62

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"VISUALIZACION DE LA REFLEXION DE ONDAS DE CHOQUE DEBILES EN EL INTERIOR DE DIFERENTES REFLECTORES ELIPTICOS"

FALLA DE ORIGEN

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

LUIS LAGAR PONCE

DIRECTOR DE TESIS:
DR. FERNANDO ENRIQUE PRIETO CALDERON

MEXICO, D. F. 1995

VISUALIZACION DE LA REFLEXION DE ONDAS DE CHOQUE DEBILES EN EL INTERIOR DE DIFERENTES REFLECTORES ELIPTICOS.

A Dios por ser el centro de mi vida.

A mi abuela Carmen, con
todo el amor del mundo.

A mis padres.

A mis hermanas.

A mis tíos.

A mis primos.

En memoria de mi
abuelo Ignacio.

"Hay hombres que luchan un día y son buenos.
Hay quienes luchan un año y son mejores.
Hay quienes luchan muchos años y son muy buenos.
Pero hay los que luchan toda la vida; esos son
los imprescindibles."

B. BRECHT

AGRADECIMIENTOS

Antes que nada quiero agradecer a una persona muy especial para mí, alguien a quien admiro y respeto; un hombre con una gran trayectoria en las ciencias: al Dr. Fernando E. Prieto Calderón, por su paciencia, dedicación y disposición para la dirección de esta tesis.

Agradezco también al Dr. Achim M. Loske Mehling por dirigir esta tesis, seguir muy de cerca la elaboración del sistema que desarrollé y por todo el apoyo que me brindó siempre.

A Cecilia Aguilar por su apoyo en la bibliografía de este trabajo, y sobre todo por su sincera amistad.

Al Director de la Facultad de Ingeniería de la U.N.A.M., Ing. José Manuel Covarrubias Solís y al Jefe de la División de Ingeniería Eléctrica, Electrónica y en Computación de la Facultad de Ingeniería de la U.N.A.M., Ing. Salvador Landeros Ayala por todo su apoyo en esta tesis.

A la Ing. Dora Carmen Gálvez por aclarar algunas de mis dudas.

Al Instituto de Física de la U.N.A.M., a Foxboro S.A., al Fís. Raymundo Hugo Rangel, al personal del Laboratorio de Cómputo de la División de Ingeniería Eléctrica, Electrónica y en Computación de la Facultad de Ingeniería de la U.N.A.M., a Roberto Renato Jiménez Cabrera y a María Guadalupe Urzúa de la Cruz por las facilidades en el uso de los equipos de cómputo.

A mi tío, el Ing. Sergio Dávila Sánchez por todos sus consejos en mis inicios dentro del ambiente profesional.

A mis sinodales: Ing. Adolfo A. Millán Nájera, Ing. Jaquelina López Barriéntos, Ing. Alberto Templos Carbajal y Fís. Raymundo Hugo Rangel Gutiérrez por la revisión y sugerencias acerca de esta tesis.

A la Sra. Juanita Jasso el apoyo en los trabajos de tipo secretarial y por el apoyo que me ha brindado durante mi estancia como profesor en la Facultad de Ingeniería. A Noé Villalobos por el apoyo en la impresión de este trabajo.

A todos mis amigos, por haber creído siempre en que algún día lograría finalizar este trabajo.

De un modo muy especial a Dios, por haber infundido en mí el espíritu de servicio hacia los demás por medio de esta carrera. Quiero agradecer a Carmen Sánchez Sierra, mi abuela por todos estos años en que dió lo mejor de su vida hacia mí. A mis padres, a mis hermanas, a mis tíos y a mis primos por tener su fé puesta en mí.

Desde luego a la Facultad de Ingeniería de la U.N.A.M., a todos mis maestros por su tiempo y dedicación para formarme como ingeniero, a todos mis compañeros y a todos mis alumnos.

CONTENIDO

CAPITULO I

INTRODUCCION

La computación tiene un sinúmero de aplicaciones en las ciencias como por ejemplo en la física, la química, las matemáticas, la medicina, las ciencias humanas y sociales, la administración y la ingeniería.

La aplicación de este trabajo será en el campo de la física, específicamente en la *GENERACION DE ONDAS DE CHOQUE DEBILES EN AGUA POR ROMPIMIENTO ELECTRICO*.[1,2]

Este tema es novedoso dentro del área de la computación, por lo que en esta introducción se explicará de manera breve qué es una onda de choque, cómo se genera, cuáles son sus aplicaciones más comunes y cuál es la tarea del programa de cómputo desarrollado.

Una onda de choque es una onda mecánica que se propaga a través de un medio, ésto significa que requiere de algún material para poder propagarse. No hay que confundirse con la luz que se propaga en el vacío y es una onda electromagnética.[3] Una onda de choque débil en agua puede producirse de diferentes formas: por ejemplo, por medio de microexplosivos, rayo laser o como en el caso tratado aquí, por rompimiento eléctrico en agua.

Esta onda es una discontinuidad de presión, es decir, un cambio repentino de una presión baja a una presión muy alta (compresión) y que decrece en un tiempo sumamente corto a su valor original, pasando antes de ello por un estado de rarefacción o presión por debajo de la presión de equilibrio (figura I.1).[4]

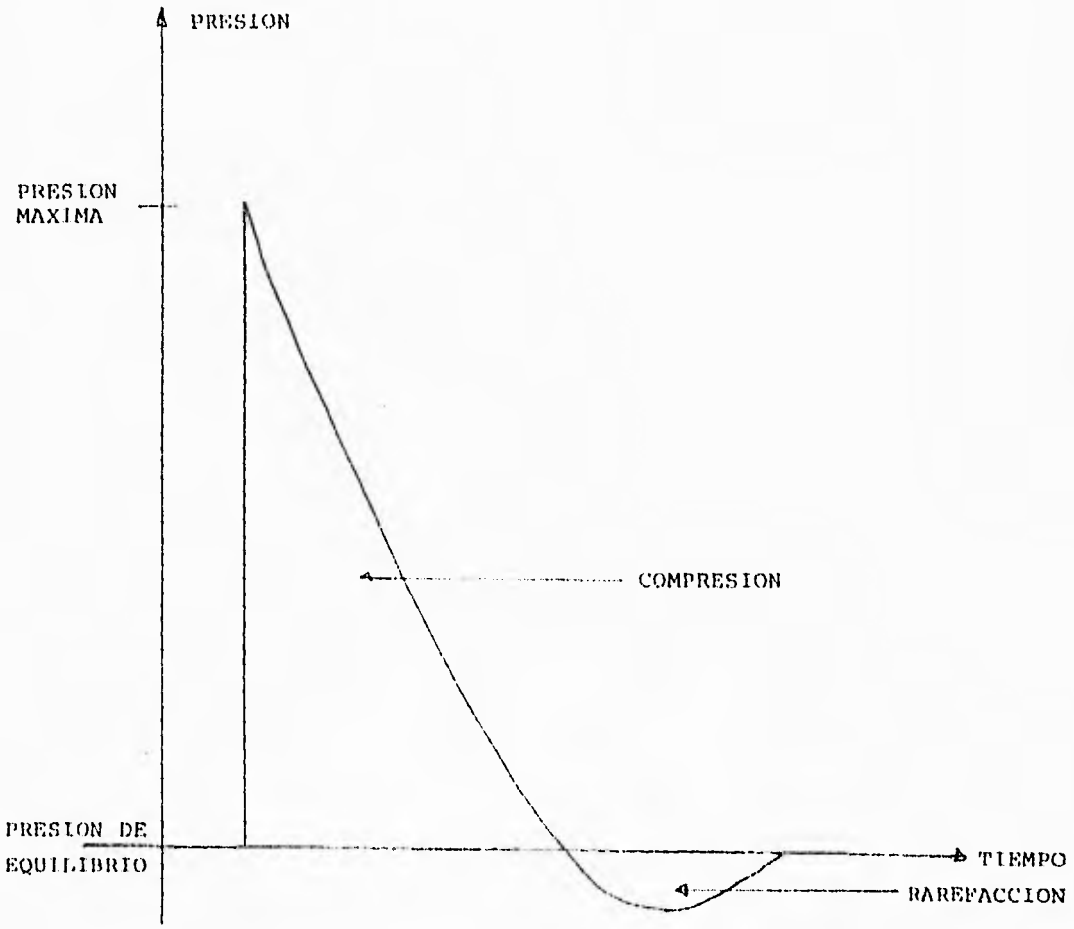

Figura 1.1 Perfil de una onda de choque

¿ En qué forma puede generarse una onda de choque ? En una explosión (de una bomba, por ejemplo), el material que es inicialmente sólido se convierte en un gas muy caliente en un tiempo tan corto, que ocupa el mismo volumen del material sólido. La presión que es muy grande hace que el gas se expanda a velocidades muy altas, impulsando y comprimiendo al medio circundante. Cuando este gas se expande se va enfriando poco a poco y origina que el medio comprimido se separe del elemento gaseoso y que la discontinuidad de presión producida se propague a través del aire. Si hay algún objeto o alguna persona dentro del área donde ésto se produce, puede sufrir daños. A medida que se va expandiendo, la presión va decreciendo de tal forma que la onda de choque se convierte en una onda sonora que no provoca daños, es decir, si alguien se encuentra en esa área solo escuchará la explosión.[5]

La generación de ondas de choque por rompimiento eléctrico se produce mediante una descarga eléctrica entre dos electrodos. La descarga forma un canal de agua evaporada entre los electrodos, que al expanderse va comprimiendo al medio (agua). Con ello surge una discontinuidad de presión que se va propagando por el agua.

Cuando la burbuja producida por la descarga eléctrica crece, su presión interna y su velocidad decrecen, por lo cual se separa de la onda de choque. En ese instante la burbuja comienza a colapsarse (figura I.2).

Si esta onda de choque es reflejada, los daños pueden producirse en un punto lejano al de la generación.[6]

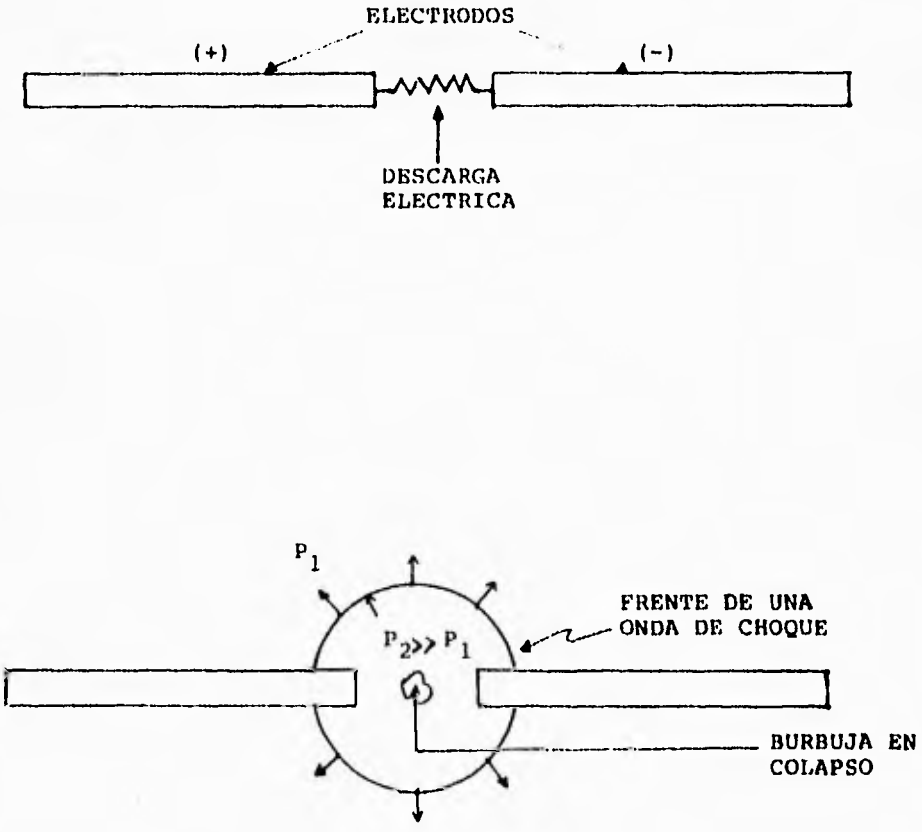

Figura I.2 Descarga eléctrica y generación de una onda de choque en agua.

En el caso descrito aquí, la onda de choque es reflejada mediante un espejo en forma de elipsoide de revolución, de acuerdo a las leyes de la acústica.

La descarga se produce en el foco más cercano al elipsoide, de manera que gran parte de la energía se concentra en el segundo foco, formándose en ese sitio una región de máxima energía, que es aquella cuya presión tiene en todo punto un valor mayor o igual al 90 % de la presión máxima, y puede provocar daños a algún objeto situado en ese punto (figura I.3).

Las discontinuidades de presión producidas tienen valores por debajo de 1 kbar (aproximadamente) y se denominan *ondas de choque débiles*.

El voltaje con el que se generan las descargas es del orden de las decenas de kilovoltios. La separación entre los electrodos es de milímetros o fracciones de milímetros y el tiempo en el que se produce la descarga eléctrica y la onda de choque es del orden de los microsegundos.[7]

Una de las aplicaciones más comunes se da en el campo de la medicina. Por medio de ondas de choque débiles se destruyen cálculos renales y biliares. Con ello disminuyen los casos en que se requiere de otras técnicas invasivas o de intervención quirúrgica. A esta aplicación se le conoce como *litotripsia extracorporal*.[8]

El objetivo del programa de cómputo descrito aquí es el de simular la propagación y reflexión de las ondas de choque débiles en agua por rompimiento eléctrico.

Para mayor información acerca del tema de la generación de ondas de choque, se proporciona la bibliografía al final de este trabajo.

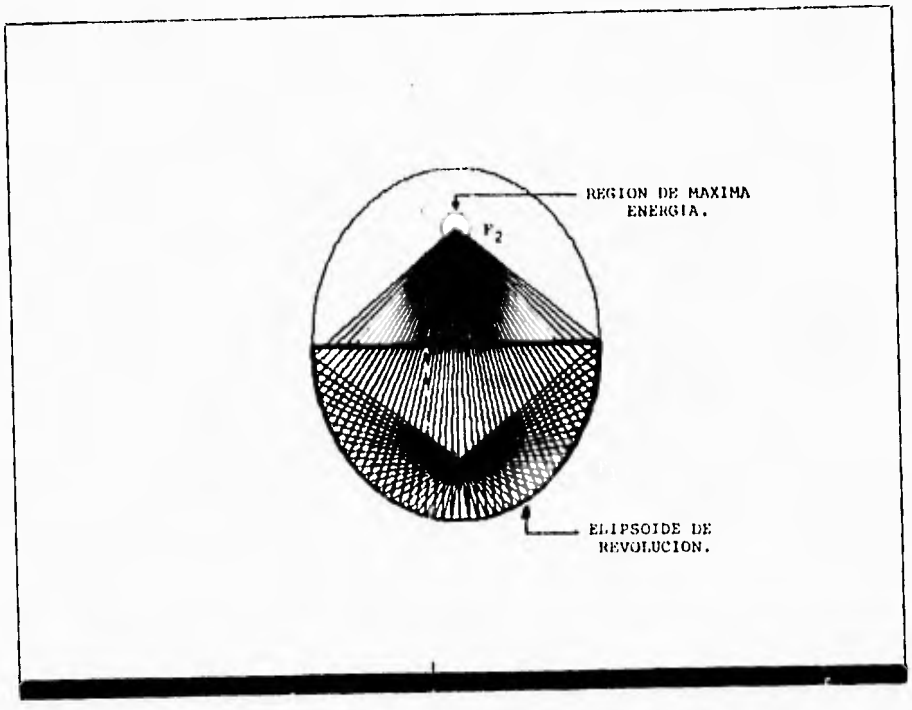

Figura I.3 Diagrama de rayos reflejados en un elipsoide de revolución truncado.

CAPITULO II

PLANEACION DEL PROYECTO

II.1 Introducción a la problemática que se desea resolver con el programa de cómputo.

Para el diseño de los reflectores descritos en el capítulo anterior es útil contar con una herramienta para mostrar la forma de los diferentes elipsoides de revolución y la reflexión de las ondas de choque débiles en ellos. Es también conveniente tener un programa de computación que calcule los parámetros necesarios para su construcción.

El objetivo de esta tesis es crear un programa que proporcione información al usuario sobre la forma y los parámetros de un reflector elíptico y la reflexión de las ondas de choque en su interior.

II.2 Estructura del programa.

La elaboración de este programa tuvo las siguientes fases[9,10]:

PLANEACION

ANALISIS DE REQUERIMIENTOS

DISEÑO

- Diseño externo.
- Diseño arquitectónico.
- Diseño detallado.

CONSTRUCCION

PRUEBAS

INSTALACION.

En la fase de planeación se especificaron los alcances del programa, los ambientes de operación, desarrollo y mantenimiento, los tipos de resultados que se esperaron, las aplicaciones específicas, las estrategias de solución para resolver el problema en cuestión, así como un glosario de términos empleados durante el desarrollo del programa.[11]

En la fase del análisis de requerimientos se especificaron las ecuaciones matemáticas que se emplearon para el desarrollo del programa, se describió el funcionamiento del programa a grandes rasgos y se dieron los criterios de aceptación.[12,13]

En la fase de diseño se describió la estructura del programa, la forma como se comunicaron los parámetros internos y se describirieron todas las entradas y salidas. El diseño se realizó desde tres puntos de vista: externo (estructura general del programa), arquitectónico (carta estructurada) y detallado (pseudocódigo).[14,15]

En la fase de construcción se implementó el programa en un lenguaje estructurado, mencionando las características de dicho lenguaje. Se adjuntó el listado del programa, así como su documentación dentro del mismo.[16,17]

En la fase de pruebas se llevó a cabo la ejecución del programa para diferentes valores de entrada, depurándolo y dándole refinamiento. Dicha prueba se realizó en diferentes tarjetas de gráficos para comprobar su portabilidad.[18,19]

Por último, se instaló el sistema probado y aceptado adjuntando los archivos de manejo de gráficos para su correcto funcionamiento. Se proporcionó un respaldo protegido contra escritura, evitando así alguna destrucción accidental o intencional, así como la posible presencia de algún virus informático.[20]

II.3 Alcances.

Además de ahorrar tiempo, esfuerzo y costos, se desea que el programa funcione en la mayoría de los equipos de cómputo personales, especialmente en sus tarjetas de gráficos(VGA, CGA, EGA, HERCULES).[21]

Es posible que el programa, además de trabajar con el sistema operativo MS-DOS[22] se adapte para que funcione en el sistema UNIX V[23] y, en ámbos casos el sistema sea fácil de manejar.

II.4 Aplicaciones específicas.

El programa es una herramienta auxiliar para el diseño de reflectores elípticos así como una información gráfica preliminar de la reflexión de una onda de choque en ellos.

II.5 Tipos de resultados.

Existen dos tipos de resultados: gráfico y numérico. En los resultados gráficos encontramos el desplegado de la forma tridimensional del reflector elíptico y la simulación de la reflexión de las ondas de choque en el interior de éste, visto en un plano bidimensional.

En los resultados numéricos encontramos los valores de las medidas características del elipsoide, tales como: su diámetro, su volumen, la magnitud del radiovector, el ángulo formado entre éste y los ejes y su factor de aprovechamiento (ver figura II.1).

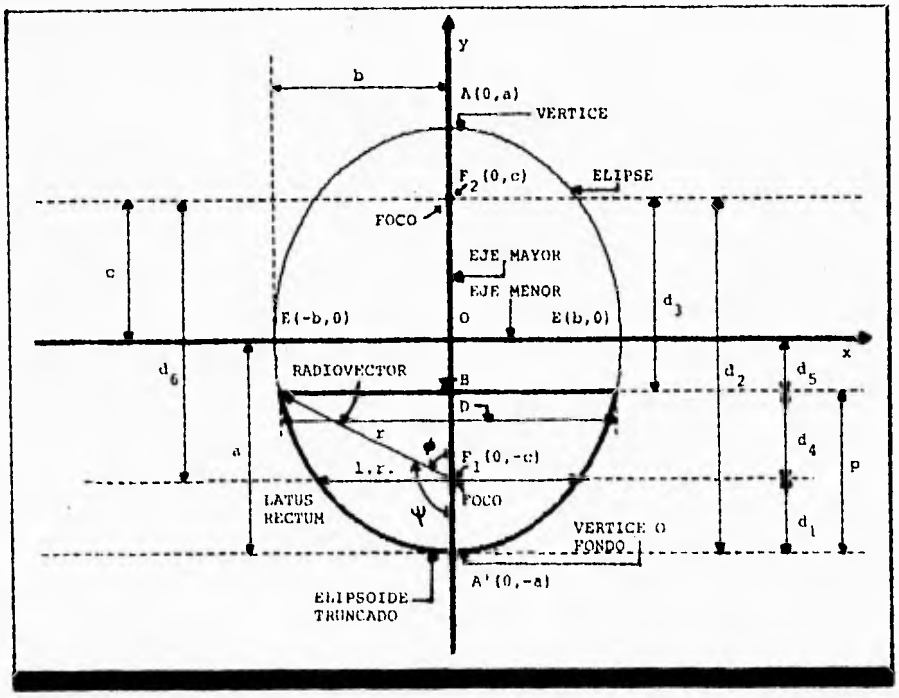

Figura II.1 Corte de un elipsoide truncado con sus parámetros.

II.6 Ambiente de desarrollo, operación y mantenimiento.

Mencionamos en el punto I.3 que el sistema podría trabajar en MS-DOS* y UNIX V **, pero, debido a que la mayoría de los usuarios emplean sistemas personales (PC), el programa se desarrolló en el ambiente MS-DOS.

El programa fue editado, compilado, depurado y ejecutado en TURBO C (Versión 2.0)***.[24] Este paquete tiene la característica de compilar, ligar y ejecutar los programas en un tiempo muy corto y tiene además un contenido rico en cuanto a la biblioteca de funciones predefinidas.

Por otra parte, se manejaron las tarjetas CGA (Color Graphics Adapter), EGA (Enhanced Graphics Adapter), VGA (Video Graphics Adapter) y HERCULES.

II.7 Estrategia de solución.

Se realizaron primero cálculos numéricos en cada una de las ecuaciones matemáticas para probar el correcto funcionamiento del programa.

Una vez logrado este paso se hizo un procedimiento de conversión de los valores obtenidos de las ecuaciones matemáticas a unidades de pantalla (pixeles) y se realizaron también cálculos numéricos para ajustar los límites de la pantalla.[25]

* MS-DOS es una marca registrada de Microsoft Inc.

** UNIX es una marca registrada de Bell Laboratories Inc.

*** TURBO C es una marca registrada de Borland International Inc.

Los valores numéricos en pixeles se desplegaron en pantalla; para ésto se probó para diferentes intervalos hasta lograr el resultado deseado.

Se complementó el programa mediante funciones predefinidas en el lenguaje C.[26]

La vista tridimensional se construyó por la intersección de dos elipses (figura II.2). Una de ellas representa el contorno del reflector elíptico (elipse 2), la otra representa el borde superior que, aunque en realidad es circular, aquí se grafica como una elipse para dar la sensación tridimensional (elipse 1). Como puede observarse en la figura, el diámetro "D" en la elipse 2 es igual al semieje mayor "a'" de la elipse 1.

Los rayos se generararon mediante rectas dirigidas del primer foco al reflector y del reflector al segundo foco, tomando en cuenta las leyes de reflexión de la acústica geométrica.

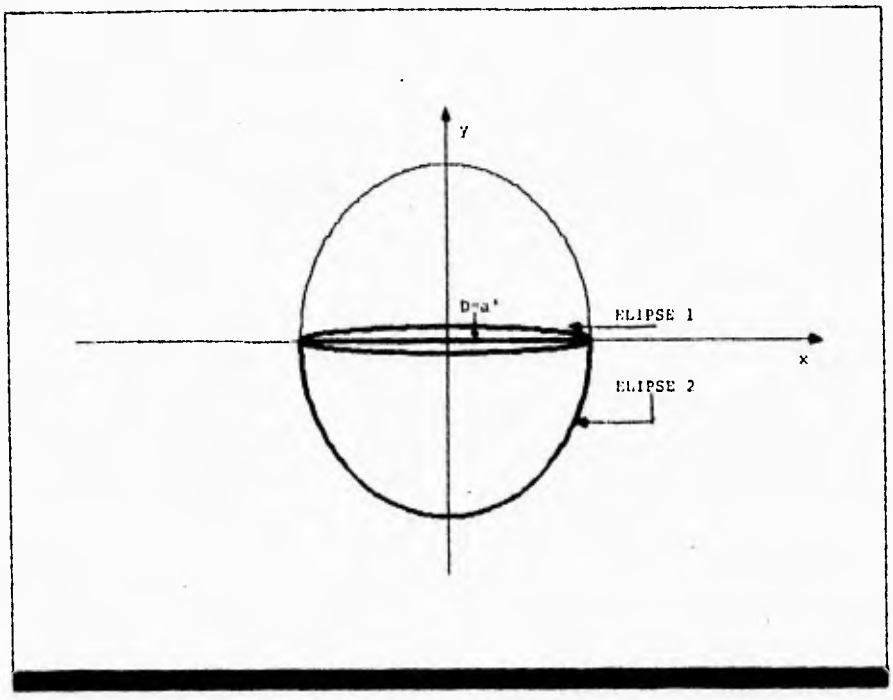

Figura II.2 Estrategia para dibujar la vista tridimensional del reflector elíptico.

II.8 Glosario de términos.

Elipse.

Curva generada por la intersección de un cono recto de base circular y un plano oblicuo al eje del cono. Es el lugar geométrico de los puntos de un plano cuya suma de distancias a dos puntos fijos F_1 y F_2 llamados *focos*, es constante. La recta AA' que pasa por los focos y está limitada por la curva es el *eje mayor*; la perpendicular EE' en el punto medio -*centro*- es el *eje menor*; los puntos A,A',E,E' de intersección de los ejes con la curva son los *vértices* de ésta; los segmentos que unen cada punto de la elipse con los focos son los *radios vectores* del punto; el segmento FF' se llama *distancia focal* y se representa por 2c y los ejes mayor y menor por 2a y 2b, respectivamente (ver figura II.1).[27]

Elipsoide.

Figura geométrica tridimensional generada por la rotación de una elipse alrededor de uno de sus semiejes.[28]

Excentricidad.

Es una constante que define la forma de una curva cónica. En el caso de la elipse su ecuación es $e=c/a$, donde "c" es la distancia del origen a los focos (ver elipse), y "a" es el semieje mayor. La excentricidad de la elipse es mayor que cero y menor que uno. Cuando la excentricidad es igual a cero, la figura es una circunferencia. Cuando la excentricidad es igual a uno, resulta ser una línea recta.[29]

Factor de aprovechamiento de un reflector elipsoidal.

Este factor da información acerca de la cantidad de energía concentrada en el foco F_1.[30]

Foco.

(ver elipse)

Frente de onda.

Es la región (discontinuidad) en la cual la presión aumenta del valor inicial al máximo (figura I.1, capítulo I).

Mexilit soft.

Es el programa de cómputo para el diseño de reflectores elípticos y visualización de ondas de choque reflejadas en su interior.

Su nombre fué tomado del generador de ondas de choque *Mexilit I*, del Instituto de Física de la U.N.A.M., en el cual se generan ondas de choque débiles en agua por rompimiento eléctrico.[31]

Mexilit-soft proviene de las palabras *Mexico-Litotripsia* y *Software*.

Latus rectum.

Es el diámetro del círculo generado por la intersección del elipsoide de revolución con un plano perpendicular al eje de simetría a la altura de los focos.[32]

Onda de choque.

Es una discontinuidad de presión que se propaga a través de un medio.

Ondas de choque débiles.

Son aquellas ondas de choque cuyas presiones son del orden de los cientos de bars.

Origen.

Punto de partida para medir distancias. Punto de intersección de los ejes coordenados.[33]

Profundidad de un reflector elipsoidal.

Es la distancia del vértice a un punto que se encuentra sobre el eje de simetría a la altura del borde del reflector elíptico (ver figura II.3).[34]

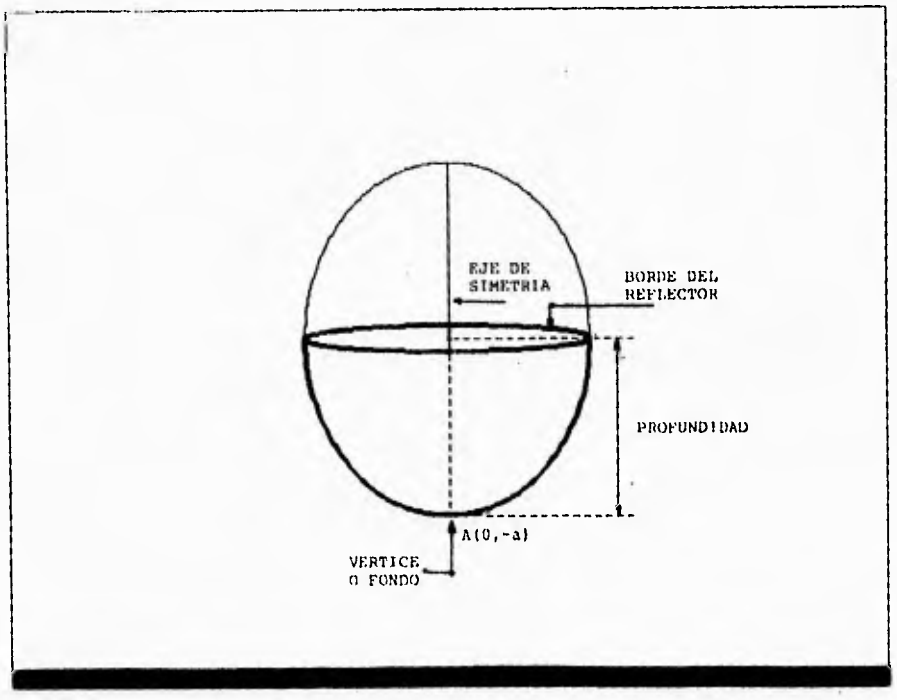

Figura II.3 Profundidad de un reflector elipsoidal.

Rayos perpendiculares al frente de onda.

Son líneas perpendiculares al frente de una onda de choque. En este caso van desde un foco del elipsoide hacia la pared interna del mismo.

La mayoría de esos rayos son reflejados hacia un segundo foco, creándose una región de máxima energía (figura I.3, capítulo I).

(*véase también frente de onda*).

Reflector elíptico.

Es un espejo metálico en forma de elipsoide de revolución truncado que es empleado para concentrar las ondas de choque hacia el segundo foco.[35]

Radián.

Arco de circunferencia cuya longitud es igual al radio. La circunferencia tiene, por tanto, 2pi radianes, donde pi = 3.14159. La medida del radián en grados es 57^{0} 17' 44''.[36]

Radiovector.

Es la recta comprendida entre el borde interior del elipsoide y el foco interior del mismo.

Semieje mayor.

(ver elipse).

Semieje menor.

(ver elipse).

Vértice.

(ver elipse)

Volumen.

En esta tesis se refiere a la capacidad que tiene el reflector si se llena de agua.

CAPITULO III

ANALISIS DE REQUERIMIENTOS

III.1 Limitaciones del programa.

Todo sistema tiene sus ventajas y sus desventajas. Comentamos en el capítulo anterior que el programa podría ser adaptable para el ambiente operativo UNIX V. Sin embargo, las funciones predefinidas para el manejo de gráficos dentro de las librerías del lenguaje C de UNIX no son iguales y, para este caso, tendrían que hacerse ciertas modificaciones al programa para poder ejecutarse en el ambiente antes mencionado.[37]

Otra limitación del proyecto es el manejo de adaptadores gráficos. Debido a que cada adaptador gráfico tiene diferente resolución, se debe crear una versión del programa para cada caso.

III.2 Ecuaciones y parámetros que se utilizarán y se manejarán en el programa.

La forma del reflector (como se mencionó en la introducción) es la de un elipsoide, por lo tanto el programa tendrá que graficar la ecuación de la elipse, teniendo como parámetros:

"a" = semieje mayor de la elipse.

"b" = semieje menor de la elipse.

"p" = profundidad del elipsoide.

(Ver la figura II.1, capítulo II)

Para este caso la ecuación de la elipse está dada por

$$\frac{x^2}{b^2} + \frac{y^2}{a^2} = 1 \quad ; a > b, \quad \text{(III.1)}$$

donde el semieje menor "b" se encuentra sobre el eje horizontal, y el semieje mayor "a" se encuentra sobre el eje vertical.

Para graficar el borde del reflector se empleará de igual modo el caso contrario: cuando el semieje mayor "a" se encuentra sobre el eje horizontal, y el semieje menor "b" sobre el eje vertical:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad ; \quad a > b, \qquad \text{(III.2)}$$

Despejando la variable "y" de las ecuaciones (III.1) y (III.2), para la graficación resultarán las ecuaciones (III.3) y (III.4) respectivamente:

$$y_1 = \pm b \left(1 - x^2/a^2\right)^{1/2} \qquad \text{(III.3)}$$

$$y_2 = \pm a \left(1 - x^2/b^2\right)^{1/2} \qquad \text{(III.4)}$$

para a > b.

La ecuación (III.3) representa el borde del reflector en la forma tridimensional. Como se mencionó anteriormente, en realidad la vista superior es circular, pero se dibuja en forma de elipse para dar la sensación tridimensional, por lo cual la ecuación (III.3) se escribe así:

$$y' = \pm b \left(1 - x^2/a'^2\right)^{1/2} \qquad \text{(III.5)}$$

donde "a'" es igual al diámetro del borde del reflector (ver figura II.2, capítulo II).

El cálculo de la distancia "c" del origen a los focos cuyas coordenadas son $F_1(0,-c)$ y $F_2(0,c)$ está dado por:

$$c = (a^2 - b^2)^{1/2} \quad ; \; a > b. \qquad \text{(III.6)}$$

El *latus rectum*, es decir, el diámetro del reflector a la altura de los focos está dado por:

$$lr = 2b^2/a \; . \qquad \text{(III.7)}$$

La distancia del fondo del reflector al punto focal $F_1(0,-c)$ está dada por:

$$d_1 = a - c \; . \qquad \text{(III.8)}$$

La distancia comprendida del fondo del reflector al punto focal $F_2(0,c)$ está dada por:

$$d_2 = a + c \; . \qquad \text{(III.9)}$$

La distancia de la parte superior del reflector al foco $F_2(0,c)$ está dada por:

$$d_3 = d_2 - p \; . \qquad \text{(III.10)}$$

La distancia entre el foco $F_1(0,-c)$ y el punto "B" del reflector está dada por:

$$d_4 = p - d_1 \; . \qquad \text{(III.11)}$$

La distancia entre el punto "B" y el origen está dada por:

$$d_5 = a - p \; . \qquad \text{(III.12)}$$

La distancia entre los dos focos de la elipse está dada por:

$$d_6 = 2c \; . \qquad \text{(III.13)}$$

La distancia entre los puntos de intersección de la recta $y = -d_5$ con la elipse, es el máximo diámetro interior del reflector dado por:

$$D = 2b/a\ (a^2 - d^2_5)^{1/2} . \qquad \text{(III.14)}$$

La magnitud del radiovector "r" que va del punto focal F_1 hasta el borde interior del reflector es

$$r = (D^2/4 + d^2_4)^{1/2} . \qquad \text{(III.15)}$$

El ángulo formado por este vector con el eje "y" se denominará "fi", mientras que su complementario será "psi".

$$fi = \arctan (D/2d_4) . \qquad \text{(III.16)}$$

Cuando "p" sea mayor que "a", el valor d_5 se vuelve negativo, lo cual significa que el borde del reflector sobrepasa el eje "x".

El factor de aprovechamiento "Fa" se calcula de la siguiente forma:

$$Fa = 1/2(\ 1 - \cos (psi)); \quad 0 <= psi <= pi, \qquad \text{(III.17)}$$

$$pi = 3.14159.$$

En la figura II.1 se muestra un elipsoide truncado con las variables descritas anteriormente.[38]

Dentro del programa, los valores reales obtenidos de las ecuaciones antes mencionadas deberán ser truncados y convertidos a unidades de pantalla (pixeles), dentro un marco de proyección o ventana (window).[39]

El dibujo de los rayos perpendiculares al frente de onda será generado por las ecuaciones paramétricas de la elipse[40]:

$$x = b \cos (teta) \qquad (III.18)$$

$$y = a \operatorname{sen} (teta). \qquad (III.19)$$

Los parámetros que definen los límites superiores e inferiores del sistema cartesiano de coordenadas (x,y) son (ver figura III.1):[41]

xwr = límite a la derecha x_0

xwl = límite a la izquierda x_1

ywt = límite hacia arriba y_1

ywb = límite hacia abajo y_0

Los parámetros de las coordenadas de pantalla son:

xvr = límite a la derecha

xvl = límite a la izquierda

yvt = límite hacia arriba

yvb = límite hacia abajo

Las coordenadas de pantalla son llamadas (xpc,ypc).

Para convertir las coordenadas (x,y) del sistema cartesiano (también llamadas coordenadas reales) a coordenadas de pantalla, es necesario encontrar primero la relación entre los intervalos $(x_1 - x_0)$ y $(y_1 - y_0)$ para el caso del sistema cartesiano y los intervalos de pantalla.

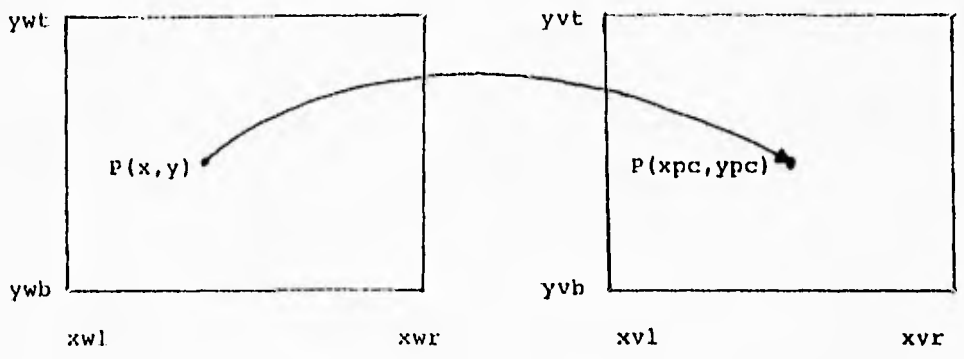

Figura III.1 Conversión de coordenadas reales a coordenadas de pantalla.

Para el caso del eje "x" los intervalos se obtienen restando el valor del límite a la derecha real "xwr" menos el límite a la izquierda "xwl" para el sistema cartesiano:

$$xwr - xwl. \qquad \text{(III.20)}$$

Lo mismo se hace para las coordenadas de pantalla:

$$xvr - xvl. \qquad \text{(III.21)}$$

La relación está dada por (III.21) entre (III.20):

$$A = xvr-xvl/xwr-xwl. \qquad \text{(III.22)}$$

"A", que está dada por la ecuación (III.22) determina la relación entre el intervalo "x" real y el intervalo "x" de pantalla.

Ahora hay que encontrar una traslación para convertir datos reales a pixeles, por lo que la ecuación (III.22) se multiplica por "xwl" y se le resta el producto a "xvl". Como resultado se tiene la ecuación (III.23):

$$B = xvl - (xvl-xvr/xwl-xwr)\ xwl. \qquad \text{(III.23)}$$

Simplificando (III.22) se obtiene:

$$B = xvl - Axwl. \qquad \text{(III.24)}$$

Para convertir un valor "x" real a un valor "x" en pixeles, se obtiene el producto de la ecuación (III.22) por "x" y sumándole la traslación "B" (ecuación (III.24)):

$$xpc = (xvr-xvl/xwr-xwl)\ x + (xvl - Axwl). \qquad \text{(III.25).}$$

Simplificando se obtiene:

$$xpc = Ax + B. \qquad \text{(III.26).}$$

El mismo procedimiento se usa para convertir un valor "y" real a un valor "y" en pixeles:

La relación entre los intervalos de "y" real y "y" en pixeles es:

$$ywt - ywb. \qquad \text{(III.27)}$$

Luego:

$$yvt - yvb. \qquad \text{(III.28)}$$

La relación denominada "C" se obtiene dividiendo (III.28) entre (III.27):

$$C = yvt-yvb/ywt-ywb. \qquad \text{(III.29)}$$

La traslación para "y" real se obtiene con los valores de las cotas inferiores de "y" real y "y" en pixeles respectivamente:

$$D = yvb - (yvt-yvb/ywt-ywb)\ ywb. \qquad \text{(III.30)}$$

Simplificando (III.29) se obtiene:

$$D = yvb - Cywb. \qquad \text{(III.31)}$$

Para convertir "y" a pixeles se realiza el mismo procedimiento que con "x" :

$$ypc = (yvt-yvb/ywt-ywb)y + ywb - Cywb. \qquad \text{(III.32)}$$

Simplificando (III.32) queda:

$$ypc = Cy + D. \qquad \text{(III.33)}$$

III.3 Descripción funcional

El sistema está constituido por un programa principal, una rutina que leerá los datos de entrada por parte del usuario. Dichos valores, validados por la misma rutina serán almacenados en memoria en donde serán empleados en otras tres rutinas que a continuación se mencionarán.

El sistema también tiene una rutina que calcula los parámetros y dibuja un elipsoide de revolución en su forma tridimensional. Esta rutina se llama *elipsoide*.

Otra rutina que calcula los principales parámetros, dibuja el elipsoide en un plano bidimensional y muestra una representación gráfica de la reflexión de los rayos perpendiculares al frente de onda en su interior se llama *ondas-choque*.

La cuarta rutina calcula otros valores de parámetros necesarios para su construcción. Esta rutina se llama *información*.

Las rutinas *elipsoide*, *ondas-choque* e *información* pueden ejecutarse cada una tantas veces como se quiera, para los mismos valores vigentes, hasta que se introduzcan nuevos valores en los parámetros (si se desea). La ventaja es que no hay que introducir los valores de entrada cada vez que se ejecute cada una de las rutinas.

El programa cuenta con ayudas en bloques, de manera general y particular, así como una demostración gráfica de un elipsoide truncado con todos sus parámetros. Todas las ayudas se encuentran almacenadas en un programa llamado *ayudas.c*.

El programa cuenta también con un sistema de menús que hará su empleo más fácil; este sistema se encuentra almacenado en un programa llamado *menu.c*. Hablaremos con mayor detalle acerca de la interacción hombre-máquina del sistema de menús en el capítulo IV.

La figura III.2 muestra el diagrama de flujo de datos (DFD) que describe el funcionamiento general del programa.[42]

III.4 Criterios de aceptación.

Se realizarán pruebas para comprobar su eficiencia y confiabilidad.

El plan de pruebas consistirá en probar el sistema para diferentes valores de entrada.

Con seguridad, en el momento de realizar las pruebas se encontrarán anomalías durante la ejecución, lo cual permitirá que vaya mejorándose el sistema a fín de que el usuario no tenga que enfrentarse a ningún tipo de problemas al emplearlo.

También se realizarán pruebas previas a su instalación. En ellas se hará una verificación del programa en diferentes tarjetas de gráficos a fín de hacer el sistema adaptable a dichas tarjetas.

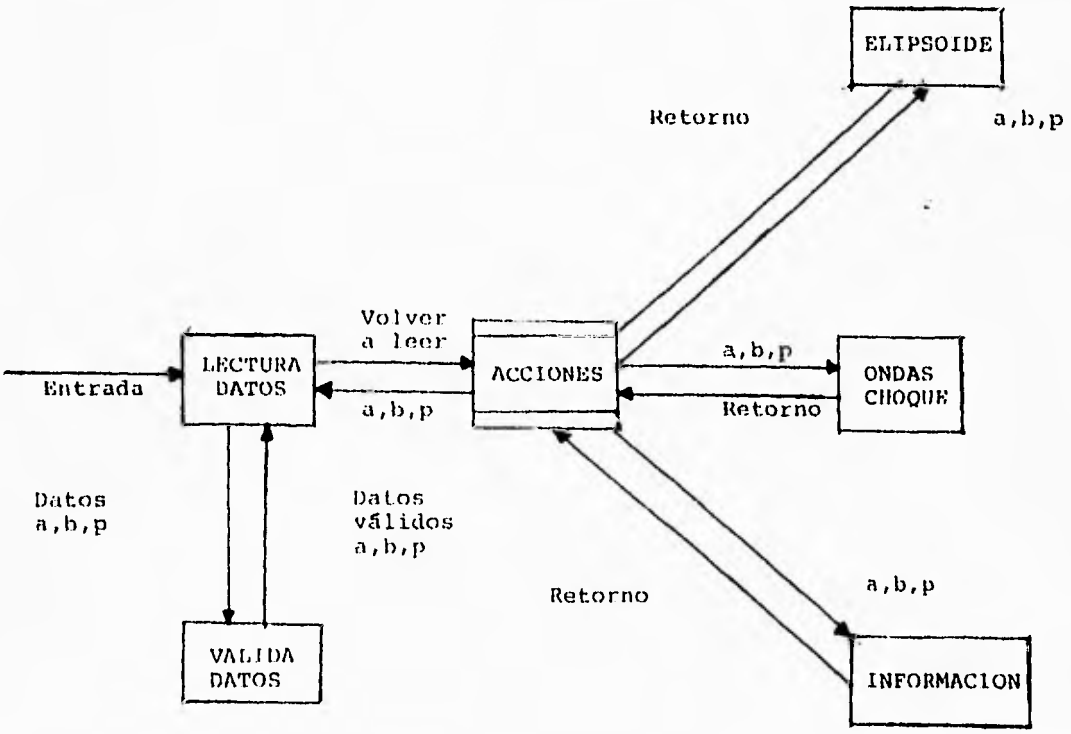

Figura III.2 Diagrama de flujo de datos (DFD) del programa.

CAPITULO IV.

DISEÑO DEL PROGRAMA

IV.1 Descripción del diseño.

Antes de pasar de lleno a la fase de diseño del sistema, se hará de nuevo énfasis en el objetivo del sistema. Como se dijo en el capítulo II, el objetivo es crear un sistema que facilite el diseño de reflectores elípticos y proporcione información correspondiente a la reflexión de las ondas de choque, la forma de un reflector elíptico y sus datos técnicos.

Los principales componentes de "software" y "hardware" con que se cuenta para la realización del programa son:

"Software": Las funciones de librerías, el compilador y el editor de Turbo C.

"Hardware": Un equipo PC sin importar el microprocesador con que se cuente. Se trabajará principalmente con tarjeta gráfica HERCULES, y las pruebas se harán en función de dicha tarjeta dadas las necesidades específicas del usuario.

Las interfases que van a emplearse en la realización del programa (como se ya se mencionó), son los programas de extensión BGI (Borland Graphics Interface).[43,44] Dichos programas los proporciona Turbo C y su funcionamiento consiste en tener acceso a las tarjetas de gráficos HERCULES, CGA, VGA y EGA, para graficar en pantalla.

En cuanto a sus limitaciones, se mencionó en el capítulo III que se presentan problemas en cuanto a las librerías de gráficos entre UNIX y Turbo C: su incompatibilidad.

Ahora se mencionará otro problema. Parece contraproducente mencionar los alcances del sistema por un lado y, por otro lado, decir que el sistema trabajará en función de la tarjeta HERCULES. ¿ Qué sucede ?

La cuestión es que las diferentes tarjetas de memoria varían en su resolución de imagen, como se mencionó en el capítulo anterior. Mientras en una tarjeta se puede ver (por ejemplo) un círculo perfecto, en otras puede verse alargado o aplastado, como una elipse. De hecho cualquier sistema gráfico funciona en cualquiera de las tarjetas antes mencionadas pero de diferente forma. Otra situación es que las llamadas a color en las tarjetas CGA, EGA y VGA no funcionan en la tarjeta HERCULES, salvo los colores blanco y negro. La razón de lo anterior se debe a que HERCULES es una tarjeta estrictamente monocromática, por así decirlo, y no emplea tonos grises como lo hacen otras tarjetas monocromáticas. Una de las ventajas que tiene la tarjeta HERCULES es su buena resolución de imagen.

Dentro de la fase de diseño (como ya se mencionó en el capítulo II), existen dos niveles: *diseño externo y diseño interno.*[45] El diseño externo se mencionó en el capítulo anterior, cuando nos referimos a la descripción funcional del sistema por medio del diagrama de flujo de datos (DFD).

El diseño interno se compone a su vez de dos niveles: *diseño arquitectónico y diseño detallado.*

A continuación se hará énfasis al diseño arquitectónico. El diagrama de flujo de datos (DFD) visto en el capítulo III (figura III.2), es dividido por secciones para describir cuáles serán las entradas y cuáles serán las salidas.

Como puede apreciarse en esta figura, los bloques de lectura y validación de datos forman parte de un nuevo bloque que ahora será global. Ese bloque constituirá lo que se refiere a las entradas. Los bloques *elipsoide*, *ondas-choque* e *información*, constituirán el bloque de salidas.

El diseño arquitectónico se desarrollará en este caso, a través de la técnica del *diseño estructurado*, desarrollado por Larry L. Constantine.[46] Dicha técnica consiste en convertir el DFD en cartas de estructura. Para guiar el proceso, se utilizan heurísticas de diseño tales como el acoplamiento y la cohesión.

El acoplamiento mide la comunicación entre datos y elementos de control. En este caso, se puede decir que se trata de un acoplamiento por zonas de elementos, ya que existen elementos globales (semieje mayor "a", semieje menor "b" y profundidad "p"), los cuales son compartidos en forma selectiva entre las diversas rutinas que requieren los datos (*elipsoide*, *ondas-choque* e *información*).

La cohesión mide la relación interna de los elementos de un módulo. En este caso, de trata de una cohesión funcional, que representa un tipo fuerte de cohesión, dado que los elementos de cada módulo se encuentran relacionados al desempeño de una sola función.[47] Tal es el cálculo de cada una de las ecuaciones de la elipse, así como su graficación y los cálculos de los parámetros necesarios para la construcción de un reflector elíptico. Son instrucciones específicas concretas.

Una vez explicadas ambas heurísticas de diseño, se describirá la conversión del DFD a una carta estructurada.

Nótese en la figura IV.1 que existen cuatro elementos que no estaban descritos anteriormente en el DFD, tales son los bloques de control (menús), y los bloques de ayudas. Esto da una visión más detallada de qué es a lo que se pretende llegar.

El bloque del menú de acciones fué descrito en el DFD como un bloque en donde se mantienen los datos de entrada validados almacenados en memoria listos para ser procesados en el bloque de salidas; ahora es descrito formalmente como bloque de control.

El menú principal llama a cualquiera de las tres opciones: *el dibujo del elipsoide, la simulación de ondas de choque , ayuda general* y *salida del sistema*.

La primera opción envía una bandera para llamar a la lectura de datos, en donde el usuario introducirá valores. Si los datos son correctos (valores numéricos mayores de cero, dentro del intervalo de 30 a 500 mm), se envía una bandera para llamar al menú de acciones en donde (como ya se dijo) cada una de las opciones (*elipsoide, ondas-choque* e *información*) se pueden ejecutar una y otra vez con los mismos datos en memoria. Se puede probar con otros valores o regresar al menú principal.

El diseño detallado es también llamado *diseño procedural*.[48] Es el desgloce de todos los bloques descritos aquí y consta de una serie de pasos previos a su implementación en algún lenguaje; esto se hará por medio de *pseudocódigos*.

A continuación se mostrarán los pseudocódigos para el bloque de *lectura-datos*, para el bloque *elipsoide*, para el bloque *ondas-choque* y para el bloque *información*.

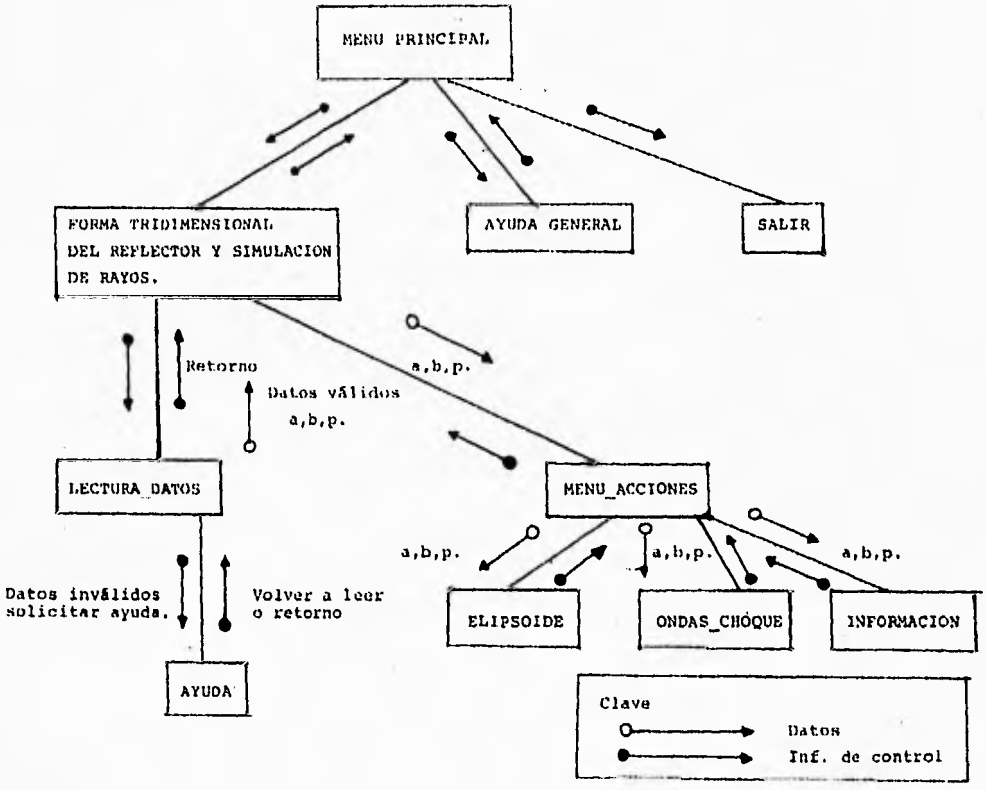

Figura VI.1 Diseño arquitectónico del programa.

Cabe mencionar que las palabras reservadas empleadas en los peudocódigos tienen cierta semejanza con las que tienen *Pascal*,[49] *ADA*[50] *y MODULA-2*.[51] Ello no significa que el programa vaya a implementarse en alguno de estos lenguajes. Las palabras reservadas son empleadas para darle cierta formalidad al diseño.

Este es el diseño procedural de la rutina *elipsoide* del programa NEXILIT-SOFT. En este procedimiento se inicializa la pantalla en modo gráfico. Se calculan los valores de "y" convirtiéndose éstos a pixeles para que posteriormente sean dibujados, formando así la forma tridimensional. Lo anterior se logra interponiendo la elipse calculada con otra elipse que representa la vista superior del elipsoide.

A continuación se muestra el pseudocódigo *elipsoide*:

```
/* ELIPSOIDE */
/* Algoritmo para visualizar la forma tridimensional
de  un reflector elíptico.
     Por Luis Lagar Ponce */

PRINCIPIO
INICIALIZA modo gráfico
INICIALIZA coordenadas de pantalla
CALCULA ecuaciones de conversión
CALCULA distancia d5
CALCULA diámetro del reflector
SI a es mayor que b ENTONCES
     DIBUJA focos del elipsoide
SI a es mayor que p ENTONCES
     PRINCIPIO

/* Procedimiento para posicionamiento en  pantalla  */
/*  Todos  los cálculos de la elipse se realizan tomando
en cuenta al semieje mayor a en el eje "y" */

     PARA cada x
          PRINCIPIO
          CALCULA discriminante
          SI discriminante >= 0
              PRINCIPIO
              CALCULA punto y de la elipse
              CONVIERTE x,y a pixeles
              SI y en pixeles >= d5
                  PRINCIPIO
                  POSICIONATE en x,y en pixeles
                  ROMPE proceso
                  FIN
              SI NO continúa
              FIN
          SI NO continúa
          FIN
```

```
/* Procedimiento para calcular y dibujar el elipsoide */

     PARA cada x
         PRINCIPIA
         CALCULA discriminante
         SI discriminante >= 0 ENTONCES
             PRINCIPIO
             CALCULA punto y de la elipse
             CONVIERTE x,y
             SI y en pixeles > d5
                 DIBUJA punto x,y
             SI NO continúa
             FIN
         FIN
         SI NO continúa
     FIN


     REALIZA mismo procedimiento para posicionamiento

/* Procedimiento para dibujar la parte negativa del reflector */

     PARA cada x
         PRINCIPIA
         CALCULA discriminante
         SI discriminante >= 0 ENTONCES
             PRINCIPIO
             CALCULA punto y negativo de la elipse
             CONVIERTE x,y a pixeles
             DIBUJA punto x,y
             FIN
         SI NO continúa
         FIN

     REALIZA procedimiento para posicionamiento

/* Procedimiento para dibujar la parte  positiva
de  la  elipse  para complemento del dibujo */

     PARA cada x
         PRINCIPIO
         CALCULA discriminante
         SI discriminante >=0 ENTONCES
             PRINCIPIO
             CALCULA punto y positivo de la elipse
             CONVIERTE x,y a pixeles
             DIBUJA punto x,y
             FIN
         SI NO continúa
         FIN
     FIN
```

```
SI NO
     PRINCIPIO
     REALIZA procedimiento para posicionamiento
     DIBUJA parte negativa del reflector
     REALIZA procedimiento para posicionamiento nuevamente
     DIBUJA complemento del reflector

/* Procedimiento para posicionamiento  para  una
parte  positiva  del reflector */

     PARA cada x
         PRINCIPIO
         CALCULA discriminante
         SI discriminante >= 0 ENTONCES
             PRINCIPIO
             CALCULA punto y negativa de la elipse
             CONVIERTE x,y  a pixeles
             POSICIONATE en punto x,y
             ROMPE proceso
             FIN
         SI NO continúe
         FIN

/* Procedimiento para dibujar parte positiva del reflector */

     PARA cada x de izquierda a derecha
          PRINCIPIO
          CALCULA discriminante
          SI discriminante >= 0 ENTONCES
              PRINCIPIO
              CALCULA y positiva de la elipse
              CONVIERTE punto x,y en pixeles
              SI punto y en pixeles <= d5 pixeles
                  ROMPE proceso
              SI NO continúa
              FIN
          SI NO continúa
          FIN

/* Procedimiento para posicionamiento para dibujar
la otra  parte  del reflector */

     PARA cada x
          PRINCIPIO
          CALCULA discriminante
          SI discriminante >= 0 ENTONCES
              PRINCIPIO
              CALCULA y negativa de la elipse
              CONVIERTE punto x,y
              FIN
          SI NO continúa
          POSICIONA punto x,y

     REALIZA     procedimiento     para     dibujar     parte
positiva pero en orden inverso
     FIN
```

```
SI a >= p ENTONCES
     PRINCIPIO
     REALIZA procedimiento para posicionamiento para parte negativa
     FIN
SI NO
     PRINCIPIO
     REALIZA  procedimiento  para  posicionamiento
empleando los cálculos y positiva de la elipse
     FIN

/* Procedimento para dibujar la vista
tridimensional del reflector por medio de
intersección entre elipses */

     PARA cada x
         PRINCIPIO
         CONVIERTE x a pixeles
         CALCULA discriminante tomando en cuenta  el
                  diámetro  como semieje menor
         SI discriminante >= 0
              PRINCIPIO
              CALCULA y negativa de la elipse
                           tomando   en   cuenta   b
                           en el eje y
              CONVIERTE punto y a pixeles
              SI y en pixeles >= d5 en pixeles ENTONCES
                 DIBUJA punto x,y
              SI NO continúa
              FIN
         SI NO continúa
         FIN

     REALIZAR procedimiento anterior calculando "y" positiva
     FIN
IMPRIME mensajes
TERMINA modo gráfico
FIN
```

En la rutina *ondas-choque*, al igual que en el caso anterior, se inicializa el modo gráfico, se calculan las coordenadas (x,y), convirtiéndose a pixeles, y en el borde del reflector se dibuja el limite superior del mismo. Posteriormente se dibujan los rayos perpendiculares al frente de onda, que comprende cada una del foco F_1 a la pared del reflector y de la pared del reflector al foco F_2.

A continuación se mostrará el algoritmo *ondas-choque*.

```
/*  ONDAS-CHOQUE  */

/* Algoritmo para simulación de rayos
perpendiculares al  frente  de onda
tomando en cuenta procedimientos del
algoritmo ELIPSOIDE */


/* Por Luis Legar Ponce */


PRINCIPIO
INICIALIZA modo gráfico
INICIALIZA coordenadas de pantalla
INICIALIZA coordenadas reales
CALCULA d5
CALCULA c
CONVIERTE c a pixeles
SI a > p ENTONCES
     REALIZA procedimiento de posicionamiento
     DIBUJA parte negativa del reflector
     POSICIONATE otra vez al mismo punto
     REALIZA los mismos cálculos para parte negativa
               guardando los valores  de  las  coordenadas
               de  los extremos para dibujar  el
               borde del mismo.
     DIBUJA   la   parte   negativa   de   la   elipse   para
              complemento del dibujo
     POSICIONA para complemento de la elipse
     DIBUJA parte positiva de la elipse para complemento del dibujo
     FIN
SI NO
     POSICIONATE para dibujar parte negativa del reflector
     DIBUJA la parte negativa del reflector
     POSICIONATE para la parte positiva
     DIBUJA una parte positiva del reflector
     POSICIONA para la otra parte positiva
     DIBUJA la otra parte positiva del reflector
     DIBUJAR borde del reflector
     POSICIONA de nuevo para parte positiva de la elipse
     DIBUJA parte positiva de la elipse para complemento
     FIN

/* Procedimiento para dibujar los rayos perpendiculares
al  frente  de onda */

PARA cada valor del ángulo teta de 0 a 2*PI
     donde PI = 3.14159
     PRINCIPIO
     CALCULA valor y paramétrico
     DIBUJA linea del primer foco al punto x,y vigente
     DIBUJA linea del punto x,y vigente al segundo foco
     FIN
```

```
IMPRIME mensajes
TERMINA modo gráfico
FIN
```

El bloque *información* calcula (como ya se mencionó) los parámetros necesarios para su construcción.

A continuación se mostrará el algoritmo *información*.

```
/* INFORMACION */
/*  Algoritmo  para  calcular  parámetros  necesarios  para  la  construcción
de un reflector elíptico */

 /*  Por  Luis  Legar Ponce */
PRINCIPIO
CALCULA c
CALCULA e IMPRIME excentricidad e
IMPRIME coordenadas focales
CALCULA e IMPRIME lactus rectum lr
CALCULA e IMPRIME distancia d1
CALCULA e IMPRIME distancia d2
CALCULA e IMPRIME distancia d3
CALCULA e IMPRIME distancia d4
CALCULA e IMPRIME distancia d6
CALCULA e IMPRIME diámetro DM
CALCULA e IMPRIME radiovector r
CALCULA e IMPRIME angulo fi
CALCULA e IMPRIME angulo psi
CALCULA e IMPRIME volumen del reflector V en centímetros cúbicos
CALCULA e IMPRIME factor de aprovechamiento Fa
FIN
```

El bloque *lectura-datos* valida o invalida los valores de entrada por el usuario. Cabe mencionar que el intervalo de valores establecido por el sistema, está basado en que los reflectores son por lo regular construidos dentro de dicho intervalo.

A continuación se moetrará el algoritmo *lectura-datos*.

```
/* LECTURA-DATOS */
/* Algoritmo para lectura de datos */
/* Por Luis Legar Ponce */

PRINCIPIO
LEE semieje mayor a
LEE semieje menor b
```

```
LEE profundidad p
SI p=0 O a=0 O b=0 ENTONCES
     PRINCIPIO
     IMPRIME No existe el elipsoide
     SOLICITAR ayuda si se desea
     FIN
SI a<b ENTONCES
     PRINCIPIO
     IMPRIME Los valores no satisfacen la ecuación
     SOLICITAR ayuda si se desea
     FIN
SI a<0 O b<0 O p<0 ENTONCES
     PRINCIPIO
     IMPRIME ¡¡Cuidado!! Estás introduciendo distancias negativas
     SOLICITAR ayuda si se desea
     FIN

/* Se selecciona un intervalo de 31 a 500 mm., en base a que
   por lo regular se construyen los reflectores
   dentro del intervalo antes indicado. */

SI a>500 O a<31 O es dos veces mayor que la misma a ENTONCES
     PRINCIPIO
     IMPRIME Valores fuera del intervalo
     SOLICITAR ayuda si se desea
     FIN
SI NO
     ESPECIFICAR valores de las coordenadas reales para
diferentes intervalos de valores de a y b.
FIN
```

IV.2 Descripción de los datos.

Ante la necesidad de crear rutinas que se ejecuten con los mismos datos de entrada y el manejar diferentes escalas, se recurrió al uso de las variables globales.

Las variables globales son (ver figura II.1):

a Semieje mayor del elipsoide.

b Semieje menor del elipsoide.

p Profundidad del elipsoide del vértice o fondo a un punto ubicado en el eje de simetría de dicho elipsoide a la altura del borde.

DM Diámetro máximo interior del reflector.

d5 Distancia del borde del reflector al origen sobre el eje de simetría.

c,c0 y cnot:

Valores de las coordenadas focales
del elipsoide $F_1(c0,c)$ y $F_2(c0,cnot)$.

Donde c0 = 0,
cnot = -c.

escala, bc1, bc2:

Valores para la línea de escala.

driver:

Tarjeta de gráficos a utilizarse.

mode:

Modo de la tarjeta de gráficos.

Estos dos últimos son variables correspondientes al manejo de gráficos del lenguaje C.

Las variables locales empleadas en la rutina *elipsoide* son:

d5c: Distancia 5 en pixeles.

disc1, disc2, disc3, disc4 y disc5:

Discriminantes.

xpc: Valor de "x" en pixeles.

ypc1, ypc2, ypc3 y ypc4:

valores de "y" en pixeles.

A,B,C y D: Valores de la conversión en pixeles.

y1, y2, y3, y4:

Valores reales de la función.

b2: Semieje menor de la elipse que intersecta a la primera que se calcula para formar la vista tridimensional.

x: Valor real de "x" en la función.

xv1, xvr,yvb,yvt: Coordenadas en pixeles.

xw1, xwr, ywb y ywt: Coordenadas reales.

cf, cf0, cfn: Coordenadas de los focos en pixeles.

discf, f1, f0, fn: Valores para dibujar el punto focal F_2.

Las variables locales empleadas dentro de la rutina *ondas-choque* son semejantes en su mayoría a las de la rutina *elipsoide*:

xvl1, xvr1, yvb1, yvt1: Coordenadas en pantalla.

xwl1, xwr1, ywb1, ywt1: Coordenadas reales.

AA, BB, CC y DD: Valores de conversión en pixeles.

disc6 y disc7: Discriminantes para calcular "y".

xpc1, ypc5, ypc6: Valores de "x" y "y" en pixeles.

xini, yini, xfin, yfin: Valores para dibujar el borde del reflector.

cnpc, cpc, cp0: Coordenadas de los focos en pixeles.

x1 Valor de "x" real

yy1, yy2: Valores de "y" reales.

px1, px2, px3, px4, py1, py2, py3, py4:

Coordenadas de simulación de rayos en los extremos del reflector.

teta: ángulo para dibujar los rayos rependiculares al frente de onda.

xpar, ypar: Coordenadas paramétricas para dibujar los rayos.

xray, yray: Coordenadas para dibujar los rayos mediante rectas dirigidas.

salto: Variable auxiliar para dibujar un rayo cada "n" iteraciones.

Las variables locales dentro de la rutina *información* son (ver figura II.1):

lr: Diámetro del elipsoide a la altura del foco F_1.

d1: Distancia del fondo del reflector al primer foco (F_1).

d2: Distancia del fondo del reflector al segundo foco (F_2).

d3: Distancia entre la distancia y = - d5.

d4: Distancia del primer foco al punto (0,-d5).

d6: Distancia entre los focos del elipsoide.

disc8, disc9 y disc10: Discriminantes.

den: Denominador para calcular el ángulo "fi".

r: Magnitud del radiovector.

fi: Angulo que forman el radiovector y el eje "y".

arg: Argumento para calcular el ángulo "psi".

id: Identidad para calcular el factor de aprovechamiento.

v1, v2, v3, v4, v5, v6, v7, v8 y v9:

Variables auxiliares para calcular el volumen del reflector.

V: Volumen máximo del interior del reflector cuando a este se le llena con agua.

Fa: Factor de aprovechamiento.

e: Excentricidad de elipsoide.

Las variables locales para la rutina *lectura-datos* son:

car: Decisión para solicitar ayuda.

doble_a: Dos veces el valor del semieje mayor.

Las funciones predefinidas para menús y ayudas son:

MENU_PRINCIPAL.

MENU_ACCIONES.

Ambos residen en un archivo de textos llamado *menús.c*.

ayuda_datos1: muestra un texto de ayuda cuando el elipsoide no existe.

ayuda_datos2: muestra un texto cuando los valores de entrada no satisfacen la ecuación.

ayuda_datos3: muestra un texto cuando alguno (s) de los datos de entrada es (son) negativo (s).

ayuda_datos4: muestra un texto cuando los valores están fuera del rango o son inválidos.

Todas estas rutinas residen dentro de un programa llamado *ayuda.c*. Todas las rutinas, tanto los menús como las ayudas son declaradas en *declafun.c*

IV.3 Descripción de entradas y salidas.

Las entradas principales del sistema son: el semieje mayor "a", el semieje mayor "b" y la profundidad "p".

Como ya se mencionó en el diseño procedural, el usuario puede verificar los datos dentro de *lectura-datos*. Estos tres datos de entrada son válidos para *elipsoide*, *ondas-choque*, e *información*.

Elipsoide tiene como salida la gráfica de la vista tridimensional del elipsoide.

Ondas-choque tiene como salida la gráfica de la simulación de los rayos perpendiculares al frente de onda.

Información tiene como salida las coordenadas de los focos F_1 y F_2, los valores de las distancias d_1, d_2, d_3, d_4, d_5 y d_6, el diámetro del reflector, la magnitud del radiovector, todos en milímetros, los ángulos "psi" y "fi" en radianes, el volumen del reflector en centímetros cúbicos y el factor de aprovechamiento.

Al finalizar cada una de las tres rutinas, se retorna al menú de acciones; de igual modo al validar los datos de entrada.

Las cinco opciones del menú de acciones son:

```
VISTA (T)RIDIMENSIONAL DEL REFLECTOR.

(S)IMULACION DE LOS RAYOS PERPENDICULARES AL FRENTE
DE ONDA.

(I)NFORMACION   TECNICA   EN   BASE   A
LOS DATOS DE ENTRADA.

PROBAR PARA (O)TROS VALORES.

VOLVER AL (M)ENU PRINCIPAL.
```

Hay que seleccionar alguna opción dándole la letra que se indica con los paréntesis.

Seleccionando la opción *N* tenemos que aparece el menú principal, donde se muestran las siguientes opciones:

(F)ORMA TRIDIMENSIONAL Y SIMULACION DE RAYOS PERPENDICULARES AL FRENTE DE ONDA.

(A)YUDAS.

(S)ALIR DEL SISTEMA.

La primera opción llama a *lectura-datos* para la ejecución de cualquiera de las primeras tres opciones del menú de acciones.

La segunda opción proporciona ayuda general al usuario.

La tercera opción finaliza la sesión dentro del sistema, saliendo de nuevo a MS-DOS.

A su vez, la segunda opción del menú principal muestra otro menú:

ELIPSOIDE (T)RUNCADO Y SUS PARAMETROS.

OPCIONES DEL (M)ENU PRINCIPAL.

(E)RRORES.

La primera opción de dicho menú muestra una figura típica de un elipsoide truncado con cada uno de sus parámetros y la descripción de cada uno de ellos.

La segunda opción muestra un texto con la descripción de cada una de las opciones del menú principal.

La tercera opción muestra un texto sobre los tipos de errores que pueden ocurrir al introducir los valores de los datos de entrada en forma incorrecta; también se proporciona una serie de pasos a seguir para salir de un bloqueo dentro del sistema.

Al finalizar la ejecución de cada una de las opciones se muestra otro menú:

MAS A(Y)UDA.

VOLVER AL (M)ENU PRINCIPAL.

La primera opción muestra de nuevo el menú de ayudas.

La segunda opción retorna el sistema al menú principal.

Dentro de la comunicación usuario-sistema no se deben excluir los tipos de mensajes que se muestran al introducir incorrectamente los datos de entrada:

Error 1: NO EXISTE EL ELIPSOIDE.

Esto sucede cuando alguno (s) de los datos de entrada es (son) cero.

Error 2: LOS VALORES NO SATISFACEN LA ECUACION.

Esto sucede cuando el semieje menor es más grande que el semieje mayor.

Error 3: !! CUIDADO !! ESTAS INTRODUCIENDO
DISTANCIAS NEGATIVAS.

Esto sucede cuando alguno(s) de los datos de entrada es (son) negativo(s).

Error 4: VALORES FUERA DEL INTERVALO O DATOS INVALIDOS.

Esto sucede cuando alguno(s) de los datos de entrada está(n) fuera del intervalo establecido por el diseñador del sistema o cuando los datos son inválidos (caracteres no numéricos).

Junto a cada uno de estos mensajes de error emitidos, se muestra otro mensaje:

NECESITAS AYUDA ? (S/N)

Si el usuario le da la opción *S* el sistema muestra el texto de ayuda correspondiente; en caso contrario, envía al usuario a leer datos nuevamente.

Al finalizar el texto de ayuda se muestra el siguiente menú:

(I)NTENTARLO DE NUEVO.

(S)ALIR DEL SISTEMA.

La primera opción envía al usuario a introducir datos nuevamente.

La segunda opción termina la sesión dentro del programa enviando al usuario a MS-DOS.

El programa *mexilit* por tratarse de un programa meramente de cálculos y graficación para fines científicos, no emplea bases de datos pues no lo amerita.

Los archivos descritos en el punto IV.2 son incluídos dentro del archivo fuente principal *mexilit.c* y son llamadas por medio de funciones definidas dentro de cada uno.

Existen también otros archivos predefinidos dentro de la biblioteca de *Turbo C*:

<stdio.h> Entrada y salida o *archivo cabecera*.

<conio.h> Control de pantalla en modo texto.

<graphics.h> Control de pantalla en modo gráfico.

<math.h> Archivo de funciones matemáticas tracedentales.[52]

Estos archivos forman parte del archivo fuente.

CAPITULO V

CONSTRUCCION DEL PROGRAMA.

V.1 Criterios de selección del lenguaje de programación.

Las ventajas que tiene el diseño descrito en el capítulo anterior son: la modularidad, es decir, que el programa está dividido por procesos, los cuales se intercomunican y son administrados por un menú principal. En el diseño procedural, cada uno de los pseudocódigos están estructurados de tal modo que las estructuras lógicas de control son modulares, lo que significa, que para cada estructura se realiza un pequeño programa (cuando se trata de un conjunto de acciones). Además, existen algunas condiciones en las cuales es necesaria la recursividad y las llamadas a las funciones se encuentran anidadas.[53]

Con todo ello, es necesario el empleo de un lenguaje estructurado para la construcción del programa. Para esto el lenguaje debe tener las siguientes características:

* Modular.
* De fácil programación.
* De fácil mantenimiento.
* Portable.

Un programa de fácil mantenimiento es aquel que puede actualizarse, modificarse o mejorarse sin problema alguno.

Dentro de los lenguajes estructurados a nivel comercial se encuentran contemplados los de alto nivel como ADA, MODULA_2 o Pascal; y los de nivel medio como C.

En el caso de los lenguajes estructurados de alto nivel, se tienen contempladas las características antes mencionadas, sin embargo, no cuentan con la flexibilidad en cuanto a las operaciones con datos de diferente tipo, como sería en este caso la conversión de datos reales a datos en unidades de pantalla para graficar.

C es un lenguaje medio, estructurado que permite combinaciones de diferentes tipos de datos en operaciones matemáticas y también permite la fácil conversión de datos de un tipo a otro.

Por todo lo anterior, se escogió C para implementar el programa.

V.2 Características del lenguaje de programación seleccionado.

El lenguaje C es considerado como un lenguaje de empleo general.

Fue desarrollado a principios de los años 70's como una mejora al lenguaje BCPL por Ken Thompson, Brian Kernigan y Dennis Ritchie.[54]

C (como ya se ha mencionado) es un lenguaje que trabaja con la misma clase de datos que la mayoría de las computadoras: caracteres, números y direcciones, que pueden ser combinados con operadores aritméticos y lógicos, utilizados normalmente en las máquinas.

El lenguaje C no contiene operaciones para trabajar directamente con objetos compuestos, como cadenas de caracteres, conjuntos, listas, arreglos o vectores considerados como un todo.

No tiene definido otro tipo de asignaciones de memoria aparte de las definiciones estáticas y el manejo de pilas para las variables locales de las funciones.

El lenguaje no cuenta con operadores directos de entrada y salida como lo son de lectura y escritura, sino que deben ser definidos aparte por funciones contenidas en archivos de librería.

Posee estructuras lógicas de control, al igual que cualquier lenguaje de alto nivel, como lo son if (decisiones), ciclos iterativos ("while", "for"), realizar tareas condicionadas ("do") y menús ("switch").

Realiza funciones de un lenguaje de máquina (bajo nivel) y es capaz de manejar direcciones, bits y bytes asistido por funciones de alto nivel.

Tiene también la cualidad de que sus funciones pueden ser llamadas en forma recursiva y de crear las variables locales cada vez que se llama. Las definiciones de función no pueden anidarse pero las variables pueden ser declaradas con una estructura de bloques.

Las funciones de C pueden compilarse por separado.

Las variables pueden ser internas o externas (globales) siempre y cuando pertenezcan al código fuente presente.

C no es un lenguaje "fuertemente estructurado" como lo son Pascal, ADA o MODULA_2. No tiene restricciones para el manejo de diferentes tipos de datos, sino que es muy flexible para la conversión de datos, combinaciones u operaciones con tipos diferentes de datos.[55]

Al igual que cualquier lenguaje, se debe reconocer que tiene sus desventajas. Una de estas desventajas es que la nomenclatura manejada en la sintaxis en algunas estructuras lógicas no es del todo legible para el programador.

Para este caso, el lenguaje C será muy viable para la creación del código fuente, así como para el manejo de interfases gráficas y elementos externos como los adaptadores gráficos.

V.3 Listado y documentación del programa.

En el disco anexo a este trabajo se encuentra el código fuente del programa construído en lenguaje C.

Este código fuente corresponde solamente al programa *mexilit.c*

Dentro del código fuente son incluídos los programas *menús.c*, *declafun.c* y *ayuda.c*.

En *declafun.c* están contenidas (como ya se mencionó) las declaraciones de la funciones, en *menús.c* los menús del sistema y en *ayudas.c* los textos de ayuda los cuales ninguno de ellos será incluído dentro de este trabajo escrito dado que solo son archivos de texto.

Cabe mencionar que el archivo cabecera de entrada y salida <*stdio.h*> y el archivo cabecera de control de pantalla en modo texto <*conio.h*> son incluídos dentro del archivo *menús.c*, en tanto que el archivo <*graphics.h*> es incluído dentro del programa *ayudas.c*. Estos archivos de librería también serán válidos cuando se realice la función main() dentro del archivo principal.

En el capítulo III, se mencionó la descripción del diseño del programa, así como la de sus datos, sus entradas y sus salidas, los puntos mencionados forman la mayor parte de la documentación del programa. Lo anterior ayuda a conocer con mayor detalle el funcionamiento del sistema.

V.4 Tiempo de ejecución.

Otra de las limitantes del programa en lo que se refiere a su ejecución es la velocidad con la que cuenta la máquina. Se requieren por lo menos 10 MHz para su ejecución satisfactoria , sobre todo, si el monitor maneja un sistema de rastreo al azar, es decir, que dibuja punto por punto en un orden específico.[56]

Para este primer caso tomando en cuenta que la tarjeta HERCULES maneja un sistema de rastreo al azar y la velocidad de la computadora es de 10 MHz:

El tiempo mínimo es de 1 seg.

El tiempo máximo es de 18 seg.

Estos tiempos varían según los valores del semieje mayor "a", el semieje menor "b" y la profundidad "p" que se introduzcan; entre más pequeños sean, su tiempo será más pequeño.

Si se maneja la misma velocidad pero ahora con sistema de rastreo con rastreador, es decir, que muestre el objeto como un conjunto de puntos a través de cada línea de rastreo en la pantalla, como en los adaptadores CGA y VGA , el tiempo puede variar del orden de los milisegundos a los 5 segundos (aproximadamente).

Por lo tanto, para una misma velocidad, el sistema de rastreo con rastreador es más rápido que el sistema de rastreo al azar.

En lo que se refiere a procesos numéricos como en el bloque *información*, su tiempo de ejecución varía del orden de los milisegundos a 1 seg.

CAPITULO VI

PRUEBAS DEL PROGRAMA.

VI.1 Ejemplo con diferentes parámetros de entrada.

En el capítulo I se mencionó una estrategia de solución con la cual principiaría el proceso de construcción y pruebas del programa.

En primer lugar, se realizó el diseño estructurado del sistema. Una vez terminado dicho paso se procedió al diseño procedural tomando en cuenta solo cálculos numéricos; a partir de ese punto se procedió a construir el programa y de ahí que se realizaron las pruebas numéricas para checar el correcto funcionamiento del sistema con resultados reales. Acto seguido, se realizó la conversión de datos reales a unidades de pantalla, para ello se partió del diseño procedural nuevamente y, posteriormente se construyó esta nueva modificación dentro del programa; de ahí qua se realizaron las pruebas correspondientes para checar su correcto funcionamiento.

La misma serie de pasos se siguió para construir y probar la representación gráfica de los cálculos realizados, con sus respectivas modificaciones cuando era necesario.

La etapa siguiente de pruebas consistió en corregir pequeños detalles para hacer el sistema lo más confiable posible.

Cabe hacer algunas observaciones en lo que se refiere a las pruebas. En ellas se realizó una evaluación de su funcionamiento. Dentro de esta evaluación, se contempla que el usuario no tenga que enfrentarse a situaciones difíciles como el de bloquear accidentalmente el sistema y no saber que hacer en ese momento.

Para ello se realizaron pruebas para bloquear el sistema; un caso particular, cuando se introducen datos no válidos como caracteres. El introducir datos no válidos provoca que el programa no pueda leer datos de nuevo después de solicitar ayuda. Para ésto se proporcionan dos tipos de ayudas: general y particular; en ellas se dan al usuario una serie de pasos para salir del bloqueo sin tener que apagar la máquina o reajustarla ("reset").

La primera serie de pasos consiste en solicitar ayuda (si el programa queda bloqueado dentro del mensaje de error correspondiente), al terminar la ayuda elegir la opción de salir del programa para salir al sistema MS-DOS.

Otro caso se da cuando no es la primera vez que se introducen datos dentro de la sesión del programa; se despliegan en pantalla los datos anteriores. Para este caso, se sigue la segunda serie de pasos:

- Dar la opción *S* cuando pregunte si los datos son correctos.
- Dar la opción *N* dentro del menú de acciones.
- Elegir la opción *S* para salir de la sesión.

Cabe señalar que si se vuelve a dar la opción *T* dentro del menú principal, no podrá leer ya que el proceso se saltará y se desplegarán los datos anteriores.

Para ambos casos, después de salir de la sesión, volver a llamar al programa.

Estando dentro del ambiente operativo MS-DOS, se invocará al sistema escribiendo *mexilit*:

```
A>MEXILIT
```

En el caso del disco duro se invocará de la siguiente forma:

```
C:\MLS>MEXILIT
```

donde MLS es el directorio creado durante la instalación del sistema.

Haciendo referencia al capítulo IV, el programa realiza los cálculos y las graficaciones únicamente cuando los datos de entrada (semieje mayor, semieje menor y profundidad total) son válidos.

Dentro de la sesión en *mexilit* se realizarán tres pruebas con diferentes valores.

En la figura VI.1 encontramos la forma tridimensional del reflector elíptico cuando el semieje mayor "a" es de 130.0 mm, el semieje menor "b" es de 90.0 mm, y la profundidad máxima "p" es 58.0 mm.

La figura VI.2 muestra la simulación de los rayos perpendiculares al frente de onda para estos mismos valores de entrada.

La segunda prueba se realiza con el semieje mayor "a" igual a 120.0 mm, el semieje menor "b" es de 86.0 mm, y la profundidad máxima "p" es de 120.0 mm, tanto para la forma tridmensional del reflector elíptico (figura VI.3), como para la simulación de los rayos perpendiculares al frente de onda (figura VI.4).

La tercera prueba se realiza con un semieje mayor "a" igual a 120.0 mm, un semieje menor "b" igual a 86.0 mm, y una profundidad máxima "p" igual a 228.7 mm. Sus respectivas gráficas se presentan en las figuras VI.5 y VI.6.

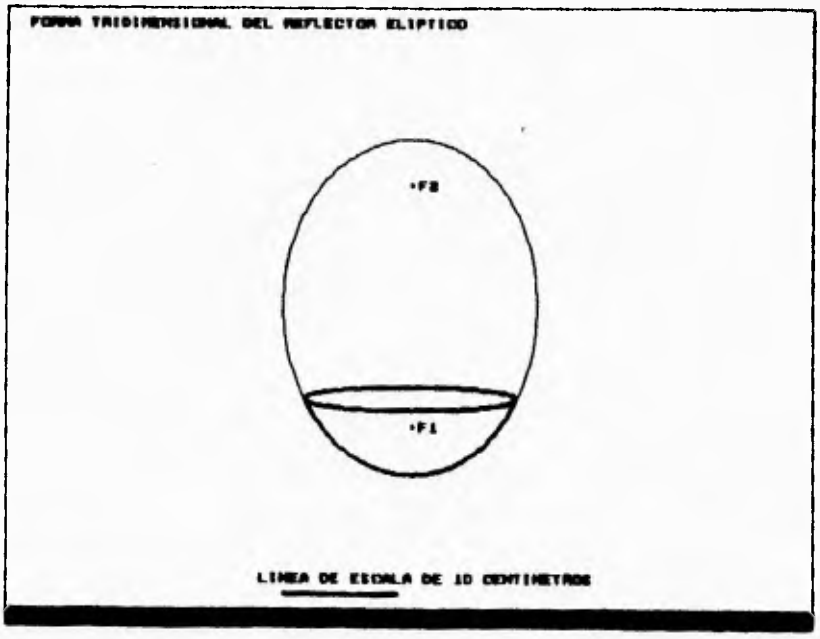

Figura VI.1 Forma tridimensional del reflector para semieje mayor "a" = 130.0 mm, semieje menor "b" = 90.0 mm y profundidad máxima "p" = 58.0 mm.

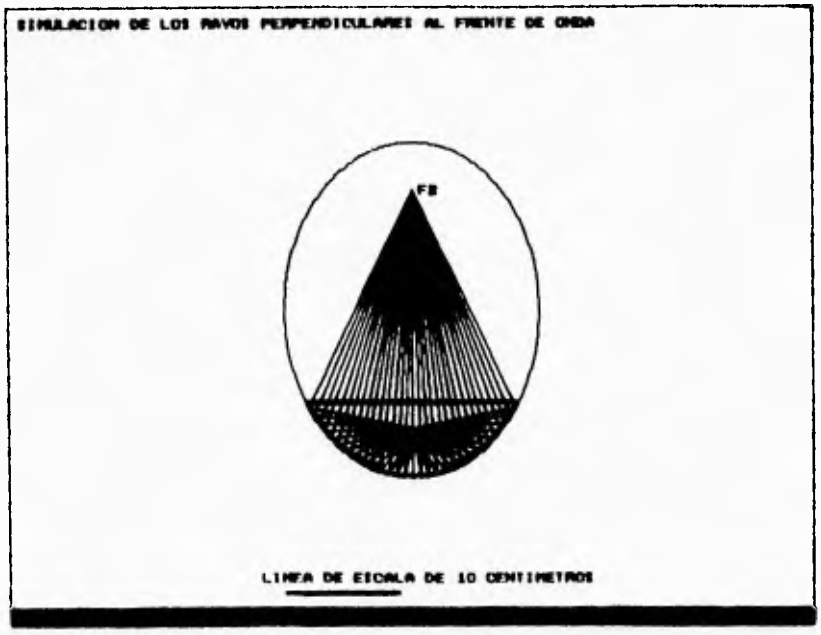

Figura VI.2 Simulación de rayos perpendiculares al frente de onda para semieje mayor "a" = 130.0 mm, semieje menor "b" = 90.0 mm y profundidad máxima "p" = 58.0 mm.

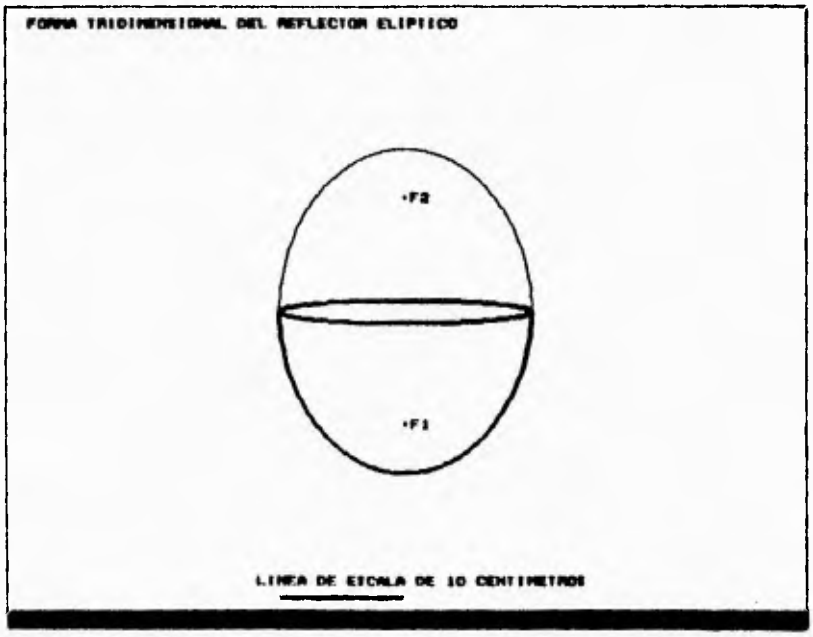

Figura VI.3 Forma tridimensional del reflector para semieje mayor "a" = 120.0 mm, semieje menor "b" = 86.0 mm y profundidad máxima "p" = 120.0 mm.

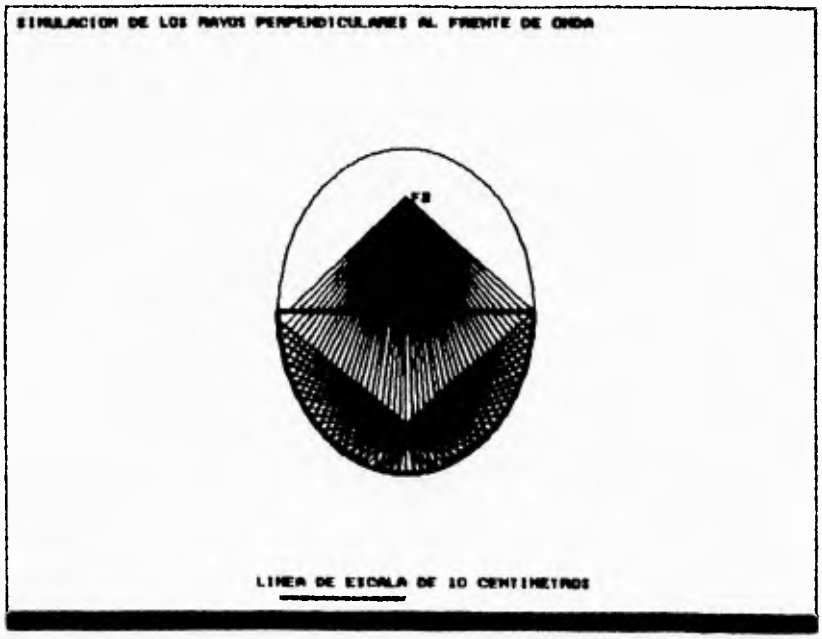

Figura VI.4 Simulación de rayos perpendiculares al frente de onda para semieje mayor "a" = 120.0 mm, semieje menor "b" = 86.0 mm y profundidad máxima "p" = 120.0 mm.

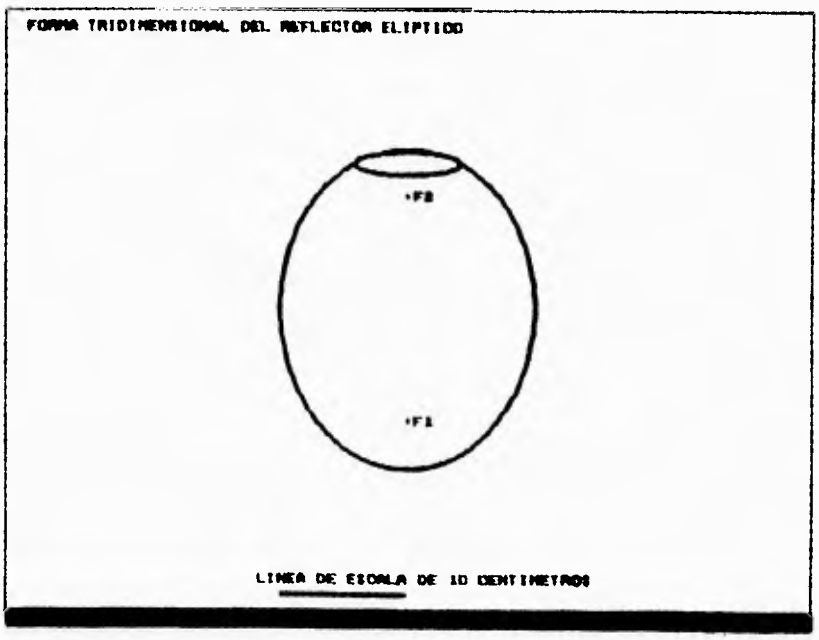

Figura VI.5 Forma tridimensional del reflector para semieje mayor "a" = 120.0 mm, semieje menor "b" = 86.0 mm y profundidad máxima "p" = 228.7 mm.

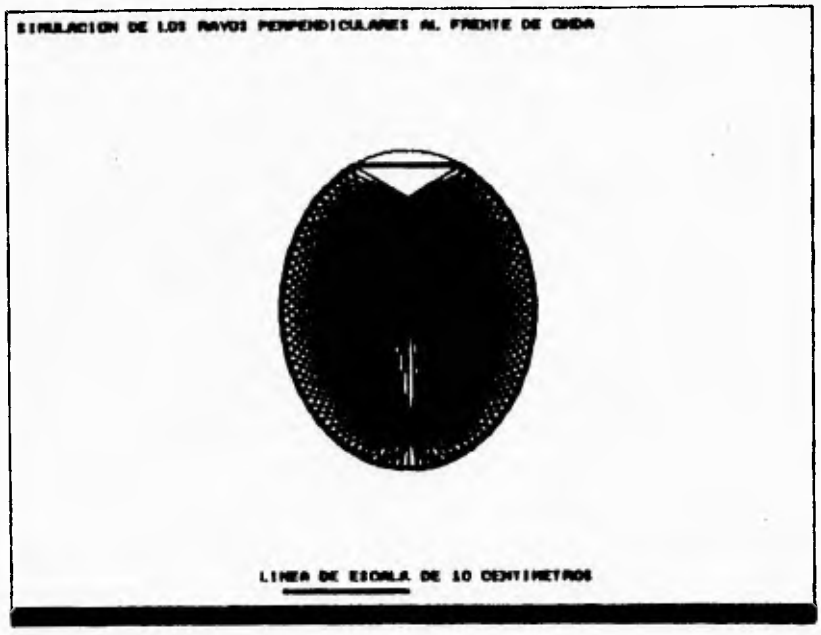

Figura VI.6 Simulación de rayos perpendiculares al frente de onda para semieje mayor "a" = 120.0 mm, semieje menor "b" = 86.0 mm y profundidad máxima "p" = 228.7 mm.

Los datos correspondientes a los datos de salida numéricos, se encuentran en la tabla VI.1. En ella aparecen las coordenadas de los focos $F_1(0,c)$ y $F_2(0,c)$, el diámetro máximo "D", la excentricidad "e", el *latus rectum* "lr", las distancias d_1, d_2, d_3, d_4, d_5, y d_6, la magnitud del radiovector "r", los ángulos "fi" y "psi", el volumen máximo "V", y el factor de aprovechamiento "Fa". Todos los valores de salida anteriormente mencionados son obtenidos de los datos de entrada con que se probaron las rutinas para la forma tridimensional del reflector elíptico y la simulación de los rayos perpendiculares al frente de onda.

DATOS DE ENTRADA	PRUEBA 1	PRUEBA 2	PRUEBA 3
SEMIEJE MAYOR a	130.0	120.0	120.0
SEMIEJE MENOR b	90.0	86.0	86.0
PROFUNDIDAD p	58.0	120.0	228.7
DATOS DE SALIDA			
EXCENTRICIDAD e	0.722	0.697	0.697
FOCO 1 F1 (0,-c)	F1 (0,-93.8)	F1 (0,-83.7)	F1 (0,-83.7)
FOCO 2 F2 (0,c)	F2 (0,93.8)	F2 (0,83.7)	F2 (0,83.7)
DISTANCIA d1	36.2	36.3	36.3
DISTANCIA d2	223.8	203.7	203.7
DISTANCIA d3	165.8	83.7	-25.0
DISTANCIA d4	21.8	83.7	192.4
DISTANCIA d5	72.0	0.0	-108.7
DISTANCIA d6	187.6	167.4	167.4
DIAMETRO MAXIMO D	149.9	172.0	72.9
LATUS RECTUM lr	124.6	123.3	123.3
RADIOVECTOR r	78.0	120.0	195.8
ANGULO fi	1.288	0.799	0.187
ANGULO psi	1.854	2.43	2.954
VOLUMEN MAXIMO V	560.5	1858.8	3693.6
F.DE APROV. fa	0.640	0.849	0.991

NOTA: Todas las distancias están en milímetros, el volumen en centímetros cúbicos y los ángulos en radianes.

TABLA VI.1 Datos numéricos de salida en la rutina INFORMACION.

VI.2 Interpretación de los resultados.

Los dibujos anteriormente mostrados, tanto de la forma tridimensional de los reflectores como de la visualización de las ondas de choque en su interior, se realizan a diferentes escalas según los valores de los parámetros de entrada. El intervalo de valores para introducirse al sistema varía de 30 a 500 mm.

De igual modo para diferentes datos, el elipsoide truncado toma obviamente diferentes formas: muy abiertas (en forma de plato), muy estrechas (en forma de tubo de ensayo), semiredondas, o semicerradas. Por consiguiente, sus respectivos focos pueden resultar muy cercanos o muy distantes.

Por otra parte, la visualización de los rayos imaginarios perpendiculares a los frentes de las ondas de choque débiles proporciona al usuario un panorama del comportamiento del dispositivo a construir. Cabe mencionar que los dibujos de los rayos perpendiculares al frente de onda se realizan despreciando efectos no lineales. En realidad estos efectos pueden originar una región de máxima presión que no coincide exactamente con F_2.[57] Por lo tanto esta primera versión del sistema sólo es de índole demostrativa para que el interesado tenga una visión de la forma que puede adquirir el reflector y, en base a los resultados obtenidos podrá tomar la decisión de construirlo o no.

Los resultados numéricos obtenidos son útiles igualmente para la construcción del dispositivo y para pruebas posteriores del mismo.

No obstante las características anteriores el programa *mexilit-soft* es considerado como un sistema de diseño preliminar.

CAPITULO VII.

I N S T A L A C I O N .

El programa ejecutable *mexilit-soft* se compone de 4 archivos:

MEXILIT.EXE

CGA.BGI

VGA.BGI

EGA.BGI

En *mexilit.exe*, como ya se mencionó, se encuentra contenido el programa gráfico para diseño de reflectores elípticos y visualización de ondas de choque débiles en sus interiores. En el capítulo anterior se abarcó todo lo referente a las pruebas del programa y en el capítulo IV se abarcó todo lo referente a la comunicación con los menús dentro de la sesión.

Ambos aspectos se encuentran contemplados dentro de *mexilit.exe*. Este programa ejecutable contiene el archivo fuente *mexilit.c*; junto con él están ligados los programas fuentes *declafun.c*, *ayuda.c* y *menús.c*, así como los programas *stdio.h*, *math.h*, *graphics.h*, *conio.h*, que forman parte de la bibioteca de Turbo C.[58]

Todos estos archivos se encuentran compilados y ligados debidamente.

Como se mencionó en el capítulo IV, los archivos de extensión BGI son programas de interfaz entre el software que es *mexilit.exe* en este caso y los adaptadores gráficos de las computadoras, es decir, el elemento de "hardware". Lo anterior permite que el sistema trabaje en modo gráfico. Sin los archivos BGI no se puede realizar ninguna graficación. En este caso el programa únicamente realizaría los procesos correspondientes al modo texto, como lo son los cálculos

numéricos y la impresión de mensajes de error, ayudas, menús y demás textos. Es por eso que los archivos BGI se incluyen en la instalación del programa ejecutable *mexilit* en el disco duro de la computadora.

Son dos las formas de instalar el programa *mexilit* en el disco duro ("hard disk").

La primera forma se realiza a través del manual de los comandos del sistema MS-DOS.

El primer paso es crear un directorio dentro del directorio raíz del disco duro:

```
Unidad de disco>MKDIR MEXILIT
```

Ejemplo:

```
C>MKDIR MEXILIT
```

Invocar el comando `CD MEXILIT` para comprobar que fué creado.

Sin cambiar de directorio cambiar el "drive" A con `A:`, insertar el disco del sistema en dicha unidad y dar la orden

```
COPY *.* C:
```

Todos los archivos, tanto el ejecutable como los de interfaz se copiarán en la unidad especificada.

Accesar a la unidad especificada para probar que los archivos han sido correctamente copiados; dar la orden `DIR`.

La segunda forma de instalar el sistema y, digamos la más elegante es a través del disco de instalación que viene incluído junto con el disco del sistema. El disco de instalación contiene un programa ejecutable que realiza en forma automática los mismos comandos de MS-DOS de la serie de pasos anterior.

Es deseable que se tenga un sistema MS-DOS en español debido a que dentro de la ejecución hay diálogos con el ambiente operativo, y se dice que es deseable no solo para darle uniformidad a la comunicación dentro de la ejecución, sino para que resulte más explícito, sobre todo si el usuario no sabe inglés.

Para instalar el sistema *mexilit*, se deberá insertar el disco de instalación en la unidad A y desde ahí invocar:

```
A>INSTALA
```

Se mostrará una carátula similar a la del programa ejecutable *mexilit*.

Al pulsar cualquier tecla se muestra otro texto indicando en qué consiste el funcionamiento del sistema de instalación.

Se pide al usuario retirar el disco de instalación, insertar el disco del sistema en la unidad A y pulsar <ENTER> cuando esté listo.

Se muestra un menú triopcional en el que se puede elegir la unidad C, D o E.

Al elegir alguna de las tres opciones se accesa a la unidad, se crea en forma automática el directorio, se accesa a él, se copia el contenido del disco del sistema de A a la unidad especificada, también en forma automática. Cuando se realiza la instalación, se van imprimiendo en la pantalla los archivos que se van copiando al disco duro.

Una vez realizada la instalación, se imprime en la pantalla el número de archivos copiados y un mensaje proveniente del sistema de instalación que dice:

```
|| SISTEMA MEXILIT_SOFT INSTALADO ||

   Accesa al drive <unidad/path>

      con <unidad>>CD MEXILIT.

      Teclea MEXILIT y pulsa <ENTER>.
```

En el caso de pulsar <ENTER> sin haber insertado el disco del sistema, MS-DOS enviará un mensaje de error de falta de disco, ofreciendo al usuario tres opciones: reintentar (R), abandonar (A) o ignorar (I). Si ésto se da, insertar el disco del sistema y dar la opción (R).

Igualmente se puede accesar a *mexilit* desde el mismo disco del sistema.

CONCLUSIONES

Cuando se inició con esta tesis, se pensó sólo en un visualizador de reflexión de ondas de choque débiles, que fuese una herramienta que permitiese al usuario realizar cálculos y graficar automáticamente, con el fin de encontrar un reflector elíptico adecuado para su aplicación posterior. Durante la realización del proyecto se descubrió que el sistema era algo más: un sistema para diseño preliminar de reflectores elípticos, además de ser un simulador muy modesto. Lo anterior no significa que con el programa se pueda realizar un diseño de un reflector elíptico, sino que el programa servirá de auxiliar al diseñador para conocer en primer lugar qué forma podría tener el reflector.

Las ventajas de *mexilit-soft* son:

Que es una herramienta auxiliar para obtener información acerca de la forma de un reflector elíptico para semiejes y profundidades dadas.

Que los cálculos y la graficación automática reducen tiempo y esfuerzo al usar el programa para un gran número de parámetros diferentes.

Las desventajas que tiene el programa son:

Que los rayos mostrados por el programa son aproximaciones. Así por ejemplo, los rayos en realidad no deberían coincidir todos con el foco F_2, ya que ésto implicaría una energía infinita en este punto.

En este caso particular, el programa funciona correctamente con computadoras con tarjeta HERCULES; en otro tipo de monitor, la figura sale incompleta o "aplastada".

Esta es sólo la primera versión del programa *mexilit-soft* y como se mencionó, tiene mucho por mejorar y perfeccionarse, como el sistema para pantallas a color, como es el caso de CGA y VGA, así como ajustarse a sus diferentes resoluciones. Otra de ellas es que sea completamente adaptable para diferentes tarjetas de gráficos, sin importar su tipo y su resolución.

El sistema *mexilit-soft* tiene ciertamente una proyección a futuro que puede resultar interesante. El sistema tiene una característica muy importante: el programa está construído en lenguaje C. Este lenguaje tiene la propiedad de ser un lenguaje medio, es decir, con las instrucciones de un lenguaje de alto nivel pueden realizarse tareas de lenguaje ensamblador, y por consiguiente puede trabajarse con elementos de "hardware" (capítulo V).

****

GLOSARIO GENERAL

Acoplamiento.

Procedimiento de diseño que mide la comunicación entre datos y elementos de control.[59]

Acoplamiento por zonas compartidas.

Procedimiento de diseño en donde los módulos son atados en forma conjunta por medio de zonas globales para las estructuras de datos.[60]

Acoplamiento por zonas de datos.

Procedimiento de diseño parecido al acoplamiento por zonas compartidas, excepto que los elementos son en forma selectiva, entre las diversas rutinas que requieren de los datos.[61]

Ambiente operativo.

(Vease SISTEMA OPERATIVO).

B G I (Borland Graphics Interface).

Programas que funcionan como interfases para controlar diferentes tipos de monitores a partir de la programación.[62]

Carta estructurada.

Es una estructura jerárquica en forma de árbol que se emplea para el diseño de programas. En dicha estructura, los nodos del árbol son considerados como rutinas, las cuales se encuentran entrelazadas comunicándose a través de datos o banderas.

Código fuente.

Archivo que contiene las instrucciones de un programa.

C G A (Color Graphics Interface).

Controlador de video para pantallas a color cuyas resoluciones son de 320 x 200 y 640 x 200 puntos o pixeles.[63]

Cohesión.

Patrón de diseño que mide la relación de elementos de un módulo con otro.[64]

Cohesión funcional.

Patrón de diseño en el cual los elementos de un módulo se encuentran relacionados al desempeño de una sola función.[65]

Comando.

Orden especifica que se le da a la computadora dentro de algún ambiente de operación.

Compilador.

Es un programa que lee un programa escrito en un lenguaje alto o medio nivel (programa fuente) y lo traduce a un lenguaje de bajo nivel o de máquina (programa objeto). Como parte importante de este proceso, el compilador informa al usuario de la presencia de errores en el programa fuente.[66]

Criterios de aceptación.

Son aquellos que especifican las pruebas funcionales y de rendimiento que se deben realizar y los estándares que se aplicarán al código fuente, documentación interna, documentos externos tales como las especificaciones de diseño, plan de pruebas, manual de usuario, prinicipios de operación y procedimientos de instalación y mantenimiento.[67]

Depuración.

Ejecución de un programa paso a paso por bloques o segmentos.

Diagrama de flujo de datos (DFD).

Son gráficas dirigidas en donde los nodos especifican las actividades de proceso y los arcos o líneas la transferencia de datos entre nodos de proceso.[68]

Diseño arquitectónico.

Es aquél que se ocupa de la afinación de la vista conceptual del sistema, identificando funciones internas del proceso, descomposición de funciones de alto nivel y subfunciones, así como la definición de las cadenas de datos locales y su almacenamiento.[69]

Diseño detallado.

Es aquél que especifica los algoritmos, la instrumentación de las funciones, las estructuras de datos específicos que instrumentan el almacenamiento de los mismos, las iteraciones entre datos y funciones y el empacado del sistema.[70]

Diseño externo.

Es aquél que requiere de concebir, planear y especificar las características de un producto de programación.[71]

Drive.

Es un dispositivo de entrada y salida en donde se leen y se almacenan datos en disco duro o disco flexible.

E G A (Enhaced Graphics Adapter).

Es un controlador de video cuya resolución en la pantalla son 640 X 350 y 640 X 200 puntos o pixeles.[72]

Hardware.

Componentes físicos, tangibles y permanentes de una computadora o de un sistema de proceso de datos.[73]

HERCULES.

Controlador monocromático cuya resolución es de 720 X 348 puntos o pixeles.[74]

Heurística.

En términos del diseño de un programa, la heurística es una regla que se aplica a dicho diseño para que los datos que contiene el programa se intercomuniquen adecuadamente.

Interfase.

Dispositivo que sirve para comunicar elementos de *software* y *hardware*.

Lenguaje de alto nivel.

Es aquél cuya estructura e instrucciones son legibles para el programador.

Lenguaje de bajo nivel.

Es aquél cuyas instrucciones son poco legibles para el programador y directamente entendibles para la máquina.

Lenguaje de medio nivel.

Es aquél cuyas instrucciones son legibles para el usuario, es estructural y que pueden tambien realizar funciones directas a nivel de máquina.

Ligado.

Es también conocido como *"link"*. Es un programa que se encarga de enlazar todos los archivos involucrados en un código fuente.

Modo gráfico.

Es una faceta de la pantalla en donde se pueden desplegar letras góticas, gráficas y dibujos.

Modo texto.

Es una faceta de la pantalla en donde los caracteres se despliegan en forma normal.

Modularidad.

Característica del diseño estructurado o de la programación estructurada en donde el código fuente es dividido y ejecutado por bloques que se intercomunican entre sí.

Palabras reservadas.

Conjunto de instrucciones propias de un lenguaje de programación que no pueden ser empleadas para nombrar archivos, variables y ningún tipo de datos.

Pixel.

Conocido también como punto. Es la unidad mínima de la pantalla.

Pseudocódigo.

Es un conjunto de instrucciones detalladas empleando frases cortas y concisas en español (o cualquier idioma), las cuales se encuentran estructuradas por medio del uso de palabras clave como *si-entonces-si no*, *mientras*, *repetir* y *fin*.[75]

Reset.

Reajuste de la computadora.

Sistema de rastreo al azar.

Es aquél que traza las líneas componentes de una figura en cualquier orden que se le especifique.[76]

Sistema de rastreo con rastreador.

Es aquél que despliega el objeto como un conjunto de puntos a través de cada línea de rastreo en la pantalla.[77]

Sistema operativo.

Conjunto de programas que controla y organiza las actividades de la computadora. Realiza un enlace entre la computadora y los usuarios y los equipos periféricos. Aprovecha los recursos de la computadora.[78]

Software.

Conjunto de programas utilizable en una clase de computadoras, junto con la documentación asociada a la computadora o a los programas tales como manuales, diagramas e instrucciones de funcionamiento.[79]

Variables globales.

Son aquellas que son declaradas fuera de cualquier función y son conocidos a lo largo de todo el programa. Guardan sus valores durante la ejecución entera del programa y pueden usarse en cualquier segmento del mismo código.[80]

V G A (Video Graphics Adapter).

Es un controlador cuyas resoluciones pueden ser de 640 X 200, 640 X 350 ó 640 X 480 puntos o pixeles.[81]

Virus informático.

Son programas que provocan la pérdida de datos o archivos en los medios de almacenamiento de información, hasta daños al sistema y, algunas veces incluyen instrucciones que pueden ocasionar daños al equipo.[82]

BIBLIOGRAFIA

1. Prieto, Fernando E. y Achim M. Loske; **Underwater shock waves and extracorporal lithotripsy, an introduction to the bibliography,** Coordinación de la Investigación Científica, Universidad Nacional Autónoma de México, 1990.

2. Loske, Achim M.; **Generación de ondas de choque débiles en agua por rompimiento eléctrico,** tesis de Maestría en Ciencias (Física), Universidad Nacional Autónoma de México, Facultad de Ciencias, México D.F., 1990.

3. Loske, Achim M. y Fernando E. Prieto; "Las ondas de choque en la litotripsia extracorporal", **Revista de la Facultad de Medicina, U.N.A.M.**, vol. 33, Num. 2, marzo-abril 1990, pp. 113-120.

4. Loske, Achim M., **Ref. 2**, p. 23.

5. Loske, Achim M.y Fernando E. Prieto, **Ref. 3**, p. 115.

6. Loske, Achim M., **Ref. 2**, pp. 22-24.

7. Loske, Achim M. y Fernando E. Prieto; "Mexilit 1, generador de ondas de choque en agua", **Revista Ciencia y Desarrollo, CONACYT**, vol. XVII, Num. 101, noviembre-diciembre 1991, pp. 82-89.

8. **Ibid**, pp. 82-83.

9. Fairley, Richard E. ; **Ingeniería de software**, Ed. Mc.Graw-Hill, primera edición en español, México 1988.

10. Pressman, Roger S.; **Software engineering: a practitioners approach**, second edition, Mc.Graw-Hill, International Editions, Computer Science Series, 1987.

11. Fairley, Richard E., **Ref. 9**, pp.32-64.

12. **Ibid**, pp. 92-93.

13. Pressman, Roger S., **Ref. 10.**, pp. 139-158.

14. Fairley, Richard E., **Ref. 9**, pp. 144-160.

15. Pressman, Roger S., **Ref. 10**, pp. 214-257.

16. Fairley, Richard E., **Ref. 9**, pp. 205-239.

17. Pressman, Roger S., **Ref. 10**, pp. 404-428.

18. Fairley, Richard E., **Ref. 9**, pp. 304-326.

19 Pressman, Roger S., **Ref. 10**, pp. 499-522.

20. Ferreyra Cortés, Gonzalo; **Virus en las computadoras**, Ed. Macrobit, México 1990.

21. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw-Hill, Madrid 1990, pp. 316-318.

22. **Microsoft MS-DOS, operating system (MS-DOS Versión 3.2)**, Microsoft Corporation, 1986.

23. Kernighan, Brian W. y Bob Pike; **El entorno de programación UNIX**, primera edición en español, México 1987, Prentice-Hall Hispanoamericana S.A.

24. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1990, pp. 3-9.

25. Hearn, Donald y M. Pauline Baker; **Gráficas por computadora**, primera edición en español, México 1988, Prentice-Hall Hispanoamericana S.A., p. 53.

26. Kernighan, Brian y Dennis Ritchie; **The C programming language**, second edition, Prentice-Hall, 1988.

27. Vera, Francisco; **Matemática**, Lexicón Kapelusz, Ed. Kapelusz, segunda edición, marzo 1987, Buenos Aires, p. 239.

28. **Diccionario Durvan de la lengua española**, Durvan S.A. de Ediciones, Bilbao.

29. Solís, Rodolfo, Jesús Nolasco, Angel Victoria; **Geometría Analítica**, Programa del libro de texto universitario, Facultad de Ingeniería, U.N.A.M., México 1984.

30. Loske, Achim M., **Ref. 2**, pp 113-116.

31. Loske, Achim M. y Fernando E. Prieto, **Ref. 7**, pp. 84-86.

32. Loske, Achim M., **Ref. 2**, p. 108.

33. Vera, Francisco, **Ref. 27**, p. 493.

34. Loske, Achim M., **Ref. 2**, p. 105.

35. Loske, Achim M. y Fernando E. Prieto, **Ref. 7**, p. 85.

36. Vera, Francisco, **Ref. 27**, p. 534.

37. **VENIX programming guide**, 80 193CD, caps. 4-5, The Foxboro Company, January 13, 1989.

38. Loske, Achim M., **Ref. 2**, pp. 105-116.

39. Weiskamp, Keith, Loren Heiny, Namir Shammas; **Power Graphics using turbo C**, John Wiley and Sons Inc, 1989, pp.107-126.

40. Solís, Rodolfo *et al.*, **Ref. 29**, p. 149.

41. Weinkamp, Keith *et al.*, **Ref. 39**, pp. 109-111.

42. Pressman, Roger S., **Ref. 10**, pp. 166-172.

43. Weinkamp, Keith *et al.*, **Ref. 39**, pp. 1-24.

44. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1990, pp. 316-317.

45. Fairley, Richard E., **Ref. 9**, pp. 144-145.

46. Yourdon, Edward y Larry L. Constantine; **Structured design, fundamental of a discipline of computer program and systems design**, Prentice-Hall Inc., 1979.

47. Fairley, Richard E., **Ref. 9**, pp. 156-160.

48. Pressman, Roger S., **Ref. 10**, pp. 238-254.

49. Grogono, Peter; **Programación en Pascal**, Adisson-Wesley Iberoamericana, pp. 337-347.

50. Elbert, Theodore F.; **Embedded programming in Ada**, Van Nostran Reinhold Company, pp. 48-49.

51. Schildt, Herbert; **Programación en Modula-2**, Osborne / Mc Graw Hill, pp. 337-349.

52. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid, 1990, pp. 363-411.

53. Fairley, Richard E., **Ref. 9**, pp. 191-196.

54. Kernighan, Brian y Dennis Ritchie; **The C programming languaje**, second edition, Prentice Hall, 1988, pp. xi, 1.

55. Kernighan, B.W. y D.M. Ritchie; **El lenguaje de programación C**, primera edición en español, Prentice-Hall, 1985, pp. 3-4.

56. Hearn, Donald *et al.*, **Ref. 25**, pp. 51-54.

57. Loske, Achim M., **Ref. 2**, pp. 61-86.

58. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1990, pp. 303-327.

59. Fairley, Richard E., **Ref. 2**, p. 178.

60. **Ibid**, p. 157.

61. **Ibid**, p. 157.

62. Weinkamp, Keith *et al.*, **Ref. 39**, pp. 1-5.

63. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1990, p. 317.

64. Fairley, Richard E., **Ref.** 9, p. 178.

65. **Ibid**, pp. 158-159.

66. Aho, Alfred V., Ravi Sethi y Jeffrey D. Ullman; **Compiladores, principios, técnicas y herramientas**, Addison-Wesley Iberoamericana.

67. Fairley, Richard, **Ref.** 9, p.92

68. **Ibid**, pp. 161-163.

69. **Ibid**, p. 145.

70. **Ibid**, p. 145.

71. **Ibid**, p. 144.

72. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1990, p. 318.

73. **Diccionario Mc. Graw-Hill de Computación**, p. 237.

74. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1991, p. 318.

75. Fairley, Richard E., **Ref.** 9, p. 168.

76. Hearn, Donald *et al.*, **Ref.** 25, p. 34.

77. **Ibid**, p. 34.

78. UNISYS; **Basic UNIX usage workshop, student guide**, september 1988, UE-7829, U.S. America, module 1, p. 2.

79. **Diccionario Mc. Graw Hill de Computación**, p. 485.

80. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1991, pp. 168-169.

81. **Ibid**, p. 318.

82. Ferreyra Cortés, Gonzalo, **Ref.** 20, p. MF 2-1.

****

63. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1990, p. 317.

64. Fairley, Richard E., **Ref.** 9, p. 178.

65. **Ibid**, pp. 158-159.

66. Aho, Alfred V., Ravi Sethi y Jeffrey D. Ullman; **Compiladores, principios, técnicas y herramientas**, Addison-Wesley Iberoamericana.

67. Fairley, Richard, **Ref.** 9, p.92

68. **Ibid**, pp. 161-163.

69. **Ibid**, p. 145.

70. **Ibid**, p. 145.

71. **Ibid**, p. 144.

72. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1990, p. 318.

73. **Diccionario Mc. Graw-Hill de Computación**, p. 237.

74. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1991, p. 318.

75. Fairley, Richard E., **Ref.** 9, p. 168.

76. Hearn, Donald *et al.*, **Ref.** 25, p. 34.

77. **Ibid**, p. 34.

78. UNISYS; **Basic UNIX usage workshop, student guide**, september 1988, UE-7829, U.S. America, module 1, p. 2.

79. **Diccionario Mc. Graw Hill de Computación**, p. 485.

80. Schildt, Herbert; **Programación en Turbo C**, segunda edición, serie programación, Borland Osborne / Mc. Graw Hill, Madrid 1991, pp. 168-169.

81. **Ibid**, p. 318.

82. Ferreyra Cortés, Gonzalo, **Ref.** 20, p. MF 2-1.

****