar el manejo de algunas componentes de nivel intermedio.

- Cuando se desee utilizar comentarios, se debe efectuar mediante el uso del símbolo "*", cuando inicie y termine el comentario.

- Información acerca de la composición de los datos va en el DD (que son los componentes y como se relacionan)
- Información acerca del contenido y procesamiento de los datos va en la descripción del proceso (como son establecidos los valores). Información acerca de la ruta de los datos va dentro del DFD.
- Los términos definidos por sí mismos son aquellos que no se encuentran definidos en alguna parte del diccionario.
  Este término puede ser entendido sin ambigüedad desde su propio nombre. Por ejemplo: "edad".
- Alias
  Un alias es un sinónimo previamente definido.
  Los Aliases ocurren por diversas razones:
  Diferentes usuarios tienen nombres diferentes para el mismo documento.
  Un desarrollador introduce inadvertidamente un alias cuando baja de nivel.
  Dos desarrolladores trabajan independientemente con el mismo flujo de datos, pero dan a éste diferentes nombres.
  Es recomendable documentar los Aliases para tener una pista del trabajo realizado.

Nota: Se debe de tener en mente la siguiente oración en la elaboración de un DD: "Es la definición de un item lo que dice lo que realmente es, y no su nombre".

Para efectuar las definiciones en el diccionario de datos se pueden utilizar los siguientes formatos:

Flujos de datos:

- Nombre del flujo de datos
- Alias
- Composición
- Notas

Elemento de datos, es un elemento que es indivisible, es decir, que no es descompuesto dentro de flujos de datos subordinados.

- Nombre del elemento de datos.
- Alias
- Valores y significados
- Notas

NOTA: Los elementos de datospueden ser continuos y discretos. Si estos son continuos podrían ser definidos mediante rangos válidos y/o inválidos de valores, en muchos casos, un elemento de datos continuo puede ser dejado como un término definido por el mismo, es decir, no se necesita definir éste a menos que exista una característica no usual acerca de la distribución de valores.

Archivos (files)

- Nombre del archivo o base de datos
- Alias
- Composición
- Organización
- Notas

Procesos

- Nombre del proceso
- Número del proceso
- Descripción del proceso

3.1.2.1 Diagramas de estructuras de datos (DED)

Un DED ha sido proporcionado como un medio para definir archivos complejos.

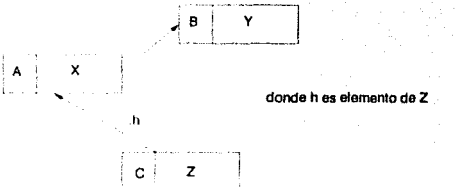
Características:

- Hay un bloque para cada archivo componente.
- La llave de acceso para cada archivo es escrita dentro del bloque
- Las flechas indican que existen apuntadores que ligan a los archivos.
- La dirección del apuntador muestra la dirección de la liga.
- Cuando el nombre del elemento apuntador es diferente a el nombre de la llave de acceso apuntada, el nombre del elemento

apuntador es escrito a lo largo de la flecha, de otra forma es el mismo y no se escribe.

- Hay tantos modos de acceso como llaves hay.

Ejemplo:



Usos:

El desarrollador hace dos diferentes usos del DED:

- Descripción del medio ambiente actual de tal forma que el usuario pueda entender y verificar éste.
- Ayudar al usuario a reconstruir sus verdaderos requerimientos de almacenamiento de datos, divorciado de la composición arbitraria de su almacenamiento actual. (probablemente manual).

3.1.3.- Procesos

Para comenzar hay que definir a las miniespecificaciones:

Es la descripción que gobierna la política de transformación de los flujos de datos de entrada en flujos de datos de salida.

Existen dos metas específicamente definidas por las miniespecificaciones:

- Debe haber una miniespecificación por cada primitiva funcional en el conjunto de DFD's
- Cada miniespecificación debe describir las reglas que gobiernen la transformación de los flujos de datos que llegan a la primitiva, dentro de los flujos de datos que dejan ésta.

Existen tres alternativas clásicas para la descripción de una narrativa, las cuales son:

- Español estructurado
- Tablas de decisión
- Arboles de decisión

3.1.3.1.- Español estructurado

Es un lenguaje de especificación que hace uso de un vocabulario y sintaxis limitado. El vocabulario del español estructurado consiste únicamente de:

- Verbos en lenguaje imperativo
- Términos definidos en el DD
- Ciertas palabras reservados para una formulación lógica.

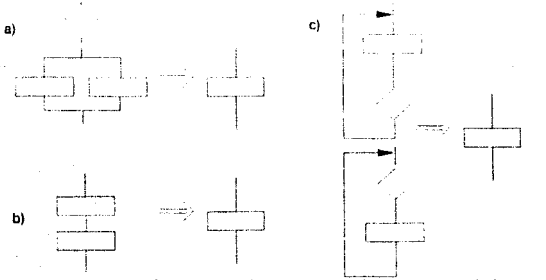
La sintaxis del español estructurado esta limitado a:

- Sentencias declarativas simples. Consiste de una o más piezas subordinadas, las cuales son aplicadas una después de la otra sin interrupción.
- Sentencias de decisión. Dos o más políticas subordinadas, en donde sólo una de las piezas aplica en un caso dado.
- Sentencias de repetición. Una política subordinada que es realizada una y otra vez dentro de un límite
- La combinación de las sentencias anteriores

Algoritmo para prueba de sintaxis

Este algoritmo es útil para determinar si únicamente se están utilizando las tres sentencias anteriores en el español estructurado.

- Cambiar cada una de las estructuras permitidas por una instrucción de secuencia, es decir:



- Realizar éste cambio hasta que se obtenga una sola instrucción de secuencia o no se puedan efectuar más cambios.

- Si al terminar el proceso se tiene una instrucción de secuencia única, el proceso cumple con la sintaxis del español estructurado.

Reglas que gobiernan la implementación lingüística de las estructuras de control.

- Una construcción de secuencia es compuesta por una o más sentencias imperativas simples.
- Una construcción de decisión puede ser implementada por una oración condicional (tipicamente IF), seguida por una política subordinada la cual aplica si la condición es conocida y otra la cual aplica si no lo es. La segunda de éstas (OTHERWISE) puede ser omitida.
- Una construcción de decisión puede ser implementada usando el formato Select-Case o equivalente
- La construcción de repetición puede ser implementada por una condición que gobierne la repetición continua seguida por la política subordinada a ser repetida.
- Una descripción que combine políticas subordinadas sintáticamente válidas es por si misma sintácticamente válida.

Vocabulario del español estructurado

- Verbos - Que pueden ser seleccionados del conjunto del español evitando verbos tales como procesa, pareja.
- Objetos.- que deben ser tomados del conjunto de nombres dados a archivos, flujos de datos y elementos de datos del DD.
- Calificadores.- que pueden ser tomados del conjunto de nombres dados a valores de elementos de datos en el DD.
- Conjunciones.- (tales como "if", "while", "until") que deben ser tomados del conjunto de palabras reservadas para efectuar las tres construcciones sintácticas.
- Atributos de selección - (tales como "IGUAL", "AND", "OR") deben ser tomados del conjunto de palabras reservadas.

3.1.3.2.- Tablas de decisión

Una tabla de decisión es usada cuando la selección de una subpolítica depende de la combinación de condiciones

Las tablas de decisión estan formadas fundamentalmente por tres partes principales:

- Condiciones
- Acciones a tomar en base a las condiciones

- Reglas.- son los valores que pueden tomar cada una de las condiciones.

NOTA:El número de reglas requeridas para completar una tabla de decisión es igual al producto del número de valores para todas las variables de decisión.

Una vez que se conoce el número de reglas, se puede construir la matriz de condiciones de la tabla de decisión.

Existe una regla para llenar rápidamente ésta matriz

1.- Colocar la primera condición y los valores que ésta tenga (reglas) formando un conjunto.

2.- Repetir cada conjunto de reglas en el siguiente espacio disponible tantas veces como valores tenga la siguiente condición

3.- Colocar la siguiente condición

4.- Llenar cada conjunto de reglas con un valor diferente de la siguiente condición

5.- Repetir cada conjunto de reglas de los dos renglones anteriores

6.- Colocar la siguiente condición

7.- Y así sucesivamente

Por ejemplo:

	A		C		F
condición 1	B	condición 2	D	condición 3	G
			E		H

Número de reglas = 2X3X3 = 18

1) Condiciones Reglas
12
Condición 1 A B

2,3) Condiciones Reglas
1 2 3 4 5 6
Condición 1 A B A B A B
Condición 2

4) Condiciones Reglas
1 2 3 4 5 6
Condición 1 A B A B A B
Condición 2 C C D D E E

5,6) Condiciones Reglas
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Condición 1 A B A B A B A B A B A B A B A B A B
Condición 2 C C D D E E C C D D E E C C D D E E
Condición 3 etc....

7) etc...

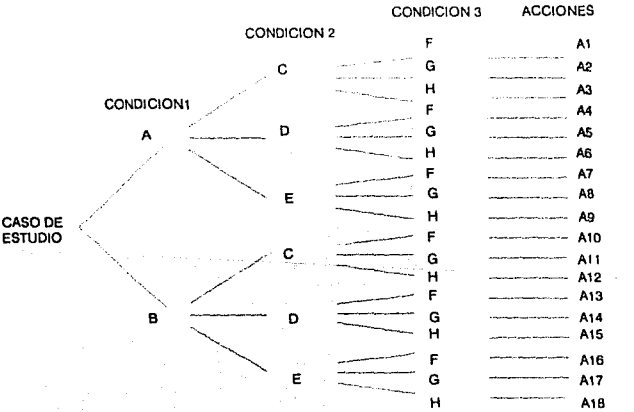
Las tablas de decisión pueden ser combinadas con el español estructurado colocando en las acciones una sentencia variable en base a cada una de las reglas establecidas (acciones).

Cuando se utilicen éstas con los usuarios, se debe de tener cuidado en no llamarles tablas de decisión, sino tablas que describen una situación o política en particular.

3.3.1.3 Arboles de decisión

Un arbol de decisión no es otra cosa que una representación gráfica de una tabla de decisión. El uso de este tipo de herramienta está orientado a un usuario al cual no se le puede persuadir en el manejo de tablas de decisión, algunas veces ésto es mucho mejor para un usuario porque se le hace más familiar.

Tomando el ejemplo anterior tendría:



3.2 .- DISEÑO ESTRUCTURADO

La técnica mostrada a continuación fué desarrollada por Constantine-Yourdon, la cual explica en más detalle en su libro "Structured design: Fundamentals of a discipline of computer program and system design"

Diseño estructurado:

Es el arte de diseñar los componentes de un sistema y la interrelación entre estos componentes en el mejor camino posible.

O bién:

El proceso de decidir cuales componentes interconectados de que manera resolverán algún problema "bien definido".

El diseño estructurado permite una comoda transición de las representaciones de la información (el diagrama de flujos de datos) a una descripción de diseño de la estructura del programa.

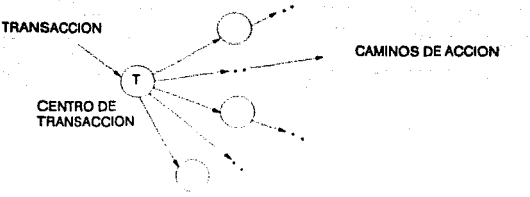
La transición desde el flujo de información a la estructura, se realiza como parte de un proceso de cinco pasos:

- Se establece el tipo de flujo de información
- Se indican los límites del flujo
- El DFD se convierte en la estructura del programa
- Se define la jerarquía de control mediante factorización
- Se refina la estructura resultante usando medidas heurísticas de diseño

Flujo de información.

La información entra al sistema mediante caminos que transforman los datos externos en una forma interna y se identifica como flujo de llegada. En el núcleo del software ocurre una transición. Los datos de llegada se pasan a través del centro de transformación y comienza a moverse a lo largo de caminos que van hacia la salida del software. Cuando un segmento de un diagrama de flujos de datos presenta éstas características se tiene un flujo de transformación.

El modelo fundamental del sistema implica un flujo de transformación; por tanto, todo el flujo de datos puede colocarse en ésta categoría. No obstante, el flujo de información se caracteriza frecuentemente por un elemento de datos sencillo llamado transacción, que desencadena otro flujo de datos a lo largo de uno de los muchos caminos. Cuando un DFD toma la forma mostrada a continuación, se presenta un flujo de transacción.



El flujo de transacción se caracteriza por datos que se mueven a lo largo de un camino de llegada que convierte información del mundo exterior en una transacción. La transacción es evaluada y basandonos en su valor, el flujo se inicia por uno de los muchos caminos de acción. El centro de flujo de información desde el que emanan muchos caminos de acción se llama control de transacción.

Un diagrama de flujos de datos para un sistema, puede presentar los dos tipos de flujos mencionados anteriormente.

3.2.1 Análisis de transformación

El análisis de transformación o diseño centrado a transformaciones, es una estrategia para derivar una estructura inicial de diseño que usualmente es totalmente buena (con respecto a modularidad) y generalmente requiere sólo una modesta reestructuración para llegar a un diseño final. Es una forma particular de una estrategia "top-down", la cual toma ventaja de la perspectiva global.

Los pasos para el diseño utilizando análisis de transformación en forma general son:

1.- Reorganizar el problema como un diagrama de flujo de datos (que ésto ya lo tendría si utiliza la técnica de análisis estructurado mostrada en la sección anterior)

2.- Identificar los elementos de datos aferentes y los elementos de datos eferentes, así como el centro de transformación.

Se define como elementos de datos aferentes a aquellos elementos de datos de alto nivel que están lo más lejanamente removidos de la entrada física y que continuan siendo entradas del sistema.

Se define como elementos de datos eferentes a aquellos elementos de datos de alto nivel que estan lo más lejanamente removidos de la salida física y que continuan siendo salidas del sistema.

Para identificar los elementos de datos aferentes, se inicia en las entradas físicas del sistema y moviendonos hacia adentro a lo largo del DFD hasta que se identifique un flujo que no puede ser considerado más como una entrada. Esto se encuentra determinado al juicio del desarrollador, pero lo que se busca es ir lo más lejos de

las entradas físicas como sea posible. Este proceso es realizado para cada flujo de entrada.

A menudo se puede encontrar que varios flujos de entrada pueden finalizar en el mismo elemento de datos aferente.

Para encontrar los elementos de datos eferentes se realiza lo mismo, pero comenzando de las salidas hacia el centro.

Realizando los pasos anteriores, se tienen algunas transformaciones en la mitad, entre los elementos de datos aferentes y los elementos de datos eferentes. Ellas son designadas como la transformación central.

3.- Primer nivel de factorización.

Habiendo identificado los elementos de datos aferentes y eferentes del sistema, se especificará un módulo principal el cual, al ser activado, realizará la tarea completa del sistema llamando a sus subordinados.

Para cada elemento de datos eferente que emerge de una transformación central se define un módulo eferente subordinado al módulo principal.

Finalmente, generar un módulo de transformación central que contendra como módulos subordinados a todos los módulos que forman parte de la transformación central.

El módulo principal es un control general o ejecutor de los procesos. Su función es controlar y coordinar los módulos aferentes, eferentes y de transformación. El módulo principal llama a sus módulos subordinados aferentes para obtener las entradas principales, pasar estás a sus módulos de transformación apropiados y entregar los resultados finales a los módulos eferentes.

4.- Factorización de módulos aferentes, eferentes

a) Módulos aferentes

Se toma el módulo aferente que se encuentra subordinado al módulo principal, se identifica la transformación (o computaciones) requerida para producir éste módulo aferente. Está última transformación (en sentido de transformación de datos) se vuelve la función de un nuevo módulo de transformación inmediatamente subordinado al módulo aferente inicial. Obviamente esta nueva transformación requerirá una nueva entrada: para cada entrada a esta última transformación se especificará un nuevo módulo aferente inmediatamente subordinado al primer módulo aferente. Cada uno se éstos nuevos módulos son factorizados recursivamente de la misma manera hasta que la última entrada física es alcanzada o el proceso es de otra forma terminado.

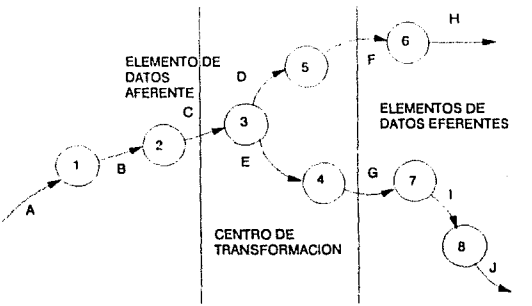
b) Módulos eferentes

La factorización de módulos eferentes es esencialmente lo mismo que se tiene en módulos aferentes. Para un módulo eferente dado, se debe buscar la siguiente transformación a ser aplicada, la cual traerá los datos más cercanos a su última forma física.

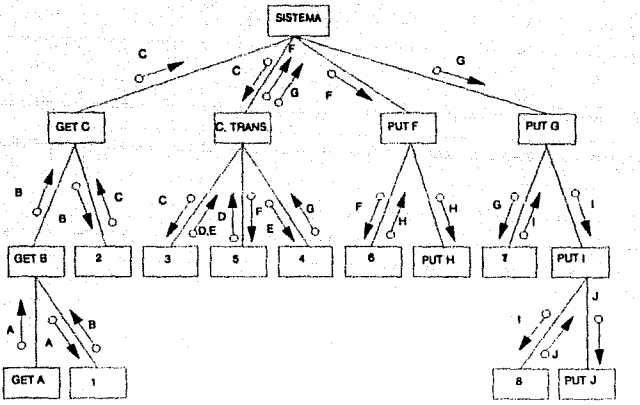
Resumiendo la descripción anterior se podría dar una "receta de cocina" para el uso de la técnica de análisis de transformación.

a) Obtener el DFD del sistema
b) Generar un módulo principal cuyo nombre será la identificación del sistema
c) Generar un módulo subordinado para cada elemento de datos aferente, para cada elemento de datos eferente y uno para la transformación central
d) Realizar la factorización de cada módulo aferente, manejando una estructura binaria recursiva para cada uno de éstos módulos (módulo "consigue (get)" y "transforma (transform)"
e) Realizar la factorización de cada módulo eferente, manejando una estructura binaria recursiva para cada uno de éstos módulos (módulo "transforma (transform)" y "pone (put)")
f) Los niveles inferiores del módulo de transformación central corresponden uno a uno a los procesos que se encuentran en el centro de transformación.

Ejemplo:
Del siguiente diagrama de Flujo de datos



Se pude obtener la carta estructurada:



3.2.2 Análisis de transacción

Existen numerosas situaciones en que estrategias adicionales pueden ser usadas como suplemento al enfoque de análisis de transformación. Una de estas estrategias es el análisis de transacción.

La estrategia de análisis de transacción simplemente reconoce que los flujos de datos (mostrados en el esquema al principio de la sección) pueden ser mapeados dentro de una estructura modular particular. Un centro de transacción de un sistema debe poder:

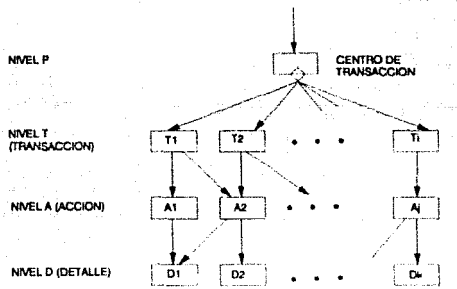
- Obtener y responder transacciones
- Analizar cada una de las transacciones para determinar su tipo
- Despachar sobre el tipo de transacción
- Completar el proceso para cada una de las transacciones

La estructura modular puede ser modelada como un sistema de 4 niveles llamados:

- Procesador de transacción (nivel P)
- Nivel de transacción (o nivel T)

- Nivel de acción (o nivel A)
- Nivel de detalle (o nivel D)

Gráficamente:



Pasos a seguir para hacer uso de la estrategia de análisis de transacción:

- Identificar los orígenes de transacciones.
  El diseñador tiene que reconocer módulos aferentes, eferentes y de transformación que generan transacciones; esto puede ser obvio después de los primeros pasos de factorización de un diseño orientado a transformación.
  Más de un flujo de transacciones puede alimentar el módulo de nivel P.
- Identificar las transacciones y su definición de acciones.
  El diseñador deberá tener cuidado de definir el proceso que se llevará a cabo cada transacción.
- Identificar situaciones potenciales en los cuales los módulos pueden ser combinados.
  Como en el caso de análisis de transformación, a menudo se pueden encontrar situaciones en las cuales un módulo de nivel intermedio puede ser creado desde módulos de bajo nivel funcionalmente cohesivos. Esta combinación probablemente sea apropiada en situaciones en las cuales la semántica o la sintaxis de varias transacciones es similar.
- Para cada transacción o colección cohesiva de transacciones especificar un módulo de transacción para efectuar este proceso.

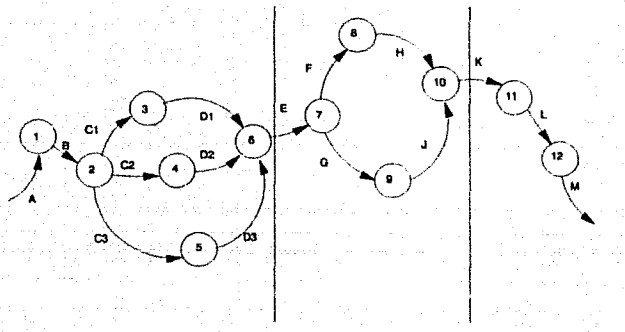
Debido a que las transacciones en un sistema son a menudo similares, se puede intentar agrupar el procesamiento de varias transacciones dentro de un módulo, esto deberá ser evitado si el módulo resultante tiene baja cohesión

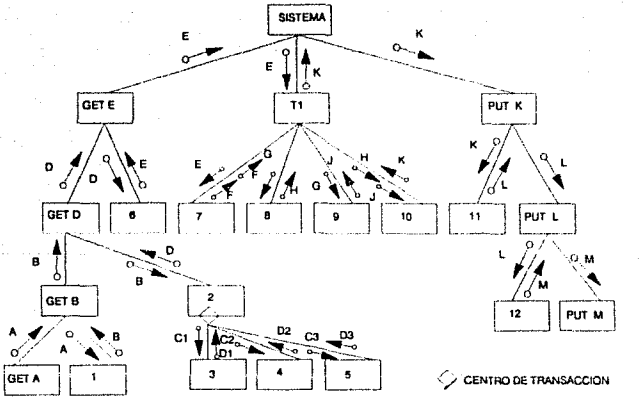
- Para cada acción en una transacción, especificar un módulo de acción subordinado apropiado al (los) módulo (s) de transacción. En esencia este es el paso de factorización discutido para el análisis de transformación. Hay que hacer notar que muchos módulos de transacción comparten módulos de acción comunes.
- Para cada paso detallado en un módulo de acción, especificar un apropiado módulo subordinado de detalle a cualquier módulo de acción que necesite éste.
  Este paso es la continuación del proceso de factorización.
  Para grandes sistemas que tienen transacciones complejas se tendrán varios niveles de detalle.

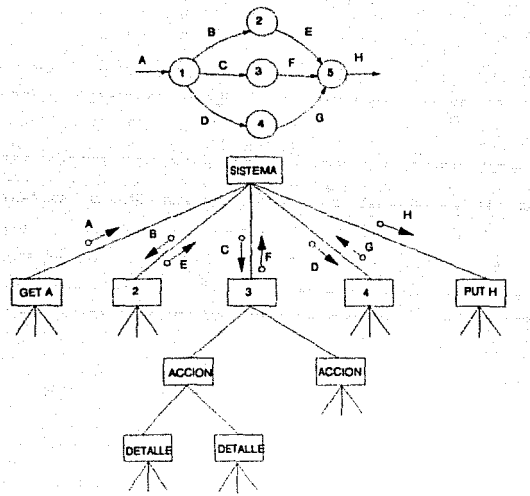
Al igual que en el análisis de transformación se podría dar una "receta de cocina" para el análisis de transacción:

a) El módulo que corresponde al proceso de centro de transacción es por si mismo un centro de transacción, es decir, éste maneja un módulo subordinado para cada uno de los procesos paralelos del DFD.
b) Cada módulo subordinado bajo el centro de transacción es asignado como uno de los procesos paralelos. Estos subordinados forman el nivel de transacción.
c) Bajo cada uno de los módulos esta un conjunto de módelos de acción (que son las funciones que combinadas describen a la transacción). Los módulos de acción pueden ser compartidos por los módulos de transacción.
d) Bajo los módulos de acción está un nivel de módulos de detalle, que son usualmente subrutinas útiles llamadas por uno o más de los módulos de acción.

Ahora que se ha definido las metodologías de análisis de transaccion y análisis de transformación, se muestra los siguiente ejemplos con la finalidad de despejar dudas existentes.


1
2
3
4
5
6
7
8
9
10
11
12
A
B
C1
C2
C3
D1
D2
D3
E
F
G
H
J
K
L
M


SISTEMA
GET E
T1
PUT K
GET D
GET B
GET A
PUT L
PUT M
CENTRO DE TRANSACCION



3.2.3.- Cohesión y acoplamiento

Uno de los principios fundamentales del diseño estructurado, es que un sistema deberá ser fragmentado en módulos manejables. Dicha fragmentación tendrá que ser efectuada de tal manera que los módulos queden tan independientes como sea posible (criterios de acoplamiento) y que cada módulo realice una simple función relacionada con el problema (criterios de cohesión).

3.2.3.1.- Acoplamiento

El acoplamiento es una medida de la firmeza de interconexiones que existe entre módulos.

El acoplamiento proporciona el grado de interdependencia entre módulos.

Idealmente se busca una interdependencia entre módulos, de tal forma que al efectuar una modificación en un módulo específico, no se tenga que modificar otros módulos, es decir, se busca que se tenga el acoplamiento más bajo posible.

Existen 5 tipos de acoplamiento que pueden ocurrir entre un par de módulos

(Que van desde el bueno o flojo hasta el malo o estrecho)

1.- Acoplamiento de datos
2.- Acoplamiento de estampa
3.- Acoplamiento de control
4.- Acoplamiento común
5.- Acoplamiento de contenido

Dos módulos pueden estar acoplados por más de un tipo de acoplamiento o por el mismo tipo varias veces.

- Acoplamiento de datos

  Dos módulos estan acoplados por datos si éstos se comunican mediante parámetros, siendo cada parámetro un sólo campo o una tabla homogenea (una tabla en la cual cada elemento contiene el mismo tipo de información).

  El acoplamiento de datos es la comunicación necesaria de datos entre módulos.

  Lo que se debe evitar en este tipo de acoplamiento son los datos "trampa" que son aquellas piezas de información que pasan sin objeto alguno (aparentemente) alrededor de un sistema sin significado para la mayoría de los módulos a través de los cuales pasa.

- Acoplamiento de estampa o estampilla

  Dos módulos estan acoplados por estampa, si ambos se refieren a la misma estructura de datos.

  Se debe entender como una estructura de datos a una pieza compuesta de datos, tal como un registro que consiste de un número de campos.

  El acoplamiento de estampa tiende a exponer a un módulo más datos que los que éste necesita, con posiblemente consecuencias desastrosas.

  Hay que eliminar de los módulos, tanto como sea posible, los datos superfluos, pero no hay que rechazar la idea de usar estructuras de datos, estas son buenas en la medida que tienen un significado en el problema original (como un todo), son una manera excelente de reducir el acoplamiento excesivo de datos.

  A la idea de coleccionar datos en una estructura artificial de datos se le llama AMASAR (building), hacer esto no ofrece más que una obscuridad innecesaria.

- Acoplamiento de control

Dos módulos estan acoplados por control, si uno pasa al otro una pieza de información con el propósito de controlar la lógica del otro. En un módulo acoplado por control, el modulo subordinado (al que se le indica cual es el trabajo que se debe realizar) no es una caja negra.
Se puede presentar que la pieza de control sea dirigida hacia arriba desde el módulo subordinado al módulo jefe, a ésto se le llama "inversión de autoridad" y se sigue considerando como acoplamiento de control.
Cuando en el acoplamiento de control se observa que una señal que es pasada de un módulo a otro con el propósito de control, significa que se tiene una pobre fragmentación, ya sea que una función ha sido definida entre los dos módulos o que un módulo tiene que estar enterado de las capacidades específicas de su jefe.

- Acoplamiento común

Dos módulos se encuentran acoplados en forma común, si ambos se refieren a la misma área global de datos.
El acoplamiento común no es bueno por las siguientes razones:
a) Un error en cualquier módulo que usa una área global, se puede manifestar en cualquier módulo que usa esa misma área global de datos.
b) Los módulos que hacen referencia a datos globales usualmente lo hacen por un nombre explícito (se debe conocer ese nombre para poder accesarlo, y no como en los módulos llamados por parámetros que no estan atados a nombres específicos en el mundo exterior)
c) Algunas veces se puede abusar drasticamente de las áreas globales por diferentes módulos, para almacenar piezas de información completamente diferentes
d) Programas con muchos datos globales son extremadamente difíciles de entender cuando se realiza el mantenimiento a dichos programas, porque es difícil saber que datos son utilizados por un modelo en particular
e) Si un valor de un área global de datos es cambiado, es difícil saber que modelos deberan ser cambiados.

- Acoplamiento de contenido
  Se dice que dos módulos tienen acoplamiento por contenido (o patológico) si uno se refiere al interior del otro de cualquier forma; por ejemplo: si un módulo busca dentro de otro, si un módulo modifica instrucciones o comando dentro del otro. Tal acoplamiento elimina el sentido de las "cajas negras".

Una forma para evaluar el acoplamiento de un diseño, es suponer que cada módulo será modificado por un diferente programador, y realizar las siguientes preguntas:

¿Qué tan independiente pueden trabajar los programadores?-
¿Existe cualquier hecho, suposición o decisión de implementación de la cuál necesita estar enterado más de un módulo?
¿Qué tan probable es que el hecho, suposición o decisión cambiará?
¿Hay alguna forma en la cuál el cambio pueda ser aislado en un módulo?

Responder a estas preguntas determinará cuales cambios de los usuarios posiblemente requerirán modificación a un gran número de módulos.

3.2.3.- Cohesión

Es una medida de la fuerza, firmeza, intensidad de asociación funcional entre elementos dentro de un módulo, entendiendose por elemento una instrucción, un grupo de instrucciones o una llamada a otro módulo, es decir, cualquier pieza de código que realiza un trabajo.

Niveles de cohesión. (Que va desde bajo hasta alto nivel de cohesión

Cohesión coincidental
Cohesión lógica
Cohesión temporal
Cohesión procedural
Cohesión comunicacional
Cohesión secuencial
Cohesión funcional

- Cohesión coincidental.
  Un módulo coincidentalmente cohesivo es uno, cuyos elementos contribuyen a actividades con ninguna relación significativa entre uno y otros.

La cohesión coincidental ocurre cuando hay poca o ninguna relación constructiva entre los elementos de un módulo. Cuando se presenta ésta, al módulo resultante se le llama "módulo aleatorio" (random module). Afortunadamente un módulo que es puramente coincidental es raro que se presente.

- Cohesión lógica.
  Un módulo es logicamente cohesivo si sus elementos contribuyen a actividades de la misma categoría general, en las cuales la actividad o actividades son selecionadas fuera del módulo.
  Los elementos de un módulo están logicamente asociadas, si uno puede pensar que ellos se encuentran dentro de la misma clase lógica de funciones similares o relacionadas.
  Un módulo logicamente cohesivo contiene un número de actividades del mismo tipo general y para usar el módulo se seleccionan solamente las partes que se necesitan. Por ejemplo: un módulo que combine todos los elementos del procesamiento que se tienen dentro de la clase "entrada" (input), es decir, logicamente relacionados en virtud de que son opciones de entrada.

- Cohesión temporal
  Es común en la práctica de la programación, colocar dentro de un lugar sencillo (algunas veces formado por una subrutina) todas los elementos que tengan que ver con una inicialización. De ésta forma se tiene un módulo de inicialización, que lee cintas, inicializa contadores, pone acumuladores a cero, etc. Tal módulo puede ser visto como asociado temporalmente. (sus elementos son relacionados en el tiempo).
  La cohesión temporal indica que todas las ocurrencias de todos los elementos de un proceso en una colección, ocurren durante el mismo período limitado de tiempo durante la ejecución del sistema.

- Cohesión procedural
  Un módulo proceduralmente cohesivo es uno cuyos elementos están involucrados en diferentes actividades y posiblemente no relacionadas entre sí, en los cuales el control fluye de cada actividad a la siguiente (contrario a lo que sucede en un módulo secuencialmente cohesivo, en el que no fluye control sino datos de una actividad a la siguiente)

Elementos de procesamiento asociados proceduralmente son elementos de una unidad procedural común, entre ellos son combinados dentro de un módulo de cohesión procedural porque ellos son encontrados en la misma unidad procedural. La unidad procedural común puede ser una iteración (loop), un proceso de decisión o una secuencia lineal de pasos.
La cohesión procedural asocia elementos de proceso en base a su relación procedural o algorítmica.

- Cohesión comunicacional
  Para decir que un conjunto de elementos está comunicacionalmente asociado, se tiene que todos los elementos operan sobre el mismo conjunto de datos de entrada y/o producen la misma salida de datos.
  La cohesión comunicacional, a pesar de no ser máxima, es suficientemente alta para ser aceptada.

- Cohesión secuencial
  Un módulo secuencialmente cohesivo es aquel cuyos elementos están involucrados en actividades tales que los datos de salida de una actividad sirven como datos de entrada a la siguiente.
  En la asociación secuencial los datos de entrada (o resultados), sirven como datos para el siguiente procesamiento.
  Un módulo secuencial puede contener una función, solamente parte de una función, o más de una función.

- Cohesión funcional
  Un módulo funcionalmente cohesivo contiene elementos que contribuyen a una tarea relacionada con un y sólo un problema.
  En un módulo funcional cada elemento de procesamiento es una parte integral de y es esencialmente para, la realización de una simple función.

Reglas para ayudar a distinguir módulos no funcionales.

- Si el único camino razonable para describir la operación de un módulo es, una sentencia compuesta a una sentencia que contiene una coma o una sentencia que contenga más de un verbo,

entonces el módulo no es funcional.

- Si la sentencia descriptiva contiene palabras orientadas al tiempo tales como "primero", "siguiente", "después", "entonces", "inicio", "paso", "cuando", "hasta", o "para todos" entonces el módulo tiene probablemente cohesión temporal o procedural; algunas veces, pero no muy a menudo, tales palabras son indicativas de cohesión secuencial.
- Si el predicado de las sentencias no contiene un objetivo específico simple seguido del verbo, el módulo es probablemente cohesivo lógicamente.
- Palabras tales como "inicializa"y "limpia" en la sentencia, implican cohesión temporal.

3.2.3.3.- Heurísticas de diseño

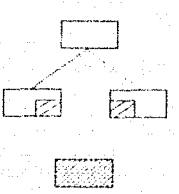
Por heurísticas se debe entender ciertos trucos los cuales, en general, tienen el efecto de incrementar la modularidad de un sistema. Ellas no están garantizadas para trabrajar, no ayudan mucho en la generación de la estructura inicial de un sistema, ninguna constituye una regla rigurosa y rápida.

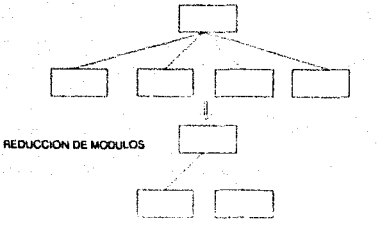
A pesar de lo anterior las heurísticas son valiosas debido a que ellas sirven como checadores o indicadores por medio de las cuales una estructura puede ser examinada para mejoramientos potenciales.

Cada indicador es una pista de que una estructura puede tener una configuración óptima. Sin embargo el juicio final de si una heurística deberá ser aplicada lo determinará la estructura intrínseca del problema que el sistema resolverá. Si el sistema refleja la estructura del problema, entonces existe una defensa en contra de aplicar cambios a la estructura basado en el uso de heurísticas.

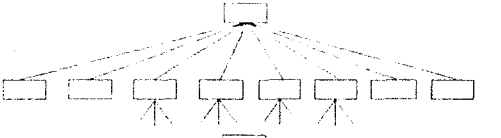
Algunas heurísticas que podrían ser aplicadas son:

- Evaluar la estructura del sistema para reducir el acoplamiento y la cohesión. Una vez que se tiene la estructura del sistema los módulos pueden expandirse o reducirse con la mirada puesta en la mejora de la independencia de los módulos. Se examina la descripción del procesamiento de cada módulo para determinar si una componente común del proceso puede expandirse a dos o más módulos y redefinirse como un módulo separado coherente. Por ejemplo:

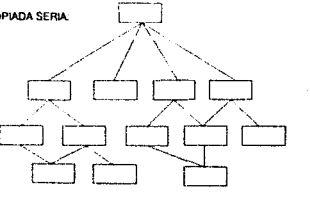




- Intentar minimizar las estructuras con un abanico de salida ancho y fomentar los abanicos de entradas conforme aumenta la profundidad.
  La estructura que se muestra en la figura contiene un abanico de salida muy grande (no hace uso efectivo de la factorización).



UNA DISTRIBUCION MAS APROPIADA SERIA



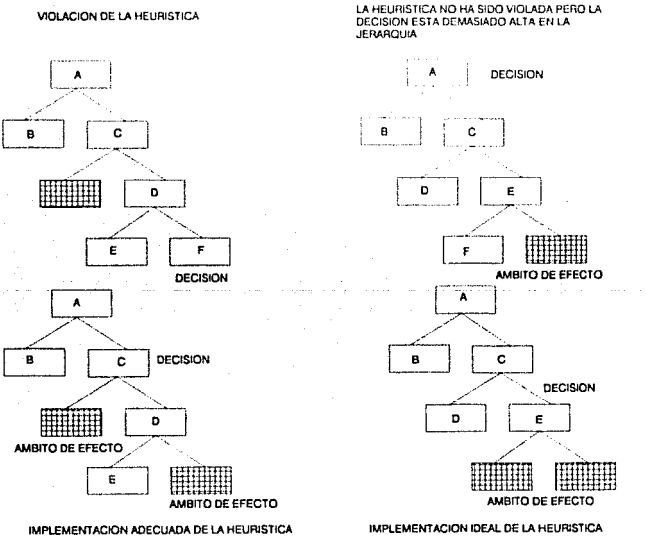
La estructura tiene una forma oval, idicando varias capas de control y una alta utilidad de los módulos de los niveles inferiores (fomenta los abanicos de entrada).

- Mantener el efecto de un módulo dentro del ámbito de control de ese módulo.

  El ámbito de efecto de un módulo m se define como todos los módulos que quedan afectados por una decisión hecha en el módulo m.

  El ámbito de control del módulo m lo constituyen todos los módulos que son sus subordinados (directos uno), es decir, el ámbito de control de un módulo es el módulo por si mismo y todos sus subordinados. El ámbito de control es un parámetro puramente estructural independiente de las funciones del módulo.

  Las siguientes figuras ilustran el ámbito de efecto y el ámbito de



control.

Las heurísticas presentadas (tamaño de un módulo, abanico entrada/salida y ámbito de control/efecto) pueden se extremadamente valiosas si son usadas apropiadamente, pero pueden conducir a un pobre diseño si son interpretadas literalmente. Si éstas son aplicadas ciegamente los resultados pueden ser catastróficos.

3.3.- Conceptos básicos para la construcción de una analizador léxico y un analizador sintáctico.

A continuación se presentaran algunos conceptos básicos para la construcción de un analizador.

3.3.1. Gramática y lenguajes

Un sistema formal es un cálculo o sistema lógico sin interpretación definida, que posee alfabeto, conjunto de palabras (axiomas) conjunto de relaciones (reglas de inferencia)

Los sistemas formales se componen de oraciones, las cuales son series o cadenas de símbolos pertenecientes a un alfabeto o vocabulario determinados.

Dado un alfabeto T cualquiera, se puede combinar sus símbolos para construir cadenas y es obvio que este número es infinito. Un lenguaje L sobre un alfabeto T es un conjunto de cadenas de T que a su vez es un subconjunto del conjunto de todas las posibles combinaciones de T.

La parte mínima de un lenguaje es un símbolo. Los símbolos se concatenan para formar cadenas o series , las cuales pueden o no pertenecer al lenguaje.

Un alfabeto se define como un conjunto finito de símbolos. La concatenación finita de los elementos del conjunto T se le denomina cerradura transitiva y se denota por T^*, a esta serie de concatenaciones se denominan con letras griegas minúsculas y se utiliza la letra para denotar la serie nulo o vacía, es decir una serie que no contiene elemento alguno.

Sea A un conjunto.La cerradura positiva (A +) del conjunto, es la unión de todas las potencias no nulas del conjunto, formalmente.

$A+ = A^1 U A^2 U A^3$

$A+ = A^* - \lambda$ Donde:

$A^0 = 0.1$

$A^1 = A$

$A^2 = AA$

$A^N = A^{N-1} - A$

Una gramática G es un conjunto finito de reglas para determinado lenguaje. Una gramática formal es un cuadruple que está formado por:

$G = (V_n, V_T, S, P)$

En donde:

Vn = es el vocabulario o símbolos introducidos, como elementos auxiliares para la definición de la gramática y que no figuran en las sentencias del lenguaje.

VT = Es un vocabulario terminal. Son todos aquellos elementos que forman parte del alfabeto.

S = Es el símbolo distinguido y que es elemento del conjunto de variables no terminales, es conocido como símbolo inicial.

P = Es el conjunto de producciones o reglas de derivaciones que tienen una construcción de la forma

X = > Y(X deriva a Y)

donde,

$X,Y \in ((V_T \cup V_N)^*)^2$

Es decir X e Y son cadenas que pueden construirse con la unión de los alfabetos V_T y V_n. X se llama consecuente o parte izquierda de la producción mientras que Y se llama parte derecha de la producción.

Una gramática de estructura de frases es un conjunto finito de reglas de la forma X = > Y

Ejemplo:

Oración Sujeto predicado

Sujeto artículo, sustantivo

Predicado verbo complemento

Artículo El/la/etc.

Reglas definidas para derivar oraciones del lenguaje definido por una gramática (usando un ejemplo)

VT = {a,b,c} Vn = {Z,A,B,C} G = {VT,Vn,Z,P}

1)Z = > ABC

2)AB = > BA

3)AC = > CA

4)BC = > CB

5)A = > a

6)A = > b

7)C = > c

1) Se selecciona una regla de producción y se usa el consecuente de dicha regla como la cadena inicial.

2) Se barren los antecedentes de la regla, buscando uno que sea una subcadena de la cadena inicial, si la búsqueda no tiene éxito termina el proceso.

3) Substituir la cadena encontrada por el consecuente de la regla.

4) Substituir la cadena inicial por la cadena derivada en el paso 3) y se repiten los pasos 2,3 y 4) hasta encontrar una oración con elementos terminales (letras minúsculas).

Ejemplo:

$Z => ABC => BAC => BCA => bCa => bca$

3.3.2.- Expresiones regulares

Una expresión regular permite representar en una forma compacta un conjunto de cadenas de caracteres agrupados.

Hay tres operaciones básicas que utilizan las expresiones regulares, que son la concatenación, la alteración y la cerradura.

- La concatenación es una operación binaria asociativa no conmutativa. El símbolo para la concatenación es la yuxtaposición de las dos expresiones. Si E_1 y E_2 son expresiones regulares, entonces E_1E_2 es la concatenación de las dos.

  Si E_1 y E_2 denotan el conjunto de caracteres S_1 y S_2 respectivamente, entonces E_1E_2 denotan el conjunto.

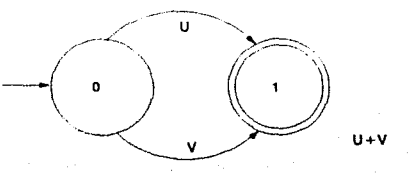
  $S = \{uv/u \in S_1 \ Y \ v \in S_2\}$

  Gráficamente:

- La alteración es una operación binaria asociativa conmutativa representada por el símbolo / ó +. Si E_1 y E_2 son expresiones regulares denotando el conjunto de caracteres S_1 y S_2, entonces E_1/E_2 da como resultado una expresión denotada por:

  S1US2

  Gráficamente:



- La cerradura es la tercera operación. Es una operación unaria si E es una expresión regular denotando a S como el conjunto de caracteres de esa expresión, entonces {E} es una expresión regular, llamada la cerradura de E también se denota por E* y esto representa al grupo de todas las cadenas formadas por la concatenación de los elementos S junto con la cadena vacia.
  El símbolo vacío se denota como ϵ , así E* de una expresión regular, E es una forma compacta de escribir la siguiente expresión regular infinita:
  $E^* = \epsilon / E / EE / EEE / EEEE / \ldots$
  formalmente:
  $\{E\} = \{XY / X \epsilon E \cap Y \epsilon \{E\}\} \cup \{\epsilon\}$

Por convención, la concatenación tiene mayor precedencia que la alteración y la cerradura, que tiene mayor precedencia que la alteración o la concatenación.

a/bc* (a)/(b(c*))

Los símbolos para la alteración y cerradura son llamados metasímbolos y no pueden ser del lenguaje regular.

Los elementos de una expresión regular son los símbolos del alfabeto S, el símbolo vacío y conjunto nulo 0.

El conjunto nulo es aquel que no contiene cualquier cadena de ni el símbolo

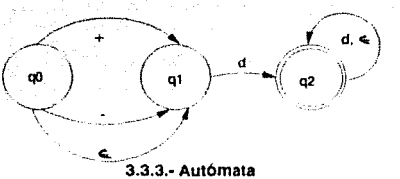
Identidades de las expresiones regulares

Las expresiones regulares satisfacen un número de identidades las cuales pueden ser usadas para reducir la complejidad de una expresión regular.

1) A + B = B + A	Conmutatividad de alteración
2) {0} = ϵ	cerradura del conjunto vacio
3) A + (B + C) = (A + B) + C	asociatividad de alteración
4) A(BC) = (AB)C	asociatividad de concatenación
5) A(B + C) = AB + AC	distributividad concatenación sobre la alteración
6) A ϵ = ϵ A = A	identidad de la concatenación
7) 0E = E0 = 0	cero de la concatenación
8) E* = E + E*	
9) (E*)* = E*	
10) E + $\emptyset$ = E	

Ejemplo de una expresión regular y su expresión en diagrama de estado.

(+/-/ϵ)dd*



3.3.3.- Autómata

Un autómata reconoce una cadena de la línea de entrada, empezando en la configuración inicial y después de una serie de movimientos (navegar) se llega a una configuración final.

Los autómatas finitos son los que reconocen lenguajes regulares, en donde el apuntador a la línea de entrada solo se mueve en un sentido y no posee memoria auxiliar.

Un autómata de estado finito es un sistema que se define como:

$M = \{Q, \Sigma, \delta, q0, F\}$

donde:

Q es un conjunto finito de estados que determina la conducta del autómata

Σ alfabeto de entrada

δ es la función de transición entre estados, que mapea un estado y la entrada de un símbolo a otro estado.

q0 estado inicial y pertenece al conjunto finito de estados

F estados finales cuando el autómata a llegado a un final lógico en el proceso de aceptación.

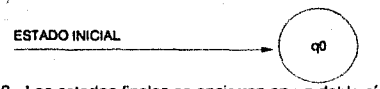
Tal como se ha definido el autómata se denomina no determinístico. Cuando por el contrario la función de transición δ es capaz de asignar un (estado, símbolo) = > estado siguiente único, entonces se dice que el autómata es finito y determinístico.

La función de transición de estados se entiende como $\delta = Qx\Sigma$ la cual va ha producir la dupla $\delta(q,a) = > q0$ donde:

q es un estado inicial cualquiera, a símbolo de entrada y q0 estado siguiente $(q,q0) \in Q$

Reglas para la construcción del diagrama de autómata de estado finito.

1.- Cada estado del conjunto Q (conjunto finito de estados) se dibuja como un círculo y una etiqueta



2.- Los estados finales se encierran en un doble círculo

3.- Para cada transición se dibuja un arco o flecha para la conexión del estado del dominio con el rango de la función y también se etiqueta con el mismo símbolo definido en el dominio de la función. Así (q1,a) = q2; se representa como:

3.3.4.- Analizador léxico

La tarea del analizador léxico-gráfico es identificar los símbolos del texto de entrada correspondientes a los símbolos especificados en las reglas gramaticales y transformarlos de una manera que correspondan a las abstracciones de las reglas gramaticales.

Además lleva a cabo la reconstrucción de las líneas de texto fuente, es decir, elimina símbolos de continuación de línea, concatena las partes formando una misma proposición o instrucción, detecta los finales de instrucción, etc.

Un token es una cadena de uno o más caracteres que tiene un significado gramatical preciso. Los tokens de los lenguajes de programación típicos pueden agruparse en las siguientes clases:

- Signos de puntuación
- Palabras reservadas
- Identificadores
- Constantes
- Operadores
- Separadores o terminadores

El analizador léxico-gráfico debe identificar cada token del texto de entrada y traducirlo a una representación interna que permita a las etapas posteriores distinguir cada token de los demás.

Las palabras reservadas forman un subconjunto de los identificadores,

debido que estos pueden ser una cadena de letras y normalmente estas no contienen dígitos. Esta particularidad dificulta la distinción entre palabras reservadas y variables manejadas en el programa.

Hay dos enfoques más comunes para hacer la distinción:

a) Distinguir a las palabras reservadas de los identificadores con una marca léxica. Por ejemplo: exigiendo que las palabras reservadas sean encerradas entre comillas o que esten precedidas por un símbolo especial.

b)Construir un diccionario inicializadolo con las palabras reservadas y tratarlas como si fueran identificadores, después se buscan éstos en el diccionario de palabras para poder decir a que clase pertenece el símbolo a identificar.

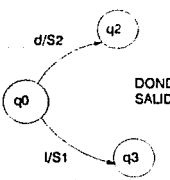
Cualquier token puede describirse como una gramática regular y por lo tanto puede ser ser reconocido por un autómata.

Se puede representar cada uno de los autómatas reconocedores de cada uno de los tokens mediante un arreglo y utilizar un único algorítmo para identificar a cualquier token.

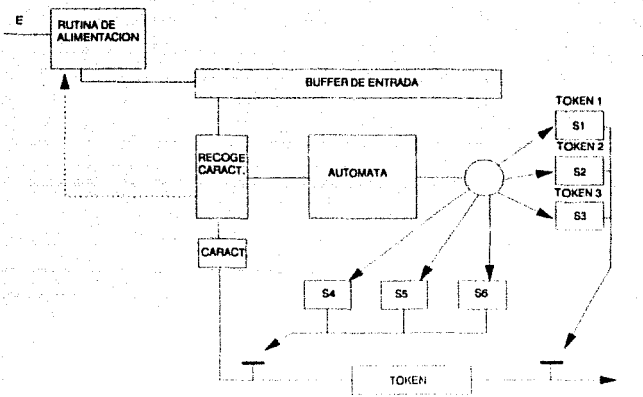
El funcionamiento de éste autómata puede ser descrito en términos muy sencillos, su comportamiento es cíclico y principia con el autómata en su estado inicial Q = q0

En cada nuevo ciclo, el autómata recibe la fuente de entrada, el siguiente caracter del texto, cambia a un nuevo estado siguiendo alguna de las transacciones de la figura, y envía una señal de salida (estas señales identifican el número de token asignado).

Ejemplo



Un esquema sencillo para representar lo anterior es el mostrado a continuación:



La rutina de alimentación proporciona información de entrada en el buffer y de ahí se va obteniendo cada uno de los caracteres de entrada del autómata. Cuando el autómata genera señales como S4, S5 y S6 se va formando cada uno de los tokens, y cuando se tienen las señales S1, S2 y S3 se obtiene una salida para cada uno de los tokens.

3.3.1.5 Analizador sintáctico (parser)

Una técnica para generar un analizador sintáctico es, diseñar un programa descendente que sea específico para un lenguaje dado y construirlo sistemáticamente según un conjunto de reglas que transforman la gramática en una serie de instrucciones.

reglas

gramática => instrucciones

Aquí interesa representar la sintáxis dada por un gráfico llamado "gráfico sintáctico o de reconocimiento".

Este gráfico representa el flujo de control durante el proceso, estos gráficos son llamados gráficos de Conway.

Se construye un analizador parcial para cada símbolo no terminal.

Cada analizador parcial tiene el objetivo de reconocer una subfrase generable a partir de su correspondiente símbolo no terminal.

Como se desea construir un gráfico que representa el analizador completo, entonces se hará corresponder una subgráfica con cada símbolo terminal.

A partir de los diagramas de Conway es fácil deducir un programa que acepte y analice sintácticamente un lenguaje; el diagrama representa el flujo de control del programa. Sin embargo al desarrollar éste, es muy importante seguir un conjunto de reglas de traducción; estas reglas son aplicables a un marco específico, el cual consiste de un programa principal que contiene una rutina para avanzar al siguiente símbolo y de los procedimientos correspondientes a los distintos objetivos parciales.

Conversión.

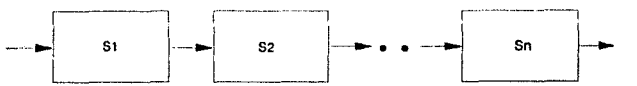
El programa principal está formado por una instrucción inicial que lee el primer caracter, seguida de una instrucción que activa el procedimiento del objetivo principal del análisis, obteniendo con ésto las rutinas individuales correspondientes a los objetivos parciales o gráficos sintácticos. Esto se logra aplicando las siguientes reglas.

Se denomina T(S) a la instrucción que se debe obtener al traducir el gráfico.

1.- Por medio de sustituciones apropiadas reducir el sistema de gáficos a un número de gráficos individuales, lo más pequeño posible.

2.- Traducir cada gráfico resultante a una declaración de procedimiento siguiendo las reglas 3 a 7

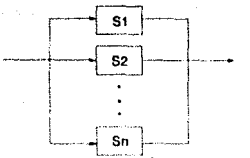
3.-Una secuencia de elementos



Se traduce con la instrucción compuesta:

```
Begin
      T(S1);
      T(S2);
      ....
      T(Sn);
End
```

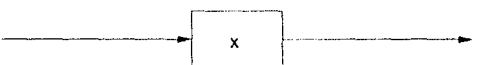
4.- Una bifurcación de elementos:



Se traduce como la instrucción selectiva o condicional.
Case ch of
L1: T(S1); IF ch IN L1 THEN T(S1) ELSE
L2: T(S2); IF ch IN L2 THEN T(S2) ELSE
Ln: T(SN); IF ch IN LnTHEN T(Sn) ELSE ERROR.
donde Li designa el conjunto de símbolos iniciales de la construcción. Si Li esta formado por un único símbolo a, entonces debe expresarse "ch en Li" como "ch = a"
5.- Un ciclo de la forma:

Se traduce por la instrucción
While ch in L do T(S)
donde T(S) es la traducción de S según las reglas 3 a 7 y L es el conjunto L = primero(s)
6.- Un elemento gráfico que designe a otro gráfico A

Se traduce por la instrucción de llamada de procedimiento.
7.- Un elemento gráfico que designe un símbolo terminal X

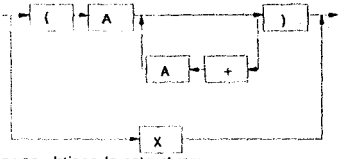


Se traduce por la instrucción:

```
if ch = x then
   read (ch)
Else
   error
```

Donde error es una rutina que se activa cuando se encuentra un error sintáctico.

Ejemplo: Traducir el gráfico A:



Del gráfico anterior se obtiene la estructura:

Procedimiento llamado:

```
Procedimiento A
Inicio procedimiento
if ch < >x then
        if ch'(' then
                error
        else
                read(ch)
                A
                read(ch)
                while ch = '+' do
                inicio while
                        read (ch)
                        A
                        read (ch)
                fin de while
                if ch< >')' then
                        error
                fin de if
        fin de if
fin de if
fin de procedimiento.
```

Procedimiento de llamado

```
Programa Analizar
Inicio
        read (ch)
        A;
fin
```

IV.- ESTANDARES DE DOCUMENTACION.

Durante el desarrollo del software, nunca se dá la importancia a la documentación que se debe generar, es por ello que en éste capítulo se presentará algunos puntos que pueden ser tomados como guías para la elaboración de esa documetación.

Cuando se concluye el desarrollo de un sistema, se obtiene un documento de especificación estructurada, el cual deberá ser incluido dentro del manual de referencia.

A continuación se establecen algunos puntos que podrían tener los manuales de referencia técnica y de usuario. Se debe tener en cuenta que estos puntos han sido establecidos en base a lecturas, otros manuales y experiencias laborales, y que por lo tanto pueden no ser los más óptimos.

4.1.- Manual de referencia técnica

I.- Indice.

El manual deberá contener la lista de los elementos que integran al mismo, de tal forma que reflejen una organización lógica.

Se debe establecer una referencia (número de hoja por ejemplo) para la localización de cada uno de los temas contenidos en ésta sección.

II.- Introducción.

En ésta sección se detallará el objetivo esencial del sistema describiendo el funcionamiento del mismo de una manera concreta y condensada.

Se describirán las secciones del manual con textos, cartas, con el fin de que el usuario del manual no trabaje con un documento denso y difícil de entender.

III.- Características técnicas.

En ésta sección quedarán integrados los aspectos técnicos más relevantes del sistema, relacionado con los recursos de software utilizados para su construcción y los requerimientos mínimos de hardware para poner al sistema en operación.

a) Tipo del sistema.- Se refiere a la forma de trabajo del sistema (interactivo, batch o ambos).

b) Lenguaje de programación utilizado en la implementación del sistema.

c) Herramientas de software utilizadas.

Se debe realizar una descripción explicando los paquetes de software que se utilizan en el sistema, explicando las funciones que desempeñan.

d) Tipo de organización de archivos utilizados.

e) Requerimientos de equipo de computo (hardware).

IV.- Descripción general y funcional del sistema.

En ésta sección se incluirá el "documento de especificación estructurada; que como se recordara esta formado por DFD's, flujos de datos, archivos, diccionario de datos, descripción de procesos y las cartas estructuradas generadas.

El tener estos documentos incluidos dentro del manual es de gran utilidad, debido a que antes de efectuar un cambio en el sistema (mantenimiento) los diagramas nos pueden servir para efectuar los cambios dentro de ellos, antes de efectuar los cambios a nivel de programación.

En ésta sección, al tener los elementos descritos se está teniendo una descripción general (análisis) y específica (diseño) del sistema.

V.- Descripción de programas.

En ésta sección se encuentra una descripción tanto funcional como estructural de cada uno de los programas.

Para ello hay que dividir la descripción en las siguientes secciones:

a) Comentarios.

Muchas personas tratan los comentarios como algo que debe ser agregado después de que un programa ha sido terminado y no hay nada más equivocado que ésto. Un programa, módulo, función o subrutina debe ser documentado, debido a que en la mayoría de las ocasiones, la persona que realizó el programa ya no trabaja en la institución, y en consecuencia, la persona que está realizando el mantenimiento puede no comprender la lógica del mismo.

El comentario debe efectuarse al comienzo de cada unidad (programa principal, subrutina, procedimiento, función).

El contenido del comentario debe ser:

1.- Nombre de la unidad

2.- Nombre del programador

3.- Fecha de elaboración/modificación
4.- Descripción de la unidad
5.- Condiciones de entrada
6.- Condiciones de salida
7.- Breve historia de las modificaciones realizadas y del porque fueron necesarias.

La finalidad de escribir los comentarios es la de establecer un registro de quién elaboró la unidad, así como el de todas las modificaciones efectuadas.

b) Subprogramas por programa.

Se refiere a los programas (subprogramas) que son llamados dentro de la rutina principal. Esto es útil para determinar que programas (archivos físicos) forman el sistema completo.

En este punto no se está hablando de los módulos que se generarón durante el proceso de diseño del sistema.

La relación que se puede tener es:

Archivo Tamaño Fecha (última actualizacion)

c) Módulos por programa.

Se refiere a los procedimientos, funciones, subrutinas que se encuentran declaradas en cada programa (subprograma).

La relación que se puede tener es:

Nombre de la unidad Tipo ProgramaPágina (archivo)

d) Alcance de módulos.

Esta opción es similar a la anterior, sólo que en aquí se identifican el inicio, terminación y proceso de cáda módulo.

La relación que se puede tener es:

Nombre de la unidad De la línea A la línea sec. ejecutable. Archivo(programa)

e) Estructura modular.

Establecimiento de las llamadas (de las unidades) en forma modular, mostrando una estructura jerárquica; indicando a su vez los módulos que son llamados a partir de las llamadas iniciales, indentando cada uno de los niveles de llamada.

Esto es muy interesante porque nos muestra de una manera sencilla la lógica del programa (Esto es si utilizan algunas guías de estilo de programación).

Por ejemplo:

llamada rutina A
    llamada rutina C son llamadas por
    llamada rutina D la rutina A
llamada rutina B

f) Referencias cruzadas

En el establecimiento de las relaciones que existen entre los módulos.

1) Módulo llamado vs módulo llamador.
Para ello se tiene una estructura

Nombre del módulo	llamado por
módulo A	módulo B
	módulo C
	módulo D
	módulo E
módulo F	módulo B
	módulo L

2) Módulo llamador vs módulo llamado
Para ello se tiene la estructura:

Nombre del módulo	módulos que llama
módulo B	módulo A
	módulo F
módulo C	módulo A
	módulo F

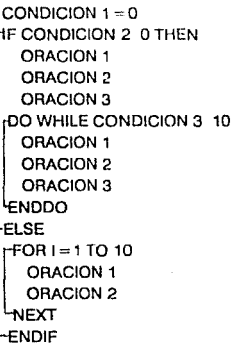
g) Impresión editada de programas.

Cuando un desarrollador se encuentra revisando un programa, puede tener problemas para identificar las estructuras que son utilizadas en la programación estructurada (secuenciación, selección e iteración). Esto es debido al anidamiento que existe en las diferentes estructuras de control.

El problema de entendimiento se presenta aún y si se realiza la identación de cada una de las estructuras de control.

Para ello, es necesario anexar a la documentación programas en los que se pueda identificar, en forma simple, cada una de las estructuras.

El utilizar símbolos gráficos para denotar el inicio y terminación de estructuras es una buena práctica,por ejemplo:



h) Modulos contenidos.

Se refiere a la relación existente entre un módulo y aquellos que son incluidos dentro de éste. La relación que se puede tener es:

Nombre del módulo Módulos contenidos

VI.- Formas de compilación.

Establecimiento de los pasos necesarios para realizar la compilación y ligado de todos los módulos, indicando el lugar donde se encuentren los archivos de trabajo, así como las ecuaciones necesarias para poder efectuarla.

VII.- Identificación de los procesos entrada/salida.

Se refiere a los formatos de pantallas de captura y de reportes.

VII.- Apéndices.

4.2 .- Manual de usuario.

El manual de usuario es aquel documento que debe contener todo lo que se requiere para el entendimiento del sistema. Tiene que proporcionar los elementos suficientes para que el usuario del mismo pueda operarlo adecuadamente.
Los puntos que debe contener el manual de usuario son:

I.-Indice.

El manual deberá contener la lista de los elementos que integran al mismo, de tal forma que reflejen una organización lógica.

II.- Introducción.

En está sección se explicará el objetivo del sistema describiendo el funcionamiento del mismo de una manera concreta y condensada.

III.- Sobre el manual.

En ésta sección se describirán cada una de las partes que forman éste manual. La descripción de cada parte incluirá una pequeña narrativa de la sección indicando la función, así como las partes que constituyen a la sección.

IV.- Organización.

La finalidad de ésta sección, es la de mostrar una estructura jerárquica entre las diferentes opciones del sistema, así como el manejo de teclas de uso común.

V.- Operación.

En ésta sección se explicará de manera detallada, cada una de las opciones que formarán parte del sistema.

Para la descripción de cada una de éstas opciones se utilizará el siguiente formato.

a) Función: Establece la función que realiza la opción.

b) Camino: Muestra las acciones que deberán realizarse para llegar a la opción que se está seleccionando.

c) Opciones: Muestra las operaciones que pueden efectuarse, así como las restricciones y consideraciones a tomar para su correcto funcionamiento.

VI.- Apéndices.

En esta parte se pueden mostrar los términos utilizados (glosario de términos), mensajes de error, reportes (los formatos de entrada deberán haber sido descritos en la sección anterior) e instrucciones de operación

V.- DESARROLLO DE LA HERRAMIENTA.

En éste capítulo se muestran los elementos involucrados en la elaboración del documentador para pascal, así como los alcances del mismo.

La documentador se desarrolló con la finalidad de que proporcionara una referencia lo suficientemente completa del funcionamiento de cualquier sistema desarrollado en pascal, además de que en el desarrollo de la misma se pude demostrar que conceptos que han sido adquiridos en forma separada pueden reunirse para formar un todo (compiladores, estructura de datos, programación estructurada e ingeniería de software).

El documentador se desarrolló utilizando el lenguaje de programación Pascal y se encuentra orientado a sistemas que hayan sido elaborados en el mismo lenguaje.

En este momento, el lector podría estar pensando en el porque desarrollar ésta herramienta en pascal y para pascal, lo cual se podría contestar con las siguientes razones:

1.- Es un lenguaje totalmente didáctico que se basa en los tipos básicos de estructura utilizados en la programación estructurada, y en consecuencia, muy acorde a las técnicas de análisis y diseño estructurado.

2.- Si la herramienta es desarrollada para generar la documentación de un sistema, ¿Porqué no probarla con un sistema lo suficientemente extenso como lo es el documentador por sí mismo?

3.- El manejo de estructuras dinámicas es lo bastante sencillo, completo y claro para el desarrollo del documentador.

4.- El lenguaje es muy amigable, lo cuál es de gran importancia debido a que los programas desarrollados puden ser entendidos con gran facilidad.

Alcances:

El documentador para pascal básicamente cubrirá los puntos señalados en la sección de descripción de programas del manual de referencia técnico, descritos en el capítulo 4, que son:

- Relacion de subprogramas por programa
- Relación de módulos por programa
- Relación de módulos contenidos
- Relación de módulos llamados vs. módulo llamador
- Relación de módulo llamador vs. módulo llamado
- Alcance de módulos
- Estructura modular
- Impresión editada de programas

El lenguaje de programación utilizado es pascal.

Los sistemas a documentar deberán estar desarrollados en pascal.

El pascal utilizado es la versión 5.0 de Borland.

Análisis.

Considerando los alcances de la herramienta, se elaboró el análisis de la misma utilizando la técnica de análisis estructurado de Tom Demarco. (para mayor información referirse capítulo 3)

El documento generado está formado por:

Diagramas de Flujos de Datos.- Diagramas que detallan el funcionamiento de la herramienta y que se encuentran organizados en base a una nivelación.

Miniespecificaciones.- Establecen una narrativa de los procesos terminales (primitivas funcionales), los cuales están formados a través de un diccionario de datos y un conjunto de palabras reservadas que identifican un tipo de estructura o acción a realizar.

Diccionario de datos.- Establece el conjunto de elementos que

pueden ser utilizados tanto en la descripción de un proceso, como en la generación de diagramas de flujos de datos.

Automata.- Establece la manera en que deben ser construidos los Tokens, que son utilizados para identificar las relaciones exitentes entre el código, con la finalidad de generar una documentación.

Los diagramas que se presentan a continuación, son el resultado de un conjunto de iteraciones efectuadas para refinar un diagrama de flujos de datos, partiendo de una aproximación inicial.

La organización de la primera parte del documento de especificación estructurada es:

Diagrama de contexto

Diagrama de nivel 0

Diagramas Nivelados
- Diagrama 3
- Diagrama 6
- Diagrama 9
- Diagrama 13
- Autómata
- Descripción de salidas del autómata.

Diccionario de Datos
- Descripción de Procesos (Miniespecificaciones)
- Palabras reservadas miniespecificaciones

Para obtener la impresión editada de programas, en cuanto a miniespecificaciones se refiere, utilice la técnica mostrada en el capítulo 3 para generar un analizador sintáctico partiendo de los gráficos de Conway de las instrucciones.

NOTA: La segunda parte del documento es generada durante el diseño.

Los diagramas mostrados contienen los elementos suficientes para poder entenderlos sin ninguna dificultad.

DIAGRAMA DE CONTEXTO

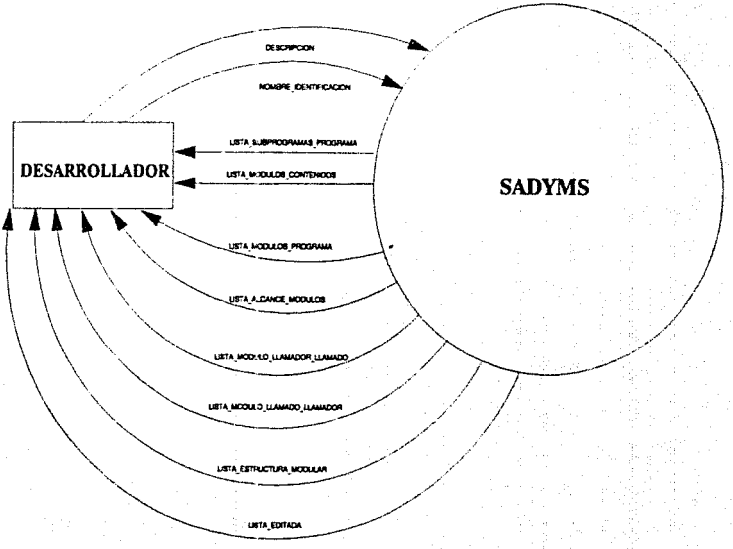


DIAGRAMA 0

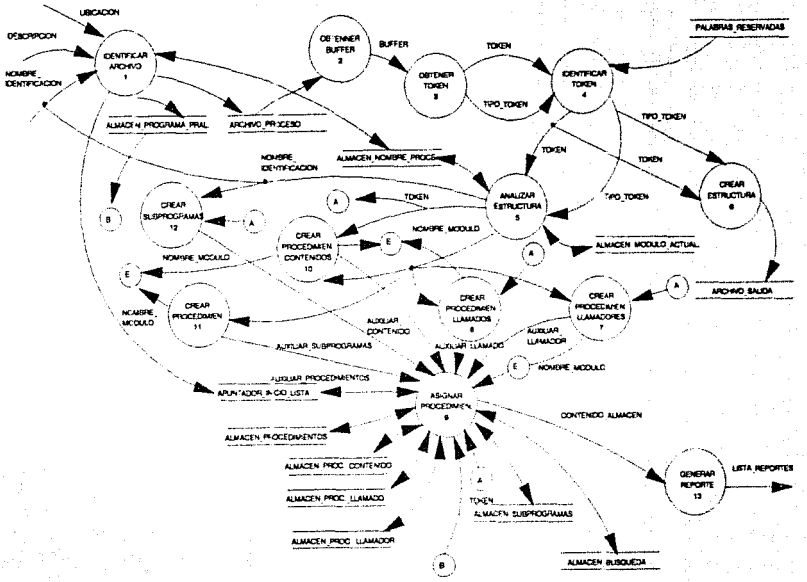

UBICACION
DESCRIPCION
NOMBRE_IDENTIFICACION
IDENTIFICAR ARCHIVO 1
ALMACEN PROGRAMA PRIAL
ARCHIVO_PROCESO
OBTENNER BUFFER 2
BUFFER
OBTENER TOKEN 3
TOKEN
TIPO_TOKEN
IDENTIFICAR TOKEN 4
PALABRAS RESERVADAS
TIPO_TOKEN
NOMBRE_IDENTIFICACION
ALMACEN_NOMBRE_PROCE
ANALIZAR ESTRUCTURA 5
TOKEN
TIPO_TOKEN
CREAR ESTRUCTURA 6
ALMACEN MODULO_ACTUAL
ARCHIVO SALIDA
CREAR SUBPROGRAMAS 12
CREAR PROCEDIMIEN CONTENIDOS 10
NOMBRE_MODULO
CREAR PROCEDIMIEN 11
CREAR PROCEDIMIEN LLAMADOS 8
CREAR PROCEDIMIEN LLAMADORES 7
AUXILIAR CONTENIDO
AUXILIAR LLAMADO
AUXILIAR LLAMADOR
AUXILIAR SUBPROGRAMAS
AUXILIAR_PROCEDIMIENTOS
APUNTADOR_INICIO_LISTA
ASIGNAR PROCEDIMIEN 9
CONTENIDO ALMACEN
ALMACEN_PROCEDIMIENTOS
ALMACEN PROC CONTENIDO
ALMACEN PROC LLAMADO
ALMACEN PROC LLAMADOR
GENERAR REPORTE 13
LISTA REPORTES
ALMACEN SUBPROGRAMAS
ALMACEN BUSQUEDA

DIAGRAMA 3

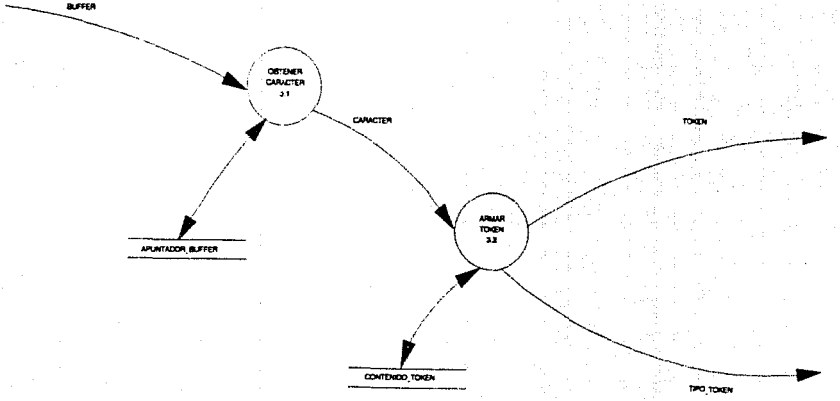

BUFFER
OBTENER CARACTER 3.1
CARACTER
ARMAR TOKEN 3.2
TOKEN
APUNTADOR_BUFFER
CONTENIDO_TOKEN
TIPO_TOKEN

DIAGRAMA 6

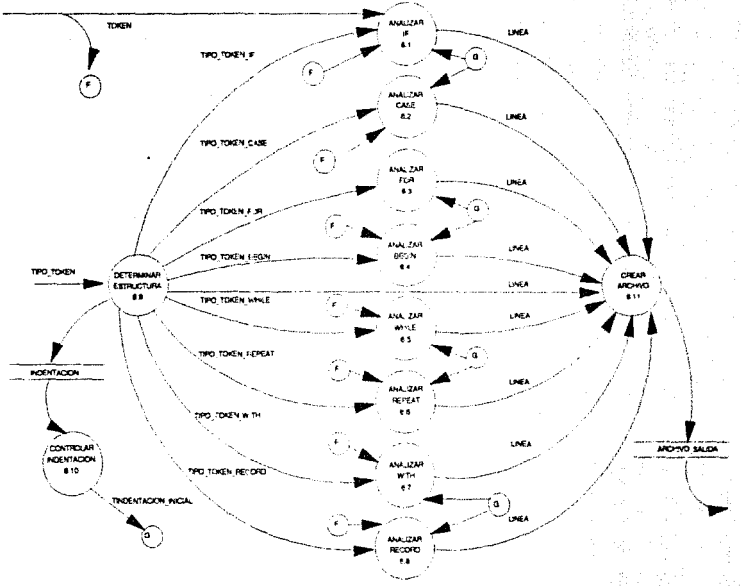

TOKEN
F
TIPO_TOKEN_IF
ANALIZAR IF 6.1
LINEA
G
TIPO_TOKEN_CASE
ANALIZAR CASE 6.2
TIPO_TOKEN_FOR
ANALIZAR FOR 6.3
TIPO_TOKEN_BEGIN
ANALIZAR BEGIN 6.4
TIPO_TOKEN
DETERMINAR ESTRUCTURA 6.0
TIPO_TOKEN_WHILE
ANALIZAR WHILE 6.5
TIPO_TOKEN_REPEAT
ANALIZAR REPEAT 6.6
TIPO_TOKEN_WITH
ANALIZAR WITH 6.7
TIPO_TOKEN_RECORD
ANALIZAR RECORD 6.8
CREAR ARCHIVO 6.11
INDENTACION
CONTROLAR INDENTACION 6.10
TINDENTACION_INICIAL
ARCHIVO SALIDA

DIAGRAMA 9

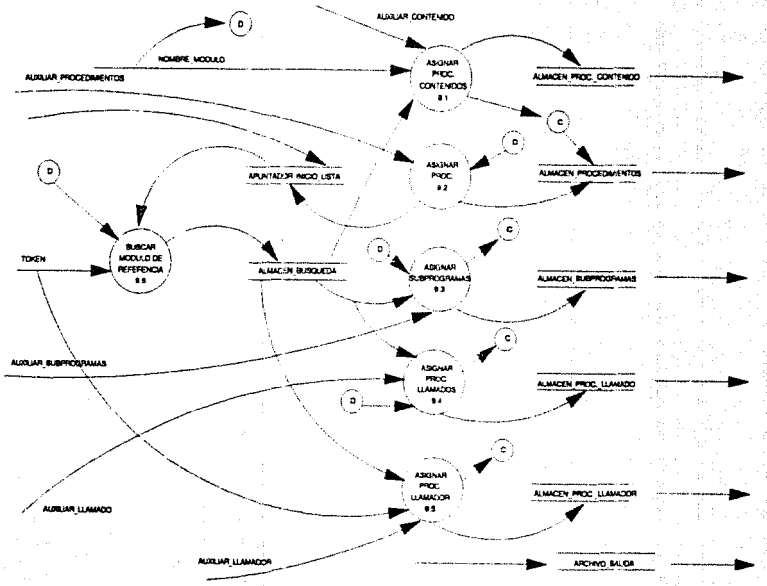
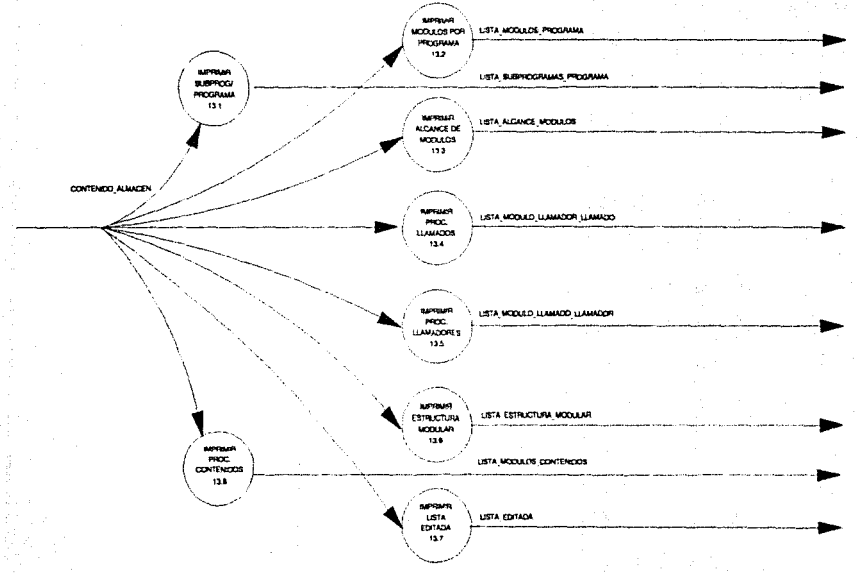
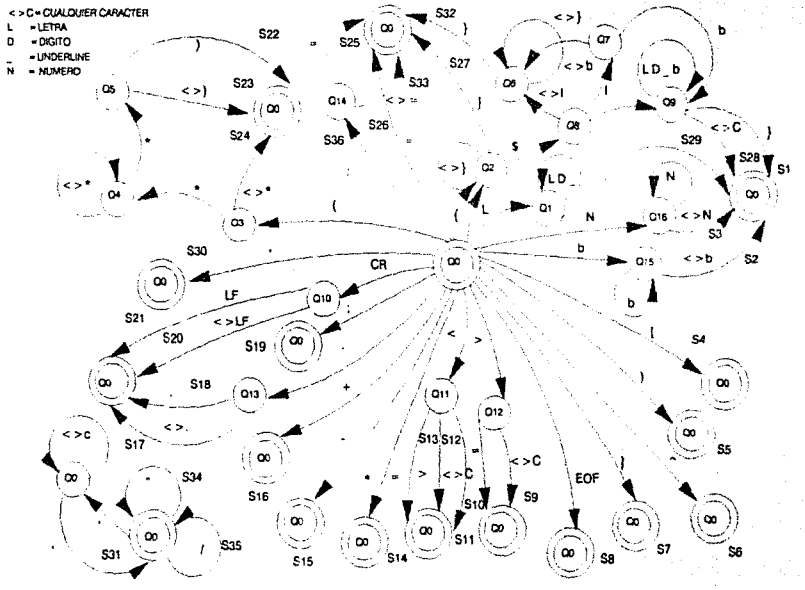

AUXILIAR CONTENIDO
D
NOMBRE MODULO
AUXILIAR PROCEDIMIENTOS
ASIGNAR PROC. CONTENIDOS 9.1
ALMACEN PROC. CONTENIDO
C
D
ASIGNAR PROC. 9.2
ALMACEN PROCEDIMIENTOS
APUNTADOR INICIO LISTA
D
BUSCAR MODULO DE REFERENCIA 9.5
TOKEN
ALMACEN BUSQUEDA
D
C
ASIGNAR SUBPROGRAMAS 9.3
ALMACEN SUBPROGRAMAS
C
AUXILIAR SUBPROGRAMAS
ASIGNAR PROC. LLAMADOS 9.4
ALMACEN PROC. LLAMADO
D
C
AUXILIAR LLAMADO
ASIGNAR PROC. LLAMADOR 9.5
ALMACEN PROC. LLAMADOR
AUXILIAR LLAMADOR
ARCHIVO SALIDA

DIAGRAMA 13


CONTENIDO_ALMACEN
IMPRIMIR SUBPROGS PROGRAMA 13.1
IMPRIMIR MODULOS POR PROGRAMA 13.2
IMPRIMIR ALCANCE DE MODULOS 13.3
IMPRIMIR PROC. LLAMADOS 13.4
IMPRIMIR PROC. LLAMADORES 13.5
IMPRIMIR ESTRUCTURA MODULAR 13.6
IMPRIMIR PROC. CONTENIDOS 13.8
IMPRIMIR LISTA EDITADA 13.7
LISTA_MODULOS_PROGRAMA
LISTA_SUBPROGRAMAS_PROGRAMA
LISTA_ALCANCE_MODULOS
LISTA_MODULO_LLAMADOR_LLAMADO
LISTA_MODULO_LLAMADO_LLAMADOR
LISTA_ESTRUCTURA_MODULAR
LISTA_MODULOS_CONTENIDOS
LISTA EDITADA

AUTOMATA


< > C = CUALQUIER CARACTER
L = LETRA
D = DIGITO
_ = UNDERLINE
N = NUMERO
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
S34
S35
S36
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
Q9
Q10
Q11
Q12
Q13
Q14
Q15
Q16
LD_b
LD_
CR
LF
EOF

Diccionario de datos.

- ALMACEN_ARCHIVO_PROCESO = {ARCHIVO_LECTURA}
- ALMACEN_BUSQUEDA = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS *
- ALMACEN_MODULO_ACTUAL = {NOMBRE_MODULO + NIVEL_END + LINEA_FINAL + LINEA_INICIAL + SECCION_EJECUTABLE + NOMBRE_ARCHIVO}
- ALMACEN_NOMBRE_PROCESO = {NOMBRE_ARCHIVO + CONTADOR_LINEAS}
- ALMACEN_PROCEDIMIENTOS = {ULTIMO_LISTA_CONTENIDO + PRIMERO_LISTA_CONTENIDO + ULTIMO_LISTA_LLAMADOR + PRIMERO_LISTA_LLAMADOR + PRIMERO_LISTA_LLAMADO + ULTIMO_LISTA_LLAMADO + LISTA_ALCANCE_MODULOS + NUMERO_ELEMENTOS_CONTENIDO +NUMERO_ELEMENTOS_LLAMADOR + NUMERO_ELEMENTOS_LLAMADO + PROCESO +SIGUIENTE_LISTA_PROCEDIMIENTO}
- ALMACEN_PROCEDIMIENTOS_CONTENIDO = {NOMBRE_MODULO + SIGUIENTE_LISTA_CONTENIDO}
- ALMACEN_PROCEDIMIENTOS_LLAMADO = {NOMBRE_MODULO + SIGUIENTE_LISTA_LLAMADO + IDENTIFICADOR_RECURSIVO}
- ALMACEN_PROCEDIMIENTOS_LLAMADOR = NOMBRE_MODULO + SIGUIENTE_LISTA_LLAMADOR + IDENTIFICADOR_RECURSIVO
- ALMACEN_PROGRAMA_PRINCIPAL = PRIMERO_LISTA_SUBPROGRAMA + ULTIMO_LISTA_SUBPROGRAMA + NUMERO_ELEMENTOS_SUBPROGRAMA + NOMBRE_ARCHIVO
- ALMACEN_SALIDA_DOCUMENTADA = LISTA_EDITADA
- ALMACEN_SUBPROGRAMAS = {NOMBRE_ARCHIVO + TAMAÑO + FECHA_ACTUALIZACION + SIGUIENTE_LISTA_SUBPROGRAMA}
- APUNTADOR_BUFFER = NUMEROS
- APUNTADOR_INICIO_LISTA = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS*
- APUNTADOR_INICIO_SUBPROGRAMAS = * MISMO TIPO DE ALMACEN_PROGRAMA_PRINCIPAL*
- APUNTADOR_MOVIMIENTO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS *
- ARCHIVO_LECTURA = UBICACION + UNIDAD + DIRECTORIO
- ARCHIVO_PROCESO = { CARACTERES + [CR + LF/EOF]}
- ARCHIVO_SALIDA= {CARACTERES + [CR + LF-EOF]}
- AUXILIAR PROCEDIMIENTOS = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS *
- AUXILIAR_CONTENIDO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_CONTENIDO *
- AUXILIAR_LLAMADO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADO *
- AUXILIAR_LLAMADOR= * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADOR *
- AUXILIAR_PROGRAMA_PRINCIPAL = * MISMO TIPO DE ALMACEN_PROGRAMA_PRINCIPALAUXILIAR_SUBPROGRAMAS = * MISMO TIPO DE ALMACEN_SUBPROGRAMAS *
- BLANCOS = {b}
- BUFFER = {[LETRA/DIGITO/SIMBOLOS/CR/LF/EOF]}
- BUSCA_PROCEDIMIENTO = * MISMO TIPO_ALMACEN_PROCEDIMIENTOS *
- CARACTER = [LETRA/DIGITO/SIMBOLOS/CR/LF/EOF]
- CARACTERES = {[LETRA/DIGITO/SIMBOLOS]}
- COMPARA_CONTENIDO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_CONTENIDOS *
- COMPARA_LLAMADOR = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADOR *
- COMPARA_SUBPROGRAMA = * MISMO TIPO DE ALMACEN_SUBPROGRAMAS *
- CONTADOR_LINEAS = NUMEROS
- CONTENIDO_ALMACEN = {ALMACEN_PROCEDIMIENTOS_CONTENIDO / ALMACEN_PROCEDIMIENTOS / ALMACEN_PROCEDIMIENTOS_LLAMADO / ALMACEN_PROCEDIMIENTOS_LLAMADOR / ALMACEN_SUBPROGRAMAS / ARCHIVO_SALIDA}
- CR = * CARRIAGE RETURN *
- DIGITO = {0/1/2/3/4/5/6/7/8/9}
- DIRECTORIO = "\" + NOMBRE_IDENTIFICACION
- EOF = * END OF FILE *
- ESTADO_SIGUIENTE = [q0,q1,q2,.. qn]
- FECHA_ACTUALIZACION = DIA + MES + AÑO
- FINAL_BUFFER = NUMEROS
- GRAFICO = ["= "/" | | "/" =>"]
- IDENTIFICADOR_RECURSIVO = [*/b]
- INDENTACION = GRAFICOS + BLANCOS
- INDENTACION_INICIAL = GRAFICOS + BLANCOS

- LETRA = [A/B/C/..../Z/a/b/c/ ..../z]
- LF = * LINE FEED *
- LINEA = INDENTACION + {GRAFICO} + TOKEN
- LINEA_FINAL = NUMEROS
- LINEA_INICIAL = NUMEROS
- LISTA_ALCANCE_MODULOS = {LINEA_INICIAL + LINEA_FINAL + NOMBRE_ARCHIVO + SECCION_EJECUTABLE}
- LISTA_COMENTARIOS = {CARACTERES + CR + LF + NOMBRE_MODULO}
- LISTA_EDITADA = {GRAFICO + INDENTACION + ORACION + CR + LF}
- LISTA_ESTRUCTURA_MODULAR = {INDENTACION + NOMBRE_MODULO}
- LISTA_MODULO_LLAMADO_LLAMADOR = {NOMBRE_MODULO_LLAMADO + {NOMBRE_MODULO_LLAMADOR}}
- LISTA_MODULO_LLAMADOR_LLAMADO = {NOMBRE_MODULO_LLAMADOR + {NOMBRE_MODULO_LLAMADO}}
- LISTA_MODULOS_PROGRAMA = {TIPO + NOMBRE_PROGRAMA + PAGINA + NOMBRE_MODULO}
- LISTA_SUBPROGRAMAS_PROGRAMA = {NOMBRE_ARCHIVO + TAMAÑO + FECHA_ACTUALIZACION
- MOVIMIENTO_PROCEDIMIENTO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS *
- NIVEL_END = NUMEROS
- NOMBRE_ARCHIVO = NOMBRE_IDENTIFICACION
- NOMBRE_IDENTIFICACION = LETRA + {[LETRA/DIGITO/UNDERLINE]}
- NOMBRE_MODULO = NOMBRE_IDENTIFICACION
- NOMBRE_MODULO_BUSCAR = NOMBRE_IDENTIFICACION
- NOMBRE_MODULO_LLAMADO = NOMBRE_IDENTIFICACION
- NOMBRE_MODULO_LLAMADOR = NOMBRE_IDENTIFICACION
- NOMBRE_PROGRAMA = NOMBRE_IDENTIFICACION
- NOMBRE_REFERENCIA = [NOMBRE_MODULO/TOKEN]
- NOMBRE_REFERENCIA = [TOKEN/NOMBRE_MODULO]
- NOMBRE_RUTINA_PRINCIPAL = NOMBRE_IDENTIFICACION + " PAS"
- NOMBRE_SISTEMA = NOMBRE_IDENTIFICACION
- NUMERO_ELEMENTOS_CONTENIDO = NUMEROS
- NUMERO_ELEMENTOS_LLAMADO = NUMEROS
- NUMERO_ELEMENTOS_LLAMADOR = NUMEROS
- NUMERO_ELEMENTOS_SUBPROGRAMA = NUMEROS
- NUMEROS = {DIGITOS}
- ORACION = * UNA ESTRUCTURA DE PROGRAMACION VALIDA *
- PAGINA = NUMEROS
- PALABRAS_RESERVADAS = [ABSOLUTE/AND/ARRAY/BEGIN/CASE/CONST/DIV/DO/DOWN /ELSE/END/EXTERNAL/FILE/FOR/FORWARD/FUNCTION/GOTO/IF/IMPLEMENTATION/IN/INLINE/INTERFACE/INTERRUPT/LABEL/MODE/NIL/NOT/OF/OR/PACKED/PROCEDURE/PROGRAM/RECORD/REPEAT/SET/SHL/SHR/STRING/THEN/TO/TYPE/UNIT/UNTIL/USE/VAR/WHILE/WITH/XOR]
- PRIMERO_LISTA_CONTENIDO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_CONTENIDO *
- PRIMERO_LISTA_LLAMADO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADO *
- PRIMERO_LISTA_LLAMADOR = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADOR *
- PRIMERO_LISTA_SUBPROGRAMA = * MISMO TIPO DE ALMACEN_SUBPROGRAMAS *
- REFERENCIA = * MISMO TIPO DE APUNTADOR_INICIO_LISTA *
- REFERENCIA_LLAMADOR = * MISMO TIPO DE PROCEDIMIENTOS_LLAMADOR *
- REFERENCIA_PROCEDIMIENTO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS *
- SECCION_EJECUTABLE = NUMEROS
- SIGUIENTE_LISTA_CONTENIDO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_CONTENIDO *
- SIGUIENTE_LISTA_LLAMADO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADO *
- SIGUIENTE_LISTA_LLAMADOR = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADOR *
- SIGUIENTE_LISTA_PROCEDIMIENTO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS *
- SIGUIENTE_LISTA_SUBPROGRAMA = * MISMO TIPO DE ALMACEN_SUBPROGRAMAS *SIMBOLOS = *CUALQUIER CARACTER DIFERENTE DE LETRA Y DIGITO*
- TAMAÑO = NUMEROS
- TIPO = [PROCEDIMIENTO/FUNCION]
- TIPO_TOKEN = NOMBRE_IDENTIFICACION
- TOKEN = {LETRA/DIGITO/SIMBOLO/CR/LF/EOF}
- UBICACION = UNIDAD + ":" + DIRECTORIO + "\"
- ULTIMA_LISTA_LLAMADOR = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADOR *

- ULTIMO_LISTA_CONTENIDO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_CONTENIDO *
- ULTIMO_LISTA_LLAMADO = * MISMO TIPO DE ALMACEN_PROCEDIMIENTOS_LLAMADO *
- ULTIMO_LISTA_SUBPROGRAMA = * MISMO TIPO DE ALMACEN_SUBPROGRAMAS *
- UNDERLINE = '_'
- UNIDAD = (A/B/C/D/E/F)

Descripción de las salidas del Autómata.

Salida	Token
S1	Directiva include
S2	Blancos
S3	Número.
S4	[
S5	)
S6	^
S7	]
S8	EOF
S9	>
S10	> =
S11	<
S12	< >
S13	< =
S14	*
S15	-
S16	+
S17	..
S18	.
S19	;
S20	Error
S21	CRLF
S22	Comentario
S23	Error
S24	(
S25	: =
S26	=
S27	Comentario
S28	Error
S29	Identificador
S30	,
S31	String
S32	Directiva
S33	:
S34	"
S35	/

MINIESPECIFICACIONES

```
1.-IDENTIFICAR ARCHIVO
BEGIN
 IF INICIO_PROCESO
  CREATE (ALMACEN_PROGRAMA_PRINCIPAL)
  CREATE (APUNTADOR_INICIO_LISTA)
  NOMBRE_MODULO = ""
  CONTADOR_LINEAS = 1
  NIVEL_END = 0
  LINEA_INICIAL = 0
  LINEA_FINAL = 0
  SECCION EJECUTABLE = 0
  READ UBICACION,DESCRIPCION,NOMBRE_IDENTIFICACION
  WRITE UBICACION,DESCRIPCION,NOMBRE_IDENTIFICACION IN ARCHIVO_LECTURA
  READ ARCHIVO_LECTURA
  IF ARCHIVO_LECTURA NO EXISTE
   WRITE "EL ARCHIVO NO EXISTE ERROR"
  ELSE
   WRITE ARCHIVO_LECTURA IN ARCHIVO_PROCESO
  ENDIF
 ELSE
  READ ALMACEN_NOMBRE_PROCESO
  WRITE ALMACEN_NOMBRE_PROCESO IN ARCHIVO_PROCESO
 ENDIF
END
2.-OBTENER BUFFER
BEGIN
 READ BUFFER DE ARCHIVO_PROCESO
 COLOCAR APUNTADOR_BUFFER AL INICIO DEL BUFFER
END
3.1 OBTENER CARACTER
BEGIN
 IF APUNTADOR_BUFFER< >FINAL_BUFFER
  READ CARACTER OF BUFFER
 ELSE
  ORTENER BUFFER
  OBTENER CARACTER
 ENDIF
END


3.2 ARMAR TOKEN
BEGIN
 TOKEN = ""
 TIPO_TOKEN = ""
 OBTENER_CARACTER
 ESTADO_SIGUIENTE = q0
 DO WHILE NO SE TENGA UN TOKEN
  DO CASE
   CASE ESTADO_SIGUIENTE = q0
    DO CASE
     CASE CARACTER = "("
      ESTADO_SIGUIENTE = q2
      TOKEN = TOKEN + CARACTER
     CASE CARACTER IN LETRA
```

```
    ESTADO_SIGUIENTE = q1
    TOKEN = TOKEN + CARACTER
  CASE CARACTER = "CR"
    ESTADO_SIGUIENTE = q10
    TOKEN = TOKEN + CARACTER
  CASE CARACTER = ">"
    ESTADO_SIGUIENTE = q12
    TOKEN = TOKEN + CARACTER
  CASE CARACTER = "<"
    ESTADO_SIGUIENTE = q11
    TOKEN = TOKEN + CARACTER
  CASE CARACTER = "."
    ESTADO_SIGUIENTE = q13
    TOKEN = TOKEN + CARACTER
  CASE CARACTER = "("
    ESTADO_SIGUIENTE = q3
    TOKEN = TOKEN + CARACTER
  CASE CARACTER = ","
    ESTADO_SIGUIENTE = q14
    TOKEN = TOKEN + CARACTER
  CASE CARACTER = "b"
    ESTADO_SIGUIENTE = q15
    TOKEN = TOKEN + CARACTER
  CASE CARACTER = ";"
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_PUNTO_COMA
    ESTADO_SIGUIENTE = q0
  CASE CARACTER = "+"
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_ADICION
    ESTADO_SIGUIENTE = q0
  CASE CARACTER = "-"
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_RESTA
    ESTADO_SIGUIENTE = q0
  CASE CARACTER = "*"
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_PRODUCTO
    ESTADO_SIGUIENTE = q0
  CASE CARACTER = "/"
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_DIVISION
    ESTADO_SIGUIENTE = q0
  CASE CARACTER = "EOF"
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_EOF
    ESTADO_SIGUIENTE = q0
  CASE CARACTER = ")"
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_CIERRA_PC
    ESTADO_SIGUIENTE = q0
  CASE CARACTER = "="
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_IGUAL
  CASE CARACTER = "^"
    TOKEN = CARACTER
    TIPO_TOKEN = TOKEN_CIRC
    ESTADO_SIGUIENTE = q0
  CASE CARACTER = "]"
```

```
      TOKEN = CARACTER
      TIPO_TOKEN = TOKEN_CIERRA_PR
      ESTADO_SIGUIENTE = q0
    CASE CARACTER = "["
      TOKEN = CARACTER
      TIPO_TOKEN = TOKEN_ABRE_PC
    CASE CARACTER = "'"
      TOKEN = CARACTER
      ESTADO_SIGUIENTE = q17
    CASE CARACTER = "'"
      TOKEN = CARACTER
      TIPO_TOKEN = TOKEN_COMILLA
      ESTADO_SIGUIENTE = q0
    CASE CARACTER = "@"
      TOKEN = CARACTER
      TIPO_TOKEN = TOKEN_ARR
      ESTADO_SIGUIENTE = q0
    CASE CARACTER = ","
      TOKEN = CARACTER
      TIPO_TOKEN = TOKEN_COMA
    CASE CARACTER = '$'
      TOKEN = CARACTER
      TIPO_TOKEN = TOKEN_SIGNO_DOLAR
  ENDCASE
CASE ESTADO_SIGUIENTE = q1
  DO CASE
    CASE CARACTER IN [LETRA,DIGITO,UNDERLINE]
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q1
    OTHERWISE
      TIPO_TOKEN = TOKEN_IDENTIFICADOR
      DETERMINA SI SE TRATA DE UNA PALABRA RESERVADA
      REGRESA APUNTADOR_BUFFER EN 1
  ENDCASE
CASE ESTADO_SIGUIENTE = q2
  DO CASE
    CASE CARACTER = "}"
      TOKEN = TOKEN + CARACTER
      TIPO_TOKEN = TOKEN_COMENTARIO
      ESTADO_SIGUIENTE = q0
    CASE CARACTER = "$"
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q8
    OTHERWISE
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q2
  ENDCASE
CASE ESTADO_SIGUIENTE = q3
  DO CASE
    CASE CARACTER = "*"
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q4
    OTHERWISE
      TIPO_TOKEN = TOKEN_ABRE_PR
      REGRESA APUNTADOR_BUFFER EN 1
      ESTADO_SIGUIENTE = q0
  ENDCASE
CASE ESTADO_SIGUIENTE = q4
  DO CASE
```

```
    CASE CARACTER = "*"
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q5
    OTHERWISE
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q4
  ENDCASE
CASE ESTADO_SIGUIENTE = q5
  DO CASE
    CASE CARACTER = "}"
      TOKEN = TOKEN + CARACTER
      TIPO_TOKEN = TOKEN_COMENTARIO
      ESTADO_SIGUIENTE = q0
    CASE CARACTER = "*"
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q5
    OTHERWISE
      TOKEN = "ERROR"
      TIPO_TOKEN = TOKEN_ERROR
      ESTADO_SIGUIENTE = q0
  ENDCASE
CASE ESTADO_SIGUIENTE = q6
  DO CASE
    CASE CARACTER = "}"
      TOKEN = TOKEN + CARACTER
      TIPO_TOKEN = TOKEN_DIRECTIVA
      ESTADO_SIGUIENTE = q0
    OTHERWISE
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q6
  ENDCASE
CASE ESTADO_SIGUIENTE = q7
  DO CASE
    CASE CARACTER = "b"
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q9
    OTHERWISE
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q6
  ENDCASE
CASE ESTADO_SIGUIENTE = q8
  DO CASE
    CASE CARACTER = "l"
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q7
    CASE CARACTER IN LETRA
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q9
    OTHERWISE
      TOKEN = "ERROR"
      TIPO_TOKEN = TOKEN_ERROR
      ESTADO_SIGUIENTE = q0
  ENDCASE
CASE ESTADO_SIGUIENTE = q9
  DO CASE
    CASE CARACTER IN (LETRA,DIGITO,UNDERLINE,b)
      TOKEN = TOKEN + CARACTER
      ESTADO_SIGUIENTE = q9
    CASE CARACTER = " + "
```

```
        TOKEN = TOKEN + CARACTER
        ESTADO_SIGUIENTE = q6
      CASE CARACTER = "}"
        TOKEN = TOKEN + CARACTER
        TIPO_TOKEN = TOKEN_INCLUDE
        ESTADO_SIGUIENTE = q0
      OTHERWISE
        TOKEN = "ERROR"
        TIPO_TOKEN = TOKEN_ERROR
        ESTADO_SIGUIENTE = q0
    ENDCASE
  CASE ESTADO_SIGUIENTE = q10
    DO CASE
      CASE CARACTER = "LF"
        TOKEN = TOKEN + CARACTER
        TIPO_TOKEN = TOKEN_CRLF
        ESTADO_SIGUIENTE = q0
      OTHERWISE
        TOKEN = "ERROR"
        TIPO_TOKEN = TOKEN_ERROR
        ESTADO_SIGUIENTE = q0
    ENDCASE
  CASE ESTADO_SIGUIENTE = q11
    DO CASE
      CASE CARACTER = "="
        TOKEN = TOKEN + CARACTER
        TIPO_TOKEN = TOKEN_MENOR_IGUAL
        ESTADO_SIGUIENTE = q0
      CASE CARACTER = ">"
        TOKEN = TOKEN + CARACTER
        TIPO_TOKEN = TOKEN_DIFERENTE
      OTHERWISE
        TIPO_TOKEN = TOKEN_MENOR
        ESTADO_SIGUIENTE = q0
        REGRESA APUNTADOR_BUFFER EN 1
    ENDCASE
  CASE ESTADO_SIGUIENTE = q12
    DOCASE
      CASE CARACTER = "="
        TOKEN = TOKEN + CARACTER
        TIPO_TOKEN = TOKEN_MAYOR_IGUAL
        ESTADO_SIGUIENTE = q0
      OTHERWISE
        TIPO_TOKEN = TOKEN_MAYOR
        ESTADO_SIGUIENTE = q0
        REGRESA APUNTADOR_BUFFER EN 1
    ENDCASE
  CASE ESTADO_SIGUIENTE = q13
    DO CASE
      CASE CARACTER = "."
      TOKEN = TOKEN + CARACTER
      TIPO_TOKEN = TOKEN_PUNTO_PUNTO
      ESTADO_SIGUIENTE = q0
    OTHERWISE
      TOKEN = "ERROR"
      TIPO_TOKEN = TOKEN_PUNTO
      ESTADO_SIGUIENTE = q0
    ENDCASE
  CASE ESTADO_SIGUIENTE = q14
```

```
      DO CASE
        CASE CARACTER = "="
          TOKEN = TOKEN + CARACTER
          TIPO_TOKEN = TOKEN_DOS_PUNTO_IGUAL
          ESTADO_SIGUIENTE = q0
        OTHERWISE
          TIPO_TOKEN = TOKEN_DOS_PUNTOS
          ESTADO_SIGUIENTE = q0
          REGRESA APUNTADOR_ BUFFER EN 1
      ENDCASE
    CASE ESTADO_SIGUIENTE = q15
    *ELIMINAR LOS BLANCOS
      DO WHILE CARACTER = "b"
        DO OBTIENE_CARACTER
      ENDDO
      LIMPIAR TOKEN
      REGRESAR APUNTADOR DE BUFFER EN 1
      ESTADO_SIGUIENTE = q0
    CASE ESTADO_SIGUIENTE = q16
      DO CASE
        CASE CARACTER IN DIGITO
          TOKEN = TOKEN + CARACTER
          ESTADO_SIGUIENTE = q16
        OTHERWISE
          TIPO_TOKEN = TOKEN_NUMEROS
          ESTADO_SIGUIENTE = q0
      ENDCASE
    CASE ESTADO_SIGUIENTE = q17
      DO CASE
        CASE CARACTER = """
          TOKEN = TOKEN + CARACTER
          TIPO_TOKEN = TOKEN_STRING
          ESTADOR_SIGUIENTE = q0
        OTHERWISE
          TOKEN = TOKEN + CARACTER
          ESTADO_SIGUIENTE = q17
      ENDCASE
  ENDCASE
  IF TIPO_TOKEN ES NULO
    OBTENER CARACTER
  ENDIF
 ENDDO
 REGRESA TOKEN,TIPO_TOKEN
END
5- ANALIZAR ESTRUCTURA
BEGIN
  NIVEL_END = 0
  OBTENER TOKEN
  IF TIPO_TOKEN = * TOKEN DE PROGRAM
    OBTENER TOKEN * BUSCA EL IDENTIFICADOR DE PROGRAMA
    CREAR PROCEDIMIENTOS
    ASIGNAR_PROCEDIMIENTOS
    NOMBRE_MODULO = TOKEN
  ENDIF
  OBTENER_TOKEN
  NIVEL DE ANIDAMIENTO
END
```

```
NIVEL ANIDAMIENTO
BEGIN
 TERMINA_ANIDAMIENTO = .F.
 DO WHILE NO TERMINA ANIDAMIENTO
  OBTENER_TOKEN LLAMA EL SIGUIENTE TOKEN DISPONIBLE
  DO CASE
   CASE TIPO_TOKEN IN [TOKEN_BEGIN, TOKEN_RECORD, TOKEN_CASE
   NIVEL_END = NIVEL_END + 1
   IF TIPO_TOKEN = * TOKEN BEGIN
    SECCION EJECUTABLE = CONTADOR_LINEAS
   ENDIF
   CASE TIPO_TOKEN = TOKEN_END
    NIVEL_END = NIVEL_END - 1 * CUANDO SE ENCUENTRA UN TOKEN END DISMINUYE EL NIVEL
    IF NIVEL_END = 0
     TERMINA_ANIDAMIENTO = .T.
    ENDIF
   CASE TIPO_TOKEN IN [ TOKEN_PROCEDURE , TOKEN_FUNCTION ]
   * GUARDAR EL MODULO DE PROCESO ASI COMO EL NIVEL DE ANIDAMIENTO
    WRITE NOMBRE_MODULO IN ALMACEN_MODULO_ACTUAL
    WRITE NIVEL_END IN ALMACEN_MODULO_ACTUAL
    WRITE LINEA_INICIAL ON ALMACEN_MODULO_ACTUAL
    WRITE LINEA_FINAL ON ALMACEN_MODULO_ACTUAL
    WRITE SECCION_EJECUTAR ON ALMACEN_MODULO_ACTUAL
    WRITE NOMBRE_ARCHIVO IN ALMACEN_MODULO_ACTUAL
    * OBTENER EL SIGUIENTE IDENTIFICADOR DE MODULO
    OBTENER TOKEN
    CREAR_PROCEDIMIENTO_CONTENIDO
    ASIGNAR_PROCEDIMIENTO_CONTENIDO
    NUMER_ELEMENTOS_CONTENIDO = NUMERO_ELEMENTOS_CONTENIDO + 1
    CREAR_PROCEDIMIENTO
    ASIGNAR_PROCEDIMIENTO
    NOMBRE_MODULO = TOKEN
    NIVEL_END = 0
    NIVEL ANIDAMIENTO
   CASE TIPO_TOKEN = * TOKEN DE IDENTIFICADOR
    BUSCA_MODULO_REFERENCIA
    IF ENCUENTRA IDENTIFICADOR EN ALMACEN_PROCEDIMIENTOS
     CREAR PROCEDIMIENTO LLAMADO
     ASIGNAR PROCEDIMIENTO LLAMADO
     NUMERO_ELEMENTOS_LLAMADO = NUMERO_ELEMENTOS_LLAMADO + 1
     CREAR PROCEDIMIENTO LLAMADOR
     ASIGNAR PROCEDIMIENTO LLAMADOR
     NUMERO_ELEMENTOS_LLAMADOR = NUMERO_ELEMENTOS_LLAMADOR + 1
    ENDIF
   CASE TIPO_TOKEN = * TOKEN DE CRLF
    OBTENER TOKEN
    CONTADOR_LINEAS = CONTADOR_LINEAS + 1 * INCREMENTA LINEAS
   CASE TIPO_TOKEN = * IDENTIFICADOR DE INCLUSION
    NOMBRE_IDENTIFICACION = PARTE (TOKEN)
    WRITE NOMBRE_IDENTIFICACION IN ALMACEN_NOMBRE_PROCESO
    WRITE CONTADOR_LINEAS IN ALMACEN_NOMBRE PROCESO
    CONTADOR_LINEAS = 0
    NOMBRE_ARCHIVO = NOMBRE_IDENTIFICACION
    LINEA_INICIAL = 0
    LINEA_FINAL = 0
    SECCION_EJECUTABLE = 0
    IDENTIFICARDE ARCHIVO
   OTHERWISE
    * SE MANTIENE EL MISMO NIVEL DE ANIDAMIENTO
```

```
  ENDCASE
 ENDDO
 BUSCA MODULO REFERENCIA WITH ALMACEN_BUSQUEDA,NOMBRE_MODULO
 LINEA_INICIAL OF ALMACEN_MODULO_BUSQUEDA = LINEA_INICIAL
 LINEA_FINAL OF ALMACEN_MODULO_BUSQUEDA = LINEA_FINAL
 SECCION EJECUTABLE OF ALMACEN_MODULO_BUSQUEDA = SECCION_EJECUTABLE
 NOMBRE_ARCHIVO OF ALMACEN_MODULO_BUSQUEDA = NOMBRE_ARCHIVO
END
```

6.1 ANALIZAR IF

```
BEGIN
 INDENTACION = INDENTACION_INICIAL + "  " + "=>" + "IF"
 WRITE INDENTACION IN ARCHIVO_SALIDA
 OBTENER TOKEN
 DO WHILE TIPO_TOKEN < > TOKEN_THEN AND TIPO_TOKEN< > TOKEN_CRLF
  IF TIPO_TOKEN NOT IN [TOKEN_CRLF
   WRITE TOKEN IN ARCHIVO_SALIDA
  ENDIF
  OBTENER TOKEN
 ENDDO
 WRITELN
 IDENTACION = INDENTACION_INICIAL + "  " + "=>" + "THEN"
 WRITELN INDENTACION IN ARCHIVO_SALIDA
 OBTENER TOKEN* LEE EL SIGUIENTE VALOR
 DETERMINAR ESTRUCTURA WITH INDENTACION_INICIAL + "    " + "b" + "b" + "b"
 IF TIPO_TOKEN = * TOKEN ELSE
  INDENTACION = INDENTACION_INICIAL + "  " + "=>" + "ELSE"
  WRITELN INDENTACION IN ARCHIVO_SALIDA
  OBTENER_TOKEN
  DETERMINAR ESTRUCTURA WITH INDENTACION_INICIAL + "    " + "b" + "b" + "b"
  WRITELN
 ENDIF
 INDENTACION = INDENTACION_INICIAL + "  " + "=>" + "END"
 WRITELN INDENTACION IN ARCHIVO_SALIDA
END
```

6.2 ANALIZAR CASE

```
BEGIN
 INDENTACION = INDENTACION_INICIAL + "  " + "=>" + "CASE"
 WRITE INDENTACION IN ARCHIVO_SALIDA
 DO WHILE TIPO_TOKEN < > TOKEN_OF
  WRITE TOKEN IN ARCHIVO_SALIDA
  OBTENER_TOKEN
 ENDDO
 WRITELN TOKEN IN ARCHIVO_SALIDA * TOKEN OF
 OBTENER_TOKEN
 ANLIZAR_ESTRUCTURA_CASE WITH INDENTACION_INICIAL
 IF TIPO_TOKEN = TOKEN_ELSE
  INDENTACION = INDENTACION_INICIAL + "  " + "=>" + "ELSE"
  WRITELN INDENTACION IN ARCHIVO_SALIDA
  OBTENER TOKEN
  ANALIZAR_ESTRUCTURA_CASE WITH INDENTACION_INICIAL
  WRITELN
 ENDIF
 INDENTACION = INDENTACION_INICIAL + "  " + "=>" + "END"
 WRITELN INDENTACION IN ARCHIVO_SALIDA
END
```

ANALIZAR ESTRUCTURA_CASE

```
BEGIN
```

```
  WRITE INDENTACION_INICIAL + "|_ " + " = " + " => " + TOKEN IN ARCHIVO_SALIDA
  DO WHILE TIPO_TOKEN <> TOKEN_DOS_PUNTOS
    WRITE TOKEN IN ARCHIVO_SALIDA
    OBTENER_TOKEN
  ENDDO
  WRITELN TOKEN IN ARCHIVO_SALIDA + TOKEN
  OBTENER_TOKEN
  DO WHILE TIPO_TOKEN <> TOKEN_END
    DETERMINAR ESTRUCTURA WITH INDENTACION_INICIAL + " | " + "b" + "b" + "b" + "b"
END
6.3 ANALIZAR FOR
BEGIN
  INDENTACION = INDENTACION_INICIAL + "[- " + " = " + " => " + "FOR"
  WRITE INDENTACION IN ARCHIVO_SALIDA
  DO WHILE TIPO_TOKEN <> TOKEN DE DO
    WRITE TOKEN IN ARCHIVO_SALIDA
    OBTENER TOKEN
  ENDDO
  WRITELN TOKEN IN ARCHIVO_SALIDA
  OBTENER TOKEN
  DETERMINAR ESTRUCTURA WITH INDENTACION_INICIAL + " | " + "bbbb"
  OBTENER TOKEN
  INDENTACION = INDENTACION_INICIAL + "|_ " + " = " + "=>" + "END"
  WRITELN INDENTACION IN ARCHIVO_SALIDA
END

6.4 ANALIZAR BEGIN
BEGIN
  INDENTACION = INDENTACION_INICIAL + "[- " + " => " + "BEGIN" + TOKEN BEGIN
  WRITELN TOKEN IN ARCHIVO_SALIDA
  OBTENER_TOKEN
  DO WHILE TIPO_TOKEN <> TOKEN_END
    DETERMINAR ESTRUCTURA WITH INDENTACION_INICIAL + " | " + "b" + "b" + "b"
  INDENTACION = INDENTACION_INICIAL + "|_ " + " => " + "END" + TOKEN END
  WRITELN INDENTACION IN ARCHIVO_SALIDA
END
6.5 ANALIZAR WHILE
BEGIN
  INDENTACION = INDENTACION_INICIAL + "[- " + " = " + " => " + "WHILE"
  WRITE INDENTACION IN ARCHIVO_SALIDA
  DO WHILE TIPO_TOKEN <> TOKEN_DO
    WRITE TOKEN IN ARCHIVO_SALIDA
    OBTENER TOKEN
  ENDDO
  WRITELN TOKEN IN ARCHIVO_SALIDA + TOKEN DO
  OBTENER TOKEN
  DETERMINAR ESTRUCTURA WITH INDENTACION_INICIAL + "bbbb" + " | "
  INDENTACION = INDENTACION_INICIAL + "|_ " + " = " + " => " + "END"
  WRITELN INDENTACION IN ARCHIVO_SALIDA
END
6.6 ANALIZAR REPEAT
BEGIN
  INDENTACION = INDENTACION_INICIAL + " [- " + " => " + "REPEAT"
  WRITELN INDENTACION IN ARCHIVO_SALIDA
  OBTENER_TOKEN
  DO WHILE TIPO_TOKEN <> TOKEN_UNTIL
    DETERMINAR ESTRUCTURA WITH INDENTACION_INICIAL + " | " + "b" + "b" + "b"
  INDENTACION = INDENTACION_INICIAL + "|_ " + " => " + "UNTIL"
END
```

```
6.7 ANALIZAR WITH
BEGIN
 INDENTACION = INDENTACION_INICIAL + "<" + "=" + "=>" + "WITH"
 WRITE INDENTACION IN ARCHIVO_SALIDA
 DO WHILE TIPO_TOKEN < > TOKEN_DO
  WRITE TOKEN IN ARCHIVO_SALIDA
  OBTENER TOKEN
  ENDDO
 WRITELN TOKEN IN ARCHIVO_SALIDA
 OBTENER_TOKEN
 DETERMINAR ESTRUCTURA WITH INDENTACION_INICIAL + "bbbbbb"
 INDENTACION = INDENTACION_INICIAL + "<" + "=" + "=>" + "END"
 WRITELN INDENTACION IN ARCHIVO_SALIDA
END
6.8 ANALIZAR RECORD
BEGIN
 INDENTACION = INDENTACION_INICIAL + "<" + "=>" + "RECORD" "TOKEN RECORD
 WRITELN INDENTACION IN ARCHIVO_SALIDA
 OBTENER TOKEN
 DO WHILE TIPO_TOKEN < > TOKEN_CRLF
  WRITE TOKEN IN ARCHIVO_SALIDA
  OBTENER TOKEN
 ENDDO
 WRITELN
 OBTENER TOKEN
 INDENTACION = INDENTACION_INICIAL + "  " + "bbb"
 DO WHILE TIPO_TOKEN < > TOKEN_END
  IF TIPO_TOKEN = TOKEN_CASE
   ANALIZAR CASE WITH INDENTACION
  ELSE
   DETERMINAR ESTRUCTURA WITH INDENTACION
  ENDIF
  OBTENER TOKEN
 ENDDO
 INDENTACION = INDENTACION_INICIAL + "<" + "=" + ">" + "END"
 WRITELN INDENTACION IN ARCHIVO_SALIDA
END

LLAMA ESTRUCTURA
BEGIN
 INDENTACION = ""
 DO WHILE TIPO_TOKEN < > TOKEN_EOF
  DETERMINAR ESTRUCTURA WITH INDENTACION
 ENDDO
END
6.9 DETERMINAR ESTRUCTURA
BEGIN
 DO CASE
  CASE TIPO_TOKEN = TOKEN_COMENTARIO
   WRITELN(INDENTACION_INICIAL + TOKEN)
   OBTENER TOKEN
   DO WHILE TIPO_TOKEN < > TOKEN_CRLF AND TIPO_TOKEN < > TOKEN_EOF
    WRITE INDENTACION_INICIAL + TOKEN IN ARCHIVO_SALIDA
    OBTENER TOKEN
   ENDDO
   WRITELN
  CASE TIPO_TOKEN = TOKEN_CRLF
   WRITELN(INDENTACION_INICIAL)
```

```
  CASE TIPO_TOKEN = TOKEN_IF
    ANALIZAR_IF WITH INDENTACION_INICIAL
  CASE TIPO_TOKEN = TOKEN_CASE
    ANALIZAR CASE WITH INDENTACION_INICIAL
  CASE TIPO_TOKEN = TOKEN_FOR
    ANALIZAR FOR WITH INDENTACION_INICIAL
  CASE TIPO_TOKEN = TOKEN_REPEAT
    ANALIZAR REPEAT WITH INDENTACION_INICIAL
  CASE TIPO_TOKEN = TOKEN_BEGIN
    ANALIZAR BEGIN WITH INDENTACION_INICIAL
  CASE TIPO_TOKEN = TOKEN_WITH
    ANALIZAR WITH WITH INDENTACION_INICIAL
  CASE TIPO_TOKEN IN TOKEN_VAR, TOKEN_CONST, TOKEN_TYPE, TOKEN_UNIT,
        TOKEN_USES, TOKEN_PROGRAM
    WRITE TOKEN
    OBTENER TOKEN
    DO WHILE TIPO_TOKEN< >TOKEN_CRLF
      WRITE TOKEN IN ARCHIVO_SALIDA
      OBTENER TOKEN
    ENDDO
    WRITELN
  CASE TIPO_TOKEN IN TOKEN_PROCEDURE, TOKEN_FUNCTION
    OBTENER TOKEN
    DO WHILE NO TERMINE INSTRUCCION
      WRITE TOKEN IN ARCHIVO_SALIDA
      OBTENER_TOKEN
    ENDDO
  OTHERWISE
    TERMINA_PROCESO = FALSE
    DO WHILE NOT TERMINA_PROCESO
      OBTENER TOKEN
      WRITE INDENTACION_INICIAL IN ARCHIVO_SALIDA
      DO WHILE TIPO_TOKEN< >TOKEN_CRLF AND TIPO_TOKEN< >TOKEN_RECORD AND
          TIPO_TOKEN< >TOKEN_PUNTO_COMA
        WRITE TOKEN IN ARCHIVO_SALIDA
        OBTENER_TOKEN
      ENDDO
      IF TIPO_TOKEN IN TOKEN_END, TOKEN_CRLF, TOKEN_UNTIL, TOKEN_ELSE, TOKEN_COMENTARIO
          TOKEN_PUNTO_COMA
        TERMINA_PROCESO = TRUE
      ELSE
        INDENTACION = INDENTACION_INICIAL + BLANCOS
        ANALIZAR RECORD WITH INDENTACION
        TERMINA_PROCESO = TRUE
      ENDIF
    ENDDO
    OBTENER TOKEN
  ENDCASE
END


7.- CREAR PROCEDIMIENTOS LLAMADORES
BEGIN
  CREATE (AUXILIAR_LLAMADOR)
  NOMBRE_MODULO OF AUXILIAR_LLAMADOR = NOMBRE_MODULO
  SIGUIENTE_LISTA_LLAMADOR OF AUXILIAR_LLAMADORES = NIL
  IF NOMBRE_MODULO = TOKEN
    IDENTIFICADOR_RECURSIVO OF AUXILIAR_LLAMADOR = "*"
  ELSE
```

```
  IDENTIFICADOR_RECURSIVO OF AUXILIAR_LLAMADOR = ""
 ENDIF
END

8.- CREAR PROCEDIMIENTOS LLAMADOS
BEGIN
 CREATE (AUXILIAR_LLAMADO)
 NOMBRE_MODULO OF AUXILIAR_LLAMADO = TOKEN * NOMBRE DE MODULO
 SIGUIENTE_LISTA_LLAMADO OF AUXILIAR_LLAMADO = NIL
 IF NOMBRE_MODULO = TOKEN
  IDENTIFICADOR_RECURSIVO OF AUXILIAR_LLAMADO = "*"
 ELSE
  IDENTIFICADOR_RECURSIVO OF AUXILIAR_LLAMADO = ""
 ENDIF
END
9.1 ASIGNAR PROCEDIMIENTOS CONTENIDOS
BEGIN
 IF NOMBRE_MODULO OF ALMACEN_BUSQUEDA <> NOMBRE_MODULO
  BUSCA MODULO REFERENCIA WITH ALMACEN_BUSQUEDA.NOMBRE_MODULO
 ENDIF
 IF PRIMERO_LISTA_CONTENIDO OF ALMACEN_BUSQUEDA = NIL
  PRIMERO_LISTA_CONTENIDO OF ALMACEN_BUSQUEDA = AUXILIAR_CONTENIDO
  SIGUIENTE_LISTA_CONTENIDO OF AUXILIAR_CONTENIDO = NIL
  ULTIMO_LISTA_CONTENIDO OF ALMACEN_BUSQUEDA = AUXILIAR_CONTENIDO
 ELSE
  NO SE HA ENCONTRADO LUGAR PARA EL MODULO
 COMPARA_CONTENIDO = PRIMERO_LISTA_CONTENIDO OF ALMACEN_BUSQUEDA
 DO WHILE NO SE HAYA ENCONTRADO LUGAR AND SIGUIENTE_LISTA_CONTENIDO OF
          COMPARA_CONTENIDO <> NIL
  IF NOMBRE_MODULO OF AUXILIAR_CONTENIDO <= NOMBRE_MODULO OF COMPARA_CONTENIDO AND
NOMBRE_MODULO OF AUXILIAR_CONTENIDO <> NOMBRE_MODULO OF SIGUIENTE_LISTA_CONTENIDO OF
COMPARA_CONTENIDO
   SIGUIENTE_LISTA_CONTENIDO OF AUXILIAR_CONTENIDO = SIGUIENTE_LISTA_CONTENIDO OF
    COMPARA_CONTENIDO
   SIGUIENTE_LISTA_CONTENIDO OF COMPARA_CONTENIDO = AUXILIAR_CONTENIDO
   SE ENCONTRO LUGAR PARA EL MODULO
   ULTIMO_LISTA_CONTENIDO OF ALMACEN_BUSQUEDA = AUXILIAR_CONTENIDO
  ENDIF
  COMPARA_CONTENIDO = SIGUIENTE_LISTA_CONTENIDO OF COMPARA_CONTENIDO
 ENDDO
 IF SIGUIENTE_LISTA_CONTENIDO OF COMPARA_CONTENIDO = NIL
  SIGUIENTE_LISTA_CONTENIDO OF COMPARA_CONTENIDO = AUXILIAR_CONTENIDO
  SIGUIENTE_LISTA_CONTENIDO OF AUXILIAR_CONTENIDO = NIL
  ULTIMO_LISTA_CONTENIDO OF ALMACEN_BUSQUEDA = AUXILIAR_CONTENIDO
 ENDIF
 NUMERO_ELEMENTOS_CONTENIDOS OF ALMACEN_BUSQUEDA = NUMERO_ELEMENTOS_CONTENIDOS
       OF ALMACEN_BUSQUEDA + 1
END

9.2 ASIGNAR PROCEDIMIENTOS
 IF APUNTADOR_INICIO_LISTA = NIL
  APUNTADOR_INICIO_LISTA = AUXILIAR_PROCEDIMIENTO
 ELSE
  ORDENA
 ENDIF
 ALMACEN_BUSQUEDA = AUXILIAR PROCEDIMIENTOS
```

```
ORDENA
BEGIN
 APUNTADOR_MOVIMIENTO = APUNTADOR_INICIO_LISTA
 IF NOMBRE_MODULO OF AUXILIAR_PROCEDIMIENTOS  = NOMBRE_MODULO OF
    APUNTADOR_MOVIMIENTO
    SIGUIENTE_LISTA_PROCEDIMIENTO OF AUXILIAR_PROCEDIMIENTOS = APUNTADOR_MOVIMIENTO
    APUNTADOR_INICIO_LISTA_PROCEDIMIENTOS = AUXILIAR_PROCEDIMIENTOS
 ELSE
   NO SE HA ENCONTRADO LUGAR PARA EL MODULO
   DO WHILE NO SE HA ENCONTRADO LUGAR PARA EL MODULO AND
            SIGUIENTE_LISTA_PROCEDIMIENTO OF APUNTADOR_MOVIMIENTO  < > NIL
   IF NOMBRE_MODULO OF AUXILIAR_PROCEDIMIENTOS > = NOMBRE_MODULO OF
      APUNTADOR_MOVIMIENTO  AND NOMBRE_MODULO OF AUXILIAR_PROCEDIMIENTOS  < =
      NOMBRE_MODULO OF SIGUIENTE_LISTA_PROCEDIMIENTO OF APUNTADOR_MOVIMIENTO
    SIGUIENTE_LISTA_PROCEDIMIENTO OF AUXILIAR = SIGUIENTE_LISTA_PROCEDIMIENTO OF MOVIMIENTO
    SIGUIENTE_LISTA_PROCEDIMIENTO OF MOVIMIENTO = AUXILIAR_PROCEDIMIENTOS
    SE ENCONTRO LUGAR PARA EL MODULO
   ENDIF
   MOVIMIENTO = SIGUIENTE_LISTA_PROCEDIMIENTO OF APUNTADOR_MOVIMIENTO
   ENDDO

   IF SIGUIENTE_LISTA_PROCEDIMIENTO OF APUNTADOR_MOVIMIENTO = NIL
     SIGUIENTE_LISTA_PROCEDIMIENTO OF APUNTADOR_MOVIMIENTO = AUXILIAR_PROCEDIMIENTOS
     SIGUIENTE_LISTA_PROCEDIMIENTO OF AUXILAR_PROCEDIMIENTOS = NIL
   ENDIF
 ENDIF
END
9.3 ASIGNAR SUBPROGRAMAS
BEGIN
 IF PRIMERO_LISTA_SUBPROGRAMA OF ALMACEN_PROGRAMA_PRINCIPAL = NIL
    PRIMERO_LISTA_SUBPROGRAMA OF ALMACEN_PROGRAMA_PRINCIPAL =
    AUXILIAR_SUBPROGRAMAS
   SIGUIENTE_LISTA_SUBPROGRAMA OF AUXILIAR_SUBPROGRAMAS = NIL
   ULTIMO_LISTA_SUBPROGRAMA OF ALMACEN_PROGRAMA_PRINCIPAL = AUXILIAR_SUBPROGRAMAS
 ELSE
   NO SE HA ENCONTRADO LUGAR PARA EL MODULO
   COMPARA_SUBPROGRAMA = PRIMERO_LISTA_SUBPROGRAMA OF ALMACEN_PROGRAMA_PRINCIPAL
   DO WHILE NO SE HA ENCONTRADO LUGAR AND SIGUIENTE_LISTA_SUBPROGRAMA OF
            COMPARA_SUBPROGRAMA  < > NIL
    IF NOMBRE_MODULO OF AUXILIAR_SUBPROGRAMAS > = NOMBRE_MODULO OF COMPARA_SUBPROGRAMA
       AND NOMBRE_MODULO OF AUXILIAR_SUBPROGRAMAS  < = NOMBRE_MODULO OF
       SIGUIENTE_LISTA_SUBPROGRAMA OF COMPARA_SUBPROGRAMA
     SIGUIENTE_LISTA_SUBPROGRAMA OF AUXILIAR_SUBPROGRAMAS = SIGUIENTE_LISTA_SUBPROGRAMA
     OF COMPARA_SUBPROGRAMA
     SIGUIENTE_LISTA_SUBPROGRAMA OF COMPARA_SUBPROGRAMA = AUXILIAR_SUBPROGRAMAS
     SE ENCONTRO LUGAR PARA EL MODULO
     ULTIMO_LISTA_SUBPROGRAMA OF ALMACEN_PROGRAMA_PRINCIPAL = AUXILIAR_SUBPROGRAMAS
    ENDIF
    COMPARA_SUBPROGRAMA = SIGUIENTE_LISTA_SUBPROGRAMA OF COMPARA_SUBPROGRAMA
   ENDDO
   IF SIGUIENTE_LISTA_SUBPROGRAMA OF COMPARA_SUBPROGRAMA = NIL
     SIGUIENTE_LISTA_SUBPROGRAMA OF COMPARA_SUBPROGRAMA = AUXILIAR_SUBPROGRAMAS
     SIGUIENTE_LISTA_SUBPROGRAMA OF AUXILIAR_SUBPROGRAMAS = NIL
     ULTIMO_LISTA_SUBPROGRAMA OF ALMACEN_PROGRAMA_PRINCIPAL = AUXILIAR_SUBPROGRAMAS
   ENDIF
   NUMERO_ELEMENTOS_SUBPROGRAMAS OF ALMACEN_PROGRAMA_PRINCIPAL =
   NUMERO_ELEMENTOS_SUBPROGRAMAS OF ALMACEN_PROGRAMA_PRINCIPAL + 1
END
```

```
9.4 ASIGNAR PROCEDIMIENTOS LLAMADOS
BEGIN
 IF NOMBRE_MODULO OF ALMACEN_BUSQUEDA < > NOMBRE_MODULO
  DO BUSCA_MODULO REFERENCIA WITH ALMACEN_BUSQUEDA.NOMBRE_MODULO
 ENDIF
 IF PRIMERO_LISTA_LLAMADO OF ALMACEN_BUSQUEDA = NIL
  PRIMERO_LISTA_LLAMADO OF ALMACEN_BUSQUEDA := AUXILIAR_LLAMADO
  SIGUIENTE_LISTA_LLAMADO OF AUXILIAR_LLAMADO = NIL
  ULTIMO_LISTA_LLAMADO OF ALMACEN_BUSQUEDA = AUXILIAR_LLAMADO
 ELSE
  SIGUIENTE_LISTA_LLAMADO OF ULTIMO_LISTA_LLAMADO OF ALMACEN_BUSQUEDA := AUXILIAR_LLAMADO
  ULTIMO_LISTA_LLAMADO OF ALMACEN_BUSQUEDA := AUXILIAR_LLAMADO
  SIGUIENTE_LISTA_LLAMADO OF AUXILIAR_LLAMADO := NIL
 ENDIF
 NUMERO_ELEMENTOS_LLAMADO OF ALMACEN_BUSQUEDA := NUMERO_ELEMENTOS_LLAMADO OF
 ALMACEN_BUSQUEDA + 1
END
9.5 ASIGNAR PROCEDIMIENTOS LLAMADORES
BEGIN
 IF NOMBRE_MODULO OF ALMACEN_BUSQUEDA.TOKEN
  DO BUSCA_MODULO_REFERENCIA WITH ALMACEN_BUSQUEDA.TOKEN
 ENDIF
 IF PRIMERO_LISTA_LLAMADOR OF ALMACEN_BUSQUEDA = NIL
  PRIMERO_LISTA_LLAMADOR OF ALMACEN_BUSQUEDA := AUXILIAR_LLAMADOR
  SIGUIENTE_LISTA_LLAMADOR OF AUXILIAR_LLAMADOR := NIL
  ULTIMO_LISTA_LLAMADOR OF ALMACEN_BUSQUEDA_LLAMADOR := AUXILIAR_LLAMADOR
 ELSE
 COMPARA_LLAMADOR := PRIMERO_LISTA_LLAMADOR OF ALMACEN_BUSQUEDA
 NO SE HA ENCONTRADO LUGAR
 DO WHILE NO SE HA ENCONTRADO LUGAR AND SIGUIENTE_LISTA_LLAMADOR OF
          COMPARA_LLAMADOR < > NIL
  IF NOMBRE_MODULO OF AUXILIAR_LLAMADOR > = NOMBRE_MODULO OF COMPARA_LLAMADOR AND
     NOMBRE_MODULO OF AUXILIAR_LLAMADOR < = NOMBRE_MODULO OF SIGUIENTE_LISTA_LLAMADOR
     OF COMPARA_LLAMADOR
   SIGUIENTE_LISTA_LLAMADOR OF AUXILIAR_LLAMADOR := SIGUIENTE_LISTA_LLAMADOR
   OF COMPARA_LLAMADOR
   SIGUIENTE_LISTA_LLAMADOR OF COMPARA_LLAMADOR := AUXILIAR_LLAMADOR
   ULTIMO_LISTA_LLAMADOR OF ALMACEN_BUSQUEDA := AUXILIAR_LLAMADOR
  ENDIF
  COMPARA_LLAMADOR := SIGUIENTE_LISTA_LLAMADOR OF COMPARA_LLAMADOR
 ENDDO
 IF SIGUIENTE_LISTA_LLAMADOR OF COMPARA_LLAMADOR = NIL
  SIGUIENTE_LISTA_LLAMADOR OF COMPARA_LLAMADOR := AUXILIAR_LLAMADOR
  SIGUIENTE_LISTA_LLAMADOR OF AUXILIAR_LLAMADOR := NIL
  ULTIMO_LISTA_LLAMADOR OF ALMACEN_BUSQUEDA := AUXILIAR_LLAMADOR
 ENDIF
 NUMERO_ELEMENTOS_LLAMADOR OF ALMACEN_BUSQUEDA := NUMERO_ELEMENTOS_LLAMADOR
     OF ALMACEN_BUSQUEDA + 1
END
9.6 BUSCA_MODULO_REFERENCIA
BEGIN
 REFERENCIA := APUNTADOR_INICIO_LISTA
 DO WHILE NOMBRE_MODULO < > NOMBRE_REFERENCIA * TOKEN O NOMBRE_MODULO
  REFERENCIA := SIGUIENTE_LISTA_PROCEDIMIENTOS OF REFERENCIA
 ENDDO
 REGRESA REFERENCIA
END
```

```
10.- CREAR PROCEDIMIENTOS CONTENIDOS
BEGIN
  CREATE (AUXILIAR_CONTENIDO)
  NOMBRE_MODULO OF AUXILIAR_CONTENIDO = TOKEN * NOMBRE DE MODULO
  SIGUIENTE_LISTA_CONTENIDO OF AUXILIAR_CONTENIDO = NIL
END
11.- CREAR PROCEDIMIENTOS
BEGIN
  CREATE (AUXILIAR_PROCEDIMIENTOS)
  NOMBRE_MODULO OF AUXILIAR_PROCEDIMIENTOS = TOKEN *NOMBRE DE MODULO
  PRIMERO_LISTA_CONTENIDO OF AUXILIAR PROCEDIMIENTOS = NIL
  ULTIMO_LISTA_CONTENIDO OF AUXILIAR PROCEDIMIENTOS = NIL
  PRIMERO_LISTA_LLAMADOR OF AUXILIAR PROCEDIMIENTOS = NIL
  ULTIMO_LISTA_LLAMADOR OF AUXILIAR PROCEDIMIENTOS = NIL
  PRIMERO_LISTA_LLAMADO OF AUXILIAR PROCEDIMIENTOS = NIL
  ULTIMO_LISTA_LLAMADO OF AUXILIAR PROCEDIMIENTOS = NIL
  LINEA_INICIAL = 0
  LINEA_FINAL = 0
  SECCION_EJECUTABLE = 0
  NOMBRE_ARCHIVO = 0
  NUMERO_ELEMENTOS_CONTENIDO OF AUXILIAR PROCEDIMIENTOS = 0
  NUMERO_ELEMENTOS_LLAMADOR OF AUXILIAR PROCEDIMIENTOS = 0
  NUMERO_ELEMENTOS_LLAMADO OF AUXILIAR PROCEDIMIENTO = 0
  SIGUIENTE_LISTA_PROCEDIMIENTO = NIL
END
12.- CREAR SUBPROGRAMAS
BEGIN
  CREATE (AUXILIAR_SUBPROGRAMAS)
  NOMBRE_ARCHIVO OF AUXILIAR_SUBPROGRAMAS = TOKEN * NOMBRE DE MODULO
  TAMAÑO OF AUXILIAR_SUBPROGRAMAS = TAMAÑO EN DOS
  FECHA_ACTUALIZACION OF AUXILIAR_SUBPROGRAMAS = FECHA EN DOS
END
13.1 IMPRIMIR SUBPROGRAMAS POR PROGRAMA
BEGIN
  AUXILIAR_SUBPROGRAMAS = PRIMERO_LISTA_SUBPROGRAMA OF ALMACEN_PROGRAMA_PRINCIPAL
  WRITE NOMBRE_ARCHIVO OF ALMACEN_PROGRAMA_PRINCIPAL
  AUXILIAR_SUBPROGRAMAS = SIGUIENTE_LISTA_SUBPROGRAMA OF ALMACEN PROGRAMA PRINCIPAL
  DO WHILE AUXILIAR_SUBPROGRAMAS < > NIL
    WRITE NOMBRE_ARCHIVO OF AUXILIAR_SUBPROGRAMAS
    WRITE TAMAÑO OF AUXILIAR SUBPROGRAMAS
    WRITE FECHA_ACTUALIZACION OF AUXILIAR_SUBPROGRAMAS
    AUXILIAR_SUBPROGRAMAS = SIGUIENTE_LISTA SUBPROGRAMA OF AUXILIAR SUBPROGRAMA
    WRITELN
  ENDDO
END

13.2 IMPRIMIR MODULOS POR PROGRAMA
BEGIN
  AUXILIAR_PROCEDIMIENTOS = APUNTADOR_INICIO_LISTA
  DO WHILE AUXILIAR_PROCEDIMIENTOS < > NIL
    WRITE NOMBRE_MODULO OF AUXILIAR_PROCEDIMIENTOS
    WRITE TIPO OF AUXILIAR_PROCEDIMIENTOS
    WRITE NOMBRE_ARCHIVO OF AUXILIAR_PROCEDIMIENTOS
    WRITE PAGINA OF AUXILIAR_PROCEDIMIENTOS
    WRITELN
    AUXILIAR_PROCEDIMIENTOS = SIGUIENTE_LISTA_PROCEDIMIENTO OF AUXILIAR_PROCEDIMIENTOS
  ENDDO
END
```

13.3 IMPRIMIR ALCANCE DE MODULOS

```
BEGIN
 AUXILIAR_PROCEDIMIENTOS = APUNTADOR_INICIO_LISTA
 DO WHILE AUXILIAR_PROCEDIMIENTOS < > NIL
  WRITE NOMBRE_MODULO OF AUXILIAR_PROCEDIMIENTOS
  WRITE LINEA_INICIAL OF AUXILIAR_PROCEDIMIENTOS
  WRITE LINEA_FINAL OF AUXILIAR_PROCEDIMIENTOS
  WRITE SECCION_EJECUTABLE OF AUXILIAR_PROCEDIMIENTOS
  WRITE NOMBRE_ARCHIVO OF AUXILIAR_PROCEDIMIENTOS
  WRITE NUMERO_ELEMENTOS_LLAMADO OF AUXILIAR_PROCEDIMIENTOS
  WRITE NUMERO_ELEMENTOS_LLAMADOR OF AUXILIAR_PROCEDIMIENTOS
  WRITE NUMERO_ELEMENTOS_CONTENIDO OF AUXILIAR_PROCEDIMIENTOS
  WRITELN
  AUXILIAR_PROCEDIMIENTOS = SIGUIENTE_LISTA_PROCEDIMIENTO OF AUXILIAR_PROCEDIMIENTOS
 ENDDO
END
```

13.4 IMPRIMIR PROCEDIMIENTOS LLAMADOS

```
BEGIN
 MOVIMIENTO_PROCEDIMIENTO = APUNTADOR_INICIO_LISTA
 DO WHILE MOVIMIENTO_PROCEDIMIENTO < > NIL
  NOMBRE_MODULO_BUSCAR = NOMBRE_MODULO OF MOVIMIENTO_PROCEDIMIENTO
  REFERENCIA_PROCEDIMIENTO = APUNTADOR_INICIO_LISTA
  WRITE NOMBRE_MODULO_BUSCAR
  DO WHILE REFERENCIA_PROCEDIMIENTO < >NIL
   REFERENCIA_LLAMADOR = PRIMERO_LISTA_LLAMADOR OF REFERENCIA_PROCEDIMIENTO
   DO WHILE REFERENCIA_LLAMADOR < > NIL AND NOMBRE_MODULO_BUSCAR < > NOMBRE_MODULO
    OF REFERENCIA_LLAMADOR
    REFERENCIA_LLAMADOR = SIGUIENTE_LISTA_LLAMADOR OF REFERENCIA_LLAMADOR
   ENDDO
   IF REFERENCIA_LLAMADOR < > NIL * ENCONTRO EL NOMBRE
    WRITELN NOMBRE_MODULO OF MOVIMIENTO_PROCEDIMIENTO
    SE ENCONTRO MODULO
   ENDIF
   REFERENCIA_PROCEDIMIENTO = SIGUIENTE_LISTA_PROCEDIMIENTO OF REFERENCIA_PROCEDIMIENTO
  ENDDO
  MOVIMIENTO_PROCEDIMIENTO = SIGUIENTE_LISTA_PROCEDIMIENTO OF MOVIMIENTO_PROCEDIMIENTO
  IF NO SE ENCONTRO MODULO
   WRITELN
  ENDIF
 ENDDO
END
```

13.5.- IMPRIMIR PROCEDIMIENTOS LLAMADORES

```
BEGIN
 AUXILIAR_PROCEDIMIENTOS = APUNTADOR_INICIO_LISTA
 AUXILIAR_LLAMADOR = PRIMERO_LISTA_LLAMADOR OF AUXILIAR_PROCEDIMIENTOS
 DO WHILE AUXILIAR_PROCEDIMIENTO < > NIL
  WRITE NOMBRE_MODULO OF AUXILIAR_PROCEDIMIENTOS
  DO WHILE AUXILIAR_LLAMADOR < > NIL
   WRITELN NOMBRE_MODULO OF AUXILIAR_LLAMADOR
   AUXILIAR_LLAMADOR = SIGUIENTE_LLAMADOR OF AUXILIAR_LLAMADOR
  ENDDO
  AUXILIAR_PROCEDIMIENTOS = SIGUIENTE_PROCEDIMIENTO OF AUXILIAR_PROCEDIMIENTOS
  AUXILIAR_LLAMADOR = PRIMERO_LISTA_LLAMADOR OF AUXILIAR_PROCEDIMIENTOS
 ENDDO
END
```

```
13.6 IMPRIMIR ESTRUCTURA MODULAR
BEGIN
  INDENTACION = ""
  BUSCA_MODULO_REFERENCIA WITH REFERENCIA_PROCEDIMIENTO,NOMBRE_MODULO
  IMPRIME_JERARQUIA WITH REFERENCIA_PROCEDIMIENTO,PRIMERO_LISTA_LLAMADO OF
     REFERENCIA_PROCEDIMIENTO,INDENTACION
END

IMPRIME_JERARQUIA
BEGIN
  INDENTACION = "b"
  MODIFICA_PROCESO = TRUE
  DO WHILE NOT (AUXILIAR_LLAMADO = NIL) AND EXITEN MODULOS PARA PROCESAR
    IF AUXILIAR_LLAMADO = NIL
    POP AUXILIAR_LLAMADO IN REFERENCIA_LLAMADO
    AUXILIAR_LLAMADO = SIGUIENTE_LISTA OF REFERENCIA_LLAMADO
    IF MODIFICA_PROCESO
      BUSCA MODULO REFERENCIA WITH AUXILIAR_PROCEDIMIENTOS,NOMBRE_MODULO OF REFERENCIA
            LLAMADOR
      PROCESO OF AUXILAIR_PROCEDIMIENTOS = "b"
    ELSE
      MODIFICA_PROCESO = TRUE
    ENDIF
    IF IDENTIFICADOR_RECURSIVO OF AUXILIAR_LLAMADO < > "*"
      INDENTACION = INDENTACION + "b"
      PUSH AUXILIAR_LLAMADO
      BUSCA MODULO REFERENCIA WITH AUXILIAR_PROCEDIMIENTOS,NOMBRE_MODULO AUXILIAR_LLAMADO
      IF PROCESO OF AUXILIAR_PROCEDIMIENTOS < > "*"
        * ESTABLECE SI EL MODULO YA FUE PROCESADO*
        PROCESO OF AUXILIAR_PROCEDIMIENTO = "*"
        AUXILIAR_LLAMADO = PRIMERO_LISTA OF AUXILIAR_PROCEDIMIENTOS
      ELSE
        AUXILIAR_LLAMADO = NIL
        MODIFICA_PROCESO = FALSE
      ENDIF
    ELSE
      AUXILIAR_LLAMADO = NIL
    ENDIF
  ENDDO
  BUSCA MODULO REFERENCIA WITH AUXILIAR_PROCEDIMIENTOS, NOMBRE_MODULO OF AUXILIAR_LLAMADO
  PROCESO OF AUXILIAR_PROCEDIMIENTOS = "b"
END

13.7.- IMPRIMIR LISTA EDITADA
BEGIN
  READ LINEA FROM ARCHIVO_SALIDA
  DO WHILE .NOT. EOF()
    WRITELN LINEA
    MOVER EL APUNTADOR A LA SIGUIENTE LINEA
    READ LINEA FROM ARCHIVO_SALIDA
  ENDDO
END

13.8.- IMPRIMIR PROCEDIMIENTOS CONTENIDOS
BEGIN
  AUXILIAR_PROCEDIMIENTOS = LISTA_INICIAL_PROCEDIMIENTOS
  AUXILIAR_CONTENIDO = PRIMERO_LISTA_CONTENIDO OF AUXILIAR_PROCEDIMIENTOS
  DO WHILE AUXILIAR_PROCEDIMIENTO < > NIL
    WRITE NOMBRE_MODULO OF AUXILIAR_PROCEDIMIENTOS
```

```
  DO WHILE AUXILIAR_CONTENIDO < > NIL
    WRITELN NOMBRE_MODULO OF AUXILIAR_CONTENIDO
    AUXILIAR_CONTENIDO = SIGUIENTE_CONTENIDO OF AUXILIAR_CONTENIDO
  ENDDO
  AUXILIAR_PROCEDIMIENTOS = SIGUIENTE_PROCEDIMIENTO OF AUXILIAR_PROCEDIMIENTOS
  AUXILIAR_CONTENIDO = PRIMERO_LISTA_CONTENIDO OF AUXILIAR_PROCEDIMIENTO
 ENDDO
END
```

LISTA DE TOKENS

ABSOLUTE	AND	ARRAY	BEGIN	CASE	CONST
DIV	DO	DOWNTO	ELSE	END	EXTERNAL
FILE	FOR	FORWARD	FUNCTION	GOTO	IF
IMPLEMENTATION	IN	INLINE	INTERFACE	INTERRUPT	LABEL
MOD	NIL	NOT	OF	OR	PACKED
PROCEDURE	PROGRAM		RECORD	REPEAT	SET
SHL	SHR		STRING	THEN	TO
TYPE	UNIT		UNTIL	USES	VAR
WHILE	WITH		XOR	'.'	'.'
+	'-'		COMENTARIO		
DIRECTIVA_COMPILADOR			'*'	'/'	< >
< =	> =		>	<	CRLF
EOF	'='		'.'	[	(
^	','		)	]	';'
:=	'		'"	@	b
	A..Z		a..z	0..9	ERROR

PALABRAS RESERVADAS MINIESPECIFICACIONES

PALABRA	SIGNIFICADO
AND	Y
BEGIN	INICIO DE PROCESO
CASE	SELECCIONA DE
CLOSE	CIERRA UN ARCHIVO
CREATE	GENERA UNA ESTRUCTURA
DO	LLAMAR OTRA RUTINA
DO CASE	MARCA EL INICIO DEL CASE
DO WHILE	MIENTRAS
ELSE	DE OTRA FORMA
END	FIN DE PROCESO
ENDCASE	FIN DE SELECCIONA DE
ENDDO	FIN DE MIENTRAS
ENDIF	TERMINA SI
IF	SI
NOT	NO
POP	ALMACENA UN VALOR EN UN STACK
PUSH	RECUPERA UN VALOR DE UN STACK
OPEN	ABRE UN ARCHIVO
OR	O
OTHERWISE	CUALQUIER OTRO VALOR
READ	LEER
WRITE	ESCRIBIR

Diseño.

Una vez que se obtuvo la primera parte del documento de especificación estructurada, se procedió a la elaboración del diseño.

La técnica utilizada fué la de Yourdon-Constantine (para mayor información favor de ver el capítulo 3)

La segunda parte del documento de especificación estructurada fué generada a través de:

1.- Realizar una aproximación inicial utilizando los dos tipos de análisis existentes en base al flujo de datos (análisis de transformación y análisis de transacción).

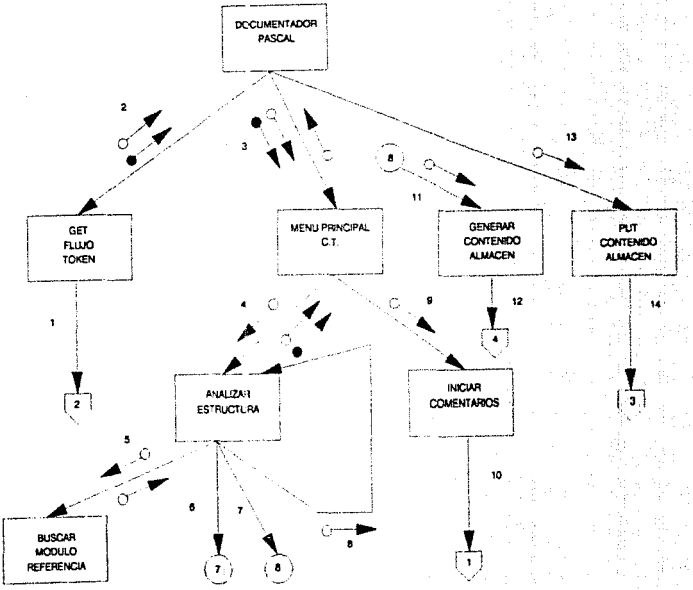
2.- Aplicar medidas de firmeza del software (Criterios de acoplamiento y cohesión)

3.- Aplicar algunas heurísticas de diseño.

Los diagramas generados, que forman parte de la segunda sección del documento de especificación estructurada, son los que se muestran en las siguientes páginas.

En estos diagramas, no se encuentran reflejados todos los flujos de datos y de control, sino que se tiene una referencia a una tabla asociada en la que se muestran dichos flujos en forma detallada. Esta organización es debida a que por la gran cantidad de información que se tiene, si ésta se coloca en los diagramas, podría causarle confusión al lector de los mismos.

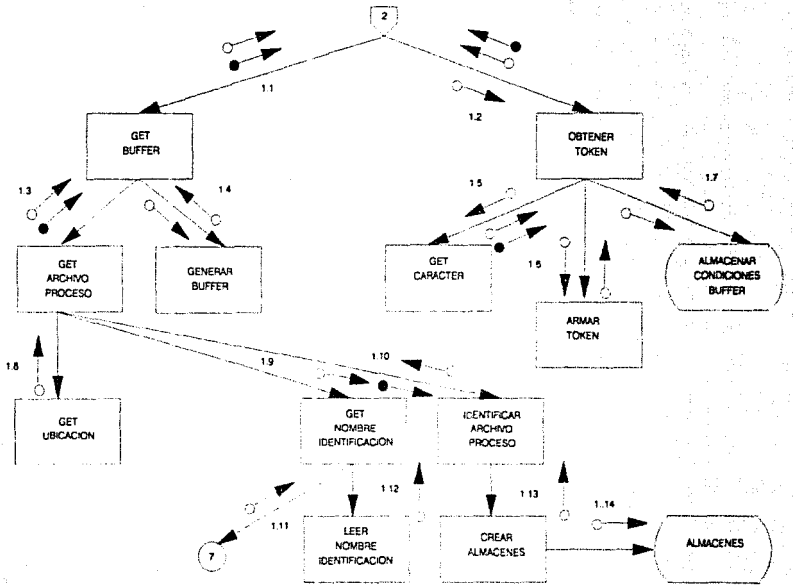
CARTA ESTRUCTURADA



PARAMETROS

PROCESO	IDENTIFICADOR	INPUT	OUTPUT
GET FLUJO TOKEN	1	CONECTOR 2	CONECTOR 2
	2		TOKEN, TIPO_TOKEN, EN_PROCESO
CENTRO TRANSFORMACION	3	TOKEN, TIPO_TOKEN, EN_PROCESO	NOMBRE_MODULO
	4	TOKEN, TIPO_TOKEN, MODULO, ACTUAL, NIVEL, END,CONTADOR, LINEAB,LINEA_INICIAL, LINEA_FINAL, SECCION, EJECUTABLE	NOMBRE_MODULO, EN_PROCESO
ANALIZAR ESTRUCTURA	5	NOMBRE_MODULO, TOKEN	ALMACEN_BUSQUEDA
	6	CONECTOR 7	CONECTOR 7
	7	CONECTOR 8	CONECTOR 8
	8	TOKEN, TIPO_TOKEN, MODULO, ACTUAL, NIVEL, END,CONTADOR, LINEAB,LINEA_INICIAL, LINEA_FINAL, SECCION, EJECUTABLE	
GENERAR INDENTACION	9	TIPO_TOKEN, TOKEN	
	10	CONECTOR 1	CONECTOR 1
GENERAR CONTENIDO ALMACEN	11	TOKEN, NOMBRE_MODULO, TIPO_CREAR	
	12	CONECTOR 5	CONECTOR 5
PUT CONTENIDO_ALMACEN	13	IDENTIFICADOR_INICIAL	
	14	CONECTOR 3	CONECTOR 3

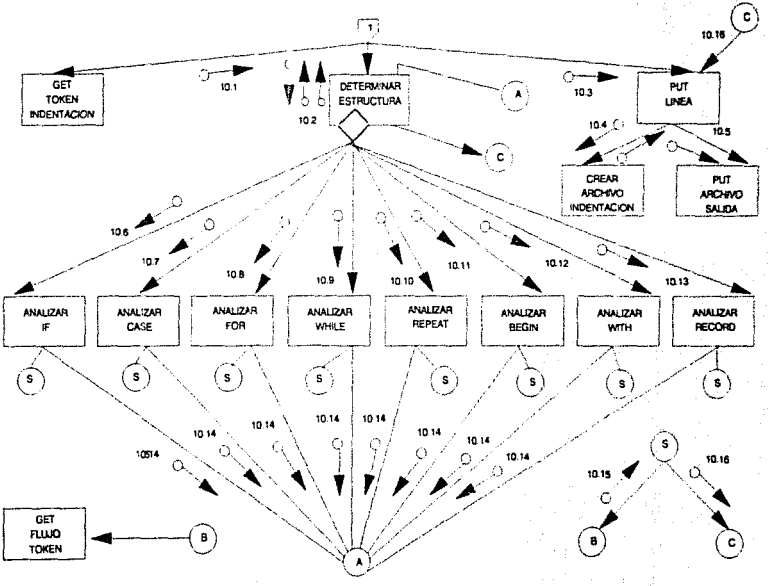
CARTA ESTRUCTURADA



PARAMETROS

PROCESO	IDENTIFICADOR	INPUT	OUTPUT
GET FLUJO TOKEN (2)	1.1	BUFFER, ARCHIVO NO EXISTENTE	
	1.2	TOKEN, TIPO_TOKEN, FIN_PROCESO	BUFFER
GET BUFFER	1.3	ARCHIVO_PROCESO, ARCHIVO NO EXISTENTE	
	1.4	BUFFER	ARCHIVO_PROCESO
GENERAR TOKEN	1.5	CARACTER, BUFFER VACIO, APUNTADOR_BUFFER	APUNTADOR_BUFFER
	1.6	TOKEN, TIPO_TOKEN	CARACTER
	1.7	BUFFER, APUNTADOR_BUFFER	BUFFER, APUNTADOR_BUFFER
GET ARCHIVO_PROCESO	1.8	UBICACION	
	1.9	NOMBRE_IDENTIFICACION, INICIO_PROCESO	
	1.10	ARCHIVO_PROCESO	UBICACION, NOMBRE_IDENTIFICACION, INICIO_PROCESO
GET NOMBRE_IDENTIFICACION	1.11	NOMBRE_IDENTIFICACION	
	1.12	NOMBRE_IDENTIFICACION	
IDENTIFICAR ARCHIVO PROCESO	1.13	INICIO_PROCESO	ALMACENES_CREADOS
	1.14		ALMACEN_PROGRAMA_PRINCIPAL, APUNTADOR_INICIO_LISTA

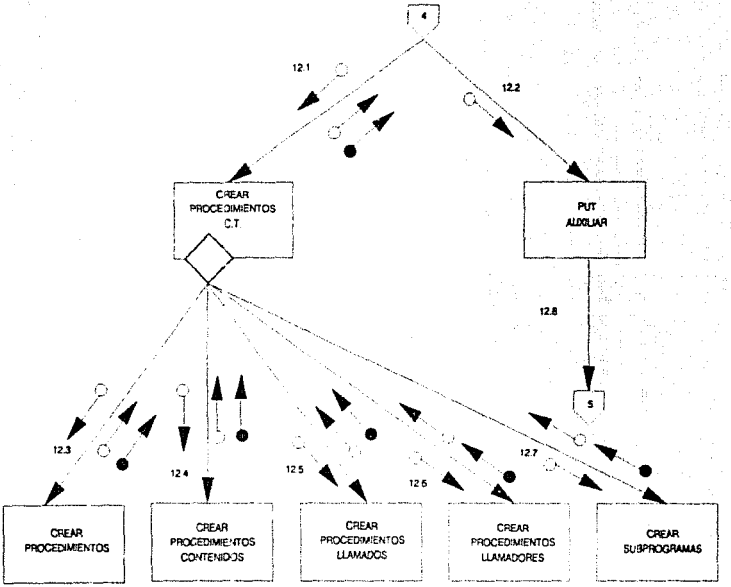
CARTA ESTRUCTURADA



PARAMETROS

PROCESO	IDENTIFICADOR	INPUT	OUTPUT
GENERAR INDENTACION(1)	13.1	INDENTACION_INICIAL	
	13.2	LINEA	INDENTACION_INICIAL, TIPO_TOKEN, TOKEN, PROCESO_TERMINADO
	13.3		LINEA
PUT LINEA	13.4	ARCHIVO_SALIDA	LINEA
	13.5		ARCHIVO_SALIDA
DETERMINAR ESTRUCTURA C.T	13.6	LINEA	INDENTACION_INICIAL, TOKEN, TIPO IF
	13.7	LINEA	INDENTACION_INICIAL, TOKEN, TIPO_CASE
	13.8	LINEA	INDENTACION_INICIAL, TOKEN, TIPO_FOR
	13.9	LINEA	INDENTACION_INICIAL, TOKEN, TIPO_WHILE
	13.10	LINEA	INDENTACION_INICIAL, TOKEN, TIPO_REPEAT
	13.11	LINEA	INDENTACION_INICIAL, TOKEN, TIPO_BEGIN
	13.12	LINEA	INDENTACION_INICIAL, TOKEN, TIPO_WITH
	13.13	LINEA	INDENTACION_INICIAL, TOKEN, TIPO_RECORD
ANALIZAR TIPOS (IF, CASE, WHILE, REPEAT, BEGIN, WITH, RECORD)	13.14		INDENTACION_INICIAL, TOKEN, ORACION, TIPO_TOKEN
	13.15	TOKEN, TIPO_TOKEN	
PUT LINEA	13.16		LINEA

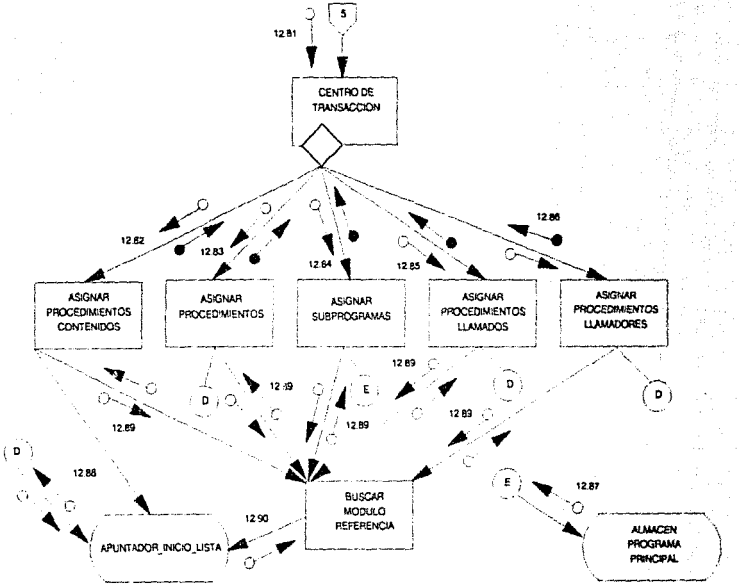
CARTA ESTRUCTURADA



PARAMETROS

PROCESO	IDENTIFICADOR	INPUT	OUTPUT
GENERAR CONTENIDO ALMACÉN (5)	12.1	AUXILIAR_PROCEDIMIENTOS, AUXILIAR_CONTENIDO, AUXILIAR_LLAMADOS, AUXILIAR_LLAMADOR, AUXILIAR_SUBPROGRAMA, PROCEDIMIENTO_CREADO	TOKEN, NOMBRE_MODULO
	12.2		AUXILIAR_PROCEDIMIENTOS, AUXILIAR_CONTENIDO, AUXILIAR_LLAMADOS, AUXILIAR_LLAMADOR, AUXILIAR_SUBPROGRAMA, NOMBRE_MODULO, TOKEN
CENTRO TRANSACCION (CREAR PROCEDIMIENTOS)	123	AUXILIAR_PROCEDIMIENTOS,PROCEDIMIENTO_REGISTRADO	TOKEN, NOMBRE_MODULO, TIPO_CREAR
	12.4	AUXILIAR_CONTENIDO, PROCEDIMIENTO_CREADO	TOKEN, NOMBRE_MODULO
	12.5	AUXILIAR_LLAMADOS, PROCEDIMIENTO_CREADO	TOKEN, NOMBRE_MODULO
	12.6	AUXILIAR_LLAMADOR, PROCEDIMIENTO_CREADO	TOKEN, NOMBRE_MODULO
	12.7	AUXILIAR_SUBPROGRAMA, PROCEDIMIENTO_CREADO	TOKEN, NOMBRE_MODULO
PUT AUXILIAR	12.8	CONECTOR 4	CONECTOR 4

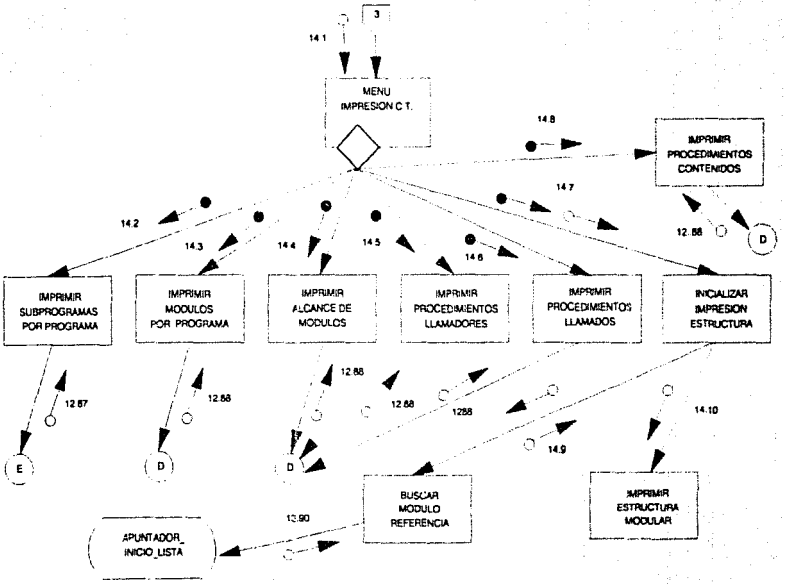
CARTA ESTRUCTURADA



PARAMETROS

PROCESO	IDENTIFICADOR	INPUT	OUTPUT
PUT AUXILIAR	12.81		AUXILIAR_PROCEDIMIENTOS, AUXILIAR_CONTENIDO, AUXILIAR_LLAMADO, AUXILIAR_LLAMADOR, AUXILIAR_SUB PROGRAMAS, TIPO_CREAR
CENTRO TRANSACCION (ASIGNAR PROCEDIMIENTOS GENERADOS)	12.82	OPERACION_CORRECTA	AUXILIAR_CONTENIDO, TOKEN, NOMBRE_MODULO
	12.83	OPERACION_CORRECTA	NOMBRE_MODULO, TOKEN, AUXILIAR_PROCEDIMIENTOS
	12.84	OPERACION_CORRECTA	NOMBRE_MODULO, TOKEN, AUXILIAR_SUBPROGRAMAS
	12.85	OPERACION_CORRECTA	NOMBRE_MODULO, TOKEN, AUXILIAR_LLAMADO
	12.86	OPERACION_CORRECTA	NOMBRE_MODULO, TOKEN, AUXILIAR_LLAMADOR
ASIGNAR PROCEDIMIENTOS LLAMADORES	12.87	ALMACEN_PROGRAMA_PRINCIPAL	
ASIGNAR PROCEDIMIENTOS CONTENIDOS, ASIGNAR PROCEDIMIENTOS, ASIGNAR PROCEDIMIENTOS LLAMADOS, ASIGNAR PROCEDIMIENTOS LLAMADORES	12.88	APUNTADOR_INICIO_LISTA	APUNTADOR_INICIO_LISTA
ASIGNAR PROCEDIMIENTOS CONTENIDOS, ASIGNAR PROCEDIMIENTOS, ASIGNAR SUBPROGRAMAS, ASIGNAR PROCEDIMIENTOS LLAMADOS, ASIGNAR PROCEDIMIENTOS LLAMADORES	12.89	ALMACEN_BUSQUEDA	TOKEN, NOMBRE_MODULO
BUSCA MODULO REFERENCIA	12.90	APUNTADOR_INICIO_LISTA	

CARTA ESTRUCTURADA



PARAMETROS

PROCESO	IDENTIFICADOR	INPUT	OUTPUT
PUT CONTENIDO ALMACEN	14.1		IDENTIFICADOR_INICIAL
CENTRO TRANSACCION (MENU IMPRESION)	14.2		INICIO_IMPRESION
	14.3		INICIO_IMPRESION
	14.4		INICIO_IMPRESION
	14.5		INICIO_IMPRESION
	14.6		INICIO_IMPRESION
	14.7		INICIO_IMPRESION, IDENTIFICADOR_INICIAL
	14.8		INICIO_IMPRESION
INICIALIZAR IMPRESION ESTRUCTURA	14.9	REFERENCIA_PROCEDIMIENTO	REFERENCIA_PROCEDIMIENTO
	14.10		REFERENCIA_PROCEDIMIENTO, INDENTACION, PRIMERO_LISTA, LLAMADOS DE REFERENCIA_PROCEDIMIENTO

Programación.

Una vez efectuado el análisis y el diseño se procedió a la programación.

En las páginas siguientes se muestra una descripción del funcionamiento de cada uno de los módulos que forman parte del documentador. Para un mayor detalle favor de referirse al disco que contiene los programas del mismo.

Nombre de la unidad: ANALIZAR_BEGIN
Descripción: Este módulo realiza el procesamiento de la estructura begin.
Entradas:
Indentacion_inicial: Contiene la indentación que ha sido construida durante el proceso de documentación.
token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar.
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos.
Genera la línea comentarizada de la instrucción en proceso.

Nombre de la unidad: ANALIZAR_CASE
Descripción: Este módulo realiza el procesamiento de la estructura case, considerando la parte else de la instrucción.
Entradas:
Indentacion_inicial: Contiene la indentación que ha sido construida durante el proceso de documentación.
token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos.
Genera la línea comentarizada de la instrucción en proceso.

Nombre de la unidad: ANALIZAR_ESTRUCTURA
Descripción: Este proceso realiza la busqueda del token program para poder efectuar el análisis de estructuras. Si no lo encuentra, es decir , si identifica que se trata de un programa con una estructura inválida, termina el proceso.
Salidas:
Relación de listas que contienen la información de los módulos declarados, referenciados a partir del apuntador_inicio_lista.

Nombre de la unidad: ANALIZAR_ESTRUCTURA_CASE
Descripción: Este módulo realiza el procesamiento de la estructura que se encuentra después de una instruccion "case caracter of".
Entradas:
indentacion_inicial: Contiene la indentación que ha sido construida durante el proceso de documentación.
token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar.
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos.
Genera la línea comentarizada de la instrucción en proceso.

Nombre de la unidad: ANALIZAR_FOR
Descripción: Este módulo realiza el procesamiento de la estructura for.
Entradas:
Indentacion_inicial: Contiene la indentación que ha sido construida durante el proceso de documentación.
token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar.
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos.
Genera la línea comentarizada de la instrucción en proceso.

Nombre de la unidad: ANALIZAR_IF
Descripción: Este módulo realiza el procesamiento de la estructura if, considerando la parte else de la instrucción.
Entradas:
Indentacion_inicial: Contiene la indentación que ha sido armada durante el proceso de documentación.
token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar.
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos.
Genera la línea comentarizada de la instrucción en proceso.

Nombre de la unidad: ANALIZAR_RECORD
Descripción: Este módulo realiza el procesamiento de la estructura record.
Entradas:
Indentacion_inicial: Contiene la indentación que ha sido construida durante el proceso de documentación.

token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar.
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos.
Genera la línea comentarizada de la instrucción en proceso.

Nombre de la unidad: ANALIZAR_REPEAT
Descripción: Este módulo realiza el procesamiento de la estructura repeat.
Entradas:
Indentacion_inicial: Contiene la indentación que ha sido construida durante el proceso de documentación.
token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos.
Genera la línea comentarizada de la instrucción en proceso

Nombre de la unidad: ANALIZAR_WHILE
Descripción: Este módulo realiza el procesamiento de la estructura while.
Entradas:
Indentacion_inicial: Contiene la indentación que ha sido construida durante el proceso de documentación.
token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos.
Genera la línea comentarizada de la instrucción en proceso.

Nombre de la unidad: ANALIZAR_WITH
Descripción: Este módulo realiza el procesamiento de la estructura with.
Entradas:
Indentacion_inicial: Contiene la indentación que ha sido construida durante el proceso de documentación.
token: Contiene el token a procesar.
tipo_token: Contiene el tipo de token a procesar
Salidas:
Proporciona la indentación que servirá como entrada a otros procesos..
Genera la línea comentarizada de la instrucción en proceso.

Nombre de la unidad: ARMAR_TOKEN
Descripción: Este proceso permite la construccion de tokens en base a un lenguaje específico, el cuál se apoya en la rutina de get_caracter para obtener los elementos necesarios en el armado.
La rutina también reconoce a los identificadores que son parte del conjunto de palabras reservadas.
Entradas:
Caracter: Caracter inicial a procesar.
Tipos de tokens.
Estado_siguiente: Determina el siguiente estado a procesar.
Salidas:
Token: Token construido en base a los caracteres leidos.
Tipo_token: Tipo de token de que se trate.
Caracter: Es el caracter que quedo pendiente de procesar cuando se regresa el apuntador de buffer en 1.
Estado_siguiente: Determina el siguiente estado a procesar.

Nombre de la unidad: ASIGNAR_PROCEDIMIENTOS
Descripción: Este proceso clasifica el módulo más recientemente creado dentro de la lista de módulos del sistema.(La clasificación es realizada en orden ascendente)
Entradas:
Apuntador_inicio_lista: Referencia a la lista de módulos definidos.
Auxiliar_procedimientos: módulo más recientemente creado.
Salidas:
Lista ordenadas de módulos declarados.
Almacen_busqueda : Módulo más recientemente referenciado.

Nombre de la unidad: ASIGNAR_PROCEDIMIENTOS_CONTENIDOS
Descripción: Este proceso clasifica al módulo contenido más recientemente creado dentro de la lista de módulos contenidos del módulo de proceso especificado.(La clasificación es realizada en forma ascendente)
Entradas:
Nombre_modulo : Nombre del módulo actual de proceso.
Apuntador_inicio_lista: Referencia a la lista de módulos definidos.
Auxiliar_contenido: Módulo contenido más recientemente creado.
Almacen_busqueda: Módulo más recientemente referenciado.
Salidas:
Listas ordenadas de módulos.

Nombre de la unidad: ASIGNAR_PROCEDIMIENTOS_LLAMADORES
Descripción: Este proceso clasifica al módulo llamador más recientemente creado dentro de la lista de módulos llamadores del módulo de proceso especificado. (La clasificación es realizada en forma ascendente)
Entradas:
Token: Nombre del módulo actual de proceso
Apuntador_inicio_lista: Referencia a la lista de módulos definidos.
Auxiliar_llamador: Módulo llamador más recientemente creado.
Almacen_busqueda: Módulo más recientemente referenciado.
Salidas:
Listas ordenadas de módulos.

Nombre de la unidad: ASIGNAR_PROCEDIMIENTOS_LLAMADOS
Descripción: Este proceso clasifica al módulo llamado más recientemente creado dentro de la lista de módulos llamados del módulo de proceso especificado. (La clasificación es realizada en la forma en que son llamados)
Entradas:
Nombre_modulo: Nombre del módulo actual de proceso
Apuntador_inicio_lista: Referencia a la lista de módulos definidos.
Auxiliar_llamado: Módulo llamador más recientemente creado.
Almacen_busqueda: Módulo más recientemente referenciado.
Salidas:
Listas ordenadas de módulos.

Nombre de la unidad: ASIGNAR_SUBPROGRAMAS
Descripción: Este proceso clasifica al subprograma más recientemente creado dentro de la lista de subprogramas. (La clasificación es realizada en forma ascendente)
Entradas:
Auxiliar_programa_principal: Referencia al programa principal.
Auxiliar_subprogramas: Subprograma más recientemente creado.
Salidas:
Listas ordenadas de módulos.

Nombre de la unidad: BORRA_HEAP
Descripción: Este proceso es utilizado para limpiar la memoria antes de procesar cualquier opción del menú principal.

Nombre de la unidad: BOX
Descripción: Dibuja un marco a pantalla en las coordenadas especificadas.
Entradas:
lsi: linea superior izquierda
csi: columna superior izquierda
lid: linea inferior derecha
cid: columna inferior derecha
Salidas:
Marco a pantalla.

Nombre de la unidad: BUSCA_MODULO_REFERENCIA
Descripción: Este proceso realiza la busqueda de un módulo en la lista de unidades almacenadas. Si lo encuentra, direcciona el apuntador de referencia a dicho módulo, de otra forma asigna el apuntador a nil.
Entradas:
Apuntador_inicio_lista: Contiene una referencia al primer módulo de la lista.
Modulo_referencia: Nombre del módulo a buscar.
Salidas:
Almacen_busqueda : Contiene una referencia al módulo solicitado.

Nombre de la unidad: CLRWINDOW
Descripción: Este proceso realiza el borrado de una sección de pantalla en particular.
La característica principal de éste procedimiento es que el borrado de la pantalla se realiza accesando directamente a la memoria de video.
Entradas:
columna1: Columna inicial de la pantalla.
renglon1: Renglón superior de la pantalla.
columna2: Columna final de la pantalla.
renglon2: Renglón inferior de la pantalla.
Salidas:
Borrado de una sección en pantalla.

Nombre de la unidad: CONFIGURACION_IMPRESORA
Descripción: Este proceso realiza un salto de página considerando el tipo de impresora que se está utilizando.
Entradas:
Tipo_impresora: Impresora seleccionada (M = matriz, L = laser.
Salidas:
Secuencia de Escape dependiendo del tipo de impresora seleccionada.

Nombre de la unidad: CONTROL_OPCIONES MENÚ.
Descripción: Controla la selección de cada una de las opciones dentro de un menú.
Entradas:
Columna: Columna de despliegue.
Renglon_inicial: Renglón actual de proceso.
Color_back : Color background menú.
Color_for : Color foreground menú.
Color_back_menu: Color background opción.
Color_for_menu : Color foreground opción.
Tamanio_menu : Número de opciones del menú.
Salidas:
Opción seleccionada.

Nombre de la unidad: CONTROL_PANTALLA
Descripción: Este proceso proporciona un control de la información que aparecerá tanto en una hoja de impresión como en la pantalla. También proporciona un mecanismo para cancelar la consulta o impresión.
Entradas:
Linea_fin_impresión: Ultima línea de impresión en pantalla y en impresora.
fin_linea: Indica si se debe de evaluar la impresión de fin de página.
linea_impresión: Primera linea de impresión en pantalla y en impresora.
Salidas:
Mensaje de cancelación de impresión/despliegue.
Mensaje de continuación de pantalla.
Control_pantalla: Determina sí se cancela el proceso.

Nombre de la unidad: CONTROL_SALIDA (EXIT PROCEDURE).
Descripción: Este proceso permite controlar los errores de terminación anormal.
Entradas:
ExitCode: Contiene el número de error.
ErrorAddr: Contiene la dirección de la oración donde ocurrió el error.
Salidas:
ErrorAdrr: Debe contener el valor de nil para no ser procesado nuevamente.

Nombre de la unidad: CREAR_PROCEDIMIENTOS
Descripción: Este proceso genera una estructura de datos dinámica que contendrá la información de un nuevo módulo de proceso.
Entradas:
Token: Nombre del módulo.
Tipo_modulo: Módulo general, procedimiento o función.
Salidas:
Auxiliar_procedimientos: Estructura de datos dinámica que contiene la referencia de un módulo.

Nombre de la unidad: CREAR_PROCEDIMIENTO_CONTENIDO
Descripción: Este proceso genera una estructura de datos dinámica que contendrá la información de un módulo contenido (declarado dentro de otro módulo).
Entradas:
Token: Nombre del módulo contenido.
Salidas:
Auxiliar_contenido: estructura de datos dinámica que contiene la referencia de un módulo contenido.

Nombre de la unidad: CREAR_PROCEDIMIENTO_LLAMADO
Descripción: Este proceso genera una estructura de datos dinámica que contendrá la información de un módulo llamado (identificando si se trata de un módulo recursivo).
Entradas:
Nombre_modulo: Nombre del módulo actual de proceso.
Token: Nombre del módulo llamado.
Salidas:
Auxiliar_llamado: Estructura de datos dinámica que contiene la referencia de un módulo llamado.

Nombre de la unidad: CREAR_PROCEDIMIENTO_LLAMADOR
Descripción: Este proceso genera una estructura de datos dinámica que contendrá la información de un módulo llamador (identificando si se trata de un módulo recursivo).
Entradas:
Nombre_modulo: Nombre del módulo actual de proceso.
Token: Nombre del módulo llamador.
Salidas:
Auxiliar_llamador: Estructura de datos dinámica que contiene la referencia de un

módulo llamador.

Nombre de la unidad: CREAR_SUBPROGRAMAS
Descripción: Este proceso genera una estructura de datos dinámica que contendrá la información de un subprograma.
Entradas:
Token: Nombre del módulo subprograma.
Salidas:
Auxiliar_subprogramas: Estructura de datos dinámica que contendrá la información de un subprograma.

Nombre de la unidad: CUENTA_CRLF
Descripción: Realiza la cuenta del número de crlf existentes (dentro de un string).
Entradas:
Token: String de entrada.
Salidas:
Número de crlf que se encuentran dentro del string.

Nombre de la unidad: DESPLIEGA_DISPOSITIVO
Descripción: Despliega en la parte superior de la pantalla la impresora que ha sido seleccionada, es decir, que se encuentra activa.
Entradas:
Dispositivo: 0 = pantalla, 1 = impresora.
Salidas:
El dispositivo seleccionado.

Nombre de la unidad: DETERMINAR_ESTRUCTURA
Descripción:
Este módulo efectúa el procesamiento de las instrucciones restantes (que no han sido consideradas en otros módulos.
Entradas:
Token: Contiene el token a procesar.
Indentacion_inicial: Contiene la indentación actual de proceso.
Tipo_token: Contiene el tipo de token a procesar.
Salidas:
Línea comentarizada.

Nombre de la unidad: DOCUMENTADOR_PASCAL
Descripción: Este proceso tiene como finalidad el llevar las siguientes fases:

a) Proporcionar la documentación que servirá de apoyo para la elaboración de un manual de referencia técnica.
b) Proporcionar la información necesaria para poder entender el funcionamiento de cada uno de los módulos que forman parte de un sistema.
c) Actualizar la documentación del sistema de una manera sencilla y rápida, cada vez que ocurran cambios en el mismo.
d) Poder efectuar un mantenimiento más rápido a los sistemas cuando no se cuente con una documentación formal de desarrollo.
e) Identificar de una forma más eficiente, cada uno de los elementos que forman parte de una programación estructurada al efectuar un mantenimiento a diversos programas.
f) Identificar de una manera más rápida los efectos que se tendrían al realizar una modificación a cualquier módulo que forma parte del sistema.

Entradas:
Archivo_proceso: Nombre del archivo principal de proceso.
Salidas:
Reportes y consultas que formarán parte de la documentación técnica.
Nota: El proceso asume que el sistema contiene un conjunto de estructuras válidas, es decir, que se trate de un sistema que se encuentra operando.

Nombre de la unidad: ENCABEZADO_ALCANCE_MODULOS
Descripción: Este proceso realiza la impresión del encabezado de la opción tanto a pantalla como a impresora.
Entradas:
Dispositivo: 0 = pantalla 1 = impresora.
Salidas:
Encabezado consulta/impresión de la opción.

Nombre de la unidad: ENCABEZADO_CONTENIDO
Descripción: Este proceso realiza la impresión del encabezado de la opción tanto a pantalla como a impresora.
Entradas:
Dispositivo: 0 = pantalla 1 = impresora.
Salidas:
Encabezado consulta/impresión de la opción.

Nombre de la unidad: ENCABEZADO_ESTRUCTURA_MODULAR
Descripción: Este proceso realiza la impresión del encabezado de la opción tanto a pantalla como a impresora.

Entradas:
Dispositivo: 0 = pantalla 1 = impresora.
Salidas:
Encabezado consulta/impresión de la opción.

Nombre de la unidad: ENCABEZADO_IDENTIFICACIÓN
Descripción: Este proceso proporciona un encabezado general utilizado durante la impresión de los reportes (secuencias de escape).
Entradas:
Incluyo_pagina: Determina sí se va a incluir el número de página durante la impresión.
Salidas:
Encabezado a impresión (secuencias de escape).

Nombre de la unidad: ENCABEZADO_LLAMADO
Descripción: Este proceso realiza la impresión del encabezado de la opción tanto a pantalla como a impresora.
Entradas:
Dispositivo: 0 = pantalla 1 = impresora.
Salidas:
Encabezado consulta/impresión de la opción.

Nombre de la unidad: ENCABEZADO_LLAMADOR
Descripción: Este proceso realiza la impresión del encabezado de la opción.
Entradas:
Dispositivo: 0 = pantalla 1 = impresora.
Salidas:
Encabezado consulta/impresión de la opción.

Nombre de la unidad: ENCABEZADO_MODULOS_PROGRAMA
Descripción: Este proceso realiza la impresión de los encabezado de la opción tanto a pantalla como a impresora.
Entradas:
Dispositivo: 0 = pantalla 1 = impresora.
Salidas:
Encabezado consulta/impresión de la opción.

Nombre de la unidad: ENCABEZADO_PROGRAMAS_NUMERADOS
Descripción: Este proceso imprime el encabezado del programa numerado.

Entradas:
Nombre_programa: Nombre del programa a imprimir
Salidas:
Encabezado.

Nombre de la unidad: ENCABEZADO_PROGRAMAS_NUMERADOS
Descripción: Este proceso imprime el encabezado del programa numerado.
Entradas:
Nombre_programa: Nombre del programa a imprimir
Salidas:
Encabezado.

Nombre de la unidad: GET_ARCHIVO
Descripción: Este proceso obtiene la ubicación del archivo a procesar. Se basa en la rutina get_nombre_identificación para realizar la construcción.
Salidas:
Ubicacion: Indica el archivo a procesar.

Nombre de la unidad: GET_CARACTEr
Descripción: Este proceso permite obtener un caracter de un archivo de entrada manteniendo el apuntador en el siguiente caracter disponible.
La variable apuntador_buffer es para determinar si se regresa un caracter de acuerdo al procesamiento de la rutina armar_token.
Entradas:
Archivo_proceso: Archivo de texto a procesar.
Caracter: Caracter a leer.
Salidas:
Archivo_proceso: Mantiene un apuntador del siguiente caracter a leer.
Caracter: Caracter leído.

Nombre de la unidad: GET_FLUJO_TOKEN
Descripción: Este proceso permite obtener un token, así como su tipo, del archivo de proceso en turno. La unidad controla el archivo que se encuentra en proceso actualmente y guarda el apuntador de buffer del archivo que no está siendo procesado. Tambien guarda el valor del siguiente caracter a procesar.
Entradas:
Caracter: Siguiente caracter a procesar.
Salidas:
Caracter: Siguiente caracter a procesar cuando se regresó en 1 el apuntador de buffer.

Token: Contiene el token actual de proceso.
Tipo_token: Contiene el tipo del token en proceso.
Archivo_proceso: Indica el archivo actual de proceso.
Error:indica si ocurrió un error en la lectura.

Nombre de la unidad: GET_NOMBRE_IDENTIFICACION
Descripción: Este proceso permite obtener y validar el nombre de la rutina principal que se tomará como referencia para iniciar el proceso de documentación.
Salidas:
Nombre_identificacion: Nombre del archivo a procesar.

Nombre de la unidad: IMPRIME_ENCABEZADO
Descripción: Este proceso realiza el control de la impresión del encabezado.

Nombre de la unidad: IMPRIME_JERARQUIA
Descripción: Este proceso realiza la consulta/impresión de los módulos llamados durante el flujo de operación del sistema, estableciendo una estructura jerárquica de dichos módulos.
Entradas:
Apuntador_inicio_lista: Contiene la referencia del primer módulo ordenado que forma parte del sistema. Si su valor es nil, indica que no se ha procesado la opción de análisis de estructuras.
Salidas:
Impresión jerárquica de las llamadas de módulos

Nombre de la unidad: IMPRIMIR_ALCANCE_MODULOS
Descripción: Este proceso realiza la consulta/impresión de los módulos contenidos en cada programa (subprograma), indicando la lócalización en detalle de cada uno de los módulos que forman parte del sistema.
Entradas:
Apuntador_inicio_lista: Contiene la referencia del primer módulo ordenado que forma parte del sistema. Si su valor es nil, indica que no se ha procesado la opción de análisis de estructuras.
Salidas:
Relación de alcance de módulos.

Nombre de la unidad: IMPRIMIR_ESTRUCTURA_MODULAR
Descripción: Este proceso realiza la consulta/impresión de los módulos llamados durante el flujo de operación del sistema, estableciendo una estructura jerárquica de los módulos que son llamados. El procedimiento se basa en el uso del módulo

imprime_jerarquia.

Entradas:

Apuntador_inicio_lista: Contiene la referencia del primer módulo ordenado que forma parte del sistema. Si su valor es nil, indica que no se ha procesado la opción de análisis de estructuras.

Nombre de la unidad: IMPRIMIR_MODULOS_PROGRAMA

Descripción: Este proceso realiza la consulta/impresión de la módulos contenidos en cada programa (subprograma), indicando la página de impresión en donde se encuentra dicho módulo.

Entradas:

Apuntador_inicio_lista: Contiene la referencia del primer módulo ordenado que forma parte del sistema. Si su valor es nil, indica que no se ha procesado la opción de análisis de estructuras.

Salidas:

Relación de módulos por programa.

Nombre de la unidad: IMPRIMIR_PROCEDIMIENTOS_CONTENIDO

Descripción: Este proceso realiza la consulta/impresión de los módulos declarados dentro de cada módulo.

Entradas:

Apuntador_inicio_lista: Contiene la referencia del primer módulo ordenado que forma parte del sistema. Si su valor es nil, indica que no se ha procesado la opción de análisis de estructuras.

Salidas:

Relación de módulos contenidos.

Nombre de la unidad: IMPRIMIR_PROCEDIMIENTOS_LLAMADOR

Descripción: Este proceso realiza la consulta/impresión de la relación de módulos y los módulos en donde ellos son llamados.

Entradas:

Apuntador_inicio_lista: Contiene la referencia del primer módulo ordenado que forma parte del sistema. Si su valor es nil, indica que no se ha procesado la opción de análisis de estructuras.

Salidas:

Relación de módulos vs. módulos llamadores.

Nombre de la unidad: IMPRIMIR_PROCEDIMIENTOS_LLAMADOS

Descripción: Este proceso realiza la consulta/impresión de la relación de módulos que llama cada uno de ellos.

La forma en que efectúa el proceso es el siguiente:
a) Obtiene un módulo de la lista ordenada de módulos.
b) Imprime el nombre del módulo.
c) Se coloca al principio de la lista y si el modulo de la lista contiene el nombre del módulo del punto a) en su lista de módulos "llamadores", se imprime el nombre del módulo de la lista.
d) mientras no se termine la lista, se realiza lo mismo que el punto anterior para los módulos restantes.
e) se repiten los pasos anteriores para cada uno de los módulos que forman parte de la lista de modulos.
Entradas:
Apuntador_inicio_lista: Contiene la referencia del primer módulo ordenado que forma parte del sistema. Si su valor es nil, indica que no se ha procesado la opción de análisis de estructuras.
Salidas:
Relación de módulos vs. módulos llamados.

Nombre de la unidad: IMPRIMIR_PROGRAMAS_NUMERADOS
Descripción: Este proceso realiza la impresión numerada tanto de los programas documentados como cada uno de los programas que forman parte del sistema.
Entradas:
Nombre_programa: Nombre del programa a imprimir.
Salidas:
Impresión del programa numerado.

Nombre de la unidad: IMPRIMIR_SUBPROGRAMAS_PROGRAMA
Descripción: Este proceso realiza la consulta/impresión de la relación de subprogramas por programa. El proceso pude ser cancelado mediante la tecla Escape.
Entradas:
Auxiliar_programa_principal: Contiene la referencia del programa principal, así como los subprogramas que éste llama.
Apuntador_inicio_lista: Contiene la referencia del primer módulo ordenado que forma parte del sistema. Si su valor es nil, indica que no se ha procesado la opción de análisis de estructuras.
Salidas:
Relación de subprogramas por programa

Nombre de la unidad: INICIA_COMENTARIOS
Descripción: Este módulo mantiene el control del archivo a comentarizar.

Entradas:
Token : Contiene el token a procesar.
Indentacion_inicial: Contiene la indentación de proceso.
Tipo_token: Contiene el tipo de token a procesar.
Salidas:
Archivo comentarizado.

Nombre de la unidad: ISPRINTER
Descripción: Verifica si la impresora se encuentra lista para recibir información.
Salidas:
True: La impresora está disponible.
False:La impresora no está disponible.

Nombre de la unidad: LEE_SIGUIENTE
Descripción: Este proceso obtiene el siguiente token a procesar.
Entradas:
Token: Contiene el token a procesar.
Tipo_token: Contiene el tipo de token a procesar.
Salidas:
Token: Contiene el token a procesar
Tipo_token: Contiene el tipo de token a procesar

Nombre de la unidad: LLENA_BLANCOS
Descripción: Este proceso es utilizado como auxiliar durante el uso de la función write(ln) a impresora. Sirve para proporcionar una indentación en la impresión, debido a que cuando se utilizan los dos puntos se colocan blancos a la izquierda del string y no a la derecha que es como se necesita.
Entradas:
Cadena: Contiene la cadena a utilizar
Longitud: Contiene el tamaño del string a imprimir. Es utilizado para determinar el numero de blancos a agregar.
Salidas:
Un string que contienen la cadena inicial + blancos.

Nombre de la unidad: LOGOTIPO
Descripción: Este proceso realiza el depliegue del logotipo de identificación.
Entradas:
Inicio: Determina si se trata del inicio del sistema o de un redespliegue.
Salidas:
Logotipo del sistema

Nombre de la unidad: MENSAJE
Descripción: Realiza el desplegado de un mensaje (centrado) en la línea 24 de la pantalla.
Entradas:
Contenido : Descripción del mensaje a desplegar.
Salidas:
Mensaje a pantalla.

Nombre de la unidad: MENU_IMPRESION
Descripción: Este proceso controla la selección de cada una de las opciones que forman parte del menú de consultas/reportes
Entradas:
Pantalla: Contiene la pantalla actual de proceso, la cuál servirá como referencia para poder realizar las funciones de save y restore screen.

Nombre de la unidad: MENU_PRINCIPAL
Descripción: Este proceso controla la selección de cada una de las opciones que forman parte del menú principal del sistema.
Entradas:
Pantalla: Contiene la pantalla actual de proceso, la cuál servirá como referencia para poder realizar las funciones de save y restore screen.

Nombre de la unidad: NIVEL_ANIDAMIENTO
Descripción: Este proceso realiza una busqueda de los módulos declarados y referenciados dentro de un programa en particular. Crea la lista de módulos, así como las listas de módulos contenidos, llamados y llamadores.
El procedimiento es llamado en forma recursiva cada vez que se procesa un nuevo módulo (programa principal, procedimiento o función).
En el mismo proceso se asignan los números de página, linea inicial, linea final y sección ejecutable de cada una de las unidades.
Inicializa los variables anteriores cada vez que un nuevo programa es llamado, colocandolas a su estado anterior una vez que se regresa del archivo de llamado (directiva include).
El módulo reconoce e identifica tanto a los módulos recursivos como aquellos que fueron declarados mediante un Forward.
Entradas:
Nombre_modulo : nombre del módulo de proceso.
contador_lineas: número de línea actual de proceso.
nombre_archivo : nombre del archivo actual de proceso.

Salidas:
Relación de listas que contienen la información de los módulos contenidos, referenciados a partir del apuntador_inicio_lista.

Nombre de la unidad: OBTENER_TOKEN
Descripción: Este proceso llama a las rutinas que leen un caracter y arman el token.
Entradas:
Caracter: Siguiente caracter a procesar.
Salidas:
Caracter: Siguiente caracter a procesar cuando se regresó en 1 el apuntador del buffer
Token: Contiene el token actual de proceso.
Tipo_token: Contiene el tipo del token en proceso.

Nombre de la unidad: SELECCIONA_DISPOSITIVO
Descripción: Selecciona el dispositivo a utilizar durante el uso del módulo de consultas y reportes.
Salidas:
Dispositivo: 0 = pantalla, 1 = impresora

Nombre de la unidad: SETCURSOR
Descripción: Este proceso permite encender y apagar el cursor durante el procesamiento de información.
Entradas:
Tope y Fondo: Dependiendo del valor de éstas variables se podrá colocar el cursor en un tamaño establecido.
Salidas:
Cambia de tamaño el cursor.

Nombre de la unidad: SOMBRA
Descripción: Produce una sombra en las coordenadas establecidas mediante el uso de la memoria de video.
Entradas:
lsi: Línea superior izquierda.
csi: Columna superior izquierda.
lid: Línea inferior derecha.
cid: Columna inferior derecha.
Salidas:
Sombra a pantalla.

Nombre de la unidad: TECLAS_MENU
Descripción: Controla el movimiento de cada uno de los menús existentes dentro del sistema.
Entradas:
Renglon_movimiento : Renglón seleccionado.
Tamanio_menu: Número de opciones dentro del menú.
Salidas:
Renglón seleccionado.

VII.- CASO PRACTICO

En éste capítulo se presentan los resultados que se obtienen al aplicar el documentador a un sistema desarrollado en Pascal. El sistema que se utilizó para generar estos resultados fué el mismo documentador.

La razón de utilizar el documentador, como caso práctico, es debido a que el sistema:

1) Considera el uso de diferentes estructuras de datos (registros, arreglos, listas, arboles)

2) Se encuentra dividido en diferentes archivos.

3) Realiza la declaración de varios módulos dentro de otro. (módulos contenidos)

4) Contiene un número amplio de referencias entre módulos.

5) Identifica la función de cada módulo mediante su nombre.

6) Se encuentra comentarizado, lo cuál proporciona una visión más detallada de cada módulo cuando se realizan las impresiones editadas y numeradas.

7)Contiene llamadas recursivas y declaraciones forward,

es decir, proporciona los elementos necesarios para obtener una documentación lo bastante amplia para ilustrar el funcionamiento del documentador para pascal.

En las siguientes páginas se muestran los listados que proporcionó el documentador.

Estructura modular

```
CONTROL_SALIDA
 MENSAJE
 SETCURSOR
SETCURSOR
BOX
LOGOTIPO
 MENSAJE
MENU_PRINCIPAL
 DESPLIEGA_DISPOSITIVO
 SOMBRA
 BOX
 CONTROL_OPCIONES_MENU
  TECLAS_MENU
 BORRA_HEAP
 ANALIZAR_ESTRUCTURA
  GET_FLUJO_TOKEN
   SETCURSOR
   GET_ARCHIVO
    GET_NOMBRE_IDENTIFICACION
   MENSAJE
   OBTENER_TOKEN
    GET_CARACTER
    ARMAR_TOKEN
     GET_CARACTER
 CUENTA_CRLF
 SOMBRA
 BOX
 CREAR_PROCEDIMIENTOS
 ASIGNAR_PROCEDIMIENTOS
 NIVEL_ANIDAMIENTO
  GET_FLUJO_TOKEN
   SETCURSOR
   GET_ARCHIVO
    GET_NOMBRE_IDENTIFICACION
   MENSAJE
   OBTENER_TOKEN
    GET_CARACTER
    ARMAR_TOKEN
     GET_CARACTER
  BUSCA_MODULO_REFERENCIA
  CREAR_PROCEDIMIENTO_CONTENIDO
  ASIGNAR_PROCEDIMIENTOS_CONTENIDOS
   BUSCA_MODULO_REFERENCIA
  CREAR_PROCEDIMIENTOS
  ASIGNAR_PROCEDIMIENTOS
  NIVEL_ANIDAMIENTO  *
 MENSAJE
INICIA_COMENTARIOS
 GET_FLUJO_TOKEN
  SETCURSOR
  GET_ARCHIVO
   GET_NOMBRE_IDENTIFICACION
  MENSAJE
  OBTENER_TOKEN
   GET_CARACTER
   ARMAR_TOKEN
    GET_CARACTER
 SOMBRA
 BOX
 DETERMINAR_ESTRUCTURA  F
  GET_FLUJO_TOKEN
   SETCURSOR
   GET_ARCHIVO
    GET_NOMBRE_IDENTIFICACION
   MENSAJE
   OBTENER_TOKEN
    GET_CARACTER
    ARMAR_TOKEN
     GET_CARACTER
  LEE_SIGUIENTE
   GET_FLUJO_TOKEN
```

Estructura modular

```
    SETCURSOR
    GET_ARCHIVO
     GET_NOMBRE_IDENTIFICACION
    MENSAJE
    OBTENER_TOKEN
     GET_CARACTER
     ARMAR_TOKEN
      GET_CARACTER
  ANALIZAR_IF
   GET_FLUJO_TOKEN
    SETCURSOR
    GET_ARCHIVO
     GET_NOMBRE_IDENTIFICACION
    MENSAJE
    OBTENER_TOKEN
     GET_CARACTER
     ARMAR_TOKEN
      GET_CARACTER
   DETERMINAR_ESTRUCTURA  F
  ANALIZAR_CASE
   GET_FLUJO_TOKEN
    SETCURSOR
    GET_ARCHIVO
     GET_NOMBRE_IDENTIFICACION
    MENSAJE
    OBTENER_TOKEN
     GET_CARACTER
     ARMAR_TOKEN
      GET_CARACTER
   ANALIZAR_ESTRUCTURA_CASE
    DETERMINAR_ESTRUCTURA  F
    GET_FLUJO_TOKEN
     SETCURSOR
     GET_ARCHIVO
      GET_NOMBRE_IDENTIFICACION
     MENSAJE
     OBTENER_TOKEN
      GET_CARACTER
      ARMAR_TOKEN
       GET_CARACTER
   DETERMINAR_ESTRUCTURA  F
  ANALIZAR_FOR
   GET_FLUJO_TOKEN
    SETCURSOR
    GET_ARCHIVO
     GET_NOMBRE_IDENTIFICACION
    MENSAJE
    OBTENER_TOKEN
     GET_CARACTER
     ARMAR_TOKEN
      GET_CARACTER
   DETERMINAR_ESTRUCTURA  F
  ANALIZAR_REPEAT
   GET_FLUJO_TOKEN
    SETCURSOR
    GET_ARCHIVO
     GET_NOMBRE_IDENTIFICACION
    MENSAJE
    OBTENER_TOKEN
     GET_CARACTER
     ARMAR_TOKEN
      GET_CARACTER
   DETERMINAR_ESTRUCTURA  F
  ANALIZAR_WHILE
   GET_FLUJO_TOKEN
    SETCURSOR
    GET_ARCHIVO
     GET_NOMBRE_IDENTIFICACION
    MENSAJE
    OBTENER_TOKEN
```

Estructura modular

```
      GET_CARACTER
      ARMAR_TOKEN
       GET_CARACTER
    DETERMINAR_ESTRUCTURA  F
   ANALIZAR_BEGIN
    GET_FLUJO_TOKEN
     SETCURSOR
     GET_ARCHIVO
      GET_NOMBRE_IDENTIFICACION
     MENSAJE
     OBTENER_TOKEN
      GET_CARACTER
      ARMAR_TOKEN
       GET_CARACTER
    DETERMINAR_ESTRUCTURA  F
   ANALIZAR_WITH
    GET_FLUJO_TOKEN
     SETCURSOR
     GET_ARCHIVO
      GET_NOMBRE_IDENTIFICACION
     MENSAJE
     OBTENER_TOKEN
      GET_CARACTER
      ARMAR_TOKEN
       GET_CARACTER
    DETERMINAR_ESTRUCTURA  F
   ANALIZAR_RECORD
    GET_FLUJO_TOKEN
     SETCURSOR
     GET_ARCHIVO
      GET_NOMBRE_IDENTIFICACION
     MENSAJE
     OBTENER_TOKEN
      GET_CARACTER
      ARMAR_TOKEN
       GET_CARACTER
    ANALIZAR_CASE
     GET_FLUJO_TOKEN
      SETCURSOR
      GET_ARCHIVO
       GET_NOMBRE_IDENTIFICACION
      MENSAJE
      OBTENER_TOKEN
       GET_CARACTER
       ARMAR_TOKEN
        GET_CARACTER
     ANALIZAR_ESTRUCTURA_CASE
      DETERMINAR_ESTRUCTURA  F
      GET_FLUJO_TOKEN
       SETCURSOR
       GET_ARCHIVO
        GET_NOMBRE_IDENTIFICACION
       MENSAJE
       OBTENER_TOKEN
        GET_CARACTER
        ARMAR_TOKEN
         GET_CARACTER
      DETERMINAR_ESTRUCTURA  F
    DETERMINAR_ESTRUCTURA  F
  MENSAJE
 MENU_IMPRESION
  SOMBRA
  BOX
  CONTROL_OPCIONES_MENU
  TECLAS_MENU
 IMPRIMIR_MODULOS_PROGRAMA
  SELECCIONA_DISPOSITIVO
   SOMBRA
   BOX
   CONTROL_OPCIONES_MENU
```

Estructura modular

```
  TECLAS_MENU
  ISPRINTER
  MENSAJE
 CONTROL_PANTALLA
  CONFIGURACION_IMPRESORA
 CLRWINDOW
 BOX
 ENCABEZADO_MODULOS_PROGRAMA
  ENCABEZADO_IDENTIFICACION
  LLENA_BLANCOS
 LLENA_BLANCOS
 CONFIGURACION_IMPRESORA
 MENSAJE
IMPRIMIR_ALCANCE_MODULOS
 SELECCIONA_DISPOSITIVO
  SOMBRA
  BOX
  CONTROL_OPCIONES_MENU
   TECLAS_MENU
  ISPRINTER
  MENSAJE
 CONTROL_PANTALLA
  CONFIGURACION_IMPRESORA
 CLRWINDOW
 BOX
 ENCABEZADO_ALCANCE_MODULOS
  ENCABEZADO_IDENTIFICACION
  LLENA_BLANCOS
 LLENA_BLANCOS
 CONFIGURACION_IMPRESORA
 MENSAJE
IMPRIMIR_PROCEDIMIENTOS_CONTENIDO
 SELECCIONA_DISPOSITIVO
  SOMBRA
  BOX
  CONTROL_OPCIONES_MENU
   TECLAS_MENU
  ISPRINTER
  MENSAJE
 CONTROL_PANTALLA
  CONFIGURACION_IMPRESORA
 CLRWINDOW
 BOX
 ENCABEZADO_CONTENIDO
  ENCABEZADO_IDENTIFICACION
  LLENA_BLANCOS
 LLENA_BLANCOS
 CONFIGURACION_IMPRESORA
 MENSAJE
IMPRIMIR_ESTRUCTURA_MODULAR
 SELECCIONA_DISPOSITIVO
  SOMBRA
  BOX
  CONTROL_OPCIONES_MENU
   TECLAS_MENU
  ISPRINTER
  MENSAJE
 BUSCA_MODULO_REFERENCIA
 CONTROL_PANTALLA
  CONFIGURACION_IMPRESORA
 IMPRIME_JERARQUIA
  BUSCA_MODULO_REFERENCIA
  CLRWINDOW
  BOX
  ENCABEZADO_ESTRUCTURA_MODULAR
   ENCABEZADO_IDENTIFICACION
  LLENA_BLANCOS
  CONTROL_PANTALLA
   CONFIGURACION_IMPRESORA
 CONFIGURACION_IMPRESORA
 MENSAJE
```

Estructura modular

```
IMPRIMIR_SUBPROGRAMAS_PROGRAMA
 SELECCIONA_DISPOSITIVO
  SOMBRA
  BOX
  CONTROL_OPCIONES_MENU
   TECLAS_MENU
  ISPRINTER
  MENSAJE
 CLRWINDOW
 BOX
 ENCABEZADO_SUBPROGRAMAS_PROGRAMA
  ENCABEZADO_IDENTIFICACION
 CONTROL_PANTALLA
  CONFIGURACION_IMPRESORA
 CONFIGURACION_IMPRESORA
 MENSAJE
ISPRINTER
MENSAJE
IMPRIMIR_PROGRAMAS_NUMERADOS
 CONTROL_PANTALLA
  CONFIGURACION_IMPRESORA
 ENCABEZADO_PROGRAMAS_NUMERADOS
  ENCABEZADO_IDENTIFICACION
 CONFIGURACION_IMPRESORA
 MENSAJE
SETCURSOR
GET_ARCHIVO
 GET_NOMBRE_IDENTIFICACION
IMPRIMIR_PROCEDIMIENTOS_LLAMADOS
 SELECCIONA_DISPOSITIVO
  SOMBRA
  BOX
  CONTROL_OPCIONES_MENU
   TECLAS_MENU
   ISPRINTER
  MENSAJE
 CONTROL_PANTALLA
  CONFIGURACION_IMPRESORA
 IMPRIME_ENCABEZADO
  CLRWINDOW
  BOX
  ENCABEZADO_LLAMADO
   ENCABEZADO_IDENTIFICACION
   LLENA_BLANCOS
 LLENA_BLANCOS
 CONFIGURACION_IMPRESORA
 MENSAJE
IMPRIMIR_PROCEDIMIENTOS_LLAMADOR
 SELECCIONA_DISPOSITIVO
  SOMBRA
  BOX
  CONTROL_OPCIONES_MENU
   TECLAS_MENU
  ISPRINTER
  MENSAJE
 CONTROL_PANTALLA
  CONFIGURACION_IMPRESORA
 CLRWINDOW
 BOX
 ENCABEZADO_LLAMADOR
  ENCABEZADO_IDENTIFICACION
  LLENA_BLANCOS
 LLENA_BLANCOS
 CONFIGURACION_IMPRESORA
 MENSAJE
SELECCIONA_TIPO_IMPRESORA
 SOMBRA
 BOX
 CONTROL_OPCIONES_MENU
  TECLAS_MENU
MENSAJE
```

Módulos por programa

Nombre de la unidad	Tipo	Página	Programa(Archivo)	Llamados	Llamadores	Contenidos
ANALIZAR_BEGIN	P	8	ESTRUC5	2	1	0
ANALIZAR_CASE	P	3	ESTRUC5	3	2	0
ANALIZAR_ESTRUCTURA	P	10	ESTRUC3	8	1	0
ANALIZAR_ESTRUCTURA_CASE	P	1	ESTRUC5	2	1	0
ANALIZAR_FOR	P	6	ESTRUC5	2	1	0
ANALIZAR_IF	P	4	ESTRUC5	2	1	0
ANALIZAR_RECORD	P	10	ESTRUC5	3	1	0
ANALIZAR_REPEAT	P	9	ESTRUC5	2	1	0
ANALIZAR_WHILE	P	7	ESTRUC5	2	1	0
ANALIZAR_WITH	P	11	ESTRUC5	2	1	0
ARMAR_TOKEN	F	1	ESTRUC2	1	1	0
ASIGNAR_PROCEDIMIENTOS	P	7	ESTRUC8	0	2	0
ASIGNAR_PROCEDIMIENTOS_CONTENIDOS	P	4	ESTRUC8	1	1	0
ASIGNAR_PROCEDIMIENTOS_LLAMADORES	P	5	ESTRUC8	1	1	0
ASIGNAR_PROCEDIMIENTOS_LLAMADOS	P	6	ESTRUC8	1	1	0
ASIGNAR_SUBPROGRAMAS	P	6	ESTRUC8	0	1	0
BORRA_HEAP	P	10	ESTRUC7	0	1	0
BOX	P	3	ESTRUC9	0	14	0
BUSCA_MODULO_REFERENCIA	P	5	ESTRUC3	0	6	0
CLRWINDOW	P	3	ESTRUC4	0	7	0
CONFIGURACION_IMPRESORA	P	1	ESTRUC4	0	9	0
CONTROL_OPCIONES_MENU	F	5	ESTRUC9	1	4	0
CONTROL_PANTALLA	F	3	ESTRUC4	1	9	0
CONTROL_SALIDA	P	12	ESTRUC3	2	1	0
CREAR_PROCEDIMIENTOS	P	3	ESTRUC8	0	2	0
CREAR_PROCEDIMIENTO_CONTENIDO	P	2	ESTRUC8	0	1	0
CREAR_PROCEDIMIENTO_LLAMADO	P	1	ESTRUC8	0	1	0
CREAR_PROCEDIMIENTO_LLAMADOR	P	1	ESTRUC8	0	1	0
CREAR_SUBPROGRAMAS	P	2	ESTRUC8	0	1	0
CUENTA_CRLF	F	3	ESTRUC9	0	2	0
DESPLIEGA_DISPOSITIVO	P	5	ESTRUC6	0	1	0
DETERMINAR_ESTRUCTURA	P	13	ESTRUC5	10	10	0
DOCUMENTADOR_PASCAL	G	1	ESTRUC1	6	0	62
ENCABEZADO_ALCANCE_MODULOS	P	12	ESTRUC4	2	1	0
ENCABEZADO_CONTENIDO	P	4	ESTRUC7	2	1	0

Módulos por programa

Nombre de la unidad	Tipo	Página	Programa(Archivo)	Llamados	Llamadores	Contenidos
ENCABEZADO_ESTRUCTURA_MODULAR	P	17	ESTRUC4	1	1	0
ENCABEZADO_IDENTIFICACION	P	2	ESTRUC4	0	8	0
ENCABEZADO_LLAMADO	P	7	ESTRUC7	2	1	0
ENCABEZADO_LLAMADOR	P	14	ESTRUC4	2	1	0
ENCABEZADO_MODULOS_PROGRAMA	P	9	ESTRUC4	2	1	0
ENCABEZADO_PROGRAMAS_NUMERADOS	P	5	ESTRUC4	1	1	0
ENCABEZADO_SUBPROGRAMAS_PROGRAMA	P	6	ESTRUC4	1	1	0
GET_ARCHIVO	P	11	ESTRUC2	1	3	0
GET_CARACTER	P	1	ESTRUC2	0	2	0
GET_FLUJO_TOKEN	P	12	ESTRUC2	4	14	0
GET_NOMBRE_IDENTIFICACION	P	10	ESTRUC2	0	1	0
IMPRIME_ENCABEZADO	P	8	ESTRUC7	3	1	0
IMPRIME_JERARQUIA	P	1	ESTRUC7	6	1	0
IMPRIMIR_ALCANCE_MODULOS	P	11	ESTRUC4	8	1	1
IMPRIMIR_ESTRUCTURA_MODULAR	P	3	ESTRUC7	6	1	0
IMPRIMIR_MODULOS_PROGRAMA	P	9	ESTRUC4	8	1	1
IMPRIMIR_PROCEDIMIENTOS_CONTENIDO	P	4	ESTRUC7	8	1	1
IMPRIMIR_PROCEDIMIENTOS_LLAMADOR	P	14	ESTRUC4	8	1	1
IMPRIMIR_PROCEDIMIENTOS_LLAMADOS	P	7	ESTRUC7	6	1	2
IMPRIMIR_PROGRAMAS_NUMERADOS	P	4	ESTRUC4	4	1	1
IMPRIMIR_SUBPROGRAMAS_PROGRAMA	P	6	ESTRUC4	7	1	1
INICIA_COMENTARIOS	P	16	ESTRUC5	5	1	0
ISPRINTER	F	1	ESTRUC9	0	2	0
LEE_SIGUIENTE	P	12	ESTRUC5	1	1	0
LLENA_BLANCOS	P	1	ESTRUC4	0	11	0
LOGOTIPO	P	1	ESTRUC6	1	1	0
MENSAJE	P	2	ESTRUC9	0	16	0
MENU_IMPRESION	P	2	ESTRUC6	15	1	0
MENU_PRINCIPAL	P	6	ESTRUC6	9	1	0
NIVEL_ANIDAMIENTO	P	6	ESTRUC1	14	2	0
OBTENER_TOKEN	P	11	ESTRUC2	2	1	0
SELECCIONA_DISPOSITIVO	P	5	ESTRUC9	5	7	0
SELECCIONA_TIPO_IMPRESORA	P	5	ESTRUC6	3	1	0
SETCURSOR	P	6	ESTRUC3	0	4	0
SOMBRA	P	1	ESTRUC9	0	6	0
TECLAS_MENU	P	4	ESTRUC9	0	1	0

Nombre de la unidad	Alcance de módulos De la línea	A la línea	Sec. eje.	Programa
ANALIZAR_BEGIN	524	559	525	ESTRUC5
ANALIZAR_CASE	119	226	147	ESTRUC5
ANALIZAR_ESTRUCTURA	639	714	640	ESTRUC3
ANALIZAR_ESTRUCTURA_CASE	27	113	36	ESTRUC5
ANALIZAR_FOR	367	420	370	ESTRUC5
ANALIZAR_IF	251	343	255	ESTRUC5
ANALIZAR_RECORD	643	690	646	ESTRUC5
ANALIZAR_REPEAT	583	619	584	ESTRUC5
ANALIZAR_WHILE	444	500	447	ESTRUC5
ANALIZAR_WITH	714	768	717	ESTRUC5
ARMAR_TOKEN	61	615	121	ESTRUC2
ASIGNAR_PROCEDIMIENTOS	456	497	460	ESTRUC8
ASIGNAR_PROCEDIMIENTOS_CONTENIDOS	199	248	201	ESTRUC8
ASIGNAR_PROCEDIMIENTOS_LLAMADORES	273	322	277	ESTRUC8
ASIGNAR_PROCEDIMIENTOS_LLAMADOS	347	364	348	ESTRUC8
ASIGNAR_SUBPROGRAMAS	387	432	391	ESTRUC8
BORRA_HEAP	615	655	621	ESTRUC7
BOX	133	161	137	ESTRUC9
BUSCA_MODULO_REFERENCIA	295	309	299	ESTRUC3
CLRWINDOW	151	160	154	ESTRUC4
CONFIGURACION_IMPRESORA	60	66	61	ESTRUC4
CONTROL_OPCIONES_MENU	272	301	276	ESTRUC9
CONTROL_PANTALLA	169	223	192	ESTRUC4
CONTROL_SALIDA	741	870	742	ESTRUC3
CREAR_PROCEDIMIENTOS	155	174	156	ESTRUC8
CREAR_PROCEDIMIENTO_CONTENIDO	127	132	128	ESTRUC8
CREAR_PROCEDIMIENTO_LLAMADO	57	67	59	ESTRUC8
CREAR_PROCEDIMIENTO_LLAMADOR	23	33	25	ESTRUC8
CREAR_SUBPROGRAMAS	89	105	94	ESTRUC8
CUENTA_CRLF	180	191	183	ESTRUC9
DESPLIEGA_DISPOSITIVO	274	292	277	ESTRUC6
DETERMINAR_ESTRUCTURA	824	1033	834	ESTRUC5
DOCUMENTADOR_PASCAL	42	900	873	ESTRUC3
ENCABEZADO_ALCANCE_MODULOS	750	794	751	ESTRUC4
ENCABEZADO_CONTENIDO	230	256	231	ESTRUC7

Nombre de la unidad	Alcance de módulos De la línea	A la línea	Sec. eje.	Programa
ENCABEZADO_ESTRUCTURA_MODULAR	1087	1104	1088	ESTRUC6
ENCABEZADO_IDENTIFICACION	88	126	99	ESTRUC6
ENCABEZADO_LLAMADO	437	463	438	ESTRUC7
ENCABEZADO_LLAMADOR	920	946	921	ESTRUC6
ENCABEZADO_MODULOS_PROGRAMA	582	626	583	ESTRUC6
ENCABEZADO_PROGRAMAS_NUMERADOS	270	284	271	ESTRUC6
ENCABEZADO_SUBPROGRAMAS_PROGRAMA	367	400	368	ESTRUC6
GET_ARCHIVO	662	675	666	ESTRUC2
GET_CARACTER	24	33	25	ESTRUC2
GET_FLUJO_TOKEN	728	795	732	ESTRUC2
GET_NOMBRE_IDENTIFICACION	634	643	636	ESTRUC2
IMPRIME_ENCABEZADO	477	494	478	ESTRUC7
IMPRIME_JERARQUIA	25	127	33	ESTRUC7
IMPRIMIR_ALCANCE_MODULOS	725	870	736	ESTRUC6
IMPRIMIR_ESTRUCTURA_MODULAR	149	180	155	ESTRUC7
IMPRIMIR_MODULOS_PROGRAMA	557	700	628	ESTRUC6
IMPRIMIR_PROCEDIMIENTOS_CONTENIDO	204	371	258	ESTRUC7
IMPRIMIR_PROCEDIMIENTOS_LLAMADOR	894	1066	947	ESTRUC6
IMPRIMIR_PROCEDIMIENTOS_LLAMADOS	404	600	496	ESTRUC7
IMPRIMIR_PROGRAMAS_NUMERADOS	245	315	286	ESTRUC6
IMPRIMIR_SUBPROGRAMAS_PROGRAMA	342	532	402	ESTRUC6
INICIA_COMENTARIOS	1055	1085	1056	ESTRUC5
ISPRINTER	17	29	18	ESTRUC9
LEE_SIGUIENTE	791	801	792	ESTRUC5
LLENA_BLANCOS	26	37	29	ESTRUC4
LOGOTIPO	21	67	32	ESTRUC6
MENSAJE	83	111	87	ESTRUC9
MENU_IMPRESION	86	254	92	ESTRUC6
MENU_PRINCIPAL	385	484	390	ESTRUC6
NIVEL_ANIDAMIENTO	381	619	398	ESTRUC3
OBTENER_TOKEN	697	701	698	ESTRUC2
SELECCIONA_DISPOSITIVO	321	386	326	ESTRUC9
SELECCIONA_TIPO_IMPRESORA	316	366	321	ESTRUC6
SETCURSOR	330	341	333	ESTRUC3
SOMBRA	52	62	55	ESTRUC9
TECLAS_MENU	212	246	216	ESTRUC9

Módulo	Módulo va. Módulo contenido	Módulo contenido
ANALIZAR_BEGIN		
ANALIZAR_CASE		
ANALIZAR_ESTRUCTURA		
ANALIZAR_ESTRUCTURA_CASE		
ANALIZAR_FOR		
ANALIZAR_IF		
ANALIZAR_RECORD		
ANALIZAR_REPEAT		
ANALIZAR_WHILE		
ANALIZAR_WITH		
ARMAR_TOKEN		
ASIGNAR_PROCEDIMIENTOS		
ASIGNAR_PROCEDIMIENTOS_CONTENIDOS		
ASIGNAR_PROCEDIMIENTOS_LLAMADORES		
ASIGNAR_PROCEDIMIENTOS_LLAMADOS		
ASIGNAR_SUBPROGRAMAS		
BORRA_HEAP		
BOX		
BUSCA_MODULO_REFERENCIA		
CLRWINDOW		
CONFIGURACION_IMPRESORA		
CONTROL_OPCIONES_MENU		
CONTROL_PANTALLA		
CONTROL_SALIDA		
CREAR_PROCEDIMIENTOS		
CREAR_PROCEDIMIENTO_CONTENIDO		
CREAR_PROCEDIMIENTO_LLAMADO		
CREAR_PROCEDIMIENTO_LLAMADOR		
CREAR_SUBPROGRAMAS		
CUENTA_CRLF		
DESPLIEGA_DISPOSITIVO		
DETERMINAR_ESTRUCTUTA		
DOCUMENTADOR_PASCAL		ANALIZAR_BEGIN ANALIZAR_CASE ANALIZAR_ESTRUCTURA ANALIZAR_ESTRUCTURA_CASE ANALIZAR_FOR ANALIZAR_IF

[illegible]

Módulo	Módulos llamados
[illegible]	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	[illegible]
	CONFIGURACION_IMPRESORA
	CONTROL_VENTANA_MENU
	CONTROL_PANTALLA
	CONTROL_SALIDA
	CREAR_PROCEDIMIENTOS
	CREAR_PROCEDIMIENTOS_CONTENIDOS
	CREAR_PROCEDIMIENTO_LLAMADO
	CREAR_PROCEDIMIENTOS_LLAMADOS
	CREAR_SUBPROGRAMAS
	CUENTA_CRLF
	DESPLIEGA_DISPOSITIVO
	DETERMINAR_ESTRUCTURA
	ENCABEZADO_ESTRUCTURA_MODULAR
	ENCABEZADO_IDENTIFICACION
	GET_ARCHIVO
	GET_CARACTER
	GET_FILENO_TOKEN
	GET_NOMBRE_IDENTIFICACION
	IMPRIMIR_JERARQUIA
	IMPRIMIR_ALCANCE_MODULOS
	IMPRIMIR_ESTRUCTURA_MODULAR
	IMPRIMIR_MODULOS_PROGRAMA
	IMPRIMIR_PROCEDIMIENTOS_CONTENIDOS
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOS
	IMPRIMIR_PROCEDIMIENTO_LLAMADO
	IMPRIMIR_PROGRAMAS_NUMERADOS
	IMPRIMIR_SUBPROGRAMAS_PROGRAMA
	INICIA_COMENTARIOS
	IMPRIMIENDO
	LEE_SIGUIENTE
	LLENA_BLANCOS
	LOGOTIPO
	MENSAJE
	MENU_IMPRESION
	MENU_PRINCIPAL
	NIVEL_ANIDAMIENTO
	OBTIENE_TOKEN
	SELECCIONA_DISPOSITIVO
	SELECCIONA_TIPO_IMPRESORA
	[illegible]
	VENTANA
	TECLAS_MENU
ENCABEZADO_ALCANCE_MODULOS	
ENCABEZADO_CONTENIDO	
ENCABEZADO_ESTRUCTURA_MODULAR	
ENCABEZADO_IDENTIFICACION	
ENCABEZADO_LLAMADO	
ENCABEZADO_LLAMADOS	
ENCABEZADO_MODULOS_PROGRAMA	

Módulo vs. Módulo contenido

Módulo	Módulo contenido
ENCABEZADO_PROGRAMAS_NUMERADOS	
ENCABEZADO_SUBPROGRAMAS_PROGRAMA	
GET_ARCHIVO	
GET_CARACTER	
GET_FLUJO_TOKEN	
GET_NOMBRE_IDENTIFICACION	
IMPRIME_ENCABEZADO	
IMPRIME_JERARQUIA	
IMPRIMIR_ALCANCE_MODULOS	ENCABEZADO_ALCANCE_MODULOS
IMPRIMIR_ESTRUCTURA_MODULAR	
IMPRIMIR_MODULOS_PROGRAMA	ENCABEZADO_MODULOS_PROGRAMA
IMPRIMIR_PROCEDIMIENTOS_CONTENIDO	ENCABEZADO_CONTENIDO
IMPRIMIR_PROCEDIMIENTOS_LLAMADOR	ENCABEZADO_LLAMADOR
IMPRIMIR_PROCEDIMIENTOS_LLAMADOS	ENCABEZADO_LLAMADO IMPRIME_ENCABEZADO
IMPRIMIR_PROGRAMAS_NUMERADOS	ENCABEZADO_PROGRAMAS_NUMERADOS
IMPRIMIR_SUBPROGRAMAS_PROGRAMA	ENCABEZADO_SUBPROGRAMAS_PROGRAMA
INICIA_COMENTARIOS	
IBPRINTER	
LEE_SIGUIENTE	
LLENA_BLANCOS	
LOGOTIPO	
MENSAJE	
MENU_IMPRESION	
MENU_PRINCIPAL	
NIVEL_ANIDAMIENTO	
OBTENER_TOKEN	
SELECCIONA_DISPOSITIVO	
SELECCIONA_TIPO_IMPRESORA	
SETCURSOR	
SOMBRA	
TECLAS_MENU	

Archivo	Tamaño	Subprogramas por programa	Fecha ultima actualización (DD/MM/AAAA)	
* ESTRUC3	46342	22	10	1993
ESTRUC2.PAS	31137	4	10	1993
ESTRUC4.PAS	50275	17	10	1993
ESTRUC5.PAS	47648	29	9	1993
ESTRUC6.PAS	18972	17	10	1993
ESTRUC7.PAS	28229	17	10	1993
ESTRUC8.PAS	28336	30	9	1993
ESTRUC9.PAS	17519	4	10	1993

Módulo llamador vs. Módulo llamado

Módulo	Módulos que llama
ANALIZAR_BEGIN	DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ANALIZAR_CASE	ANALIZAR_ESTRUCTURA_CASE DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ANALIZAR_ESTRUCTURA	ASIGNAR_PROCEDIMIENTOS BOX CREAR_PROCEDIMIENTOS CUENTA_CRLF GET_FLUJO_TOKEN MENSAJE NIVEL_ANIDAMIENTO SOMBRA
ANALIZAR_ESTRUCTURA_CASE	DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ANALIZAR_FOR	DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ANALIZAR_IF	DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ANALIZAR_RECORD	ANALIZAR_CASE DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ANALIZAR_REPEAT	DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ANALIZAR_WHILE	DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ANALIZAR_WITH	DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN
ARMAR_TOKEN	GET_CARACTER
ASIGNAR_PROCEDIMIENTOS	
ASIGNAR_PROCEDIMIENTOS_CONTENIDOS	BUSCA_MODULO_REFERENCIA
ASIGNAR_PROCEDIMIENTOS_LLAMADORES	BUSCA_MODULO_REFERENCIA
ASIGNAR_PROCEDIMIENTOS_LLAMADOS	BUSCA_MODULO_REFERENCIA
ASIGNAR_SUBPROGRAMAS	
BORRA_HEAP	
BOX	
BUSCA_MODULO_REFERENCIA	
CLRWINDOW	
CONFIGURACION_IMPRESORA	
CONTROL_OPCIONES_MENU	TECLAS_MENU
CONTROL_PANTALLA	CONFIGURACION_IMPRESORA
CONTROL_SALIDA	MENSAJE SETCURSOR
CREAR_PROCEDIMIENTOS	
CREAR_PROCEDIMIENTO_CONTENIDO	

Módulo llamador vs. Módulo llamado

Módulo	Módulos que llama
CREAR_PROCEDIMIENTO_LLAMADO	
CREAR_PROCEDIMIENTO_LLAMADOR	
CREAR_SUBPROGRAMAS	
CUENTA_CRLF	
DESPLIEGA_DISPOSITIVO	
DETERMINAR_ESTRUCTURA	ANALIZAR_BEGIN ANALIZAR_CASE ANALIZAR_FOR ANALIZAR_IF ANALIZAR_RECORD ANALIZAR_REPEAT ANALIZAR_WHILE ANALIZAR_WITH GET_FLUJO_TOKEN LEE_SIGUIENTE
DOCUMENTADOR_PASCAL	BOX CONTROL_SALIDA LOGOTIPO MENSAJE MENU_PRINCIPAL SETCURSOR
ENCABEZADO_ALCANCE_MODULOS	ENCABEZADO_IDENTIFICACION LLENA_BLANCOS
ENCABEZADO_CONTENIDO	ENCABEZADO_IDENTIFICACIGN LLENA_BLANCOS
ENCABEZADO_ESTRUCTURA_MODULAR	ENCABEZADO_IDENTIFICACION
ENCABEZADO_IDENTIFICACION	
ENCABEZADO_LLAMADO	ENCABEZADO_IDENTIFICACION LLENA_BLANCOS
ENCABEZADO_LLAMADOR	ENCABEZADO_IDENTIFICACION LLENA_BLANCOS
ENCABEZADO_MODULOS_PROGRAMA	ENCABEZADO_IDENTIFICACION LLENA_BLANCOS
ENCABEZADO_PROGRAMAS_NUMERADOS	ENCABEZADO_IDENTIFICACION
ENCABEZADO_SUBPROGRAMAS_PROGRAMA	ENCABEZADO_IDENTIFICACION
GET_ARCHIVO	GET_NOMBRE_IDENTIFICACION
GET_CARACTER	
GET_FLUJO_TOKEN	GET_ARCHIVO MENSAJE OBTENER_TOKEN SETCURSOR
GET_NOMBRE_IDENTIFICACION	
IMPRIME_ENCABEZADO	BOX CLRWINDOW ENCABEZADO_LLAMADO
IMPRIME_JERARQUIA	BOX BUSCA_MODULO_REFERENCIA CLRWINDOW CONTROL_PANTALLA

Módulo llamador vs. Módulo llamado

Módulo	Módulos que llama
IMPRIME_JERARQUIA	ENCABEZADO_ESTRUCTURA_MODULAR LLENA_BLANCOS
IMPRIMIR_ALCANCE_MODULOS	BOX CLRWINDOW CONFIGURACION_IMPRESORA CONTROL_PANTALLA ENCABEZADO_ALCANCE_MODULOS LLENA_BLANCOS MENSAJE SELECCIONA_DISPOSITIVO
IMPRIMIR_ESTRUCTURA_MODULAR	BUSCA_MODULO_REFERENCIA CONFIGURACION_IMPRESORA CONTROL_PANTALLA IMPRIME_JERARQUIA MENSAJE SELECCIONA_DISPOSITIVO
IMPRIMIR_MODULOS_PROGRAMA	BOX CLRWINDOW CONFIGURACION_IMPRESORA CONTROL_PANTALLA ENCABEZADO_MODULOS_PROGRAMA LLENA_BLANCOS MENSAJE SELECCIONA_DISPOSITIVO
IMPRIMIR_PROCEDIMIENTOS_CONTENIDO	BOX CLRWINDOW CONFIGURACION_IMPRESORA CONTROL_PANTALLA ENCABEZADO_CONTENIDO LLENA_BLANCOS MENSAJE SELECCIONA_DISPOSITIVO
IMPRIMIR_PROCEDIMIENTOS_LLAMADOR	BOX CLRWINDOW CONFIGURACION_IMPRESORA CONTROL_PANTALLA ENCABEZADO_LLAMADOR LLENA_BLANCOS MENSAJE SELECCIONA_DISPOSITIVO
IMPRIMIR_PROCEDIMIENTOS_LLAMADOS	CONFIGURACION_IMPRESORA CONTROL_PANTALLA IMPRIME_ENCABEZADO LLENA_BLANCOS MENSAJE SELECCIONA_DISPOSITIVO
IMPRIMIR_PROGRAMAS_NUMERADOS	CONFIGURACION_IMPRESORA CONTROL_PANTALLA ENCABEZADO_PROGRAMAS_NUMERADOS MENSAJE
IMPRIMIR_SUBPROGRAMAS_PROGRAMA	BOX CLRWINDOW CONFIGURACION_IMPRESORA CONTROL_PANTALLA ENCABEZADO_SUBPROGRAMAS_PROGRAMA MENSAJE SELECCIONA_DISPOSITIVO
INICIA_COMENTARIOS	BOX DETERMINAR_ESTRUCTURA GET_FLUJO_TOKEN MENSAJE

Módulo llamador vs. Módulo llamado

Módulo	Módulos que llama
INICIA_COMENTARIOS	SOMBRA
ISPRINTER	
LEE_SIGUIENTE	GET_FLUJO_TOKEN
LLENA_BLANCOS	
LOGOTIPO	MENSAJE
MENSAJE	
MENU_IMPRESION	BOX CONTROL_OPCIONES_MENU GET_ARCHIVO IMPRIMIR_ALCANCE_MODULOS IMPRIMIR_ESTRUCTURA_MODULAR IMPRIMIR_MODULOS_PROGRAMA IMPRIMIR_PROCEDIMIENTOS_CONTENIDO IMPRIMIR_PROCEDIMIENTOS_LLAMADOR IMPRIMIR_PROCEDIMIENTOS_LLAMADOS IMPRIMIR_PROGRAMAS_NUMERADOS IMPRIMIR_SUBPROGRAMAS_PROGRAMA ISPRINTER MENSAJE SETCURSOR SOMBRA
MENU_PRINCIPAL	ANALIZAR_ESTRUCTURA BORRA_HEAP BOX CONTROL_OPCIONES_MENU DESPLIEGA_DISPOSITIVO INICIA_COMENTARIOS MENU_IMPRESION SELECCIONA_TIPO_IMPRESORA SOMBRA
NIVEL_ANIDAMIENTO	ASIGNAR_PROCEDIMIENTOS ASIGNAR_PROCEDIMIENTOS_CONTENIDOS ASIGNAR_PROCEDIMIENTOS_LLAMADOR ASIGNAR_PROCEDIMIENTOS_LLAMADOS ASIGNAR_SUBPROGRAMAS BUSCA_MODULO_REFERENCIA CREAR_PROCEDIMIENTOS CREAR_PROCEDIMIENTO_CONTENIDO CREAR_PROCEDIMIENTO_LLAMADO CREAR_PROCEDIMIENTO_LLAMADOR CREAR_SUBPROGRAMAS CUENTA_CRLF GET_FLUJO_TOKEN NIVEL_ANIDAMIENTO
OBTENER_TOKEN	ARMAR_TOKEN GET_CARACTER
SELECCIONA_DISPOSITIVO	BOX CONTROL_OPCIONES_MENU ISPRINTER MENSAJE SOMBRA
SELECCIONA_TIPO_IMPRESORA	BOX CONTROL_OPCIONES_MENU SOMBRA
SETCURSOR	
SOMBRA	
TECLAS_MENU	

Módulo	Módulo llamador
ANALIZAR_BEGIN	DETERMINAR_ESTRUCTURA
ANALIZAR_CASE	ANALIZAR_RECORD DETERMINAR_ESTRUCTURA
ANALIZAR_ESTRUCTURA	MENU_PRINCIPAL
ANALIZAR_ESTRUCTURA_CASE	ANALIZAR_CASE
ANALIZAR_FOR	DETERMINAR_ESTRUCTURA
ANALIZAR_IF	DETERMINAR_ESTRUCTURA
ANALIZAR_RECORD	DETERMINAR_ESTRUCTURA
ANALIZAR_REPEAT	DETERMINAR_ESTRUCTURA
ANALIZAR_WHILE	DETERMINAR_ESTRUCTURA
ANALIZAR_WITH	DETERMINAR_ESTRUCTURA
ARMAR_TOKEN	OBTENER_TOKEN
ASIGNAR_PROCEDIMIENTOS	ANALIZAR_ESTRUCTURA NIVEL_ANIDAMIENTO
ASIGNAR_PROCEDIMIENTOS_CONTENIDOS	NIVEL_ANIDAMIENTO
ASIGNAR_PROCEDIMIENTOS_LLAMADORES	NIVEL_ANIDAMIENTO
ASIGNAR_PROCEDIMIENTOS_LLAMADOS	NIVEL_ANIDAMIENTO
ASIGNAR_SUBPROGRAMAS	NIVEL_ANIDAMIENTO
BORRA_HEAP	MENU_PRINCIPAL
BOX	ANALIZAR_ESTRUCTURA DOCUMENTADOR_PASCAL IMPRIME_ENCABEZADO IMPRIME_JERARQUIA IMPRIMIR_ALCANCE_MODULOS IMPRIMIR_MODULOS_PROGRAMA IMPRIMIR_PROCEDIMIENTOS_CONTENIDO IMPRIMIR_PROCEDIMIENTOS_LLAMADOR IMPRIMIR_SUBPROGRAMAS_PROGRAMA INICIA_COMENTARIOS MENU_IMPRESION MENU_PRINCIPAL SELECCIONA_DISPOSITIVO SELECCIONA_TIPO_IMPRESORA
BUSCA_MODULO_REFERENCIA	ASIGNAR_PROCEDIMIENTOS_CONTENIDOS ASIGNAR_PROCEDIMIENTOS_LLAMADORES ASIGNAR_PROCEDIMIENTOS_LLAMADOS IMPRIMIR_ESTRUCTURA_MODULAR NIVEL_ANIDAMIENTO
CLRWINDOW	IMPRIME_ENCABEZADO IMPRIME_JERARQUIA IMPRIMIR_ALCANCE_MODULOS IMPRIMIR_MODULOS_PROGRAMA IMPRIMIR_PROCEDIMIENTOS_CONTENIDO IMPRIMIR_PROCEDIMIENTOS_LLAMADOR IMPRIMIR_SUBPROGRAMAS_PROGRAMA
CONFIGURACION_IMPRESORA	CONTROL_PANTALLA IMPRIMIR_ALCANCE_MODULOS IMPRIMIR_ESTRUCTURA_MODULAR IMPRIME_JERARQUIA

Módulo vs. Módulo llamador

Módulo	Módulo llamador
CONFIGURACION_IMPRESORA	IMPRIMIR_MODULOS_PROGRAMA
	IMPRIMIR_PROCEDIMIENTOS_CONTENIDO
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOR
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOS
	IMPRIMIR_PROGRAMAS_NUMERADOS
	IMPRIMIR_SUBPROGRAMAS_PROGRAMA
CONTROL_OPCIONES_MENU	MENU_IMPRESION
	MENU_PRINCIPAL
	&SELECCIONA_DISPOSITIVO
	&SELECCIONA_TIPO_IMPRESORA
CONTROL_PANTALLA	IMPRIME_JERARQUIA
	IMPRIMIR_ALCANCE_MODULOS
	IMPRIMIR_ESTRUCTURA_MODULAR
	IMPRIMIR_MODULOS_PROGRAMA
	IMPRIMIR_PROCEDIMIENTOS_CONTENIDO
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOR
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOS
	IMPRIMIR_PROGRAMAS_NUMERADOS
	IMPRIMIR_SUBPROGRAMAS_PROGRAMA
CONTROL_SALIDA	DOCUMENTADOR_PASCAL
CREAR_PROCEDIMIENTOS	ANALIZAR_ESTRUCTURA
	NIVEL_ANIDAMIENTO
CREAR_PROCEDIMIENTO_CONTENIDO	NIVEL_ANIDAMIENTO
CREAR_PROCEDIMIENTO_LLAMADO	NIVEL_ANIDAMIENTO
CREAR_PROCEDIMIENTO_LLAMADOR	NIVEL_ANIDAMIENTO
CREAR_SUBPROGRAMAS	NIVEL_ANIDAMIENTO
CUENTA_CRLF	ANALIZAR_ESTRUCTURA
	NIVEL_ANIDAMIENTO
DESPLIEGA_DISPOSITIVO	MENU_PRINCIPAL
DETERMINAR_ESTRUCTURA	ANALIZAR_BEGIN
	ANALIZAR_CASE
	ANALIZAR_ESTRUCTURA_CASE
	ANALIZAR_FOR
	ANALIZAR_IF
	ANALIZAR_RECORD
	ANALIZAR_REPEAT
	ANALIZAR_WHILE
	ANALIZAR_WITH
	INICIA_COMENTARIOS
DOCUMENTADOR_PASCAL	
ENCABEZADO_ALCANCE_MODULOS	IMPRIMIR_ALCANCE_MODULOS
ENCABEZADO_CONTENIDO	IMPRIMIR_PROCEDIMIENTOS_CONTENIDO
ENCABEZADO_ESTRUCTURA_MODULAR	IMPRIME_JERARQUIA
ENCABEZADO_IDENTIFICACION	ENCABEZADO_ALCANCE_MODULOS
	ENCABEZADO_CONTENIDO
	ENCABEZADO_ESTRUCTURA_MODULAR
	ENCABEZADO_LLAMADO
	ENCABEZADO_LLAMADOR
	ENCABEZADO_MODULOS_PROGRAMA
	ENCABEZADO_PROGRAMAS_NUMERADOS
	ENCABEZADO_SUBPROGRAMAS_PROGRAMA
ENCABEZADO_LLAMADO	IMPRIME_ENCABEZADO

Módulo vs. Módulo llamador

Módulo	Módulo llamador
ENCABEZADO_LLAMADOR	IMPRIMIR_PROCEDIMIENTOS_LLAMADOR
ENCABEZADO_MODULOS_PROGRAMA	IMPRIMIR_MODULOS_PROGRAMA
ENCABEZADO_PROGRAMAS_NUMERADOS	IMPRIMIR_PROGRAMAS_NUMERADOS
ENCABEZADO_SUBPROGRAMAS_PROGRAMA	IMPRIMIR_SUBPROGRAMAS_PROGRAMA
GET_ARCHIVO	GET_FLUJO_TOKEN MENU_IMPRESION
GET_CARACTER	ARMAR_TOKEN OBTENER_TOKEN
GET_FLUJO_TOKEN	ANALIZAR_BEGIN ANALIZAR_CASE ANALIZAR_ESTRUCTURA ANALIZAR_ESTRUCTURA_CASE ANALIZAR_FOR ANALIZAR_IF ANALIZAR_RECORD ANALIZAR_REPEAT ANALIZAR_WHILE ANALIZAR_WITH DETERMINAR_ESTRUCTURA INICIA_COMENTARIOS LEE_SIGUIENTE NIVEL_ANIDAMIENTO
GET_NOMBRE_IDENTIFICACION	GET_ARCHIVO
IMPRIME_ENCABEZADO	IMPRIMIR_PROCEDIMIENTOS_LLAMADOS
IMPRIME_JERARQUIA	IMPRIMIR_ESTRUCTURA_MODULAR
IMPRIMIR_ALCANCE_MODULOS	MENU_IMPRESION
IMPRIMIR_ESTRUCTURA_MODULAR	MENU_IMPRESION
IMPRIMIR_MODULOS_PROGRAMA	MENU_IMPRESION
IMPRIMIR_PROCEDIMIENTOS_CONTENIDO	MENU_IMPRESION
IMPRIMIR_PROCEDIMIENTOS_LLAMADOR	MENU_IMPRESION
IMPRIMIR_PROCEDIMIENTOS_LLAMADOS	MENU_IMPRESION
IMPRIMIR_PROGRAMAS_NUMERADOS	MENU_IMPRESION
IMPRIMIR_SUBPROGRAMAS_PROGRAMA	MENU_IMPRESION
INICIA_COMENTARIOS	MENU_PRINCIPAL
ISPRINTER	MENU_IMPRESION SELECCIONA_DISPOSITIVO
LEE_SIGUIENTE	DETERMINAR_ESTRUCTURA
LLENA_BLANCOS	ENCABEZADO_ALCANCE_MODULOS ENCABEZADO_CONTENIDO ENCABEZADO_LLAMADO ENCABEZADO_LLAMADOR ENCABEZADO_MODULOS_PROGRAMA IMPRIME_JERARQUIA IMPRIMIR_ALCANCE_MODULOS IMPRIMIR_MODULOS_PROGRAMA IMPRIMIR_PROCEDIMIENTOS_CONTENIDO IMPRIMIR_PROCEDIMIENTOS_LLAMADOR IMPRIMIR_PROCEDIMIENTOS_LLAMADOS

Módulo vs. Módulo llamador

Módulo	Módulo llamador
LOGOTIPO	DOCUMENTADOR_PASCAL
MENSAJE	ANALIZAR_ESTRUCTURA
	CONTROL_SALIDA
	DOCUMENTADOR_PASCAL
	GET_FLUJO_TOKEN
	IMPRIMIR_ALCANCE_MODULOS
	IMPRIMIR_ESTRUCTURA_MODULAR
	IMPRIMIR_MODULOS_PROGRAMA
	IMPRIMIR_PROCEDIMIENTOS_CONTENIDO
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOR
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOS
	IMPRIMIR_PROGRAMAS_NUMERADOS
	IMPRIMIR_SUBPROGRAMAS_PROGRAMA
	INICIA_COMENTARIOS
	LOGOTIPO
	MENU_IMPRESION
	SELECCIONA_DISPOSITIVO
MENU_IMPRESION	MENU_PRINCIPAL
MENU_PRINCIPAL	DOCUMENTADOR_PASCAL
NIVEL_ANIDAMIENTO	ANALIZAR_ESTRUCTURA
	NIVEL_ANIDAMIENTO
OBTENER_TOKEN	GET_FLUJO_TOKEN
SELECCIONA_DISPOSITIVO	IMPRIMIR_ALCANCE_MODULOS
	IMPRIMIR_ESTRUCTURA_MODULAR
	IMPRIMIR_MODULOS_PROGRAMA
	IMPRIMIR_PROCEDIMIENTOS_CONTENIDO
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOR
	IMPRIMIR_PROCEDIMIENTOS_LLAMADOS
	IMPRIMIR_SUBPROGRAMAS_PROGRAMA
SELECCIONA_TIPO_IMPRESORA	MENU_PRINCIPAL
SETCURSOR	CONTROL_SALIDA
	DOCUMENTADOR_PASCAL
	GET_FLUJO_TOKEN
	MENU_IMPRESION
SOMBRA	ANALIZAR_ESTRUCTURA
	INICIA_COMENTARIOS
	MENU_IMPRESION
	MENU_PRINCIPAL
	SELECCIONA_DISPOSITIVO
	SELECCIONA_TIPO_IMPRESORA
TECLAS_MENU	CONTROL_OPCIONES_MENU

Impresión editada de programas

```
 1 {*************************************************************************}
 2 (* NOMBRE DE LA UNIDAD: ISPRINTER                                        *)
 3 (*                                                                       *)
 4 (* ELABORADO POR: MAURICIO H. VILLANUEVA CASTELLANOS                     *)
 5 (*                                                                       *)
 6 (* FECHA DE ELABORACIÓN: SEPTIEMBRE-1993                                 *)
 7 (*                                                                       *)
 8 (* DESCRIPCIÓN:                                                          *)
 9 (*                                                                       *)
10 (* VERIFICA SI LA IMPRESORA SE ENCUENTRA LISTA PARA RECIBIR INFORMACIÓN   *)
11 (*                                                                       *)
12 (* SALIDAS:                                                              *)
13 (*     TRUE. LA IMPRESORA ESTÁ DISPONIBLE.                               *)
14 (*     FALSE:LA IMPRESORA NO ETÁ DISPONIBLE.                             *)
15 {*************************************************************************}
16
17 FUNCTION ISPRINTER:BOOLEAN;
18 ┌─>BEGIN
19 │  { VERIFICA SI LA IMPRESORA SE ENCUENTRA EN LINEA MEDIANTE LA
20 │    INTERRUPCIÓN 17 Y LA FUNCION 02 }
21 │    SAVEREG.AX:=$0200;
22 │    SAVEREG.DX:=$0000;
23 │    INTR($17,SAVEREG);
24 │  ┌─>IF SAVEREG.AH=$0090
25 │  │  └─> THEN
26 │  │      ISPRINTER:=TRUE
27 │  │  ┌─> ELSE
28 │  │      ISPRINTER:=FALSE;
29 │  └─>* FIN DE IF *
30 │    SAVEREG.AX:=$0
31 └─>END ;* FIN DE BEGIN *
32
33 {*************************************************************************}
34 (* NOMBRE DE LA UNIDAD: SOMBRA                                           *)
35 (*                                                                       *)
36 (* ELABORADO POR: MAURICIO H. VILLANUEVA CASTELLANOS                     *)
37 (*                                                                       *)
38 (* FECHA DE ELABORACIÓN: SEPTIEMBRE-1993                                 *)
39 (*                                                                       *)
40 (* DESCRIPCIÓN:                                                          *)
41 (*                                                                       *)
42 (* PRODUCE UNA SOMBRA EN LAS COORDENADAS ESTABLECIDAS, MEDIANTE EL USO DE*)
43 (* LA MEMORIA DE VIDEO                                                   *)
44 (*                                                                       *)
45 (* ENTRADAS:                                                             *)
46 (*     LSI: LINEA SUPERIOR IZQUIERDA                                     *)
47 (*     CSI: COLUMNA SUPERIOR IZQUIERDA                                   *)
48 (*     LID: LINEA INFERIOR DERECHA                                       *)
49 (*     CID: COLUMNA INFERIOR DERECHA                                     *)
50 (*                                                                       *)
51 (* SALIDAS:                                                              *)
52 (*  SOMBRA A PANTALLA.                                                   *)
53 {*************************************************************************}
54 PROCEDURE SOMBRA ( LSI,CSI,LID,CID:BYTE);
55 VAR
56   I: BYTE;
57 ┌─>BEGIN
58 │  { SOMBRA HORIZONTAL }
59 │  ┌─>FOR I:=LSI TO LID DO
60 │  │    PANTALLA[I,2*CID]:=CHR(8);
```

Impresión numerada programas
ESTRUC9.PAS

```
 1 {*************************************************************************}
 2 (* Nombre de la unidad: isprinter                                        *)
 3 (*                                                                       *)
 4 (* Elaborado por: Mauricio H. Villanueva Castellanos                     *)
 5 (*                                                                       *)
 6 (* Fecha de elaboración. Septiembre-1993                                 *)
 7 (*                                                                       *)
 8 (* Descripción.                                                          *)
 9 (*                                                                       *)
10 (* Verifica si la impresora se encuentra lista para recibir información *)
11 (*                                                                       *)
12 (* Salidas:                                                              *)
13 (*      true: la impresora está disponible.                              *)
14 (*      false:la impresora no etá disponible.                            *)
15 {*************************************************************************}
16
17 Function isprinter:boolean;
18 begin
19  { verifica si la impresora se encuentra en linea mediante la
20    interrupción 17 y la función 02 }
21  savereg.ax:=$0200;
22  savereg.dx:=$0000;
23  intr($17,savereg),
24  if savereg.ah=$0090 then
25   isprinter:=true
26  else
27   isprinter:=false;
28  savereg.ax:=$0
29 end;
30
31 {*************************************************************************}
32 (* Nombre de la unidad: sombra                                           *)
33 (*                                                                       *)
34 (* Elaborado por: Mauricio H. Villanueva Castellanos                     *)
35 (*                                                                       *)
36 (* Fecha de elaboración. Septiembre-1993                                 *)
37 (*                                                                       *)
38 (* Descripción:                                                          *)
39 (*                                                                       *)
40 (* Produce una sombra en las coordenadas establecidas, mediante el uso de  *)
41 (* la memoria de video                                                   *)
42 (*                                                                       *)
43 (* Entradas:                                                             *)
44 (*      lsi: linea superior izquierda                                    *)
45 (*      csi: columna superior izquierda                                  *)
46 (*      lid: linea inferior derecha                                      *)
47 (*      cid: columna inferior derecha                                    *)
48 (*                                                                       *)
49 (* Salidas:                                                              *)
50 (*  sombra a pantalla.                                                   *)
51 {*************************************************************************}
52 Procedure sombra ( lsi,csi,lid,cid:byte);
53 var
54  i: byte;
55 begin
56  { sombra horizontal }
57  for i:=lsi to lid do
58   pantalla[i,2*cid]:=chr(8);
59  { sombra vertical   }
60  for i:=csi to cid do
```

VII.- CONCLUSIONES.

En un principio, el desarrollo de hardware tenía un crecimiento mucho mayor que el del software. En la actualidad éste desarrollo ha disminuido, comenzando a incrementarse el del software. El incremento que ha tenido éste ultimo, es debido a la necesidad de crear sistemas más complejos que tengan interacción entre ellos.

El tener sistemas más elaborados trae consigo el incremento de los costos de mantenimiento, que es una de las mayores preocupaciones en cualquier organización.

Para poder poder efectuar la reducción de estos costos, surge la necesidad de contar con mecanismos que permitan obtener, de una manera ordenada, los documentos necesarios para conocer el funcionamiento de un sistema sin perder de vista sus caracteristicas principales de operación.

Esta documentación puede ser obtenida mediante el uso de metodologías adecuadas de desarrollo. Existe una gran variedad de metodologías y aunque algunas ya no son tan recientes, tienen validez en nuestros días.

El uso de metodologías de desarrollo proporcionan los elementos para elaborar la documentación de un sistema, documentación que va desde el análisis de requerimientos hasta el mantenimiento, siguiendo un orden lógico. Cuando se efectúa lo anterior, el tiempo de mantenimiento se reduce debido a que se conoce el impacto que podría producir cualquier cambio al sistema.

Las metodologías de desarrollo no tienen que ser tomadas como un conjunto de pasos que deben de ser seguidos en un estricto orden, por el contrario, deben servir como una guía para elaborar una metodología propia, que se adapte a las políticas y procedimientos de la organización donde se esté desarrollando.

Además de las metodologías, se debe proporcionar al desarrollador las herramientas necesarias para que efectúe su trabajo en el menor tiempo posible y con la mejor calidad.

A pesar de que en el mercado de software hay muchas herramientas disponibles tales como graficadores, documentadores, generadores de código, etc., no pueden abarcar todo el software que existe. Es por ello que también se debe prestar atención a desarrollar herramientas que permitan generar nuevos sistemas con el menor esfuerzo.

El documentador de pascal se desarrolló con la finalidad de cumplir lo anterior, y aunque no es un sistema demasiado elaborado, cumple con el objetivo principal que es el de proporcionar la documentación de otros sistemas.

En la actualidad el desarrollo de sistemas ya no es sólo el crear un programa en base a los requerimientos de un usuario, sino que se trata de una disciplina fundamentada en el uso de métodos, herramientas y procedimientos, considerada de gran importancia en la área de la computación.

VIII.- BIBLIOGRAFIA

- Structured Analysis and Sistem especification
  Tom Demarco
  Yourdon Press Computing Series
  1979

- Structured Design
  Fundamentals of a discipline of computer program and system design
  Edward Yourdon
  Larry L. Constantine
  Youdon Press Computing Series
  1979

- Ingeniería del Software
  Un enfoque práctico
  Roger S. Pressman
  Segunda Edición
  Mc Graw Hill
  1988

- Ingeniería del Software
  Richard Fairley
  Mc Graw Hill
  1988

- Principles of Compiler Design
  Alfred U. Aho
  Jeffrey D. Ullman
  Adisson Wesley

- Estructura de Datos en pascal
  Aaron M. Tenenbaum
  Moshe S. Augenstein
  Prentice Hall Hispanoamericana
  1983

- Turbo Pascal
  Reference guide
  Version 5.0
  1987
  Borland International

- Programación en Pascal
  Peter Grogono
  Fondo Educativo Interamericano
  1980

- Documentation standards for computer Sistems
  Second edition
  Norman L. Enger.
  The technology Press Incorporated
  1980

- Manual de metodología de Ingeniería de sistemas
  Análisis Estructurado
  1984

- Artículo:
  Un rastreador de texto
  (Herramientas de Software)
  Enrique Calderón Alzati
  Publicación Cero uno cero
  Fundación Arturo Rosenblueth
  Agosto 1983

- Artículo:
  Guía de elaboración de Manuales técnicos.
  Elisa Gonzalez
  Armando Rivera
  Publicación Cero uno cero
  Fundación Arturo Rosenblueth
  Agosto 1984

- Artículo:
  Evolución de la documentación de sistemas
  Ricardo Lule Barrera
  Publicación Cero uno cero
  Fundación Arturo Rosenblueth
  Julio/Agosto 1984

- Apuntes de:
  Taller de Diseño Estructurado de
  Sistemas de Información
  Fundación Arturo Rosenblueth

- Apuntes del curso de:
  Advanced structured Analysis and Design
  Lawrence Peters
  Technology Training Corporation
  1990

- Apuntes de:
  Tecnicas de programación y teoría de compiladores.

I.- INDICE

II.- INTRODUCCION

El desarrollo de sistemas de computo ha evolucionado con el tiempo. Al principio no era necesario el documentar los sistemas que se realizaban debido a que las rutinas elaboradas eran de propósito especifico y en general eran corregidas, cuando se necesitaba efectuar un mantenimiento, por la misma persona que las desarrollaba.

Una de las premisas que existian eran elaborabar el mejor hardware sin prestarle mucha importancia al software, sin imaginarse que este último crecería mucho más rápido que el crecimiento que ha tenido el hardware.

Con el crecimiento del software el mantenimiento que se efectuaba a los sistemas se volvió más complejo, debido a que los programas que se desarrollaban tenían interacción unos con otros lo que hacía más complicado el realizar modificaciones a un programa sin afectar a los otros.

Otro de los problemas que se tenía era que el personal que originalmente realizaba los programas, ya no se encontraba en la organización y en consecuencia el poder entender el funcionamiento de los sistemas era muy complejo.

Al no tener una documentación de un sistema los costos de mantenimiento (el cuál siempre existirá durante la vida útil de un sistema) se incrementaban llegando a porcentajes del costo total de un sistema demasiados altos.

Es por ello que surgió la necesidad de contar con métodos, herramientas y procedimientos que regularan el desarrollo de un sistema. Existen actualmente muchas metodologías para el desarrollo de sistemas y aunque algunas de ellas ya no son tan recientes, siguen teniendo validez en nuestros días. Estas metodologías no son como una receta de cocina que van a resolver nuestros problemas en su totalidad, sino que se trata de una serie de guias que permiten elaborar una **documentación** que respalde el desarrollo del sistema.

Existen diferentes tipos de documentación que van desde el documento de definición de un sistema hasta la elaboración de manuales tecnicos, de operación y de usuario.

Hoy en día existen desarrollos que permiten utilizar las metodologías generando su propia documentación.

El desarrollo presentado en éste manual (documentador para pascal) es una de esas herramientas que permiten generar una documentación que ayudará a elaborar y actualizar el manual de referencia técnica del sistema, además que permitirá al usuario del mismo obtener, en "linea", esa documentación sin tener la necesidad de tener el manual tecnico en sus manos.

Cabe aclarar que una vez que se hayan efectuado modificaciones a un sistema estás tendrán que ser reflejadas en cada uno de los documentos que respaldan el desarrollo de un sistema, es decir, el uso del documentador de pascal no se encuentra aislado, sino que tiene relación con un gran número de herramientas existentes.

El documentador de pascal considera las siguientes salidas:

- Relación de subprogramas por programa. (Consulta/impresión)
- Relación de módulos por programa. (Consulta/impresión)
- Relación de módulo contenidos (declarados por cada módulo). (Consulta/impresión)
- Referencias cruzadas: (Consulta/impresión)
  - Relación de módulos llamados vs. módulo llamador
  - Relación de módulo llamador vs módulos llamados
- Alcance de módulos. (Consulta/impresión)
- Estructura modular (impresión jerarquica, (Consulta/impresión))
- Impresión editada de programas. (Impresión)
- Impresión numerada de programas. (Impresión)

El documentador de pascal se basa en el uso del pascal Standard considerando algunas opciones de versiones mejoradas, así como el manejo de directivas.

El documentador de pascal asume que los sistemas que se van a documentar se encuentran trabajando, es decir, que los programas a documentar se encuentran libres de errores.

III.- SOBRE EL MANUAL

El manual se encuentra organizado de la siguiente forma:

En la sección II se presenta una breve introducción acerca del documentador para pascal.

En la sección IV (organización) se presenta:

- La forma para ejecutar el documentar, así como las características que el hardware (equipo de computo) deberá reunir para el correcto funcionamiento del sistema.
- Una descripción sobre el manejo de las teclas de uso común.
- Una explicación (siguiendo una jerarquía) de cada una de las opciones que forman parte del documentador para pascal.

En la sección V (Operación) se presenta, en detalle, la forma de accesar, la función y la descripción de cada uno de los programas terminales (entiendase como programas terminales aquellas opciones que realizan una función en particular, es decir, que no se trata de una opción de menú)

Y por último en el Apéndice, se presenta una relación de los errores más comunes que se pueden tener durante la operación normal del documentador.

IV.- ORGANIZACION

En ésta sección se describen los requerimientos de hardware necesarios para ejecutar el sistema, las teclas de uso común y la forma en que se encuentran organizadas cada una de las opciones del documentador.

- Requerimientos de hardware (equipo de computo).

  - Computadora PC XT AT compatible IBM.
  - Monitor a color (esto es por el manejo de la dirección B800:0000 de memoria de video)
  - 640 K RAM (memoria de acceso aleatorio)
  - Espacio en disco de aproximadamente dos veces el espacio ocupado por los programas a documentar.

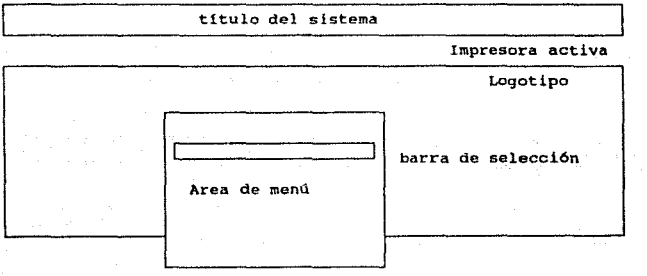
- Teclas de uso común.

El sistema se encuentra organizado mediante opciones de menú. El movimiento a través de cada una de estas opciones se realiza por medio de las siguientes teclas:

Teclas	Función
↓ →	Selecciona la siguiente opción en donde se encuentre posicionada la barra de color cyan. En el caso de que la barra se encuentre en la última opción del menú, el movimiento se efectuará a la primera opción del mismo (movimiento circular).
↑ ←	Selecciona la opción anterior en donde se encuentre posicionada la barra de color cyan. En el caso de que la barra se encuentre en la primera opción del menú, el movimiento se efectuará a la última opción del mismo (movimiento circular).
<↵	Permite seleccionar el siguiente menú disponible o ejecuta un programa terminal.
Esc	Regresa al menú de donde fué llamado o termina la operación del sistema en el caso de que se trate del menú pricipal.

Estructura jerárquica.

Las pantallas de menú se encuentran organizadas de la siguiente forma:

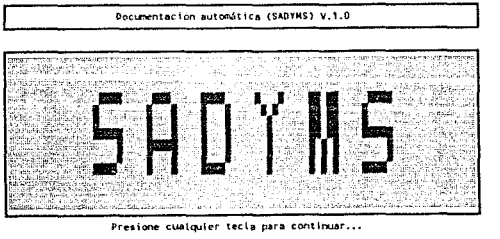


Para ejecutar el documentador se tiene que escribir el comando

DOCUMENT.EXE <Enter>

Se recomienda que el documentador de pascal se encuentre instalado en disco duro, para que el acceso al mismo y el tiempo de proceso sean menores.

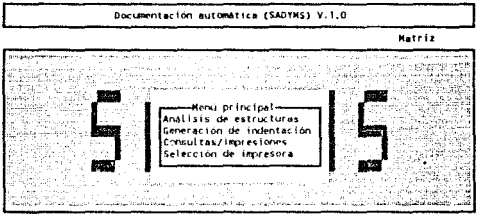
La primera pantalla que aparece al iniciar el sistema es la siguiente:



En ella se muestra el logotipo de identificacion del documentador.

Las siglas SADYMS significan sistema de apoyo para la documentación y mantenimiento de sistemas.

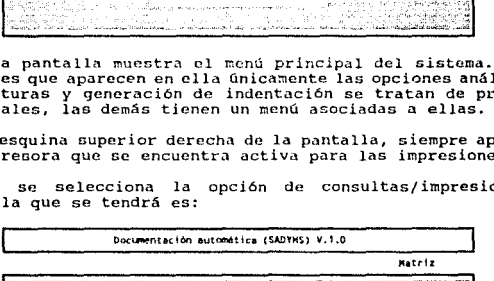
Al oprimir cualquier tecla la pantalla que aparecerá es:



Esta pantalla muestra el menú principal del sistema. De las opciones que aparecen en ella únicamente las opciones análisis de estructuras y generación de indentación se tratan de programas terminales, las demás tienen un menú asociadas a ellas.

En la esquina superior derecha de la pantalla, siempre aparecerá la impresora que se encuentra activa para las impresiones.

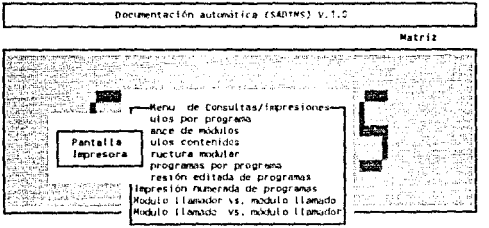
Cuando se selecciona la opción de consultas/impresiones la pantalla que se tendrá es:



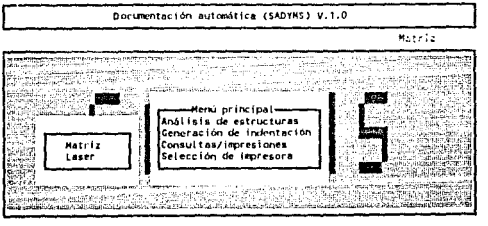
En cada una de las opciones anteriores, a excepción de las opciones de impresion editada y numerada de programas, se presenta un submenú cuando éstas son seleccionadas; Dicho submenú permite seleccionar el dispositivo en el cuál se va a obtener el resultado de la operación (pantalla/impresora).

Para las opciones restantes, la salida se efectúa directamente a impresora, tomando en cuenta la impresora que se encuentre activa al momento de elegir la opción.

La pantalla que identifica el tipo de dispositivo a utilizar es el que se muestra a continuación:



Regresando al menú principal del sistema (mediante el uso de la tecla <ESC>) y seleccionando la opción de selección de impresora se obtiene la pantalla:



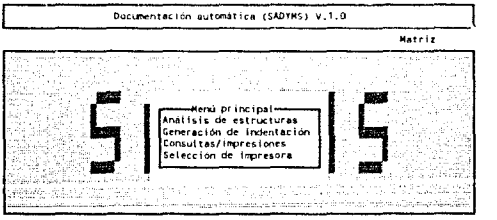
En esta pantalla se muestran dos opciones:

Laser y
Matriz

Cada una de ellas permite elaborar las secuencias de escape (conjunto de comandos mediante los cuales se le indica a la impresora las operaciones a realizar) que son utilizadas para controlar la compresión de la letra y saltos al final de la hoja de impresión (eject). En las impresiones no se utilizan secuencias de escape elaboradas y esto es porque la totalidad de las secuencias varia entre una marca y otra de impresoras.

Debido a que muchas veces el que desarrolla sistemas, y en consecuencia el usuario a quien está dirigido el documentador, no cuenta con impresoras que permitan alta calidad de impresión, por lo que tener un conjunto extenso y variado de secuencias de escape sería ridículo.

Por último, cuando el usuario del documentador termina su operación (mediante el uso de la tecla <ESC>) aparecerá el mensaje:



Operación finalizada normalmente

Cualquier otro mensaje signica que ocurrió un error durante la operación y en consecuencia la documentación generada no será confiable.

Los errores que se podrían presentar durante la operación normal del documentador, se encuentran descritos en el apéndice al final de éste manual.

V.- OPERACION

En ésta sección se describe cada uno de los programas terminales, los cuales se encuentran organizados en base a los conceptos:

Función: Establece la función que realiza la opción.

Camino: Muestra las acciones que deberan realizarse para llegar a la opción que se está consultando, partiendo del menú principal del documentador.

Opciones: Muestra las operaciones que pueden efectuarse, así como las restricciones y consideraciones a tomar para su correcto funcionamiento.

Proceso: **Análisis de estructuras.**

Función: Generar las listas que determinarán las relaciones existentes entre programas, subprogramas, módulos llamados, módulos llamadores, módulos contenidos y la estructura jerárquica del sistema a documentar.

Camino: Análisis de estructuras

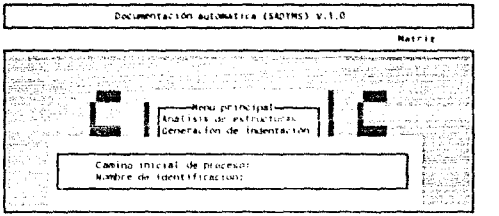
Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- El sistema a documentar se encuentre libre de errores, es decir, que se haya compilado con anterioridad.
- El nombre del programa a proporcionar debera contener una estructura válida de programa, es decir, contener las palabras reservadas PROGRAM, BEGIN END y un punto al final del archivo.
- Los programas, que forman parte del sistema a documentar, deberán encontrarse en el mismo directorio en el que se encuentra el módulo principal.

NOTA: El nombre del programa deberá ser considerado sin extensión.

Si no se cumple con alguno de los requerimientos anteriores, el documentador detendrá su operación mostrando un mensaje del porqué no puede continuar con la operación.

Cuando se accesa la opción aparecerá la siguiente pantalla:



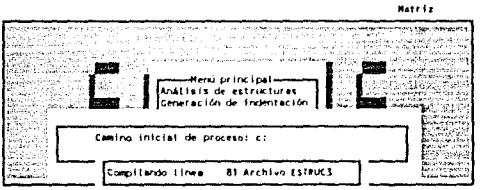
En donde:

"Camino inicial de proceso:" indica el path donde se encuentran los programas a documentar. Si se deja en blanco se toma el path donde se encuentre el documentador.

"Nombre de identificación:" indica el nombre del archivo principal de proceso.

Si se le proporciona al docementador un camino ó nombre de identificación inválido, aparecerá el mensaje "Archivo no encontrado" teniendo que ejecutar nuevamente la opción.

Por el contrario, si el archivo existe el documentador mostrará la pantalla:



En el que se indica tanto la línea como el archivo que esta procesando (el o los archivos que fueron llamados mediante una directiva de inclusión dentro del programa principal).

Una vez que se termine el proceso aparecerá el mensaje "Proceso terminado", momento en el que podrá realizar la consulta y/o impresión de las relaciones.

Proceso: **Generación de indentación.**

Función: Genera un archivo de salida, que servirá para efectuar la impresión editada de programas, identificando cada una de las instrucciones estructuradas que forman parte de pascal.

Camino: Generación de indentación.

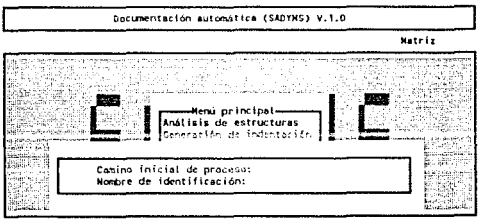
Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- El programa a documentar se encuentre libre de errores, es decir, que se haya compilado con anterioridad.

NOTA: Ningún programa debera de llevar extensión.

Si no se cumple con el requerimiento anterior, el documentador detendrá su operación, mostrando un mensaje del porqué no puede continuar con la misma.

Cuando se accesa la opción aparecerá la siguiente pantalla:



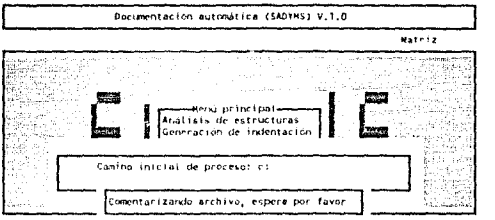
En donde:

"Camino inicial de proceso:" indica el path donde se encuentra el programa a documentar. Si se deja en blanco se toma el path donde se encuentre el documentador.

"Nombre de identificación:" indica el nombre del programa a documentar.

Si se le proporciona un camino ó nombre de identificación inválido, el documentador mostrará el mensaje "Archivo no encontrado" teniendo que ejecutar nuevamente la opción.

Por el contrario, si el archivo existe el documentador mostrará la pantalla:



Una vez que se termine la opción aparecerá el mensaje "Proceso terminado", momento en el que podrá realizar la impresión editada de programas.

Consultas.

Para poder efectuar las consultas/impresiones es necesario ejecutar primero las opciones de análisis de estructuras y/o generación de indentación según sea el caso.

Proceso: Módulos por programa.

Función: Muestra la relación existente entre cada uno de los módulos que forman parte del sistema a documentar, indicando el tipo y ubicación de cada módulo dentro de un listado generado por la opción de impresión numerada de programas.

Camino: Consultas/impresiones/módulos por programa.

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Se haya ejecutado la opción de análisis de estructuras

Cuando se accesa la opción aparecerá una pantalla en la que se solicitará la salida del reporte (pantalla/impresión).

Si se selecciona la salida a pantalla, se tendrá:

```
Documentación automática (SADTMS) V.1.0

                                                                Matriz

                        Módulos por programa
Nombre de la unidad                    Tipo Página  Programa(Archivo)
-------------------                    ---- ------  ----------------

ANALIZAR_BEGIN                         P    8       ESTRUCS
ANALIZAR_CASE                          P    3       ESTRUCS
ANALIZAR_ESTRUCTURA                    P    10      ESTRUCS
ANALIZAR_ESTRUCTURA_CASE               P    1       ESTRUCS
ANALIZAR_FOR                           P    6       ESTRUCS
ANALIZAR_IF                            P    4       ESTRUCS

Presione <Return> para continuar, <Esc> para cancelar
```

En esta pantalla se muestra la relación exitente entre cada módulo y el programa en donde se encuentra declarado.

El tipo del módulo puede tener tres valores que son:

P	Procedimiento
F	Funcion
G	Rutina principal (nombre que sigue a la palabra reservada PROGRAM)

Para continuar con el despliegue del resto de los valores basta con presionar la tecla <Return>.

Si por el contrario no se desea continuar, al presionar la tecla <ESC> la operación de consulta se cancelará.

Al igual que en la consulta la tecla <ESC> cancelará la impresión.

El formato del reporte que se obtiene cuando se selecciona la opción de impresora se muestra a continuación:

14-10-1993 Sistema de documentacion para Pascal 1

Nombre de la unidad	Módulos por programa Tipo	Página	Programa(Archivo)	Llamados	Llamadores	Contenidos
ANALIZAR_BEGIN	P	8	ESTRUC5	2	1	0
ANALIZAR_CASE	P	3	ESTRUC5	3	2	0
ANALIZAR_ESTRUCTURA	P	10	ESTRUC3	8	1	0
ANALIZAR_ESTRUCTURA_CASE	P	1	ESTRUC5	2	1	0
ANALIZAR_FOR	P	6	ESTRUC5	2	1	0
ANALIZAR_IF	P	4	ESTRUC5	2	1	0
ANALIZAR_RECORD	P	10	ESTRUC5	3	1	0
ANALIZAR_REPEAT	P	9	ESTRUC5	2	1	0
ANALIZAR_WHILE	P	7	ESTRUC5	2	1	0
ANALIZAR_WITH	P	11	ESTRUC5	2	1	0
ARMAR_TOKEN	P	1	ESTRUC2	1	1	0
ASIGNAR_PROCEDIMIENTOS	P	7	ESTRUC8	0	2	0
ASIGNAR_PROCEDIMIENTOS_CONTENIDOS	P	4	ESTRUC8	1	1	0
ASIGNAR_PROCEDIMIENTOS_LLAMADORES	P	5	ESTRUC8	1	1	0
ASIGNAR_PROCEDIMIENTOS_LLAMADOS	P	6	ESTRUC8	1	1	0
ASIGNAR_SUBPROGRAMAS	P	6	ESTRUC8	0	1	0
BORRA_HEAP	P	10	ESTRUC7	0	1	0
BOX	P	3	ESTRUC9	0	14	0

Proceso: Alcance de módulos.

Función: Muestra la localización, por número de línea, de cada uno de los módulos que forman parte del sistema, así como la declaración de la sección ejecutable y la terminación del módulo.

Camino: Consultas/impresiones/alcance de modulos.

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Se haya ejecutado la opción de análisis de estructuras

Cuando se accesa la opción aparecerá una pantalla en la que se solicitará la salida del reporte (pantalla/impresión).

Si se selecciona la salida a pantalla se tendrá:

Documentación automática (SADYMS) V.1.0

Matriz

Nombre de la unidad	Alcance de módulos De la línea	A la línea	Sec. eje.	Programa
ANALIZAR_BEGIN	524	559	525	ESTRUC5
ANALIZAR_CASE	139	226	147	ESTRUC5
ANALIZAR_ESTRUCTURA	639	714	640	ESTRUC3
ANALIZAR_ESTRUCTURA_CASE	27	113	36	ESTRUC5
ANALIZAR_FOR	367	420	370	ESTRUC5
ANALIZAR_IF	251	343	255	ESTRUC5

Presione <Return> para continuar, <Esc> para cancelar

En esta pantalla se muestra la ubicación de cada uno de los módulos que forman parte del sistema, en cuanto a inicio (declaración del programa, procedimiento o función), terminación (el último end que forma parte del módulo) y la sección ejecutable (el primer begin del módulo).

Para continuar con el despliegue del resto de los valores basta con presionar la tecla <Return>.

Si por el contrario no se desea continuar, al presionar la tecla <ESC> la operación de consulta se cancelará.

Al igual que en la consulta la tecla <ESC> cancelará la impresión.

El formato del reporte que se obtiene cuando se selecciona la opción de impresora se muestra a continuación:

14-10-1993 Sistema de documentacion para Pascal 1

Alcance de modulos

Nombre de la unidad	De la linea	A la linea	Sec. eje.	Programa
ANALIZAR_BEGIN	524	559	525	ESTRUC5
ANALIZAR_CASE	139	226	147	ESTRUC5
ANALIZAR_ESTRUCTURA	639	714	640	ESTRUC3
ANALIZAR_ESTRUCTURA_CASE	27	113	36	ESTRUC5
ANALIZAR_FOR	367	420	370	ESTRUC5
ANALIZAR_IF	251	343	255	ESTRUC5
ANALIZAR_RECORD	643	690	646	ESTRUC5
ANALIZAR_REPEAT	583	619	584	ESTRUC5
ANALIZAR_WHILE	444	500	447	ESTRUC5
ANALIZAR_WITH	714	768	717	ESTRUC5
ARMAR_TOKEN	61	615	123	ESTRUC2
ASIGNAR_PROCEDIMIENTOS	456	497	460	ESTRUC8
ASIGNAR_PROCEDIMIENTOS_CONTENIDOS	199	248	203	ESTRUC8
ASIGNAR_PROCEDIMIENTOS_LLAMADORES	273	322	277	ESTRUC8
ASIGNAR_PROCEDIMIENTOS_LLAMADOS	347	364	348	ESTRUC8
ASIGNAR_SUBPROGRAMAS	387	432	391	ESTRUC8
BORRA_HEAP	614	654	620	ESTRUC7
BOX	133	161	137	ESTRUC9
BUSCA_MODULO_REFERENCIA	295	309	299	ESTRUC3
CLRWINDOW	151	160	154	ESTRUC4
CONFIGURACION_IMPRESORA	60	66	61	ESTRUC4
CONTROL_OPCIONES_MENU	272	301	276	ESTRUC9
CONTROL_PANTALLA	189	223	192	ESTRUC4
CONTROL_SALIDA	741	870	742	ESTRUC3
CREAR_PROCEDIMIENTOS	155	174	156	ESTRUC8

Proceso: Modulo vs. módulo contenido

Función: Muestra la relación exitente entre cada uno de los módulo y los módulos declarados dentro del mismo.

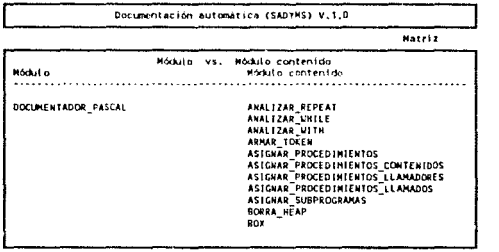
Camino: Consultas/impresiones/modulo vs. módulo contenido

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Se haya ejecutado la opción de análisis de estructuras

Cuando se accesa la opción aparecerá una pantalla en la que se solicitará la salida del reporte (pantalla/impresión).

Si se selecciona la salida a pantalla se tendrá:



En esta pantalla se muestra la relación existente entre cada módulo y sus módulos contenidos, es decir, cuales módulos se encuentran declarados dentro de que módulo.

Para continuar con el despliegue del resto de los valores basta con presionar la tecla <Return>.

Si por el contrario no se desea continuar, al presionar la tecla <ESC> la operación de consulta se cancelará.

Al igual que en la consultą la tecla <ESC> cancelará la impresión.

El formato del reporte que se obtiene cuando se selecciona la opción de impresora se muestra a continuación:

```
14-10-1993          Sistema de documentacion para Pascal          1

                          Módulo vs. Módulo contenido
Módulo                                    Módulo contenido
---------------------------------------   ---------------------------------------

ANALIZAR_BEGIN

ANALIZAR_CASE

ANALIZAR_ESTRUCTURA

ANALIZAR_ESTRUCTURA_CASE

ANALIZAR_FOR

ANALIZAR_IF

ANALIZAR_RECORD

ANALIZAR_REPEAT

ANALIZAR_WHILE

ANALIZAR_WITH

ARMAR_TOKEN

ASIGNAR_PROCEDIMIENTOS

ASIGNAR_PROCEDIMIENTOS_CONTENIDOS

ASIGNAR_PROCEDIMIENTOS_LLAMADORES

ASIGNAR_PROCEDIMIENTOS_LLAMADOS

ASIGNAR_SUBPROGRAMAS

BORRA_HEAP

BOX

BUSCA_MODULO_REFERENCIA

CLRWINDOW

CONFIGURACION_IMPRESORA

CONTROL_OPCIONES_MENU

CONTROL_PANTALLA

CONTROL_SALIDA

CREAR_PROCEDIMIENTOS

CREAR_PROCEDIMIENTO_CONTENIDO
```

Proceso: Estructura modular

Función: Muestra las llamadas efectuadas a módulos declarados manteniendo una estructura jerárquica de llamado, con la finalidad de proporcionar al lector del reporte una visión genérica de la operación del sistema.

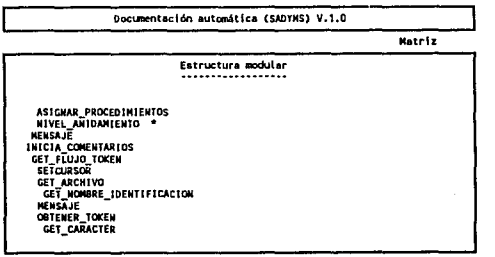
Camino: Consultas/impresiones/estructura modular

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Se haya ejecutado la opción de análisis de estructuras

Cuando se accesa la opción aparecerá una pantalla en la que se solicitará la salida del reporte (pantalla/impresión).

Si se selecciona la salida a pantalla se tendrá:



En esta pantalla se muestra una relación jerárquica de módulos llamados, agregando un blanco de indentación cada vez que se efectúa una llamada.

Junto con el nombre del módulo se pueden imprimir los valores:

*	Llamada recursiva
F	Declaración de forward.

Para continuar con el despliegue del resto de los valores basta con presionar la tecla <Return>.

Si por el contrario no se desea continuar, al presionar la tecla <ESC> la operación de consulta se cancelará.

Al igual que en la consulta la tecla <ESC> cancelará la impresión.

El formato del reporte que se obtiene cuando se selecciona la opción de impresora se muestra a continuación:

```
14-10-1993          Sistema de documentacion para Pascal          1

                                  Estructura modular
                                  ------------------

  CONTROL_SALIDA
   MENSAJE
   SETCURSOR
  SETCURSOR
  BOX
  LOGOTIPO
   MENSAJE
  MENU_PRINCIPAL
   DESPLIEGA_DISPOSITIVO
   SOMBRA
   BOX
   CONTROL_OPCIONES_MENU
    TECLAS_MENU
   BORRA_HEAP
   ANALIZAR_ESTRUCTURA
    GET_FLUJO_TOKEN
     SETCURSOR
     GET_ARCHIVO
      GET_NOMBRE_IDENTIFICACION
     MENSAJE
     OBTENER_TOKEN
      GET_CARACTER
      ARMAR_TOKEN
       GET_CARACTER
    CUENTA_CRLF
    SOMBRA
    BOX
    CREAR_PROCEDIMIENTOS
    ASIGNAR_PROCEDIMIENTOS
    NIVEL_ANIDAMIENTO  F
     GET_FLUJO_TOKEN
      SETCURSOR
      GET_ARCHIVO
       GET_NOMBRE_IDENTIFICACION
      MENSAJE
      OBTENER_TOKEN
       GET_CARACTER
       ARMAR_TOKEN
        GET_CARACTER
     BUSCA_MODULO_REFERENCIA
     CREAR_PROCEDIMIENTO_CONTENIDO
     ASIGNAR_PROCEDIMIENTOS_CONTENIDOS
      BUSCA_MODULO_REFERENCIA
     CREAR_PROCEDIMIENTOS
     ASIGNAR_PROCEDIMIENTOS
     NIVEL_ANIDAMIENTO  F
```

Proceso: Subprogramas por programa.

Función: Muestra una relación de los programas que son llamados por medio de una directiva de inclusión, indicando tamaño y fecha de la última modificación efectuada.

Camino: Consultas/impresiones/subprogramas por programa.

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Se haya ejecutado la opción de análisis de estructuras

Cuando se accesa la opción aparecerá una pantalla en la que se solicitará la salida del reporte (pantalla/impresión).

Si se selecciona la salida a pantalla se tendrá:

```
Documentación automática (SADYNS) V.1.0

                                                              Matriz

                          Subprogramas por programa
Archivo          Tamaño     Fecha ultima actualización (DD/MM/AAAA)
-------          ------     ---------------------------------------

* ESTRUC3         44344          4   10  1993
ESTRUC2.PAS      31137          4   10  1993
ESTRUC4.PAS      50119          4   10  1993
ESTRUC5.PAS      47448         29    9  1993
ESTRUC6.PAS      17608          4   10  1993
ESTRUC7.PAS      28161         30    9  1993
ESTRUC8.PAS      28336         30    9  1993
ESTRUC9.PAS      17519          4   10  1993

Proceso terminado, oprima <Return> para continuar...
```

En esta pantalla se muestran los programas que son llamados por medio de una directiva de inclusión, indicando tamaño y fecha de la última actualización efectuada.

El programa que contiene el "*" después del nombre es el programa llamador.

Para continuar con el despliegue del resto de los valores basta con presionar la tecla <Return>.

Si por el contrario no se desea continuar, al presionar la tecla <ESC> la operación de consulta se cancelará.

Al igual que en la consulta la tecla <ESC> cancelará la impresión.

El formato del reporte que se obtiene cuando se selecciona la opción de impresora se muestra a continuación:

14-10-1993 Sistema de documentacion para Pascal 1

Archivo	Tamaño	Subprogramas por programa	Fecha ultima actualización (DD/MM/AAAA)	
*ESTRUC3	44344	12	10	1993
ESTRUC2.PAS	31137	4	10	1993
ESTRUC4.PAS	50119	4	10	1993
ESTRUC5.PAS	47448	29	9	1993
ESTRUC6.PAS	18282	12	10	1993
ESTRUC7.PAS	28161	30	9	1993
ESTRUC8.PAS	28336	30	9	1993
ESTRUC9.PAS	17519	4	10	1993

Proceso: Impresión editada de programas

Función: Realiza la impresión de los programas elaborados mediante la opción de generación de indentación.

Camino: Consultas/impresiones/impresión editada de programas.

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Se haya ejecutado la opción de generación de indentación

Cuando se accesa la opción el reporte, será enviado a impresión considerando la secuencia de escape del valor mostrado en la parte superior derecha de la pantalla.
Para cancelar la impresión basta con presionar la tecla <ESC>.

El formato del reporte que se obtiene es el que se muestra a continuación:

```
1  PROCEDURE NIVEL_ANIDAMIENTO (NOMBRE_MODULO:TIPO_NOMBRE_MODULO);
2  VAR
3   NIVEL_RECORD,          (* PROCESO DE CASE DENTRO DE LA ESTRUCTURA RECORD *)
4   TERMINA_ANIDAMIENTO, (* CONTROL DE ANIDAMIENTO                        *)
5   LLAMADA_VALIDA,      (* CONTROL DE LLAMADA, NO DECLARACIÓN, DE FUNCTION*)
6   LLAMADA_FORWARD,     (* CONTROL DE LA INSTRUCCIÓN FORWARD             *)
7   TERMINA_COMPARACION       :BOOLEAN          ;
8
9   AUXILIAR_BUSQUEDA         :APUNTADOR_PROCEDIMIENTOS;
10 ( APUNTADORES DE REFERENCIA DURANTE LAS BUSQUEDAS )
11  AUXILIAR_BUSQUEDA_COM     :APUNTADOR_CONTENIDO    ;
12  AUXILIAR_BUSQUEDA_LLO     :APUNTADOR_LLAMADO      ;
13  AUXILIAR_BUSQUEDA_LLR     :APUNTADOR_LLAMADOR     ;
14  SECCION_EJECUTABLE,
15  NIVEL_END                 :INTEGER                ;
16  ANT_TOKEN,
17  TOKEN_REF                 :TI_TOKEN               ;
18   ┌->BEGIN
19   |  TERMINA_ANIDAMIENTO:=FALSE;
20   |  NIVEL_END:=0;
21   |  SECCION_EJECUTABLE:=0;
22   |  NIVEL_RECORD:=FALSE;
23   |  ( PROCESAMIENTO DE MODULO )
24   |  ┌->WHILE NOT TERMINA_ANIDAMIENTO DO
25   |  |  ┌->BEGIN
26   |  |  |  ANT_TOKEN:=TOKEN;
27   |  |  |  GET_FLUJO_TOKEN;
28   |  |  |  GOTOXY(20,3);
29   |  |  |  WRITE(CONTADOR_LINEAS:5);
30   |  |  |  GOTOXY(34,3);
31   |  |  |  WRITE(NOMBRE_ARCHIVO);
32   |  |  |  ┌->CASE  TIPO_TOKEN OF
33   |  |  |  |-  TOKEN_BEGIN,TOKEN_RECORD,
34   |  |  |  |   TOKEN_CASE:
35   |  |  |  |                ┌->BEGIN
36   |  |  |  |                |  ( CONTABILIZA EL NIVEL DE
37   |  |  |  |                |    ANIDAMIENTO )
38   |  |  |  |                |  ┌->IF NOT NIVEL_RECORD
39   |  |  |  |                |  |- THEN
40   |  |  |  |                |  |  ┌->BEGIN
```

Proceso: Impresión número de programas

Función: Realiza la impresión de programas agregandole un número al inicio de cada línea.

Camino: Consultas/impresiones/impresión numerada. programas.

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Exista el programa a imprimir en el camino establecido.

NOTA: Ningún programa debera de llevar extensión.

Cuando se accesa la opción, el reporte será enviado a impresión considerando la secuencia de escape del valor mostrado en la parte superior derecha de la pantalla.

Para cancelar la impresión basta con presionar la tecla <ESC>.

El formato del reporte que se obtiene es el que se muestra a continuación:

```
1  PROCEDURE NIVEL_ANIDAMIENTO (NOMBRE_MODULO:TIPO_NOMBRE_MODULO);
2  VAR
3    NIVEL_RECORD,          (* PROCESO DE CASE DENTRO DE LA ESTRUCTURA RECORD *)
4    TERMINA_ANIDAMIENTO,   (* CONTROL DE ANIDAMIENTO                        *)
5    LLAMADA_VALIDA,        (* CONTROL DE LLAMADA, NO DECLARACIÓN, DE FUNCTION*)
6    LLAMADA_FORWARD,       (* CONTROL DE LA INSTRUCCIÓN FORWARD             *)
7    TERMINA_COMPARACION       :BOOLEAN                ;
8
9    AUXILIAR_BUSQUEDA         :APUNTADOR_PROCEDIMIENTOS;
10 ( APUNTADORES DE REFERENCIA DURANTE LAS BUSQUEDAS )
11   AUXILIAR_BUSQUEDA_CON     :APUNTADOR_CONTENIDO    ;
12   AUXILIAR_BUSQUEDA_LLO     :APUNTADOR_LLAMADO      ;
13   AUXILIAR_BUSQUEDA_LLR     :APUNTADOR_LLAMADOR     ;
14   SECCION_EJECUTABLE,
15   NIVEL_END                 :INTEGER                ;
16   ANT_TOKEN,
17   TOKEN_REF                 :TI_TOKEN               ;
18 BEGIN
19   TERMINA_ANIDAMIENTO:=FALSE;
20   NIVEL_END:=0;
21   SECCION_EJECUTABLE:=0;
22   NIVEL_RECORD:=FALSE;
23   ( PROCESAMIENTO DE MÓDULO )
24   WHILE NOT TERMINA_ANIDAMIENTO DO
25    BEGIN
26     ANT_TOKEN:=TOKEN;
27     GET_FLUJO_TOKEN;
28     GOTOXY(20,3);
29     WRITE(CONTADOR_LINEAS:5);
30     GOTOXY(34,3);
31     WRITE(NOMBRE_ARCHIVO);
```

[illegible] [illegible]

[illegible] [illegible]

[illegible] [illegible]

Opciones: [illegible]

- [illegible] estructuras

Cuando se accesa la [illegible] en la que se [illegible] (pantalla/impresion).

Si se selecciona la [illegible]

Documentacion automatica [illegible]	
	[illegible]
[illegible]	
Módulo	[illegible]
ANALIZAR_BEGIN	[illegible]
ANALIZAR_CASE	[illegible]
ANALIZAR_[illegible]	[illegible]
[illegible]	

En este [illegible]

[illegible]

[illegible]

Proceso: Modulo llamador vs. módulo llamado

Función: Muestra la relación existente entre cada uno de los módulo y los módulos a los que llama

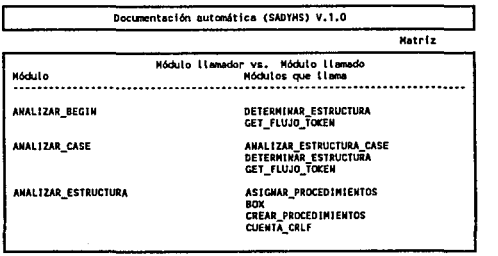
Camino: Consultas/impresiones/modulo llamador vs. modulo llamado.

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Se haya ejecutado la opción de análisis de estructuras

Cuando se accesa la opción aparecerá una pantalla en la que se solicitará la salida del reporte (pantalla/impresión).

Si se selecciona la salida a pantalla se tendrá:



En esta pantalla se muestra la relación existente entre cada módulo y sus módulos llamados, es decir, cuales módulos son llamados dentro de que módulo.

Para continuar con el despliegue del resto de los valores basta con presionar la tecla <Return>.

Si por el contrario no se desea continuar, al presionar la tecla <ESC> la operación de consulta se cancelará.

Al igual que en la consulta la tecla <ESC> cancelará la impresión.

El formato del reporte que se obtiene cuando se selecciona la opción de impresora se muestra a continuación:

14-10-1993 Sistema de documentacion para Pascal 1

Módulo llamador vs. Módulo llamado

Módulo	Módulos que llama
ANALIZAR_BEGIN	DETERMINAR_ESTRUCTURA
	GET_FLUJO_TOKEN
ANALIZAR_CASE	ANALIZAR_ESTRUCTURA_CASE
	DETERMINAR_ESTRUCTURA
	GET_FLUJO_TOKEN
ANALIZAR_ESTRUCTURA	ASIGNAR_PROCEDIMIENTOS
	BOX
	CREAR_PROCEDIMIENTOS
	CUENTA_CRLF
	GET_FLUJO_TOKEN
	MENSAJE
	NIVEL_ANIDAMIENTO
	SOMBRA
ANALIZAR_ESTRUCTURA_CASE	DETERMINAR_ESTRUCTURA
	GET_FLUJO_TOKEN
ANALIZAR_FOR	DETERMINAR_ESTRUCTURA
	GET_FLUJO_TOKEN
ANALIZAR_IF	DETERMINAR_ESTRUCTURA
	GET_FLUJO_TOKEN
ANALIZAR_RECORD	ANALIZAR_CASE
	DETERMINAR_ESTRUCTURA
	GET_FLUJO_TOKEN

Proceso: Modulo llamado vs. módulo llamador

Función: Muestra la relación existente entre cada uno de los módulo y los módulos de donde es llamado.

Camino: Consultas/impresiones/modulo llamado vs. modulo llamador.

Opciones: Antes de procesar ésta opción el usuario deberá asegurarse de que:

- Se haya ejecutado la opción de análisis de estructuras

Cuando se accesa la opción aparecerá una pantalla en la que se solicitará la salida del reporte (pantalla/impresión).

Si se selecciona la salida a pantalla, aparecera la siguiente pantalla:

```
                Documentación automática (SADYMS) V.1.0

                                                              Matriz

                        Módulo vs. Módulo llamador
Módulo                             Módulo llamador
-------------------------------------------------------------------

ANALIZAR_BEGIN                     DETERMINAR_ESTRUCTURA

ANALIZAR_CASE                      ANALIZAR_RECORD
                                   DETERMINAR_ESTRUCTURA

ANALIZAR_ESTRUCTURA                MENU_PRINCIPAL

ANALIZAR_ESTRUCTURA_CASE           ANALIZAR_CASE

ANALIZAR_FOR                       DETERMINAR_ESTRUCTURA

ANALIZAR_IF                        DETERMINAR_ESTRUCTURA

Presione <Return> para continuar, <Esc> para cancelar
```

En esta pantalla se muestra la relación existente entre cada módulo y los módulos llamadores, es decir, cuales módulos son llamados por que módulo.

Para continuar con el despliegue del resto de los valores basta con presionar la tecla <Return>.

Si por el contrario no se desea continuar, al presionar la tecla <ESC> la operación de consulta se cancelará.

Al igual que en la consulta la tecla <ESC> cancelará la impresión.

El formato del reporte que se obtiene cuando se selecciona la opción de impresora se muestra a continuación:

14-10-1993 Sistema de documentacion para Pascal 1

Módulo vs. Módulo llamador

Módulo	Módulo llamador
ANALIZAR_BEGIN	DETERMINAR_ESTRUCTURA
ANALIZAR_CASE	ANALIZAR_RECORD
	DETERMINAR_ESTRUCTURA
ANALIZAR_ESTRUCTURA	MENU_PRINCIPAL
ANALIZAR_ESTRUCTURA_CASE	ANALIZAR_CASE
ANALIZAR_FOR	DETERMINAR_ESTRUCTURA
ANALIZAR_IF	DETERMINAR_ESTRUCTURA
ANALIZAR_RECORD	DETERMINAR_ESTRUCTURA
ANALIZAR_REPEAT	DETERMINAR_ESTRUCTURA
ANALIZAR_WHILE	DETERMINAR_ESTRUCTURA
ANALIZAR_WITH	DETERMINAR_ESTRUCTURA
ARMAR_TOKEN	OBTENER_TOKEN
ASIGNAR_PROCEDIMIENTOS	ANALIZAR_ESTRUCTURA
	NIVEL_ANIDAMIENTO
ASIGNAR_PROCEDIMIENTOS_CONTENIDOS	NIVEL_ANIDAMIENTO
ASIGNAR_PROCEDIMIENTOS_LLAMADORES	NIVEL_ANIDAMIENTO
ASIGNAR_PROCEDIMIENTOS_LLAMADOS	NIVEL_ANIDAMIENTO
ASIGNAR_SUBPROGRAMAS	NIVEL_ANIDAMIENTO

APENDICE

Mensajes de error más comunes (runtime errors):

Error:
Archivo no encontrado, proceso de documentación cancelado.
Descripción:
Ocurre cuando el documentador detecta una referencia a un archivo que no se encuentra dentro del camino (path) establecido por el usuario.

Error:
Error en la lectura del disco, proceso de documentación cancelado.
Descripción:
Ocurre cuando se presentó algún problema en la lectura del disco donde se encuentra la información a documentar.

Error:
Error en la escritura del disco, proceso de documentación cancelado.
Descripción:
Ocurre cuando no existe suficiente espacio en el disco para escribir el archivo de salida (SALIDA.PDO) de la opción de generación de indentación.

Error:
Estructuras inválidas, proceso de documentación cancelado.
Descripción:
Ocurre cuando el archivo que se desea documentar no cumple con las características necesarias para procesarlo, es decir, no se encuentra libre de errores.

Error:
Unidad no lista, proceso de documentación cancelado.
Descripción:
Ocurre cuando las salida del proceso de documentación no puede ser almacenada en el disco por no encontrarse presente en la unidad.

Error:
Falla de Hardware, proceso de documentación cancelado.
Descripción:
Este es el error más serio, y se presenta cuando el equipo de computo se encuentra dañado por causas ajenas al sistema.

Error:
Overflow nivel de documentación, proceso de documentación. cancelado.
Descripción:
Ocurre cuando se ha sobrepasado el nivel máximo de anidamiento entre estructuras (aproximadamente 20).

Error:
Sobreflujo del Heap de memoria, proceso de documentación cancelado.
Descripción:
Ocurre cuando el sistema a documentar es demasido extenso y las listas de relaciones no pueden ser elaboradas por falta de espacio en memoria. El límite máximo del heap de memoria es de 64 Kbytes.