UNIVERSIDAD PANAMERICANA

ESCUELA DE INGENIERIA

CON ESTUDIOS INCORPORADOS A LA UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

SIMULADOR POR COMPUTADORA PARA
FRESADORA DE CONTROL NUMERICO
EMCO F1 CNC

T E S I S

Que para obtener el Título de:

INGENIERO MECANICO ELECTRICISTA
AREA INGENIERIA MECANICA

p r e s e n t a

LEONEL PERNUDI CONTRERAS

Director FIS. Mariano Romero Valenzuela

México, D. F. *1994*

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Dedicatorias

A mis padres:

José Leonel Pernudi Castañeda y María Guadalupe Contreras de Pernudi.

Porque debo a su amor, a su apoyo y a su ejemplo lo que hoy soy.

A mi hermana:

Montserrat Pernudi Contreras.

Agradezco infinitamente su amor y admiración, que definitivamente representan una motivación especial en mi vida.

A mi abuela:

Inés Del Valle Vda. de Contreras.

Porque no ha hecho más que verter amor y cariño en mi.

Agradecimientos

Al Lic. Alejandro González Hernández.

Por todo el conocimiento y experiencia invertidos en este trabajo.

Índice

Introducción.

El control numérico ha adquirido tal importancia como medio de producción, que las máquinas que se rigen por los sistemas de mando numérico son necesarias para la industria actual. La necesidad surge debido a las exigencias de fabricación, que se reflejan en altas precisiones de maquinado que son cubiertas eficientemente por las máquinas controladas numéricamente.

Esta nueva necesidad de la industria ha tenido efecto en el sector educativo, ya que éste es el proveedor de los recursos humanos de la industria, dado que estos recursos deben estar lo mejor capacitados para cubrir las necesidades de la industria, hoy en día, el control numérico es parte de los planes de estudio de las carreras afines a la industria, principalmente metal-mecánica.

El aprender los métodos de programación en control numérico implica práctica por parte del alumno, lo cual lleva al manejo directo de las máquinas. Estas máquinas son de un costo elevado y son bastante más delicadas que una máquina-herramienta convencional, también cabe mencionar el costo del material utilizado en las pruebas, ya que para evitar el desgaste excesivo de la herramienta, es común utilizar materiales no metálicos mucho más suaves, pero que deben tener características especiales para soportar las condiciones del maquinado, lo que por

consiguiente los hace caros. Estos factores conducen a pensar que el proceso de enseñanza-aprendizaje del control numérico puede ser costoso, y en efecto lo es.

Cuando un proceso es caro, lo que se pretende es minimizar los errores dentro de dicho proceso; pero el evitar los errores casi siempre es cuestión de práctica, y cuando se está aprendiendo, es obvio que la práctica es poca o incluso nula lo cual puede significar elevación de los costos en el proceso. Cuando se enfrentan este tipo de eventualidades, existen alternativas que pueden ayudar a amortiguar la presencia de errores, estas alternativas son los diferentes métodos de ejemplificación de los procesos. La ejemplificación, mejor conocida hoy en día como simulación, es una herramienta poderosa para eficientar los procesos costosos y difíciles de controlar. La causas que dificultan el control del proceso son múltiples, pero en este caso en particular, la causa es como ya se mencionó anteriormente, la inexperiencia del operador.

La simulación se puede realizar de diferentes maneras, es decir, el modelaje a escala es una forma de simular, la reproducción de las condiciones de operación de un proceso para observar el desarrollo del mismo es otra, entre muchas más. Una de las más sofisticadas y eficientes es la simulación por computadora, la cual se ajusta bastante bien al problema del aprendizaje, cuando el proceso o el equipo involucrado es muy costoso. Para ver más claramente el concepto es oportuno hacer una comparación. Los simuladores de vuelo reproducen las condiciones de operación de una aeronave y someten al operador a una situación similar sin necesidad de arriesgar su vida ni la nave, de este modo el aprendiz puede adquirir la práctica necesaria, mediante la simulación, para enfrentarse a la situación real minimizando los riesgos. De manera análoga un simulador de una máquina-herramienta, en este caso de una fresadora, permite al usuario experimentar programas de maquinado y observar el efecto de dichos programas, hasta obtener el resultado deseado sin desperdiciar material, sin arriesgar la máquina e incluso ahorrando tiempo, ya que el proceso en la computadora puede realizarse bastante más rápido.

Habiendo contemplado lo anterior, se hace palpable la utilidad de un simulador por computadora para una máquina-herramienta de control numérico.

El trabajo está estructurado de la siguiente manera:

En los primeros capitulos se da una idea clara y breve del control numérico, de sus posibilidades, ventajas y desventajas, así como de los métodos y estructuras de programación.

Posteriormente se plantea la estructura general del programa, así como una breve explicación de los diferentes módulos que lo componen.

En los siguientes capítulos y finales se explican a detalle las diferentes rutinas que constituyen los módulos, es decir se mencionan las tareas que realizan y la estructura lógica con las que las llevan a cabo, así como las relaciones entre las diferentes rutinas cuando éstas existan.

Capítulo 1
Control Numérico

1.1 Historia del control numérico.

La máquina-herramienta ha jugado un papel fundamental en el desarrollo tecnológico del mundo, se ha observado una relación directa entre el desarrollo de las máquinas-herramienta y la tasa de crecimiento industrial. Gracias a la utilización de las máquinas-herramienta se ha podido realizar, de forma práctica, maquinaria de todo tipo, que aunque concebida y realizada no podía ser comercializada ya que no existían medios adecuados para su construcción industrial, podemos tomar el ejemplo de la máquina de vapor, que aunque fue inventada en 1766, no tuvo un desarrollo conveniente hasta 1776 en que John Wilkinson construyó la primera mandrinadora, gracias a la cual fue posible fabricar máquinas de vapor en gran escala.

A partir de ese momento el desarrollo industrial fue espectacular. La gran variedad de máquinas de vapor que estaban siendo construidas, permitieron ampliar la variedad y velocidad para la manufactura en serie.

Paralelamente fueron evolucionando las diversas herramientas de corte utilizadas por dichas máquinas-herramienta, el primer avance significativo fue la obtención de un material llamado acero de alta velocidad, que contenía cromo y tungsteno como aleantes para mejorar su

propiedades mecánicas. Finalmente, en 1930 se introdujo el carburo de tungsteno que una vez perfeccionado, es el más utilizado en la actualidad en las naciones industrializadas del mundo.

Conforme las necesidades de producción se incrementaron, se empezaron a dar ciertas agrupaciones de máquinas-herramienta del mismo tipo en las fábricas, con el objeto de realizar un maquinado por etapas, a fin de tener una mayor eficiencia, muchos trabajos se organizan de tal manera que grupos de piezas que requieren operaciones similares se manufacturen en un grupo de máquinas localizadas adyacentemente. Así, por ejemplo, si para la mecanización total de piezas fuera necesario realizar operaciones de fresado, mandrinado y barrenado, parece lógico que se alcanzaría mayor eficacia si todas las máquinas-heramienta necesarias para realizar estas operaciones estuvieran agrupadas. La conveniencia de realizar estas tres operaciones en una única máquina-herramienta, y otras necesidades que surgían a cada momento, forzaron la utilización de nuevas técnicas que permitiesen incrementar la producción reduciendo los errores. De esta forma se introdujo la automatización en los procesos de fabricación; a continuación se enumeran los factores que condujeron a la automatización:

1.1.1) Necesidad de fabricar productos que no se podían producir en cantidad y calidad suficiente con los métodos convencionales del momento.

1.1.2) Necesidad de obtener productos hasta entonces imposibles o de muy difícil fabricación por ser excesivamente complejos para ser controlados por un operador humano.

1.1.3) Necesidad de reducir los costos de fabricación para obtener precios suficientemente bajos.

Para solucionar todos estos problemas, se idearon diversos dispositivos automáticos, ya sea, eléctricos, neumáticos, mecánicos, hidráulicos, etcétera.

Inicialmente, el factor que regía la automatización fue la productividad, pero después las necesidades de la industria demandaron factores como la rapidez, la precisión y la flexibilidad. A

partir de entonces , todos los dispositivos automáticos intentan optimar la función de cuatro variables : productividad, rapidez, precisión y flexibilidad.

Los primeros sistemas no optimaban esta función, dado que eran dispositivos para realizar tareas muy específicas y por lo tanto con una rigidez extrema.

Hacia 1942 surgió lo que se podría llamar el primer control numérico, y surgió como una necesidad impuesta por la industria aeronáutica.

La aparición del control numérico permitió por primera ocasión optimar la función antes mencionada, ya que la flexibilidad es la característica principal de este automatismo.

En este punto es necesario dar una definición general del control numérico. [1]"Se considera control numérico todo dispositivo capaz de dirigir posicionamientos de un órgano mecánico móvil, en el que las órdenes relativas a los desplazamientos del móvil son elaboradas en forma totalmente automática a partir de informaciones numéricas definidas, bien manualmente(funcionamiento semiautomático), bien por medio de un programa (funcionamiento automático)."

El primer intento para dotar a una máquina-herramienta de algún tipo de control fue el desarrollado por Jacquard Loom que en 1801 ideó una máquina textil que permitía realizar distintos tipos de tejidos sin más que variar un programa alimentado a la máquina a través de tarjetas perforadas .

Posteriormente, vinieron otros intentos, como el del piano automático, que usaba un rollo de cinta perforada como medio de introducción del programa musical.

Estos primeros ejemplos no son considerados control numérico dadas las características antes definidas, pero son importantes de mencionar ya que son los inicios de este tipo de automatismo.

[1]Cfr. Alique López José Ramón, Control Numérico, España, Marcombo, 1981, p. 3

El primer intento exitoso para obtener un control numérico, surgió de la necesidad de fabricar hélices de helicóptero de diferentes configuraciones, y fue realizado por la Parsons que ya fabricaba diversos equipos para la defensa. Se probaron diversos métodos hasta llegar a la utilización de un computadora que gobernaba una máquina fresadora, moviendo la fresa en pequeños pasos incrementales, siguiendo la línea inicialmente calculada.

Dado el interés que suscitó esta técnica, la Fuerza Aérea de los Estados Unidos de Norteamérica concedió un contrato al Instituto Tecnológico de Massachusetts (MIT) para su desarrollo. El laboratorio de sistemas electrónicos del MIT diseñó y construyó, en 1952, un primer prototipo de fresadora de control numérico que gobernaba tres ejes.

Posteriormente se han desarrollado numerosos tipos de control numérico cada vez más sofisticados, pero con el problema de ser de ejecución costosa, complicada y de difícil programación, en especial los sistemas de contorneo. El desarrollo de la automatización, de la electrónica, el surgimiento de los microprocesadores, así como de las micro computadoras, ha empezado a elevar la rentabilidad para la aplicación del control numérico y a ampliar su ámbito de aplicación.

1.2 Ámbito de aplicación del control numérico.

Como se mencionó anteriormente, las variables que definen la eficacia de un sistema automático son : su productividad, rapidez, precisión y flexibilidad. Conforme a estas variables, en la gran mayoría de los casos, el tamaño del lote a fabricar, es el parámetro que nos puede ayudar a elegir el tipo de sistema automático conveniente.

Series de fabricación:

1.2.1) Grandes series (lote mayor a 10,000 piezas).

Para responder al problema de la gran serie, se utilizan sistemas secuenciales mecánicos, neumáticos, hidráulicos o electromecánicos. Si la serie es muy grande, el sistema debe poder permitir el trabajo simultáneo de varias cabezas, que a su vez, permitan cadencias muy grandes y por lo tanto, un rendimiento muy elevado. Las máquinas que realizan hoy en dia este tipo de producción son las máquinas tipo *transfert*, realizadas por varios subsistemas automáticos que trabajan de manera simultánea en forma más o menos sincronizada. El problema de este tipo de máquinas se encuentra en sus altos costos y tiempos de preparación, que obviamente son compensados por el tamaño del lote.

En series no tan grandes, se pueden utilizar sistemas secuenciales simples, en donde las secuencias de operación no son simultáneas, sino que se realizan una después de otra.

1.2.2) Series medianas. (lote entre 50 y 10,000 piezas).

Existen tres opciones de producción para este tipo de series:

1.2.2.1) Copiadoras.

1.2.2.2) Controles pre-programados numéricamente.

1.2.2.3) Controles numéricos.

La utilización de cualquiera de los tres, dependerá de la precisión, flexibilidad y rapidez exigidas.

Cuando la precisión y el tiempo no son factores de peso en la producción, la mejor opción, son las copiadoras por su economía. Existen copiadoras mecánicas hidráulicas, electromecánicas o electrónicas con las cuales la pieza a fabricar, se realiza por un desplazamiento del útil que reproduce el desplazamiento de un palpador.

Los controles programados numéricamente incorporan numerosas ventajas, pero son inflexibles dado su limitado número de secuencias mecánicas realizables.

El control numérico es especialmente interesante cuando el lote se encuentre entre 5 y 1000 piezas. Es en este caso donde el control numérico presenta numerosas ventajas, que se analizarán posteriormente.

1.2.3) Series pequeñas (lote menor a 5 piezas).

Para estas series, la utilización del control numérico no suele ser tan rentable, a no ser que se requiera una alta complejidad y precisión, y que pueda efectuarse programación con ayuda de una computadora. De no ser así los costos de programación elevarían demasiado el costo de fabricación. Para menos de 5 piezas el maquinado en máquinas convencionales resultarán más económicas.

A continuación se muestra una gráfica del número de piezas versus costo de la pieza.

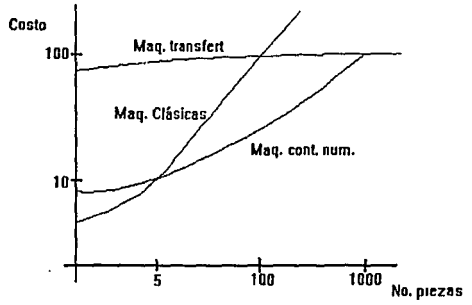


Figura 1.1 Número de piezas vs. costo.

1.3 Ventajas del control numérico.

De lo anterior se deduce, que siempre y cuando la producción se encuentre en limites medios de 5 a 1000 piezas, el control numérico encontrará una aplicación ideal, debido a las siguientes ventajas :

1.3.1) Posibilidad de fabricación de piezas imposibles o muy difíciles. Gracias al control numérico se han podido obtener piezas muy complicadas como las superficies tridimensionales necesarias en la fabricación de aviones. Es muy corriente, en la construcción aeronáutica, mecanizar piezas cuyo peso final representa 1/6 del peso de la pieza bruta inicial.

1.3.2) Seguridad. El control numérico es especialmente recomendable para trabajo con productos peligrosos.

1.3.3) Precisión. Esta ventaja es debida, en primer lugar, a la mayor precisión de la máquina-herramienta de control numérico respecto de las máquinas clásicas.

Los juegos mecánicos son menos importantes y la máquina-herramienta es en su conjunto mucho más precisa.

Otro factor que también influye en la precisión proviene del hecho de que una máquina-herramienta para control numérico es, en general, más universal que las máquinas clásicas y por tanto podrán hacerse más operaciones con la misma máquina.

Las precisiones alcanzadas en las máquinas-herramientas de control numérico van de 1 micra a 10 micras.

1.3.4) Aumento en la productividad de las máquinas. Este aumento de productividad se debe a la disminución del tiempo total de mecanización, debido a la eliminación de los tiempos de desplazamiento en vacio, y de la rapidez de los posicionamientos que proporcionan los sistemas electrónicos de control.

1.3.5) Reducción de controles y desechos. Esta reducción es debida fundamentalmente a la gran fiabilidad y repetitividad de una máquina con control numérico.

Los trabajos de mecanizado se realizan siempre siguiendo las mismas trayectorias y los juegos mecánicos de la máquina influyen siempre de la misma manera.

Esta reducción de controles, permite eliminar el retrabajo humano posterior, reduciendo así costos y tiempos de fabricación.

Por consiguiente, si las condiciones de maquinado han sido previstas adecuadamente y si las herramientas estaban bien arregladas, la máquina-herramienta obtiene piezas prácticamente idénticas y por tanto con precisión constante.

En los talleres convencionales se tiene un desperdicio del 3% al 4%, mientras que con las máquinas de control numérico, el desperdicio es tan sólo del 1%. Es evidente el ahorro que esto representa.

1.3.6) Flexibilidad. Basta cambiar el programa-pieza para que la máquina herramienta fabrique otra pieza, es decir podemos tener toda una librería de programas-pieza almacenados y simplemente alimentarlos a la máquina para fabricar las distintas piezas.

1.4 Clasificación de los sistemas de control numérico.

Hasta hace algunos años, se distinguían dos tipos fundamentales de sistemas de control numérico: los equipos de control numérico de posicionamiento, o de punto a punto, y los de contorneo o de posicionamiento continuo.

Hoy en día es más difícil hacer la misma clasificación, debido a que múltiples equipos cuentan con sistemas mixtos, es decir, algunos ejes están gobernados por contorneo y otros por posicionamiento. Aún así es conveniente tratar de explicar cada uno de estos sistemas por separado, ya que son el fundamento de los sistemas actuales.

En un sistema punto a punto el control determina, a partir de la información suministrada por el programa,es decir en función de los puntos inicial y final de la trayectoria, y antes de

iniciarse el movimiento, el camino total a recorrer. Posteriormente se realiza dicho posicionamiento, sin importar en absoluto la trayectoria recorrida, puesto que lo único que importa es alcanzar con precisión y rapidez el punto requerido. Este posicionamiento puede ser secuencial o simultáneo y se realiza normalmente a la velocidad máxima que soporta la máquina, es por ello que en muchos sistemas punto a punto no se controla la velocidad de avance ni la velocidad de rotación del útil. Sin embargo este tipo de sistemas sólo pueden realizar trayectorias rectas, debido a que sólo controlan la posición final, y la trayectoria que definen entre la posición inicial y el punto deseado es lineal. Los equipos de punto a punto son capaces de realizar trayectorias paraxiales, es decir que las trayectorias se realizan según los ejes de la máquina.

Cuando necesitamos realizar trayectorias curvas, es necesario que nuestro equipo cuente con características especiales; los equipos capaces de generar curvas reciben el nombre de equipos de contorneo.

Los sistemas de contorneo controlan no sólo la posición final sino el movimiento en cada instante de los ejes en los cuales se realiza la interpolación. En estos equipos deberá existir una sincronización perfecta entre los distintos ejes, controlándose por tanto, la trayectoria real que debe seguir la punta de la herramienta. Con estos sistemas se pueden generar recorridos, tales como: rectas con cualquier pendiente, arcos de circunferencia, cónicas o cualquier otra curva matemáticamente definible. Finalmente se observa que un equipo de contorneo puede realizar el trabajo de un equipo punto a punto, y que generalmente se utilizan para fresados complejos.

1.5 Control numérico con computadora (CNC)

Las necesidades y requerimientos de la industria, condujeron al concepto CNC.

La incorporación del computadora como elemento básico de síntesis ha revolucionado el campo del control numérico, haciendo que los equipos actuales ofrezcan numerosas posibilidades, algunas insospechadas, hasta hace sólo unos años.

Anteriormente, cualquier modificación en un diseño provocaba la modificación completa de los circuitos y sus correspondientes tarjetas impresas. Actualmente, con el CNC basta añadir memoria al programa para que el control realice nuevas funciones. Funciones tales como realizar automáticamente un diagnóstico completo del equipo de control antes de iniciarse cualquier operación, comprobar continuamente durante la mecanización las funciones de control, o realizar la lubricación de la máquina en función de su uso real, son algunas de las funciones normales para un CNC pero que anteriormente eran muy difíciles de realizar con lógica cableada.

Aún más con un mismo *hardware* es posible cumplir prestaciones diferentes sin más que cambiar la programación.

El principal inconveniente del CNC provenía del elevado costo del microprocesador y sus periféricos, pero hoy día con el avance tecnológico estos costos se han reducido notablemente, eliminando este problema.

Otro de los inconvenientes que ha sido superado es la velocidad de procesamiento de información, actualmente las velocidades de procesamiento, permiten realizar interpolaciones circulares a velocidades aceptables, estas características ya las brindan muchos procesadores comunes, lo cual representa una reducción del costo, debido a que en días pasados, sólo procesadores altamente sofisticados y costosos podían proporcionar estas velocidades de procesamiento.

Como se ha mencionado, el microprocesador abrió un nuevo campo de aplicación al CNC, desplazando los anteriores tipos de control numérico, en donde la mayoría de las variables del

proceso como velocidad de avance, velocidad de rotación del útil, holguras y otros factores eran constantes a lo largo de todo el proceso.

Capítulo 2

Programación

Los lenguajes de programación son las vías de comunicación entre la máquina y el operador. Sin importar el lenguaje, existe siempre una determinada estructura, es decir, el programador tiene que basarse en ciertos lineamientos, en general tendrá que alimentar a la máquina, las coordenadas de todas las cotas, así como las velocidades de corte y avance, para lo cual, es necesario haber realizado los cálculos pertinentes previamente.

Los lineamientos existentes se crearon por la necesidad de uniformizar la programación. En un principio, cada programador imponía sus parámetros, pero con el avance del control numérico, fue necesario estandarizar en forma estricta algunos parámetros. Esto se hizo con el fin de lograr ciertas ventajas, como : Intercambiabilidad de programas, programación flexible, en pocas palabras, compatibilidad.

Existen dos formas básicas de programación:

La programación manual o en lenguaje de máquina. En este caso el programa pieza se escribe únicamente por medio de razonamientos y cálculos que realiza el operario.

Programación automática. En este caso, los cálculos los realiza una computadora que después de realizar los cálculos, entrega en su salida, el programa en lenguaje de máquina. Es por ello que recibe el nombre de programación asistida por computadora.

La programación en lenguaje de máquina, requiere que los extremos de todos los segmentos, arcos de circunferencia, así como las coordenadas de los centros sean calculadas previamente.

2.1 Nomenclatura de ejes y movimientos.

La siguiente nomenclatura es conforme a la norma EIA (Electronic Industries Association)RS - 267 - A, conforme a la recomendación ISO (International Standard Organization) ISO - R841 (Control numérico de máquinas-herramienta - Nomenclatura de ejes y movimientos).

Eje X. Es un eje de translación principal, horizontal y perpendicular al eje Z. En máquinas que generan superficies de revolución, el movimiento del eje X es radial, y con sentido positivo, hacia afuera del eje de revolución. Sobre máquinas con herramienta de rotación, si el eje Z es horizontal, el eje X también es horizontal (plano X Z paralelo al suelo) y su sentido positivo es hacia la derecha cuando se mira desde la herramienta hacia la pieza

Si el eje Z es vertical, el eje X es horizontal y su sentido positivo es hacia la derecha cuando se mira desde la herramienta hacia el plano X Y.

Eje Y. Es un eje de translación principal, perpendicular al plano X Z. El sentido positivo del eje Y viene dado por el avance de un sacacorchos que va desde la dirección positiva del eje Z a la dirección positiva del eje X.

Eje Z. Es un eje de translación principal, perpendicular al plano X Y.

El eje Z es paralelo al eje principal de la máquina-herramienta, eje a lo largo del cual se mueve el útil de la máquina.

Existen ejes secundarios, paralelos a los ejes principales, denominados por las letras u, v, w, que son paralelos a X, Y, Z, respectivamente.

Estos ejes también pueden ser utilizados para definir movimientos no paralelos a los ejes principales. Existen también ejes terciarios, denominados por las letras p, q, r, que tiene la misma función. La aplicación de los ejes secundarios y terciarios dependerá de la complejidad de la pieza a realizar.

Existe también ejes de rotación, a, b, c, definirán la rotación alrededor de X, Y, Z, respectivamente; el sentido positivo estará dado, en cada caso, por la regla de la mano derecha.

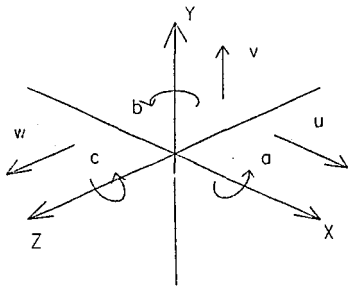


Figura 2.1 Definición de los ejes.

2.2 Programación manual.

La programación manual o en lenguaje de máquina, es todo el conjunto de datos que el control numérico necesita para realizar el mecanizado de la pieza.

Al conjunto de información que constituye una fase del mecanizado se le denomina bloque o secuencia. Cada bloque debe estar numerado para facilitar la búsqueda e interpretación.

La distribución de la información dentro de los bloque depende del formato de programación. Este formato de programación puede ser fijo o variable.

Un formato fijo es aquél en donde el número de caracteres y su función son constantes. La función del caracter está dada por la posición del mismo en el bloque. Este tipo de programación se encuentra obsoleto dada su extrema rigidez.

En un formato de programación variable, el número de caracteres es variable. Cada instrucción se compone de una letra llamada dirección y de un grupo de decimales, esta cifra da la amplitud de la dirección, esta amplitud puede ser, una velocidad de avance, una profundidad de corte, etcétera.

Se ha hablado de formatos, de bloques, de instrucciones y de caracteres, pero todavía falta mencionar que estos caracteres deben ser codificados; es decir, para manejar la información mediante los caracteres, es necesario hacerlo con un código. En un principio los códigos utilizados dependían del programador, pero con el tiempo se vió, la necesidad de uniformizar estos códigos para poder tener compatibilidad.

De los primeros códigos estandarizados que surgieron, está el propuesto por la EIA (*Electronic Industries Association*), este código utilizaba una tarjeta perforada de una pulgada, con ocho pistas. Los caracteres son codificados con un número impar de perforaciones en la cinta, es decir, con paridad impar.

Posteriormente para permitir la conexión entre los equipos de control se estudió un nuevo código, al que se le designó con el nombre de código ASCII (American Standard Code for Information Interchange).

Más tarde las necesidades condujeron al estudio de un nuevo código, que permitiera la intercambiabilidad entre máquinas análogas.

El organismo encargado del estudio fue ISO (International Organization for Standarization) como resultado se obtuvo el código ISO. Este código utiliza para información 7 dígitos binarios, de tal forma que el número de bits de cada símbolo sea impar. En este código el bit de paridad aparece en la pista número ocho de la cinta.

El código ISO consta de 50 caracteres de los cuales 40 son alfabéticos y numéricos, y el resto son caracteres especiales.

2.3 Formatos de Programación.

Un programa pieza típico tiene una estructura general que consta de las siguientes partes:

2.3.1) Texto previo. El texto previo es una parte opcional, y contiene las indicaciones técnicas de la fabricación, como son, número de pieza, número de programa, etcétera.

Si el texto previo se extiende a más de una línea, las líneas adicionales pueden escribirse entre paréntesis.

2.3.2) Principio de programa. Este se designa utilizando el símbolo correspondiente al código que se esté empleando.

2.3.3) Programa de mecanizado. El programa de mecanizado contiene todas las indicaciones necesarias para el proceso del mecanizado. Para escribir este programa es necesario seguir los lineamientos que marque el formato correspondiente.

2.3.4) Fin de programa. Este se designa utilizando el símbolo correspondiente al código que se esté empleando.

El formato de una instrucción, indica la forma correcta de escribir dicha instrucción, como se había explicado antes, las direcciones son la parte literal de una instrucción, las letras pueden

ser mayúsculas o minúsculas dependiendo del código (EIA o ISO) , las direcciones van seguidas de una o dos cifras decimales, que algunas veces están separadas por un punto.

La primera cifra decimal que sigue a la dirección indica el número máximo de cifras a la izquierda de la coma. En caso de existir una segunda, indica el número máximo de cifras a la derecha de la coma. Cuando la dirección va seguida de en signo "+ ", indica que se podrán utilizar los signos positivo y negativo en la programación absoluta.

Cuando el formato permite la omisión de ceros a la izquierda, la designación deberá ser por tres cifras, en vez de dos, es decir, deberá ir un cero antes de la primer cifra. En caso de que los ceros a la derecha puedan ser omitidos la última cifra deberá ser un cero.

Cada fabricante utiliza sus direcciones, pero algunos de los caracteres más comúnmente usados y su significado son los siguientes:

N . Es la dirección que indica el número de bloque. Esta dirección se encuentra seguida de tres o cuatro cifras decimales, según el formato.

El número de bloque es la primera instrucción de cada secuencia en un programa.

En un programa también se pueden encontrar secuencias opcionales, esto es, secuencias condicionadas que se ejecutarán dependiendo de la posición de un interruptor de secuencias opcionales. Este tipo de secuencias se caracterizan por el simbolo " / " delante del número de secuencia.

X, Y, Z. Son las direcciones correspondientes a las cotas según los ejes X, Y, Z de la máquina-herramienta.

Actualmente su formato de programación suele ser X,Y,Z + 04.3. Por tanto como la cota se expresa directamente en milímetros, utilizando este formato se deduce que la distancia máxima programable es +/- 9,999.999 mm.

G . Es la dirección correspondiente a las funciones preparatorias. Estas funciones preparatorias se utilizan para informar al control de la operación del mecanizado. dependiendo

de las diferentes casas comerciales, las funciones preparatorias se utilizan para programa funciones como : forma de la trayectoria, tipo de corrección de la herramienta, parada temporizada, ciclos automáticos y programación incremental o absoluta.

Normalmente esta dirección va seguida de dos cifras decimales, por lo que permite programar hasta 100 funciones preparatorias.

M . Esta es la dirección correspondiente a las funciones auxiliares o complementarias. Se usan para indicar a la máquina herramienta la realización de operaciones como: parada programada, sentido de rotación del husillo, cambio de herramienta, etcétera. Al igual que la dirección anterior, ésta va seguida de un número de dos cifras.

F . Es la dirección correspondiente a la velocidad de avance, normalmente se utilizan cuatro cifras para designarla y tiene unidades de mm/min. En muchos equipos la velocidad puede ser expresada en mm/vuelta o en pulgadas/minuto, a través de una función preparatoria.

S . Es la dirección que determina la velocidad de rotación del husillo principal, actualmente esta velocidad se programa directamente en revoluciones por minuto.

I, J, K. Son las direcciones para programar arcos de circunferencia. Para esto se toman a I, J, K como los ejes secundarios de X, Y, Z respectivamente.

Su formato de programación es igual al de las cotas.

T. Es la dirección que designa el número de la herramienta.

Además de estas direcciones que son prácticamente estándar para todos los equipos, existen otras direcciones particulares que varían de acuerdo al equipo y marca.

2.4 Funciones Preparatorias.

A continuación se describirán las funciones preparatorias más importantes dentro de la norma ISO 1056.

Función G00. Esta función indica que la trayectoria programada en el bloque correspondiente se debe realizar a la máxima velocidad posible.

La velocidad de avance programada es mantenida, y continuará en el siguiente bloque que no contenga G00.

Función G01. Con esta función, los ejes se gobiernan de manera que la trayectoria será una linea recta, donde la velocidad estará dada por la velocidad de avance programada.

Las coordenadas del punto final de la recta se expresan mediante las direcciones X, Y, Z , con la posibilidad de programarse en cotas absolutas o incrementales. (G90 para absolutas y G91 para incrementales).

Funciones G02 y G03. Para utilizar estas funciones un bloque deberá contener:

2.4.1) G02 una interpolación circular en sentido horario y G03 una interpolación circular en sentido antihorario.

2.4.2) Las coordenadas del punto final de la interpolación; que pueden ser expresadas en forma incremental o absoluta, referidas mediante las direcciones X, Y, Z.

2.4.3) Las direcciones de interpolación I, J, K. Con estas direcciones se definirá el centro de la interpolación, de forma incremetal, tomando como referencia el punto inicial.

Con los datos anteriores, es decir, punto en el que se encuentra, punto final de la trayectoria y centro del arco de circunferencia, el control interpola la trayectoria circular, siempre y cuando ésta sea factible.

Generalmente la mayoria de los equipos sólo interpolan por cuadrante, esto significa que si se quisiera efectuar un circulo se necesitarian cuatro interpolaciones.

Función G04. Esta función realiza una interrupción del programa durante un lapso programado, reanudando la secuencia una vez transcurrido el intervalo de tiempo. La duración del intervalo se programa de manera diferente dependiendo del equipo en cuestión.

Función G06. Esta función es para realizar una interpolación parabólica. Un arco de parábola se programa mediante el extremo del arco y el punto de intersección de las rectas tangentes a los extremos del arco.

Función G07. Con esta función se indica a la máquina que incremente la velocidad de avance hasta un valor programado.

Función G09. Esta función preparatoria indica a la máquina que debe reducir la velocidad de avance. Esto se utiliza con el propósito de alcanzar posiciones determinadas con gran precisión, es decir, para el mecanizado de contornos precisos.

Función G17-G19. Mediante estas funciones se determinan los planos en los que se llevarán a cabo las interpolaciones circulares o de corrección del útil.

Cuando se utilizan para seleccionar los planos de compensación del útil, estas funciones se programan junto con G41 y G42 , que suministran la dirección de la corrección.

Función G33. Esta es el ciclo automático de roscado. Tiene mayor aplicación en los equipos para control de tornos.

Los equipos con ciclo de roscado deben poseer un sensor acoplado al husillo principal que detecte la velocidad real de giro, esto con el fin de sincronizar perfectamente la velocidad de avance del carro con el giro de husillo.

El paso de rosca se puede programar mediante el paso de rosca en mm o en pulgadas, a través de la dirección K, o mediante la velocidad de avance de los carros en mm / vuelta o en pulgadas / vuelta.

La ecuación que determina la velocidad de avance es :

F (mm / min) = Paso (mm) n (r.p.m.)

o bien :

F (mm / min) = F (mm / vuelta) n (r.p.m.)

La dirección de la rosca, se programa mediante la dirección de giro del husillo principal.

La velocidad de giro y su sentido se programan en el bloque anterior, con el objeto de que el husillo alcance la velocidad adecuada.

Normalmente el roscado se realiza en varias pasadas, por lo que el desplazamiento del eje se inicia siempre en la misma posición angular entre pieza y herramienta.

Función G34. Esta es el roscado con incremento gradual de paso.

Función G35. Es análoga a la anterior pero con decremento en el paso.

Función G40. Esta función anula toda instrucción de corrección del útil.

Función G41. Indica a la máquina la corrección del radio de una herramienta izquierda. Entiéndase por izquierda a la herramienta que queda situada a la izquierda de la superficie a mecanizar. Se utiliza en el mecanizado de contornos exteriores.

Función G42 . Función que permite la corrección del radio para herramientas derechas, se utiliza para contornos interiores.

Función G43. Se emplea para programar una corrección positiva del útil. Esta instrucción se ve anulada por la función G40, como se mencionó anteriormente.

Función G44. Se utiliza para programar una corrección negativa de la herramienta. Al igual que la anterior se anula con G40.

Funciones G45-G52. Estas funciones indican que el valor de corrección de la herramienta debe ser sumado o restado de las cotas del bloque, o no ser tomado en cuenta.

Las correcciones usuales son las siguientes:

Función	Operación
G45	X + R , Y +R

G46	X + R , Y - R
G47	X - R , Y -R
G48	X - R , Y - R
G49	X , Y + R
G50	X , Y - R
G51	X + R , Y
G52	X - R , Y

Tabla 2.1 Correcciones de la herramienta.

Función G53. Esta función permite anular las correcciones del origen.

Funciones G54-G59. Estas funciones permiten programar correcciones del origen. Estas correcciones son necesarias en algunas ocasiones, debido a que las operaciones se realizan respecto del origen, y como pueden haber cambios en la posición de la pieza respecto a la mesa, es necesario efectuar dichas correcciones del origen.

Normalmente las correcciones son:

Función	Operación de corrección del origen en el eje:
G54	X
G55	Y
G56	Z
G57	X ,Y
G58	X ,Z
G59	Y ,Z

Tabla 2.2 Correcciones del origen.

Función G60. Sirve para programar paradas precisas.

Función G61. Se utiliza para programar paradas de precisión media.

Función G62. Esta función preparatoria se utiliza en posicionamientos rápidos que permiten ganar tiempo a cambio de alcanzar precisiones malas.

Función G63. Programa un posicionamiento rápido con un paro del husillo al final del desplazamiento.

Función G64. Esta función programa encadenamientos continuos entre bloques de programa sin reducciones de velocidad de avance ni paradas intermedias. Produce contornos redondeados, cuando hay cambios en la dirección del movimiento.

En los equipos de punto a punto y paraxiales se utilizan las funciones preparatorias G71-G75 para indicar el tipo de posicionamiento.

Funciones G71-G75. En equipos de contorneo suelen usarse para programación en pulgadas.

Función G71. Sirve para programar posicionamientos no precisos a velocidad rápida.

Función G72. Análoga a la anterior en los equipos que no utilizan desaceleración por etapas.

Función G73. Programa posicionamientos unidireccionales, de esta forma se obtiene la máxima repetitividad de posicionamiento.

Función G74. Posicionamiento a velocidad de trabajo, programada previamente F4.

Función G75. Análoga a la anterior, pero ejecuta un disminución de velocidad al final para mejorar la precisión del posicionamiento.

Actualmente el grupo anterior de instrucciones (G70) se utilizan muy poco en equipos para fresadoras - mandrinadoras.

Función G80. Esta función anula todos los ciclos automáticos G81 - G89.

Función G81. Este ciclo automático permite programar en un solo bloque una operación de taladrado, con la siguiente secuencia:

2.4.1) Desplazamiento rápido hasta la cota programada.

2.4.2) Movimiento de acercamiento de la herramienta hasta la cara a mecanizar.

2.4.3) Rotación de la herramienta.

2.4.4) Avance a velocidad de trabajo hasta la profundidad programada. Esta profundidad se programa con una instrucción especial.

2.4.5) Subida de la herramienta a velocidad rápida hasta la cara superior de la pieza.

2.4.6) Desplazamiento rápido hasta el nuevo punto a mecanizar.

Este bloque requiere de ciertos parámetros, mismos que se suministran mediante instrucciones especiales.

Función G82. Este ciclo automático es análogo al anterior y sirve para realizar barrenos pero con permanencia, se realiza con la misma secuencia, pero al finalizar el barreno a la velocidad de trabajo, se hace una parada temporizada de la herramienta, antes de iniciarse la operación siguiente.

Función G83. Este ciclo permite realizar barrenos profundos. La secuencia de operación es la siguiente.

2.4.1) Desplazamiento rápido hasta la cota programada.

2.4.2) Movimiento de acercamiento de la herramienta hasta la cara a mecanizar.

2.4.3) Rotación de la herramienta.

2.4.4) Avance a velocidad de trabajo hasta una cota intermedia.

2.4.5) Subida de la herramienta a velocidad rápida hasta la cara superior de la pieza.

2.4.6) Bajada de la herramienta a velocidad rápida hasta la cota intermedia.

2.4.7) Avance a velocidad de trabajo hasta la profundidad programada.

2.4.8) Subida de la herramienta hasta la cara superior de la pieza.

Función G84. Este ciclo permite realizar roscados interiores. La secuencia de operación es la siguiente:

2.4.1) Desplazamiento a la velocidad de avance hasta la cota programada.

2.4.2) Bajada de la herramienta en rápido, hasta la cara superior de la pieza.

2.4.3) Rotación de la herramienta.

2.4.4) Avance a velocidad de trabajo hasta la profundidad programada.

2.4.5) Inversión de rotación del sentido de la herramienta.

2.4.6) Retorno hasta la cara superior de la pieza de trabajo.

Función G85. Sirve para realizar escariados. Es similar al anterior, con la diferencia de que no se invierte el sentido de rotación de la herramienta.

Función G86. Este ciclo permite ejecutar mandrinados, su secuencia de operación es la siguiente:

2.4.1) Desplazamiento en rápido hasta la cota programada.

2.4.2) Bajada de la herramienta en rápido, hasta la cara superior de la pieza.

2.4.3) Rotación de la herramienta.

2.4.4) Avance a velocidad de trabajo hasta la profundidad programada.

2.4.5) Parada del mandrino.

2.4.6) Retorno hasta la cara superior de la pieza de trabajo.

Función G87. Es igual al anterior, con la única diferencia de que la velocidad rápida es programable.

Función G88. Es otro ciclo de mandrinado, con la diferencia de que al llegar a la profundidad programada, se realiza una parada temporizada del mandrino.

Función G89. Realiza mandrinados con la siguiente secuencia:

2.4.1) Desplazamiento en rápido hasta la cota programada.

2.4.2) Bajada de la herramienta en rápido, hasta la cara superior de la pieza.

2.4.3) Rotación de la herramienta.

2.4.4) Avance a velocidad de trabajo hasta la profundidad programada.

2.4.5) Parada temporizada del mandrino.

2.4.6) Retorno a velocidad de trabajo hasta la cara superior de la pieza de trabajo.

Funciones G90-G91. Estas dos funciones indican a la máquina el tipo de programación, ya sea absoluta o incremental. En la programación absoluta (G90), las cotas se programan con respecto al cero del sistema de referencia. En la programación incremental (G91), se indican con respecto al punto anterior programado.

Función G92. Mediante esta función se puede desplazar el cero a cualquier punto del sistema de coordenadas de la máquina. Al detectar esta función en un bloque, los registros de posicionamiento de los ejes son modificados de acuerdo con los valores escritos con las direcciones de los ejes.

Función G93. Con esta función se suministra la velocidad de avance como inversa del tiempo en minutos necesario para ejecutar los bloques.

Función G94. Esta es la función que indica las unidades en que se programó la velocidad de avance, ya sea mm / min o pulgadas / min.

Función G95. Con esta función se indica que la velocidad está programada en mm / vuelta. Es especialmente funcional para tornos.

Función G96. Esta función se utiliza para mantener la velocidad de corte constante, esto, mediante la variación de la velocidad de giro del husillo. La velocidad de giro del husillo W se modifica constantemente en función del diámetro X de la pieza, según la ecuación :

$$V = W\,R \qquad \text{donde} \qquad W\,X = cte$$

Función G97. Mediante esta función se indica que la velocidad programada mediante la dirección S esta en r.p.m

2.5 Funciones auxiliares.

Las funciones auxiliares M también reciben el nombre de complementarias.

Como se dijo anteriormente se utilizan para indicar a la máquina-herramienta que debe efectuar operaciones como: parada programada, cambio de herramienta, refrigeración de la herramienta, etcétera.

Las funciones M no asignadas pueden ser ocupadas por el operario de la máquina-herramienta para controlar operaciones a voluntad. La asignación de las funciones no reconocidas por la máquina, debe hacerla el operario mediante la codificación externa adecuada.

Las funciones auxiliares normalmente asignadas en los equipos son :

Función M00. Esta función sirve para realizar una parada incondicional del programa detiene el husillo y la refrigeración. El programa puede restablecerse mediante el control externo, una vez finalizada la operación que requería del paro.

Función M01. Esta función permite una parada opcional, misma que determina el operador mediante un control externo.

Función M02. Esta función indica el fin del programa, debe ser utilizada en el último bloque.

Función M03. Con esta función se programa la dirección de rotación del husillo en sentido horario.

Función M04. Programa el giro del husillo en sentido antihorario.

Función M05. Esta función programa una parada del husillo, también se usa para indicar el fin de la refrigeración.

Función M06. Esta función indica un cambio de la herramienta, este cambio puede realizarse manual o automáticamente.

Función M07-M08. Se emplean para indicar distintas maneras de refrigeración en la herramienta.

Función M09. Programa una detención de cualquier refrigeración que pueda existir en el útil en ese momento.

Función M17. En algunos equipos se utiliza para indicar el fin de una subrutina.

Función M30. Esta instrucción se utiliza para indicar el fin del programa, con salto al principio del mismo.

Función M36. Esta función indica al control que la velocidad será programada mediante la instrucción F4.

Función M37. Se utiliza para especificar una gama de velocidades de avance.

2.6 Saltos de programa y subrutinas.

Las funciones de salto, permiten la repetición automática de partes del programa principal o de subprogramas. Para realizar los saltos se utiliza una función preparatoria de las no asignadas.

Cuando se realiza un salto, se debe indicar a qué bloque del programa y cuántas veces se repetirá el subprograma.

Las funciones de salto habilitan la realización de subrutinas, las cuales pueden ser llamadas las veces que se requieran, desde el programa principal. En estas subrutinas se manejan ciertos parámetros que se dirigen desde el programa principal, y las convierten en ciclos automáticos totalmente definidos a gusto del operador.

Cada vez que una subrutina es llamada, se deberá indicar en el programa principal y en el mismo bloque en el que se realiza la llamada, los valores numéricos que deberán asignarse a los parámetros de dicha subrutina.

Capítulo 3
Acerca del Programa

3.1 Características generales.

El Simulador por computadora para la fresadora de control numérico F-1 CNC de EMCO es un programa realizado en Turbo Pascal V. 6.0, consta de 6814 líneas, que componen 8 unidades y el programa principal.

El objetivo de realizar dicho programa es generar una herramienta versátil para ejemplificar el proceso de maquinado de una fresadora de control numérico, a partir de la entrada de un programa en lenguaje ISO, realizado por el operador de la máquina-herramienta, que en este caso, está supuesto a ser un estudiante de ingeniería de al menos séptimo semestre. Esto significa que el programa está orientado a un tipo de usuario específico, que deberá tener conocimientos básicos de dibujo técnico y de otras áreas de la carrera de ingeniería. Este es un factor importante a considerar, ya que dadas las características del proceso de fresado, es extremadamente difícil realizar una simulación exacta del proceso, debido a que por los

diferentes movimientos de la herramienta con respecto a la pieza (ver diagrama siguiente) , es necesario considerar al menos dos planos, para tener una idea aproximada de lo que está sucediendo; pero tomando en cuenta los conocimientos del usuario es posible hacer uso de algunos conceptos de dibujo técnico, además de otros recursos que posteriormente se detallarán, para lograr una ejemplificación que de al usuario una idea clara, facilitando la realización del simulador.

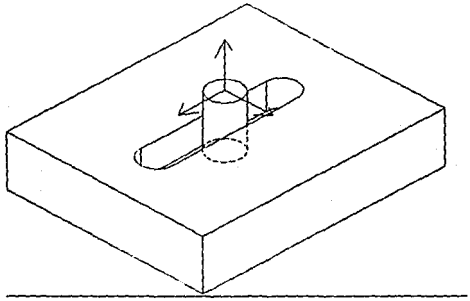


Figura 3.1 Movimientos de la herramienta con respecto a la pieza.

Otra premisa que se debe tener en cuenta es que el programa está ideado para ayudar al alumno más no para pensar por él y menos aún realizar la tarea por él. Para obtener provecho de esta herramienta, es necesario que el usuario tenga conocimiento previo del lenguaje de programación ISO, y también es totalmente recomendable que realice previamente el diseño de

la pieza deseada así como el programa correspondiente en lenguaje ISO (programa-pieza) , para que observe la simulación teniendo una idea clara del resultado deseado.

La simulación brinda otras ventajas tales como :

3.1.1) Evitar riesgos.

Existen diversos riesgos, en algunas ocasiones se programa erróneamente la profundidad de corte, resultando en daños a la mesa de la máquina, y así como este se pueden cometer una buena cantidad de errores que pueden dañar el equipo y si se tiene en cuenta el costo del equipo en cuestión, el prevenir este tipo de errores es correr menos riesgos.

3.1.2) Ahorra material. Dado que el usuario puede observar el resultado de su programa en la computadora antes de ejecutarlo físicamente en la máquina-herramienta es posible corregir las fallas sin necesidad de desperdiciar material.

3.1.3) Ahorra tiempo. Para obtener resultados óptimos e incluso para no averiar las herramientas es necesario maquinar con velocidades específicas, que repercuten en un determinado tiempo de maquinado, esto no es necesario en la computadora, por lo que el proceso puede realizarse más rápidamente.

La fresadora para la que está diseñado el programa cuenta con una interfase de conexión RS-232, la que brinda la posibilidad de transmitir datos, con el formato adecuado, desde una computadora. Esto permite que la computadora se convierta en un intermediario del usuario y la máquina-herramienta, en donde el usuario tiene la facilidad de realizar sus programas-pieza, editarlos, compilarlos (verificar sintaxis del programa), simularlos y transmitirlos a la fresadora para que ésta realice el maquinado de la pieza deseada.

El proceso se puede esquematizar mediante el siguiente diagrama:

El proceso inicia con la entrada a la computadora, la entrada es el programa-pieza, el

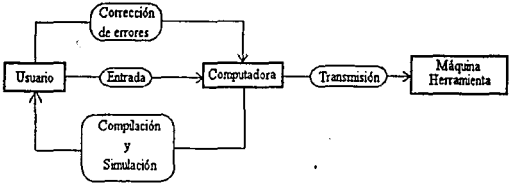


Figura 3.2 Esquema del proceso simulación - fabricación.

simulador realiza la compilación y simulación, las cuales son retroalimentadas al usuario, con esta información el usuario puede detectar sus errores, y finalmente transmitir el programa a la máquina-herramienta para que se maquine la pieza.

3.2. Características especiales.

Además de las ventajas que brinda la simulación el programa tiene algunas características especiales que ofrecen otras comodidades.

Se distinguen cuatro funciones generales del programa: Archivo, Editor, Compilador y Simulador.

Las funciones de Archivo permiten al usuario iniciar un nuevo programa, traer a memoria un programa previamente realizado y almacenado en disco flexible o disco duro, almacenar un programa e imprimir el programa en memoria.

Estas funciones brindan ventajas para el almacenamiento y manejo de los programas-pieza que se realicen, debido a que la fresadora tiene la opción de almacenar la información en cinta magnética, que es un medio más lento menos confiable y con menor capacidad que los discos flexibles o el disco duro de una computadora.

La opción de impresión del programa-pieza en memoria ofrece al usuario una visión global del programa, superando las 20 líneas que un monitor puede mostrar.

Las funciones del editor permiten al usuario insertar y borrar líneas, tiene la ventaja de poder utilizar el teclado de la computadora para efectuar la entrada y edición de los datos, que comparado con el teclado de la máqina-herramienta es bastante más cómodo. Esta comodidad comienza desde el espacio y distribución de las teclas, hasta la facilidad de movimiento en cualquier dirección de la pantalla, atributo con el que no cuenta la fresadora. La edición de datos es más rápida y práctica, ya que en la fresadora es necesario borrar el dato entero y entrarlo de nuevo, y en el editor del simulador se puede realizar la corrección de una sola cifra del dato sin borrar el resto de las cifras. Características como éstas y algunas otras serán descritas a detalle en el manual del usuario.

El editor cuenta con una ventana que despliega mensajes, ya sea de ayuda o de error según sea el caso, para orientar al usuario.

En cuanto a las funciones del compilador están la de compilar y la de transmitir.

Mediante la compilación el usuario podrá saber si su programa-pieza presenta algún error, de acuerdo con la lógica de la fresadora, es decir contiene las 18 alarmas que se presentan

en la máquina-herramienta cuando el programa-pieza tiene algún error, ya sea de sintaxis o de lógica de programación.

La opción de transmitir envía el programa en memoria a la fresadora, la transmisión de los datos se hace con el formato requerido por la máquina-herramienta y es previamente compilado para evitar los errores antes mencionados.

La función de simulador presenta dos opciones, simulación continua y simulación por pasos. La simulación continua ejecuta los comandos del programa-pieza sin interrupciones, sin embargo el usuario tiene la opción de pausar la simulación en el momento que lo desee. La simulación por pasos realiza una pausa después de ejecutar cada comando del programa-pieza, la ejecución del programa es reanudada por el usuario. Con esta opción es más fácil observar el funcionamiento de cada comando del programa-pieza.

3.3 Características de la simulación.

Como se mencionó anteriormente, como base los conocimientos del usuario, la simulación está basada en algunos conceptos de dibujo técnico y en simbología propia, que facilitan la realización del simulador.

Para comenzar es necesario decir que el simulador esquematiza tres vistas de la pieza y la herramienta. La vista superior que es totalmente controlada por el programa, y las vistas lateral y frontal que son parcialmente controladas por el operador. En el siguiente diagrama se muestra un ejemplo de una pieza con la herramienta cortando y su respectiva representación mediante las tres vistas.

Figura 3.3 Diagrama Pieza-herramienta en isométrico.

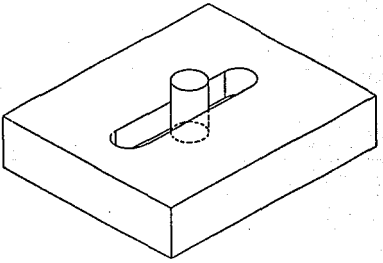
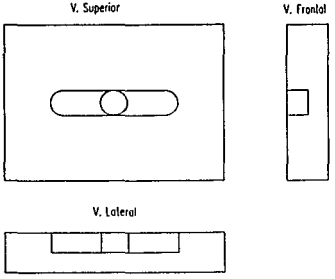


Figura 3.4 Diagrama en tres vistas.



3.3.1 Vista Superior.

La vista superior es totalmente controlada por el simulador, pero existen algunos símbolos que se deben mencionar. La herramienta solo se verá cuando se encuentre dentro del área que describe la pieza bruta, en caso de salir de la pieza por cualquiera de las fronteras el círculo que la representa no será visible, en el caso de que la herramienta desaparezca, existen unas miras que se encuentran ubicadas a un lado de la pieza y que se posicionan en lo que corresponde al centro de la fresa, y que dan la posición de la herramienta.

Para diferenciar los diferentes planos, el cortador se mueve dejando un trazo de color asignado previamente, de acuerdo con la altura a la que se encuentre. La asignación del color es realizada por el simulador, existen 14 colores disponibles. En caso de que el usuario maneje más de 14 planos el simulador asignará los colores de nuevo, esto significa que el plano 15 tendrá el mismo color que el primer plano utilizado.

El simulador no detecta profundidades, es decir realiza los trazos no importando que exista enfrente o detrás del trazo actual. Por esto es que las tres vistas trabajan en conjunto y mejoran la percepción del usuario.

En esta vista existe otro tipo de trazo, este se realiza cuando el cortador se encuentra dentro de los límites de la pieza bruta y se está desplazando pero no está cortando, en este caso el resultado es el trazo de una línea por el centro de la herramienta.

3.3.2 Vistas lateral y frontal.

Estas dos vistas son similares en cuanto su funcionamiento, la diferencia es que la vista lateral muestra el plano X - Z y la vista frontal se encarga del plano Y - Z.

Estas dos vistas son controladas parcialmente por el usuario, existen 4 interruptores accionados por las teclas F1, F2, F3 y F4. Los interruptores dan la opción de activar o no el

trazo en cualquiera de las vistas, así como de borrar los trazos anteriores en cualquiera de las vistas. Estas funciones fueron diseñadas así, debido a que el simulador no detecta las profundidades y siempre traza, más sin embargo el usuario sí tiene idea de las profundidades, por lo que el puede borrar a placer el trazo o desconectarlo, cuando lo necesite dependiendo de la profundidad a la que se encuentre la herramienta. Esto significa que suponiendo que el usuario se encuentre simulando una pieza previamente diseñada, sabrá en qué líneas del programa-pieza está maquinado los planos de interés, como tiene la opción de borrar, cada vez que se finalice un plano de interés, él podrá borrar la vista, y observar con claridad el siguiente plano, o en le caso de que no le interese observar el trazo, podrá desactivarlo para observar únicamente el paso de la herramienta a través de la pieza. El usuario también tiene la opción de interrumpir la simulación en el momento que lo desee.

Como se mencionó anteriormente, estas vistas son controladas parcialmente por el usuario, lo que indica que el simulador controla la otra parte. Las operaciones que controla el simulador son las formas de trazo, y también desactiva el trazo cuando la herramienta sale de los límites de la pieza bruta, no importando la posición de interruptores que maneja el usuario.

3.3.3 Ventana de Indicadores.

Esta ventana presenta los siguientes datos:

3.3.3.1) Las coordenadas de la herramienta, de esta manera el usuario conoce la posición de la fresa en cualquier momento.

3.3.3.2) Despliega la línea del programa que se está ejecutando, así como la línea anterior y posterior. Con esta indicación el usuario tiene noción exacta de la operación que se realiza con cada instrucción.

3.3.3.3) Se encuentran especificadas las funciones de cada tecla. Esto es barra espaciadora reanuda la ejecución, Enter interrumpe la ejecución (pausa), F1 activa o no el trazo en la vista lateral, F2 Limpia la vista lateral, F3 activa o no el trazo en la vista frontal y F4 limpia la vista frontal.

3.3.3.4) Finalmente hay un indicador que le dice al operador si el husillo se encuentra encendido o apagado.

3.4 Estructura del programa.

El programa está contenido en nueve archivos, a continuación se listan y describen dichos archivos.

3.4.1) FRS_SIM.PAS. Este archivo contiene el programa principal y todas las rutinas del editor.

3.4.2) COMP_1.PAS. En esta unidad se encuentran las rutinas comunes a todo el programa así como las variables y tipos que se utilizan en todo el programa.

3.4.3) VECTORES.PAS. Aquí se realizaron operaciones básicas para manejar vectores, estas rutinas son utilizadas para calcular los centros de los círculos.

3.4.4) MENUS_UT.PAS. En esta unidad se encuentran todas las rutinas para los menús que son de tipo colgante.

3.4.5) CMDS.PAS. Esta unidad contiene todo lo referente al compilador.

3.4.6) ARCHIVOS .PAS. Esta unidad como su nombre lo dice contiene las rutinas que efectúan el manejo de archivos, así como todos los mensajes, ya sea de ayuda o de alarma.

3.4.7) HTAS.PAS. Esta unidad asigna los movimientos y trazos básicos para el simulador.

3.4.8) GRAFPRB.PAS. Aquí se realizaron las rutinas que interpolan y realizan los diferentes trazos, así como todo el ambiente gráfico del simulador.

3.4.9) CMDS_GRA.PAS Aquí se encuentra la información específica para la ejecución de cada comando en la simulación.

En los siguientes capítulos se detallarán las rutinas significativas de cada unidad y del programa principal.

Capítulo 4
Programa principal y Editor

En este capítulo se tratarán a detalle las rutinas que constituyen el programa principal y el editor, para esto se describirán las unidades COMP_1.PAS, MENUST_UT.PAS, ARCHIVOS.PAS y el programa principal FRS_SIM.PAS.

4.1 Programa principal.

Al entrar al programa, lo primero que se ejecuta es el procedimiento Inicializa, éste como su nombre lo indica, limpia todas las variables, asignándoles el valor determinado para evitar problemas durante la corrida, borra la pantalla y genera el ambiente texto inicial.

En este momento es conveniente aclarar que la variable medular para la información del programa-pieza, es un arreglo de cadenas lo cual es una matriz de caracteres con las siguientes dimensiones [40 , 250]. Este tipo definido en la unidad COMP_1, facilitó el manejo de la información, dado que de la fresadora maneja texto plano y la longitud máxima de líneas es constante, se consideró que una variable del tipo antes mencionado era la mejor opción para estructurar la información.

Además de las variables globales para todo el programa, la unidad COMP_1 tiene los siguientes procedimientos:

4.1.1) Suena. Esta rutina emite un sonido por determinado tiempo y frecuencia.

4.1.2) Pinta. Este procedimiento reconstruye los primeros seis renglones de la pantalla de edición, esta rutina es utilizada por los menús, debido a que cuando la opción colgante se repliega, hay que repintar lo que estaba atrás.

4.1.3) Pantalla. A través de un parámetro entero, pantalla genera el ambiente de edición, el renglón de indicadores o el renglón con las opciones del menú dependiendo de los requerimientos de la situación.

4.1.4) Lee. Esta es una función que lee un carácter y verifica que esté contenido en un conjunto predefinido, en caso de no estar el conjunto emite un sonido indicando error de lectura.

Estas cuatro rutinas componen básicamente la unidad COMP_1 y son comunes a todo el programa.

Regresando al programa principal, después de inicializar el programa entra a un ciclo, en donde básicamente se lee la opción elegida por el usuario y se ejecuta, el fin de ciclo se da cuando la opción de salida es seleccionada.

La lectura de las opciones es realizada por el procedimiento Menu_opc, el cual se encuentra en la unidad MENUS_UT.PAS.

Ahora es necesario describir la operación de las rutinas de la unidad antes citada.

4.1.1) Menus_opc. Este procedimiento es básicamente un ciclo que lee la tecla presionada, y dependiendo de la tecla realiza una operación. Las teclas habilitadas son: Enter, Esc , y las teclas de movimiento lateral. Con Enter se descuelga el menú en la opción actual o si el menú está descolgado se detecta la selección y se interrumpe el ciclo enviando como resultado

un número que indica la opción elegida. Las teclas de movimiento lateral realizan el cambio de las opciones principales.

Cuando los menús se repliegan hay que repintar la parte anterior de la pantalla, es aquí donde son utilizadas las rutinas de Pinta y pantalla.

4.1.2) Titula. Este procedimiento escribe en pantalla el renglón principal del menú y resalta o no la opción que se necesite. De esta manera cada vez que el usuario cambia la opción, se muestra el menú con la opción seleccionada.

4.1.3) Mensaje. Esta rutina contiene la información de todos los mensajes que se muestran en la ventana de ayuda en el editor. El texto de los mensajes están contenidos en arreglos de cadenas, para activar este procedimiento, basta llamarlo con el número del mensaje o error que se quiera desplegar. En el caso de los errores, se emite un sonido y no se permite continuar, sino hasta que se oprime ESC. Esto es con el objeto de llamar la atención del usuario.

Estas son las principales rutinas de la unidad MENUS_UT.PAS.

Dependiendo de la opción que se haya escogido, el procedimiento Menus_opc arroja un número entero indicando en el primer dígito la opción principal, y en el segundo la opción del menú colgante. Las opciones que existen son las siguientes:

4.1.1) Archivo.

4.1.2) Editor.

4.1.3) Compilador.

4.1.4) Simulador.

En lo que corresponde a la opción de archivo todas las rutinas excepto dos están contenidas en la unidad ARCHIVOS.PAS, mismas que a continuación serán explicadas.

Lo primero que se encuentra en esta unidad es el procedimiento Captura, éste tiene la función de leer una cadena de caracteres, en una determinada posición, con longitud específica y

dentro de un conjunto determinado. De esta manera es muy sencillo validar la información que se está capturando.

La rutina es básicamente un ciclo que lee una tecla y dependiendo de si es de edición o es caractér válido realiza la operación de edición o escribe el caracter, en caso de éste sea inválido, emite un sonido para indicar el error. Las operaciones de edición se realizan básicamente mediante manejo de caracteres dentro de una cadena, ya sea suprimir, mover cursor, borra todo el contenido, etcétera.

Esta rutina es utilizada por varios de los procedimientos de la unidad, ya que es necesario capturar el nombre del archivo a cargar, grabar, o el trayecto del directorio deseado.

Los procedimientos que se encuentran en la interfase de la unidad, es decir las rutinas utilizadas por el programa principal son:

4.1.1) Cargar. Este procedimiento abre el archivo que el usuario indique, el nombre del archivo es leído por la rutina de captura, es necesario aclarar que solamente se pueden abrir archivos con la extensión FRS, debido a que sólo pueden entrar ocho caracteres y para impedir al usuario abrir otro tipo de archivos, no se permite la entrada del caracter punto " . ", esta rutina está hecha con procedimientos internos de pascal como *FindFirst y Findnext* que buscan el primer archivo de un cierto directorio, y los subsiguientes, en caso de encontrarse el archivo deseado, es regresado por un parámetro del procedimiento dentro de una variable tipo Arreglo que es el tipo de matriz de caracteres que ya se mencionó.

4.1.2) Directorio. Esta rutina cambia el directorio actual por el directorio que se seleccione, antes de intentar realizar el cambio es necesario verificar que el directorio exista, y que la unidad de discos deseada esté disponible. Una vez verificadas estas condiciones se cambia el directorio con la instrucción *CHdir* y se despliegan en la ventana todos los archivos de interés, es decir con extensión FRS, del directorio

4.1.3) Salvar. Este procedimiento guarda en un archivo de nombre definido por el usuario, el programa-pieza en memoria, esto se logra con instrucciones de pascal. La rutina también verifica que el archivo no exista previamente, en caso de existir se da la opción al usuario de no realizar la operación.

4.1.4) Trans. Esta rutina realiza la transmisión de la información a la máquina herramienta, y será descrita posteriormente dado que es parte del compilador.

4.1.5) Dim. Este procedimiento realiza la lectura de las dimensiones de la pieza en bruto, así como del origen con respecto a la pieza. Consta básicamente de un ciclo que se apoya en la rutina de captura, y de esta manera habilita todas las teclas de edición para mayor comodidad del usuario.

4.1.6) Problemas. Esta rutina despliega un mensaje de error cuando los archivos gráficos no se encuentran en el directorio adecuado y enseguida pide al usuario entre el trayecto correcto con la localización de estos archivos.

4.1.7) Pro_imp. Este procedimiento muestra un mensaje de error cuando la impresora no está lista, y se ha activado la opción de impresión, esto con el objeto de que no se interrumpa la corrida por errores de operación.

Es necesario mencionar que toda interacción con el usuario en estas rutinas se realiza a través de la ventana de mensajes, y para llamar la atención del usuario se sombrea la ventana al iniciar cualquiera de las rutinas de interacción y se remueve la sombra al abandonar. Las rutinas que realizan el sombreo son Sombra y Sombra0.

4.2 El Editor

La parte central del editor es un procedimiento llamado Edita que detecta la opción que el usuario selecciona y activa las diferentes rutinas para ejecutar las rutinas pertinentes.

Como es de esperarse es un ciclo que lee, decide y ejecuta; al entrar al ciclo se lee la tecla presionada, existen dos posibilidades principales, una es que la tecla oprimida sea de edición y otra es que corresponda a un caractér válido para escribirlo.

Se analizarán primero las opciones de edición.

4.2.1) Movimiento a la derecha. Esta tecla provoca la ejecución de la rutina Movder. Esta rutina incrementa un entero la posición de X, en caso de que se encuentre en la última posición de cualquier campo, esto es el campo de instrucción, el campo de X, Y ,Z o F entonces incrementará en dos enteros. En caso de estar en la última posición del campo de instrucción, verifica que la instrucción exista, mediante la función Busca, que será descrita posteriormente, y en caso de existir realiza la restricción de los campos necesarios, debido a que algunas instrucciones no acepten valores en determinados campos.

El procedimiento que realiza la restricción será detallado más adelante, pero básicamente lo que hace es llenar el campo con asteriscos, de esta manera cuando existan asteriscos el campo está restringido y los movimientos hacia ese campo son evitados.

En el caso del movimiento a la derecha, cuando el siguiente campo contenga asteriscos, la posición de X se seguirá incrementando hasta encontrar un campo no restringido.

Por último existe la posibilidad de estar en la última posición del renglón en este caso se regresa la posición de X a 6, que es la posición inicial en pantalla, y se incrementa la posición Y en un entero.

Una vez ejecutada esta operación se regresa a leer la siguiente tecla, y se ejecuta la rutina Ilumina, este procedimiento reescribe el renglón actual y resalta el campo en el que se encuentre el cursor, invirtiendo el color del fondo y del texto, obviamente esta operación se realiza antes de la lectura de la tecla, cada vez que se inicia el ciclo.

4.2.2) Movimiento a la izquierda. Esta tecla activa el procedimiento Movizq, el cual funciona con la misma lógica que el de movimiento a la derecha, con la diferencia de restar en lugar de incrementar la posición en X.

4.2.3) Movimiento hacia arriba. La función de esta tecla es realizada por el procedimiento Movar, el cual decrementa la posición en Y, pero antes de hacerlo es necesario verificar que el campo del renglón superior esté disponible, o en otras palabras no tenga asteriscos o que exista, es decir si se encuentra en el renglón inicial del arreglo no hay renglón anterior. En caso de no presentarse dichos factores se realiza el decremento de la posición en Y. Existe otro ligero detalle, y éste es que a pesar de que el campo esté disponible, es posible que el usuario esté posicionado al inicio de la pantalla, en este caso hay que repintar la pantalla desde el renglón anterior al actual, esto lo hace la rutina *Scroll* la cual escribe 20 renglones a partir de la posición del arreglo solicitada.

Hay algunos detalles que falta mencionar, existen dos contadores para la posición Y, uno de ellos indica la posición dentro del arreglo, es decir va desde 1 hasta el número de líneas totales alimentadas, y otro contiene la posición en pantalla; éste sólo puede tener valores desde 1 hasta 20. El otro detalle es que las rutinas que escriben en pantalla, ya sea Ilumina o *Scroll* tienen el condicionante de que cuando el arreglo contenga un asterisco, se debe escribir espacio, de esta manera al limitar los campos el usuario no percibe los asteriscos.

4.2.4) Movimiento hacia abajo. Con esta tecla se pone en marcha el procedimiento Movab, el cual tiene la misma estructura que el de movimiento hacia arriba sólo que a diferencia del anterior incrementa la posición en Y. Ahora bien, en cuanto a los detalles o condiciones que debe verificar son que exista línea subsiguiente, debido a que la única manera de incrementar el número de líneas es presionando Enter, y la activación de la rutina *Scroll* se realizará cuando el cursor se encuentre en el último renglón de la pantalla.

4.2.5) Enter. Esta tecla activa el procedimiento Completa. Esta rutina ejecuta varias operaciones importantes, cuando el usuario se encuentra dentro de cualquier campo y oprime esta tecla significa la finalización de la entrada de la cifra, en este caso la rutina verifica si la cifra abarca todos los espacios del campo, de ser así la deja sin cambio alguno y salta al siguiente campo disponible, si la cifra no abarca los espacios la recorre, justificándola hacia la derecha los espacios necesarios, para que las unidades queden en la última posición, las decenas en la penúltima y así sucesivamente.

Otra función es la de avance rápido entre campos, para ejecutar esta operación es necesario que no se haya realizado ninguna corrección a la cifra, y que el cursor se encuentre en la primera posición del campo, de ser así el contador de la posición en X será incrementado el número necesario de unidades para saltar al siguiente campo disponible.

Como se mencionó antes, una de las formas de incrementar el número de líneas es oprimiendo la tecla Enter y estando en la posición final del último renglón, al cumplirse estas condiciones y por supuesto que no se encuentre en la última línea posible (221) entonces el contador que registra el número total de líneas se incrementará.

Finalmente es necesario aclarar que toda la serie de condicionales que se encuentran en estas rutinas se debe a la variación de las longitudes de los diferentes campos, es decir el campo de las instrucciones G/M tiene una longitud de tres caracteres, mientras que el campo X y Z tienen longitudes de 6 cifras que son diferentes de los campos Y y F.

4.2.6) Inicio. Esta tecla solamente mueve el cursor a la posición 6 mediante la asignación de dicho valor al contador de la posición X.

4.2.7) Fin . Al oprimir esta tecla el contador X se modifica, para asignarle el valor 33, que es la posición final del renglón, solamente que si el campo no está disponible, se activa la rutina de Movizq, para que busque el campo disponible previo.

4.2.8) Suprime. Esta opción activa el procedimiento Borra, la tarea de éste es suprimir el caracter en donde se encuentre el cursor, y concatenar los restantes caracteres de la cifra.

4.2.9) Bck Spc. Esta tecla llama al procedimiento Bor_at, que tiene dos funciones, la primera es suprimir el caracter anterior a la posición del cursor y concatenar el resto de la cifra, la segunda función es el retroceso rápido, esta opción se activa cuando se oprime la tecla estando en la primera posición del campo y no se ha editado, lo que hace es saltar a la primera posición del campo anterior disponible.

4.2.10) F9 . Esta opción activa el compilador llamando al procedimiento Checa_al, que será descrito posteriormente.

4.2.11) F5 . Con esta tecla se insertan líneas, la rutina que se encarga de esto se llama Ins_lin, la estructura es la siguiente: primero se verifica que exista espacio, de ser así se copia la línea final del arreglo en una variable auxiliar, se borra del arreglo principal y se copia en la siguiente esto se hace de forma regresiva hasta llegar a la línea donde se encuentra el cursor.

4.2.12) F4. Es similar a la anterior con la diferencia de que suprime la línea, y para esto es necesario comenzar borrando la línea actual y escribiendo la línea siguiente en la actual, y así sucesivamente hasta llegar a la línea final del arreglo.

4.2.13) Pg up . Esta tecla llama al procedimiento Pag_arr, el cual tiene la función de subir a la pantalla anterior, para esto es necesario revisar si existe espacio suficiente para realizar el cambio, es decir se necesitan recorrer 20 líneas hacia arriba en el arreglo sin alterar la posición en pantalla, esto sólo se puede hacer si el cursor se encuentra en la línea 21 o más abajo, en caso de no ser así solamente se recorre la posición de la pantalla al inicio de ésta. Es necesario utilizar la rutina de *Scroll* que fue descrita anteriormente.

4.2.14) Pg dn. Es similar a la tecla anterior sólo que el movimiento es hacia abajo en el arreglo, la estructura es la misma.

Ahora bien en caso de que la tecla corresponda a un caracter válido, se activa el procedimiento Escribe que a continuación se detallará.

Procedimiento escribe. Lo primero es señalar que sólo acepta los caracteres numéricos del 0 al 9, el signo " - " y " + " así como la letra M.

La letra M sólo se acepta si se encuentra dentro del campo de instrucciones que abarca de la posición 6 a la 9 en X, también se aceptan números.

Al terminar la entrada de una instrucción es necesario verificar que exista y en caso de que se requiera es indispensable restringir los campos que no acepte la instrucción que se alimentó.

En caso de teclear un caracter no válido activará una bandera, que a su vez termina la rutina y emite la señal de alarma correspondiente.

Existe otra bandera que indica si se realizó una edición previa en el campo con el objeto de que si no se ha editado, y el caracter es válido, se borra toda la cifra del campo, y si se ha editado sólo se sobrepone el caracter tecleado.

Los signos " + " y " - " son aceptados en los campos X, Y y Z siempre y cuando la instrucción lo permita. El signo positivo no se escribe, lo único que hace es borrar el negativo en caso de que exista.

Cuando uno teclea un caracter válido en la última posición de un campo, se realiza un salto al siguiente campo disponible.

Esta es la estructura general del Editor, de cualquier manera los listados que contienen el desarrollo de las rutinas antes descritas se incluyen en el anexo 1.

Capítulo 5
El Compilador

Dentro de lo que es el compilador hay dos importantes herramientas para el usuario, lo que es propiamente la compilación y la transmisión.

En el capítulo tercero se dio una idea general de lo que hacen estas opciones, en este capítulo se verá cómo se realizaron estas funciones.

5.1 Compilación.

Esta operación está realizada básicamente por dos procedimientos y una función. El primero, Limita tiene la tarea de restringir los campos no disponibles para cada instrucción, a continuación Checa revisa el programa pieza en busca de alguno de los errores factibles y la función Busca se encarga de verificar que las instrucciones existan. A continuación se revisarán cada una de las anteriores.

5.1.1) Función Busca. Esta rutina es de tipo entera, lo que hace es recorrer un arreglo que contiene todas la instrucciones disponibles, en caso de encontrar la instrucción, la función

adquiere el valor de la posición de la instrucción en el arreglo, en caso de no encontrar la instrucción, la función regresa el valor cero, que indicará la necesidad de activar la alarma 00 que significa instrucción incorrecta.

Enseguida se muestra una tabla con las instrucciones disponibles y su función.

Instrucción	Función
G00	Interpolación lineal (marcha rápida)
G01	Interpolación lineal
G02	Interpolación circular (Antihoraria)
G03	Interpolación circular (Horaria)
G04	Tiempo de tratamiento
G21	Línea vacía
G25	Llamada a subprograma
G27	Ir a
G40	Supresión de la compensación del radio del útil
G45	Sumar el radio del útil
G46	Restar el radio del útil
G47	Sumar dos veces el radio del útil
G48	Restar dos veces el radio del útil
G64 *	Motores de avance sin corriente
G72	Fresado de cajas
G73	Taladrado profundo
G81	Ciclo de taladrado
G82	Taladrado con tiempo de tratamiento
G83	Extracción de virutas

G85	Ciclo de escariado
G89	Ciclo de escariado con tiempo de tratamiento
G90	Indicaciones absolutas de cotas
G91	Indicaciones incrementales de cotas
G92	Desplazamiento del punto de referencia
M00	Parada intermedia
M03	Encendido de husillo (marcha derecha)
M05	Parada de husillo
M06	Entrada radio de la fresa
M17	Retorno al programa principal
M30	Fin de programa
M98 *	Corrección de holguras
M99	Parámetros de círculos graduados
Sin instrucción	Ninguna

Tabla 5.1 Instrucciones disponibles.

* Estas instrucciones son aceptadas por el compilador, pero no tienen efecto en la simulación, sólo en el proceso real.

5.1.2) Procedimiento Limita. Esta rutina utiliza el valor de la función Busca, y accesa al arreglo que contiene las instrucciones y la información de los campos disponibles o restringidos, en caso de que un campo deba ser restringido la rutina llena de asteriscos las posiciones que deban ser limitadas. Otra operación que realiza es formatear algunos campos disponibles, por ejemplo en la instrucción G27 "Ir a", en el campo de la velocidad de avance se accesa la línea deseada y el campo lleva en la primera posición el caracter L, esta rutina coloca

automáticamente ese caracter; de la misma manera da formato a cualquier campo que así lo requiera.

5.1.3) Procedimiento Checa. Este representa la parte central de la compilación, ya que es el que revisa todo el programa-pieza y detecta los errores que se traducen en mensajes de alarma para el usuario.

A continuación se anexa una tabla con las posibles alarmas que son desplegadas en caso de error.

No. Alarma	Error
A 00	Instrucción G/M incorrecta
A 01	Radio / M99 incorrecto
A 02	Valor X incorrecto
A 03	Valor F incorrecto
A 04	Valor Z incorrecto
A 05	Falta instrucción M30
A 06	Falta instrucción M03
A 15	Valor Y incorrecto
A 16	Falta instrucción M06
A 17	Subprograma incorrecto
A 18	Recorrido de compensación de fresa menor a cero
A 19	Instrucción G92 duplicada
A 20	Falta instrucción M05.

Tabla 5.2 Alarmas.

Ahora es conveniente revisar la estructura de la rutina Checa.

Este procedimiento es un ciclo con la instrucción *Repeat* , y se ejecuta hasta que se cumpla alguna de las siguientes condiciones: haber revisado el total de las líneas del arreglo, encontrar la instrucción M30 " fin de programa " o detectar alguna alarma. En caso de detectar alguna alarma el procedimiento regresa el número de línea en el que se generó el error, de esta manera el editor puede ir exactamente a esta línea y ayudar al usuario indicándole dónde está el posible error.

Al iniciar la compilación se inicializan todos las variables del procedimiento, las variables principales son : valores actuales de posición de la herramienta, valores anteriores, número de línea y algunas banderas lógicas.

Al iniciar el ciclo se revisa que el comando o instrucción no sea M98 ni comando vacío, ya que éstos no tienen ningún efecto; la revisión se realiza línea por línea.

En caso de tener comando G90 o G91 se activa o no la bandera lógica de nombre Absol, que en caso de ser verdadera indica programación absoluta y en caso contrario_indica programación incremental.

Lo primero que se hace es validar los valores numéricos de los campos para verificar que se encuentren dentro de los rangos especificados por la fresadora. En caso de que los valores de los campos sean de posicionamiento se modifican la posición actual y la anterior.

Al verificar que los valores numéricos estén dentro de los rangos permisibles se prueban las alarmas de valor incorrecto de X, Y, Z y F. En cuanto a Z existe otra posibilidad, es decir el valor no sólo debe estar dentro del rango, además se debe observar que la trayectoria descrita no sea tridimensional, para esto se compara la posición actual con la anterior y en caso de ser una trayectoria que describa movimiento en más de un plano, se emitirá la alarma 4 que es valor incorrecto de Z.

Cuando las instrucciones contienen valores en los campos de posición, pero no modifican la posición, o el ciclo que realizan regresa a la posición inicial, las variables que se modificaron son reasignadas con el valor de la posición anterior.

En caso de que el comando sea G02 o G03 es necesario verificar la línea siguiente, ya que de contener el comando M99 indicará que se trata de un círculo graduado y en caso de ser cualquier otra instrucción indicará que es un cuarto de círculo.

Cuando se trata de un cuarto de círculo la revisión es prácticamente automática, lo único que se debe revisar es que el incremento en X sea igual al incremento en Y, dado que en un cuarto de círculo se tiene lo siguiente.

Figura 5.1 En este diagrama se observa claramente la condición antes mencionada.

Cuando se trata de un círculo graduado se tiene una situación no tan simple, ya que es necesario verificar que las coordenadas del centro, que se programan mediante M99, sean correctas y que el arco esté trazado en un solo cuadrante.

Es aquí donde fue necesario emplear un poco de geometría para programar la revisión de las condiciones anteriores, la siguiente figura muestra el diagrama vectorial que describe la situación que se tiene en círculos graduados.

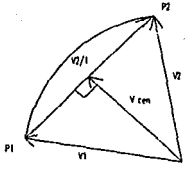


Figura 5.2 Arcos Menores
a 90 grados.

Se traza un vector entre los dos puntos P2 y P1 (V2/1), a continuación se traza el vector Vcen, para el trazo de este vector se conoce la dirección, debido a que forma un ángulo de 90 grados con el vector V2/1, su sentido estará dado a partir de el sentido del arco, esto es, si el sentido es horario se restarán los 90 grados a partir del vector V2/1 y se sumarán en caso contrario, la magnitud del vector se conoce indirectamente a partir de las magnitudes de V1 y de la mitad de V2/1, ya que los tres forman un triángulo rectángulo. Una vez obtenido Vcen se comparan cada una de sus componentes con las respectivas componentes asignadas con M99, de ser diferentes implicará que las coordenadas del centro son incorrectas, y por supuesto la activación de la alarma.

La otra condición resulta de comparar las componentes de los vectores V1 y V2 debido a que ambos vectores deben estar en el mismo cuadrante.

De esta manera se verifican los círculos graduados.

La siguiente alarma que se verifica, es la número 16 falta entrada del radio de la fresa, está alarma se activa cuando se utiliza cualquier instrucción que implique inicialización de la herramienta en pantalla sin haber entrado la instrucción M06, es decir que signifique dibujar la herramienta, debido a que para efectuar esto se debe contar con el diámetro. En realidad la fresadora sólo exige esta entrada (M06) cuando se utilizan instrucciones como G72, G45, G46,

G47 o G48, pero por cuestiones didácticas se trata de forzar al usuario a que siempre declare el diámetro de su herramienta.

Con respecto a la alarma número 18 recorrido de compensación de fresa menor a cero, se activa mediante dos condiciones. Si se ha programado G46 o G48 y la trayectoria a recorrer es menor a cero o si en el fresado de cajas la cota final no excede en al menos un 10 % a la cota inicial en la dirección X.

Finalmente es necesario describir la estructura del compilador que verifica los subprogramas.

Inicialmente al ser detectado un comando G25, se activa un contador auxiliar que hace saltar a la línea que el campo F indique, se verifica que la línea exista y posteriormente se busca el siguiente comando M17, en caso de no existir se marca error, en caso contrario, se escribe en el mismo arreglo principal, la línea a la que debe regresar, y así sucesivamente hasta finalizar los subprogramas, después el contador principal sigue el camino indicado por el programa y revisa el o los subprogramas entrados.

5.2 Transmisión

La rutina que efectúa la transmisión de datos a través de puerto serial, se encuentra en la unidad ARCHIVOS.PAS que está en el anexo 1.

Su funcionamiento es el siguiente:

Se inicializa el puerto, la clave de inicialización es propia de la fresadora, es decir cada periférico tiene su propio código, esto se realiza mediante el procedimiento InitCom, que está compuesto básicamente por la una interrupción ($14), en caso de detectar algún error se detiene el proceso.

Posteriormente se inicia la transmisión mediante los procedimiento WritecarCom y WriteStrcom, el primero transmite el caractér deseado a través del puerto y el segundo apoyado en el anterior transmite una linea completa.

Es necesario transmitir con un cierto formato que requiere la máquina, el formato es el siguiente:

Primera línea. "%, *Carriage return, line feed* ".

Segunda linea. " N'G' X ' Y ' Z ' F Cr,Lf"

Las siguientes líneas son el programa-pieza sin espacios entre campos y sin asteriscos, por lo que el compilador elimina los asteriscos sustituyéndolos por espacios, y los espacios entre campos no los envía.

La transmisión de las línea del programa se realiza, mediante una variable tipo cadena auxiliar, que se forma concatenando los caracteres del programa-pieza realizando los ajustes antes descritos y agregando siempre los caracteres *Carriage return y Line Feed* al final de cada línea.

La línea final es " MI" que indica milimetros como unidades y fresado vertical.

De esta manera se realiza la transmisión de programas-pieza a la fresadora; faltó mencionar que los programas son compilados antes de transmitirlos, y en caso de presentar errores no se realiza la transmisión.

Las rutinas que componen el compilador se encuentran en la unidad CMDS.PAS que se incluye en el anexo 2.

Capítulo 6
El Simulador

En este capítulo se describirán las rutinas que componen el ambiente gráfico, es decir la simulación. Para esto se detallarán los principales procedimientos y funciones de las unidades HTAS.PAS, GRAFPRB.PAS y CMDS_GRA.PAS los listados de dichas unidades se encuentran al final del trabajo en el anexo 3.

6.1 Trazos de la fresa.

La fresa está representada por un círculo en la vista superior y por rectángulos en las vistas lateral y frontal. En ambos casos se utilizaron objetos para generar las figuras en pantalla, aprovechando las diferentes características de este tipo de variables. Los objetos presentan tres características principales, herencia, jerarquía y polimorfismo. Estas características permitieron darle ciertas propiedades a las figuras, como la posición por ejemplo, que es una variable sumamente importante cuando se quiere simular una herramienta controlada numéricamente.

6.1.1 El Círculo.

El círculo es un objeto que hereda propiedades de locación y de punto, las propiedades que hereda son la posición en X y en Y, mediante las funciones *GetX y GetY*, se conocen sus coordenadas exactas, ya que son atributos del objeto.

Se realizaron cuatro procedimientos que se encargan de manejar la figura, a continuación se describen:

6.1.1.1) Init. Esta rutina inicializa al círculo, asignándole coordenadas en el plano y un radio determinado.

6.1.1.2) Show. Este procedimiento dibuja el círculo en las coordenadas asignadas, y con el radio requerido.

6.1.1.3) Hide. Esta rutina borra el círculo del radio asignado, sobreescribiendo otro del color con el que se esté trazando, posteriormente dibuja círculos concéntricos desde el radio asignado hasta un radio cero, todos estos círculos se dibujan del color del trazo, para generar el trazo de la herramienta en la vista superior.

6.1.1.4) Mov_cir. Este procedimiento utiliza los dos anteriores, debido a que para mover el círculo es necesario borrar el actual, pintar el que se quiere en la nueva posición y reasignar coordenadas. La lógica es la anterior, siempre y cuando el trazo sea de corte, en caso de ser de marcha rápida, este procedimiento borrará el círculo y trazará una línea del punto actual al punto requerido, y cuando sea necesario borrará el círculo y no trazará, simplemente reasignará coordenadas. Este último caso se requiere cuando la herramienta está fuera de la pieza.

6.1.2 Los rectángulos.

En cuanto a los rectángulos, ambos funcionan de la misma manera, la única diferencia es que las coordenadas se invierten, es decir los movimientos en X para la vista lateral son los mismos en Y para la vista frontal, de igual forma en el otro caso.

De forma similar al círculo existen cuatro rutinas que manejan los rectángulos, la rutina *Init y Show* trabajan de igual manera que las rutinas del círculo; sin embargo hay que mencionar que aunque las coordenadas que se usan son X e Y, en realidad simbolizan X y Z o Y y Z dependiendo de la vista. Las rutinas que varían con respecto a las del círculo y que resultan interesantes en este caso son:

6.1.2.1) Oculta. Esta rutina borra los diferentes lados del rectángulo dependiendo de los requerimientos, puede borrar todos o cada uno de los lados según se necesite. Esto es necesario debido a que las vistas lateral y frontal dejan el trazo al paso de la herramienta.

6.1.2.2) Mov_rec. Este procedimiento tiene dos formas principales de operar, puede dejar el trazo según se le ordene o puede simplemente llevar la herramienta a través de la pieza sin trazar.

La segunda opción es muy sencilla, únicamente es necesario borrar, reasignar coordenadas y pintar en las coordenadas requeridas.

En la primera opción la situación es bastante diferente, ya que es necesario utilizar ciertas condiciones que son enviadas de las rutinas de interpolación, para definir qué líneas hay que borrar y qué líneas deben permanecer para obtener el trazo adecuado.

Es pertinente analizar dichas condiciones:

6.1.2.1) Si existe un trazo diagonal es decir en X-Z o en Y-Z se borran todas las líneas y se pinta una línea a partir del vértice de ataque, excepto cuando se realiza el cambio de dirección en Z, cuando esto sucede es necesario que permanezca la línea de la base. La bandera que muestra que esta condición se dio es activada desde la interpolación de líneas rectas, el nombre de la variable es Sub_di(l o f) dependiendo de la vista.

6.1.2.2) Cuando hay movimiento a la derecha o a la izquierda, sólo permanece la línea de base, pero en caso de haber cambio de sentido, permanecerá la línea lateral contraria al sentido del movimiento.

6.1.2.3) Cuando se hace un barreno por ejemplo se activa la bandera *Down* que indica a la rutina que deberá dejar la línea de base al cambio de dirección en Z.

6.1.2.4) La última condición es la que permanezca cualquiera de las líneas laterales cuando se inicia el ascenso de la herramienta después de un trazo horizontal o cuando se inicia el trazo lateral depués de un descenso, la línea lateral que se borre dependerá del sentido del movimiento anterior, es decir, en caso de que el movimiento anterior haya sido a la derecha al subir permanecerá la línea derecha y se borrará la izquierda.

Como se mencionó anteriormente estas condiciones fueron explicadas basándose en la vista lateral, pero la vista frontal es similar, la diferencia es el cambio en las direcciones X y Y.

6.2 Interpolaciones

Las interpolaciones lineal y circular son los procedimientos relevantes de la unidad GRAFPRB.PAS, las demás rutinas inicializan el modo gráfico, generan al ambiente y realizan cuestiones hasta cierto punto simples.

6.2.1 Interpolación Lineal.

Esta rutina realiza los trazos necesarios para generar cualquier trazo lineal de la simulación, necesita los siguientes parámetros:

6.2.1.1) Coordenadas del punto final del desplazamiento (incrementales).

6.2.1.2) Color del trazo.

6.2.1.3) Diámetro de la herramienta(Medida real).

6.2.1.4) Tipo del trazo (Corte, avance rápido o ninguno).

A continuación se enumeran y describen las operaciones de la rutina:

6.2.1.1) Inicialización de las variables.

6.2.1.2) Inicialización de la herramienta en las tres vistas.

6.2.1.3) Detección del plano de movimiento, es decir, se analiza que coordenadas hay que modificar. Posteriormente se asigna la variable independiente y la dependiente para entrar a la ecuación; en esta parte también se calcula la pendiente de la recta.

6.2.1.4) Cálculo del número de ciclos en avance rápido. Esto significa conocer el número de pixeles que abarca el trayecto, debido a el número de unidades de la pieza por pixel varía de acuerdo al escalamiento de la pieza. Una vez conocida la cantidad de ciclos, se puede iniciar el ciclo con incremento de la variable independiente en pixeles, una vez que se ha alcanzado la posición final en pantalla, si es necesario se siguen incrementando las coordenadas aunque no se refleje en la pantalla.

6.2.1.5) Inicio de ciclo.

6.2.1.6) Revisión de incremento en pixeles. En esta parte se revisa si existió un incremento suficiente de las coordenadas reales, para reflejarlo a las coordenadas de pantalla, de ser así se incrementan la posición actual, la anterior y la penúltima.

Para entender mejor esta parte se describe el siguiente ejemplo: supóngase que cada pixel equivale a 10 unidades de la pieza, el trazo a realizar requiere un desplazamiento de un mil ciento tres (1103)unidades, lo cual significa desplazarse ciento diez pixeles con tres décimas de pixel (110.3) por supuesto las tres décimas son imposibles de avanzar, ya que la resolución del monitor es entera, por lo tanto se realizará un desplazamiento de 110 pixeles en incremento rápido y al llegar a esta cantidad, el contador de la variable independiente se incrementará no en 10 si no en 1, cuando esto suceda el contador modificará las coordenadas de escala real, pero no habrá incremento de las coordenadas en pantalla, es por ello que se efectúa la revisión del incremento. Otro factor es que el incremento de la variable independiente y de la dependiente en coordenadas pantalla es separado, dado que en ocasiones dependiendo de la pendiente de la recta se puede incrementar una mientras la otra no y viceversa.

6.2.1.7) Incremento de la variable independiente, ya sea en unidades equivalentes a un pixel o a unidades pieza.

6.2.1.8) Cálculo de la variable dependiente, mediante la ecuación de la recta.

6.2.1.9) Escalamiento de las coordenadas. cambio de escala real a escala pantalla.

6.2.1.10) Activación de las diferentes banderas de acuerdo con el cambio de la posición en pantalla. Estas banderas son las que indican a las rutinas de movimiento de la herramienta como deben trazar. Para ello se consideran las últimas tres posiciones en pantalla.

6.2.1.11) Movimiento de la herramienta en las diferentes vistas.

6.2.1.12) Cambio del indicador de coordenadas en pantalla. Cambio de las mirillas.

6.2.1.13) Ciclo de lectura de teclas en caso de interrupción de ciclo. Las diferentes teclas, F1 a F4, que limpian las vistas o desactivan el trazo.

6.2.1.14) Fin de ciclo.

6.2.2 Interpolación Circular.

Los parámetros que necesita este procedimiento son los mismos que los de la interpolación lineal más las coordenadas del centro.

La estructura lógica es exactamente la misma con algunas variantes.

6.2.2.1) Inicialización de las variables.

6.2.2.2) Inicialización de la herramienta en las tres vistas.

Aquí no es necesario detectar el plano de movimiento, ya que siempre será en el plano X-Y.

En esta interpolación no hay incremento en unidades de pixel, dado que el cambio no es lineal y es necesario incrementar en unidades pieza.

6.2.2.3) Inicio de ciclo.

6.2.2.4) Revisión de incremento en pixeles. En esta parte se revisa si existió un incremento suficiente de las coordenadas reales, para reflejarlo a las coordenadas de pantalla, de ser así se incrementan la posición actual, la anterior y la penúltima.

6.2.2.5) Incremento de la variable independiente.

6.2.2.6) Cálculo de la variable dependiente, mediante la ecuación del círculo.

6.2.2.7) Escalamiento de las coordenadas, cambio de escala real a escala pantalla.

6.2.2.8) Activación de las diferentes banderas de acuerdo con el cambio de la posición en pantalla. Estas banderas son las que indican a las rutinas de movimiento de la herramienta como deben trazar. Para ello se consideran las últimas tres posiciones en pantalla.

6.2.2.9) Movimiento de la herramienta en las diferentes vistas.

6.2.2.10) Cambio del indicador de coordenadas en pantalla. Cambio de las mirillas.

6.2.2.11) Ciclo de lectura de teclas en caso de interrupción de ciclo. Las diferentes teclas, F1 a F4, que limpian las vistas o desactivan el trazo.

6.2.2.12) Fin de ciclo.

Las interpolaciones son utilizadas por la rutina CMDS_GRA.PAS, que es la rutina que contiene programados los diferentes comandos que se enlistaron en el capítulo anterior.

El procedimiento central de esta unidad se llama Simula, tiene una estructura similar a la rutina Checa del compilador. El parámetro que maneja es la variable de tipo arreglo, que le envía el programa-pieza.

Posteriormente entra a un ciclo donde lee, línea a línea y ejecuta la simulación correspondiente a cada comando.

Existe una rutina que es la que asigna los diferentes colores a las diferentes alturas, este procedimiento es activado cada vez que hay un desplazamiento en Z, lo que hace es verificar si la altura o plano, ya ha sido manejado o no, en caso de encontrar la altura, regresa el color que tiene asignado, si no encuentra la altura, la registra y le asigna un nuevo color.

En cuanto a los comandos de movimiento o de corte. Lo que se hace es identificar el comando, y llamar la interpolación necesaria con los valores en incremental, para realizar el trazo en pantalla. Como los valores que aceptan las rutinas de interpolación son sólo incrementales, en caso de que el programa-pieza este en coordenadas absolutas se calcula el incremento con respecto al punto anterior.

Los comandos G02 y G03, son los únicos que requieren de la interpolación circular, estos comandos llaman a la rutina pero antes calculan las coordenadas del centro del arco que se tiene que trazar. Para esto se utiliza la misma estructura que en el compilador, la cual fue descrita en el capítulo anterior.

Los comandos como G00 y G01 sólo asignan el color del trazo, y llaman a la interpolación lineal con los parámetros necesarios.

Existen comandos como G72 "fresado de cajas" que son una combinación de G00 y G01 o en otras palabras un conjunto de interpolaciones lineales. En caso de identificar estos comandos existen subrutinas que contienen las secuencias y que ejecutarán la simulación, apoyándose en las rutinas de la unidad GRAFPRB.PAS.

Capítulo 7
Manual del usuario

7.1 Entrada al programa y recomendaciones iniciales.

Es conveniente revisar que los *drivers* de gráficos estén contenidos en el mismo directorio que el archivo ejecutable, y que se corra el programa desde este mismo directorio.

También se sugiere utilizar un monitor a colores, dadas las características del simulador.

Los comandos aceptados por el programa son los mismos que maneja la fresadora F-1 CNC de EMCO, posteriormente se da una lista de estos comandos y su función, para mayores detalles acerca de la sintaxis y formato se recomienda consultar el manual de bases de la fresadora.

Para realizar la transmisión de datos mediante la interfase RS - 232 es necesario utilizar el puerto serial de la máquina y conectarlo al tablero de control de la fresadora con una clavija especial que el fabricante provee.

El nombre del archivo ejecutable es FRS_SIM, tecleándolo se inicia la corrida del programa.

Lo primero que aparece al entrar al programa es un mensaje de presentación que se despliega en la ventana de mensajes, para continuar es necesario teclear ESC.

A continuación se inicia la rutina de captura de dimensiones y entrada del origen con respecto a la pieza bruta.

Los datos que se piden son los siguientes:

7.1.1) Longitud.

7.1.2) Ancho.

7.1.3) Altura.

7.1.4) Xo.

7.1.5) Yo.

7.1.6) Zo.

Los tres primeros son las dimensiones de la pieza bruta, y los tres últimos representan las coordenadas del origen con respecto al origen predeterminado. (ver diagrama siguiente)

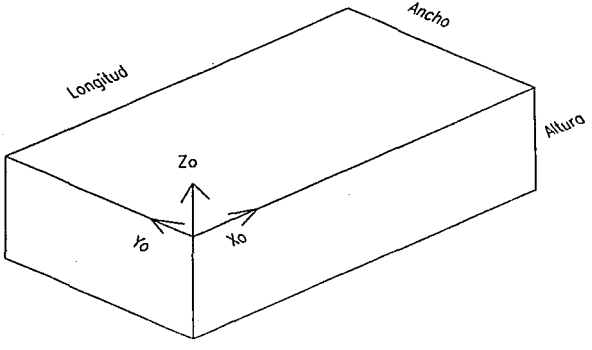


Figura 7.1 Pieza en bruto.

Para realizar la entrada de estos datos se teclea en la posición que indica el cursor, para finalizar la entrada de un dato se oprime ENTER, Todas las teclas de edición están habilitadas, de la manera tradicional. Para finalizar la captura de los datos se utiliza F1, o se finaliza automáticamente al entrar el último dato (Zo).

En caso de que algún valor esté fuera de rango, se desplegará un mensaje de error con el texto " valor incorrecto " para continuar se debe presionar ESC, y rectificar el valor en él que se posicione el cursor.

Cada vez que se despliega un mensaje de error, es necesario oprimir ESC para continuar.

Una vez entrados los datos iniciales, el programa entra al editor, y despliega en la ventana de mensajes los usos de las teclas de función.

7.1.1) F10 Menú.

7.1.2) F9 Compilar.

7.1.3) F5 Inserta Línea.

7.1.4) F4 Borra Línea.

En este momento se puede realizar la edición de un nuevo programa-pieza o ir al menú oprimiendo F10 para traer a memoria un programa-pieza previamente realizado y almacenado.

7.2. El Menú y sus opciones.

Las opciones principales del menú son las siguientes:

7.2.1) Archivo.

7.2.2) Editor.

7.2.3) Compilador.

7.2.4) Simulador.

Para seleccionar cualquiera de estas opciones basta moverse con las teclas de movimiento lateral hasta que la opción deseada sea resaltada y oprimir ENTER.

En este momento se despliega el menú colgante que muestra las opciones posibles dentro de esa selección. Para ir a cualquiera de ellas se utilizan las teclas de movimiento vertical y se oprime ENTER cuando la opción deseada esté resaltada. Para replegar el menú se oprime ESC, esto dejará al usuario dentro del menú principal, si se quiere salir y regresar al editor es necesario oprimir ESC nuevamente.

7.2.1 Las opciones que presenta archivo son las siguientes.

7.2.1.1) Cargar. Esta opción trae a memoria y da la posibilidad de editar un programa-pieza existente en el directorio actual. Para cargar el programa se oprime ENTER, en ese momento se pregunta al usuario si desea borrar el programa-pieza que esté en pantalla, para proseguir es necesario oprimir "s", para salir y no borrar oprima cualquier otra tecla.

El directorio y archivo actuales se muestran en la ventana de mensajes, cuando se está editando.

7.2.1.2) Nuevo. Esta opción borra la memoria e inicia un nuevo programa, al seleccionar esta opción, el programa vuelve a la rutina de captura de dimensiones y coordenadas del origen. Antes de borrar la memoria el programa pregunta si se está seguro, para continuar oprima "s", para interrumpir presione cualquier otra tecla.

7.2.1.3) Salvar. Al seleccionar esta operación se limpia la ventana de mensajes, y se pide la entrada del nombre del archivo para grabar el programa-pieza. No es posible entrar extensión al nombre del archivo. En caso de que el archivo exista se pregunta si se desea sobre escribir, para continuar oprima "s", el archivo se salva en el directorio actual.

7.2.1.4) Directorio. Con esta opción se cambia el directorio, se puede ir a cualquier unidad o ruta disponible, en caso de no existir se indicará en la ventana de mensajes, después de accesar el nuevo trayecto y oprimir ENTER, se despliega en la ventana de mensajes la lista de los archivos con extensión FRS, que es la extensión que asigna el simulador.

7.2.1.5) Imprime. Al presionar ENTER se imprime el archivo en memoria, en caso que se presente cualquier problema con la impresora, el papel o el puerto, se desplegará un mensaje de error, para imprimir es necesario corregir el problema y volver a teclear la opción.

7.2.1.6) Salir. Esta opción regresa al sistema operativo y borra el programa en memoria.

7.2.2 Las opciones que presenta Editor son las siguientes:

7.2.2.1) Ins Lin. Inserta línea, para efectuar esta operación se oprime ENTER, o estando en el editor se teclea F5, la línea se insertará en la posición donde se encuentre el cursor.

7.2.2.2) Borra Lin. Esto se logra ya sea desde el menú oprimiendo ENTER o desde el editor tecleando F4.

7.2.2.3) Dimensiones. Al seleccionar esta opción, el programa abre la edición de las dimensiones de la pieza bruta y de las coordenadas del origen.

7.2.2.4) Edita. Con esta opción el programa entra al editor. Las instrucciones disponibles en el editor son las siguientes:

Instrucción	Función
G00	Interpolación lineal (marcha rápida)
G01	Interpolación lineal
G02	Interpolación circular (Antihoraria)
Instrucción	Función
G03	Interpolación circular (Horaria)

G04	Tiempo de tratamiento
G21	Línea vacía
G25	Llamada a subprograma
G27	Ir a
G40	Supresión de la compensación del radio del útil
G45	Sumar el radio del útil
G46	Restar el radio del útil
G47	Sumar dos veces el radio del útil
G48	Restar dos veces el radio del útil
G64 *	Motores de avance sin corriente
G72	Fresado de cajas
G73	Taladrado profundo
G81	Ciclo de taladrado
G82	Taladrado con tiempo de tratamiento
G83	Extracción de virutas
G85	Ciclo de escariado
G89	Ciclo de escariado con tiempo de tratamiento
G90	Indicaciones absolutas de cotas
G91	Indicaciones incrementales de cotas
G92	Desplazamiento del punto de referencia
M00	Parada intermedia
M03	Encendido de husillo (marcha derecha)
M05	Parada de husillo
Instrucción	Función
M06	Entrada radio de la fresa

M17	Retorno al programa principal
M30	Fin de programa
M98 *	Corrección de holguras
M99	Parámetros de círculos graduados
Sin instrucción	Ninguna

Tabla 7.1 Instrucciones disponibles.

* Estas instrucciones son aceptadas por el compilador, pero no tienen efecto en la simulación, sólo en el proceso real.

La instrucción G92, sólo es aceptada una vez a lo largo del programa, por lo que las subrutinas (G25) sólo pueden ser programadas en modo incremental, en caso de que el programa principal se encuentre en modo absoluto, es necesario cambiar a modo incremental al principio de la subrutina.

Las teclas válidas son las siguientes:

7.2.2.1) Dígitos 0-9. Estos son aceptados en cualquier campo. Los diferentes campos disponibles para el usuario son: Instrucción G/M, X, Y, Z, F. Al estar en cualquiera de estos campos éste será resaltado automáticamente, invirtiendo los colores del texto y del fondo en los espacios que correspondan a dicho campo. En caso de entrar a un campo y oprimir cualquiera de estas teclas, automáticamente se borra la cifra. Si no se desea borrar es necesario oprimir cualquiera de las teclas de edición y después escribir.

7.2.2.2) Caracter M. Este sólo es aceptado en el campo de instrucciones G/M.

7.2.2.3) Signos + y -. Estos son aceptados en los campos X, Y y Z, siempre que la primera posición del campo esté vacía. El caracter " -" se escribe, pero el " + " solamente borra el signo negativo sin ser escrito. Es necesario aclara que al oprimir cualquiera de estas teclas el

programa considera que se ha realizado una edición a la cifra correspondiente, lo que significa que si después se oprime un caractér de escritura, este se encimará en la cifra anterior.

7.2.2.4) ENTER. Al presionar esta tecla, se entra la cifra en cualquier campo, o si se está en la primera posición del campo y no se ha realizado ningún cambio a la cifra, se saltará al siguiente campo disponible.

7.2.2.5) Bkc Spc. Esta tecla borra el caracter anterior, o salta al campo anterior si el cursor está en la posición inicial del campo

7.2.2.6) Teclas de movimiento. Estas teclas avanzarán a la posición que se indique, siempre y cuando la posición solicitada exista.

7.2.2.7) Suprime. Borra el caracter que se encuentre en la posición del cursor.

7.2.2.8) Inicio. Salta a la posición inicial del renglón.

7.2.2.9) Fin. Va al fin del renglón.

7.2.2.10) F10 .Abre el menú principal.

7.2.2.11) F9. Compila el programa en memoria.

7.2.2.12) F5. Inserta linea.

7.2.2.13) F4. Borra linea.

7.2.2.14) Pg up, Pg dn. estas teclas cambian la pantalla hacia arriba o abajo, respectivamente.

Al estar en el editor, la ventana de mensajes muestra las operaciones de las teclas de función disponibles, también presenta el nombre del archivo.

7.2.3 Opciones del compilador:

7.2.3.1) Compilar. Inicia la compilación, en caso de resultar algún error en el programa-pieza, la ventana de mensajes desplegará el texto de la alarma correspondiente, y al regresar al editor se posicionará automáticamente en la línea que haya presentado el error.

Las alarmas que se pueden presentar son las siguientes:

No. Alarma	Error
A 00	Instrucción G/M incorrecta
A 01	Radio / M99 incorrecto
A 02	Valor X incorrecto
A 03	Valor F incorrecto
A 04	Valor Z incorrecto
A 05	Falta instrucción M30
A 06	Falta instrucción M03
A 15	Valor Y incorrecto
A 16	Falta instrucción M06
A 17	Subprograma incorrecto
A 18	Recorrido de compensación de fresa menor a cero
A 19	Instrucción G92 duplicada
A 20	Falta instrucción M05.

Tabla 7.2 Alarmas.

7.2.3.2) Transmisión. Al oprimir ENTER, se inicia la compilación del programa-pieza, en caso de no encontrar errores, se transmite la información hacia la fresadora. Para iniciar la transmisión es necesario teclear en la fresadora G66, INP, INP, esto hará que la máquina cambie a modo *Load* en el que recibe la información por la interfase RS - 232.

7.2.4 Opciones del simulador:

7.2.4.1) Simulación continua. Al presionar ENTER, se compila el programa en caso de no encontrar errores, se inicia el modo gráfico y la simulación, en esta opción la simulación se realiza sin pausas a menos que el usuario lo ordene.

Para interrumpir la simulación se debe teclear ENTER, una vez en modo de pausa el usuario puede utilizar las siguientes funciones:

7.2.4.1.1) F1 Borra la vista lateral.

7.2.4.1.2) F2 Interruptor de trazo en la vista lateral.

7.2.4.1.3) F3 Borra la vista frontal.

7.2.4.1.4) F4 Interruptor de trazo en la vista frontal.

7.2.4.1.5) Spc Bar Reanuda la simulación.

Estas funciones se habilitan únicamente cuando la simulación es interrumpida, la única tecla disponible durante la simulación es ENTER.

Todas estas funciones están detalladas en la ventana de mensajes de la simulación, que corresponde al rectángulo inferior derecho de la pantalla.

Las vistas están distribuidas de la siguiente manera: vista superior cuadrante superior izquierdo, vista lateral en el inferior izquierdo y vista frontal en el superior derecha.

Cada una de las vistas cuenta con los ejes coordenados que indican el plano que muestra cada vista.

Existen unas líneas que se mueven a lado de la pieza en todas las vistas, estas líneas son mirillas que indican la posición del centro de la herramienta, esto con el fin de mostrar ubicación de la herramienta aunque ésta no aparezca. Esta condición se puede dar cuando la herramienta está fuera de los límites de la pieza bruta. El usuario debe tener precaución ya que estas mirillas

desaparecen cuando la posición de la herramienta implique salir del cuadrante de la pantalla asignado a la vista. En este caso la herramienta estará en lugares que pueden implicar un daño al equipo.

La ventana de mensajes muestra las coordenadas de la posición de la herramienta, la condición del husillo y detalla las funciones de las teclas disponibles.

7.2.4.2) Simulación por pasos. Esta opción realiza una simulación similar a la anterior, pero hace una pausa automática después de ejecutar cada comando.

7.2.4.3) Acerca de. Con esta opción se despliega en la ventana de mensajes del editor la pantalla de presentación inicial.

Conclusiones

En un trabajo de este tipo, es característico que el alumno elija el tema, y en este momento se dan situaciones especiales que probablemente nunca antes había manejado, ya que durante la vida académica los temas son impuestos por el profesorado dependiendo de las necesidades de la materia, la elección del tema implica la identificación de un problema o necesidad, debido a que la tesis deberá estar enfocada a la solución del problema o a cubrir la necesidad mediante la aplicación de algunos de los conocimientos que el alumno adquirió durante la carrera, en efecto algunos de los conocimientos dado que es prácticamente imposible aplicar todos los conocimientos adquiridos durante la carrera para resolver un solo problema.

En este particular caso la solución del problema o el cubrir la necesidad en cuestión, implica que el alumno desarrolle y profundice sus conocimientos y habilidades en las áreas de mayor interés para él.

Estos factores brindan la oportunidad al alumno de realizar un proyecto completo, es decir desde la identificación de la necesidad hasta la evaluación del mismo.

La idea de un simulador para la fresadora F1 CNC, que apoye el proceso enseñanza-aprendizaje, surge a partir de las necesidades que genera la industria sobre el sector educativo, en base a esto se diseñó y desarrolló una herramienta que facilite este proceso. Como se vió anteriormente el resultado fue un programa de computadora que ejemplifica el proceso de mecanizado de la máquina-herramienta, apoyándose en ciertas técnicas de la ingeniería.

Para la realización del programa fue necesario conjuntar los conocimientos de varias de las disciplinas que componen la carrera de ingeniería electromecánica. Algunas de estas disciplinas fueron control numérico, programación, procesos de manufactura, entre otras.

La parte más difícil en cuanto a su planeación fue la parte gráfica, dado que fue necesario buscar la manera más sencilla para ejemplificar evitando complejidad de programación. Las ideas que se generaron fueron objeto de mención en el capítulo tercero y sexto.

Una vez desarrollado el programa fue necesario realizar ciertas pruebas, en donde se identificaron algunos detalles, que fue necesario corregir. Finalmente se obtuvo el resultado esperado.

El haber concluido la tesis de licenciatura marca el fin de una etapa importante en la vida académica de una persona, ya que con ello concluye sus estudios profesionales y de alguna manera está listo para iniciar la vida profesional.

Probablemente en algún momento se piensa en una tesis de licenciatura como un mero trámite para obtener el título profesional, pero al momento de realizarla el esfuerzo involucrado elimina totalmente este sentimiento, convirtiéndose en un reto personal, el cual se traduce en una satisfacción especial al concluir el trabajo.

Bibliografía

ALIQUE LÓPEZ José Ramón, Control Numérico, Barcelona, 1981, Ed. Marcombo.

BORLAND Cía., Object-Oriented Programming, U S A, 1985, Borland.

BORLAND Cía., Turbo Pascal 5 Owner's Manual, U S A, 1988, Borland.

BORLAND Cía., Turbo Pascal 5 Reference Manual, U S A, 1988, Borland.

DOYLE E. Lawrence, et al., Materiales y Procesos de Manufactura para Ingenieros, (Trad. Fourier Gozáles Julio), México, 1988, Ed. Prentice Hall.

EMCO, Manual Basis F1-CNC, Austria, EMCO Documentación Técnica, (SP 7 700)

EMCO, Manual Edition for Software AGC 114 004, Austria, EMCO Tech. Documentation, (Ed 20019)

EMCO, Manual Visions, Austria, EMCO Documentación Técnica, (US 7 702)

GOLDSTEIN L. Joel, Turbo Pascal, (Trad. Palmas O. Alfredo), México, 1993, Ed. Prentice Hall.

PALMER D. Scott, Domine Turbo Pascal 6, México, 1992, Ed. Ventura.

Anexo 1

Archivo FRS_SIM.PAS

```
PROGRAM Frs_Sim;{ Principal y Editor}

USES
 Grafprb,cmds_gra,
 DOS,PRINTER,
 Cmds,Menus_ut,
 Comp_1,crt,Archivos;

CONST
 Yultima = 221;

VAR {general}
 posX,posY,a,b,
 Y,sel        : INTEGER;
 opc          : CHAR;
 {inst        : arreglo,}
 escribio,edito : BOOLEAN;

PROCEDURE ilumina;
VAR
 i  :  INTEGER;

BEGIN

 FOR i:=1 to 33 DO BEGIN
  GOTOXY(i,posY);
  IF inst[Y,i] <> '*' THEN WRITE(inst[Y,i])
  ELSE write(' ');
  IF i < 5 THEN TEXTCOLOR(LIGHTGRAY)
  ELSE TEXTCOLOR(YELLOW);
 END;
 GOTOXY(posX,posY);

 IF (posX > 5) AND ( posX < 9) THEN BEGIN
  TEXTCOLOR(WHITE);TEXTBACKGROUND(LIGHTGRAY);
  FOR i:=6 to 8 DO BEGIN
   GOTOXY(i,posY);WRITE(inst[Y,i]);
   GOTOXY(posX,posY);
  END;
  TEXTCOLOR(YELLOW);TEXTBACKGROUND(BLUE);
 END;
 IF (posX > 9) AND ( posX < 16) THEN BEGIN
  TEXTCOLOR(WHITE);TEXTBACKGROUND(LIGHTGRAY);
  FOR i:=10 to 15 DO BEGIN
   GOTOXY(i,posY);
   IF inst[Y,i] <> '*' THEN WRITE(inst[Y,i])
   ELSE WRITE(' ');
   GOTOXY(posX,posY);
  END;
  TEXTCOLOR(YELLOW);TEXTBACKGROUND(BLUE);
 END;
 IF (posX > 16) AND ( posX < 22) THEN BEGIN
  TEXTCOLOR(WHITE);TEXTBACKGROUND(LIGHTGRAY);
  FOR i:=17 to 21 DO BEGIN
   GOTOXY(i,posY);
   IF inst[Y,i] <> '*' THEN WRITE(inst[Y,i])
   ELSE WRITE(' ');
   GOTOXY(posX,posY);
  END;
  TEXTCOLOR(YELLOW);TEXTBACKGROUND(BLUE);
```

```
 END;
 IF (posX > 22) AND ( posX < 29) THEN BEGIN
  TEXTCOLOR(WHITE);TEXTBACKGROUND(LIGHTGRAY);
  FOR i:=23 to 28 DO BEGIN
   GOTOXY(i,posY);
   IF inst[Y,i] <> '' THEN WRITE(inst[Y,i])
   ELSE WRITE(' ');
   GOTOXY(posX,posY);
  END;
  TEXTCOLOR(YELLOW);TEXTBACKGROUND(BLUE);
 END;
 IF (posX > 29) AND ( posX < 34) THEN BEGIN
  TEXTCOLOR(WHITE);TEXTBACKGROUND(LIGHTGRAY);
  FOR i:=30 to 33 DO BEGIN
   GOTOXY(i,posY);
   IF inst[Y,i] <> '' THEN WRITE(inst[Y,i])
   ELSE WRITE(' ');
   GOTOXY(posX,posY).
  END;
  TEXTCOLOR(YELLOW);TEXTBACKGROUND(BLUE);
 END;

END;

PROCEDURE Limpia;

BEGIN
 IF (PosX>5) AND (PosX<9) THEN BEGIN
  inst[Y,6]:=' ';inst[Y,7]:=' ';inst[Y,8]:=' ';
  PosX:=6;
 END;
 IF (PosX>9) AND (PosX<16) THEN BEGIN
  IF inst[Y,10]=':' THEN inst[Y,10]:=' ';
  inst[Y,11]:=' ';inst[Y,12]:=' ';inst[Y,13]:=' ';
  inst[Y,14]:=' ';inst[Y,15]:=' ';
  PosX:=10;
 END;
 IF (PosX>16) AND (PosX<22) THEN BEGIN
  IF inst[Y,17]=':' THEN inst[Y,17]:=' ';
  inst[Y,18]:=' ';inst[Y,19]:=' ';inst[Y,20]:=' ';
  inst[Y,21]:=' ';
  PosX:=17;
 END;

 IF (PosX>22) AND (PosX<29) THEN BEGIN
  IF inst[Y,23]=':' THEN inst[Y,23]:=' ';
  inst[Y,24]:=' ';inst[Y,25]:=' ';inst[Y,26]:=' ';
  inst[Y,27]:=' ';inst[Y,28]:=' ';
  PosX:=23;
 END;
 IF (PosX>29) AND (PosX<34) THEN BEGIN
  inst[Y,31]:=' ';inst[Y,32]:=' ';inst[Y,33]:=' ';
  PosX:=31;
 END;

END;

PROCEDURE Scroll(i : Integer);
VAR
 j,k : INTEGER;

BEGIN
 {CLRSCR;}
 FOR j:=1 to 20 DO
  FOR k:=1 to 33 DO BEGIN
   GOTOXY(k,j);WRITE(' ');
  END;
```

```
 GOTOXY(1,1);
 FOR j:=i to i+19 DO
  FOR k:=1 to 33 DO BEGIN
   IF inst[j,k] <> '*' THEN gemelo[1+j-i,k]:=inst[j,k]
   ELSE gemelo[1+j-i,k]:=' ';
   GOTOXY(k,1+j-i);
   IF k < 6 THEN TEXTCOLOR(LIGHTGRAY)
   ELSE TEXTCOLOR(YELLOW);
   IF inst[j,k] <> '*' THEN
    WRITE(inst[j,k])
   ELSE WRITE(' '),
  END;
END;

PROCEDURE copia;
 VAR
  i,j,k  : INTEGER;
BEGIN

i:=Y-posY+1;

FOR j:=i to i+19 DO
  FOR k:=1 to 33 DO
   IF inst[j,k] <> '*' THEN gemelo[1+j-i,k]:=inst[j,k]
   ELSE gemelo[1+j-i,k]:=' ';
END;

PROCEDURE movaba;

VAR
 newY,l,primero   : integer;
 numero           : STRING [3];

BEGIN
 IF (Y = maxY) OR (inst[Y+1,posX] = '*') THEN BEGIN
  suena(50,100);
 END
 ELSE BEGIN
  FOR i:=1 to 33 DO BEGIN
   GOTOXY(i,posY);
   IF inst[Y,i] <> '*' THEN WRITE(inst[Y,i])
   ELSE WRITE(' ');
   IF i < 5 THEN TEXTCOLOR(LIGHTGRAY)
   ELSE TEXTCOLOR(YELLOW);
  END;
  newY := Y + 1;
  posY := posY + 1;
  IF inst[newY,posX] = '*' THEN BEGIN
   IF newY = maxY THEN BEGIN
    suena(50,100);
    newY:=Y;
    posY:=posY-1;
   END
   ELSE BEGIN
    newY := newY + 1;
    posY := posY + 1;
   END;
  END;
  IF (posY = 21)  THEN BEGIN
   scroll(Y-18);
   posY := 20;
  END;
  Y:=newY;
  Edito:=FALSE;
 END;
 STR(Y,numero);
 primero:=4;
 FOR b:=LENGTH(numero) DOWNTO 1 DO BEGIN
```

```
  inst[Y,primero]:=numero[b];primero:=primero-1;
 END;

 GOTOXY(posX,posY);
 TEXTCOLOR(LIGHTGRAY);
 WRITE(inst[Y,posX]);
 TEXTCOLOR(YELLOW);
 GOTOXY(posX,posY);
END;

PROCEDURE movar;

VAR
 newY,i : integer;

BEGIN
 IF (Y = 1) OR (inst[Y-1,posX] = '*') THEN BEGIN
  suena(50,100);
 END
 ELSE BEGIN
  FOR i:=1 to 34 DO BEGIN
   GOTOXY(i,posY);
   IF inst[Y,i] <> '*' THEN WRITE(inst[Y,i])
   ELSE WRITE(' ');
   IF i < 5 THEN TEXTCOLOR(LIGHTGRAY)
   ELSE TEXTCOLOR(YELLOW);
  END;

  newY := Y - 1;
  posY := posY - 1;
  IF inst[newY,posX] = '*' THEN BEGIN
   newY := newY - 1;
   posY := posY - 1;
  END;
  IF (posY = 0)  THEN BEGIN
    scroll(newY);
    posY := 1;
  END;
  Y:=newY;
  Edito:=FALSE;
 END;
 GOTOXY(posX,posY);
 WRITE(inst[Y,posX]);
 GOTOXY(posX,posY);
END;

PROCEDURE movder;

VAR
 newX,checa : INTEGER;

BEGIN
 newX:=posX+1;
 case newX of
  9  : BEGIN
       newX:=posX+2;
       checa:=Busca(inst[Y,6]+inst[Y,7]+inst[y,8]);
       IF Checa <> 0 THEN
        limita(checa,inst[y])
       ELSE BEGIN
        suena(50,100);
        Mensaje('e',6,Y);
        Mensaje('m',1,Y);
        newX:=6;
        posX:=6;
       END;
      END;
  16 : newX:=posX+2;
```

```
 22 : newX:=posX+2;
 29 : newX:=posX+3;
 34 : BEGIN
      newX:=6;
      IF Y < maxY THEN movaba;
     END;
END;
REPEAT
 IF inst[y,newX] = '*' THEN BEGIN
  newX:=newX+1;
  CASE newX OF
   9  : newX:=newX+1;
   16 : newX:=newX+1;
   22 : newX:=newX+1;
   29 : newX:=newX+2;
   30 : newX:=newX+1;
   34 : BEGIN
     newX:=6;
     IF Y < maxY THEN movaba;
    END;
  END;
 END;
UNTIL inst[Y,newX] <> '*';
IF posX+1 = newX THEN Edito:=TRUE;
posX:=newX;
GOTOXY(posX,posY);
WRITE(inst[Y,posX]);
GOTOXY(posX,posY);
END;

PROCEDURE movizq;

VAR
 newX : INTEGER;

BEGIN
 newX:=posX-1;
 CASE newX OF
  9  : newX:=newX-1;
  16 : newX:=newX-1;
  22 : newX:=newX-1;
  30 : newX:=newX-2;
  5  : newX:=6;
 END;
 REPEAT
  IF inst[y,newX] = '*' THEN BEGIN
   newX:=newX-1;
   CASE newX OF
    9  : newX:=newX-1;
    16 : newX:=newX-1;
    22 : newX:=newX-1;
    30 : newX:=newX-2;
    5  : newX:=6;
   END;
  END;
 UNTIL inst[Y,newX] <> '*';
 IF posX-1 = newX THEN Edito:=TRUE;
 posX:=newX;
 GOTOXY(posX,posY);
 WRITE(inst[Y,posX]);
 GOTOXY(posX,posY);
END;

PROCEDURE Completa;
VAR
 i,ultimo,primero,oldY,checa,
 espacio,j,newX   : INTEGER;
 cop              : STRING [10];
```

```
BEGIN
 Edito:=FALSE;
 cop:='';
 NewX:=posX;
 oldY:=Y;
 IF NOT escribio THEN BEGIN
  IF posX = 6 THEN BEGIN
   newX:=10;
   checa:=Busca(inst[Y,6]+inst[Y,7]+inst[y,8]);
   IF Checa <> 0 THEN
    limita(checa,inst[y])
   ELSE BEGIN
    suena(50,100);
    Mensaje('e',6,'e');
    Mensaje('m',1,'e');
    newX:=6;
    posX:=6;
   END;
   {IF checa = 31 THEN newX:=6;}
  END;
  IF posX = 10 THEN newX:=17;
  IF posX = 17 THEN newX:=23;
  IF posX = 23 THEN newX:=31;
 END;
 escribio:=false;
 IF (posX > 6) AND ( posX < 9) THEN BEGIN
  primero := 7;
  ultimo  := 8;
  IF inst[Y,ultimo] = '' THEN BEGIN
   i:=primero;
   WHILE inst[Y,i] <> '' DO BEGIN
    cop:=cop+inst[Y,i];
    i:=i+1;
   END;
   FOR i:=LENGTH(cop) DOWNTO 1 DO
    inst[Y,ultimo-LENGTH(cop)+i]:=cop[i];
   FOR i:=primero TO ultimo - LENGTH(cop) DO
    inst[Y,i]:=' ';
  END;
  NewX:=10;
  checa:=Busca(inst[Y,6]+inst[Y,7]+inst[y,8]);
   IF Checa <> 0 THEN
    limita(checa,inst[y])
   ELSE BEGIN
    Suena(50,100);
    Mensaje('e',6,'e');
    Mensaje('m',1,'e');
    newX:=6;
   END;
   IF Checa = 31 THEN newX:=6;
 END;
 IF (posX > 10) AND ( posX < 16) THEN BEGIN
  primero := 11;
  ultimo  := 15;
  IF inst[Y,ultimo] = '' THEN BEGIN

   i:=primero;
   a:=primero;

   REPEAT
    IF inst[Y,a]=' ' THEN i:=i+1;
    a:=a+1;
   UNTIL (inst[y,a-1] <> ' ') OR (a = ultimo);


   WHILE inst[Y,i] <> '' DO BEGIN
    cop:=cop+inst[Y,i];
```

```
    i:=i+1;
   END;
   FOR i:=LENGTH(cop) DOWNTO 1 DO
    inst[Y,ultimo-LENGTH(cop)+i]:=cop[i];
   FOR i:=primero TO ultimo - LENGTH(cop) DO
    inst[Y,i]:=' ';
  END;
  Newx:=17;
 END;
 IF (posX > 17) AND ( posX < 22) THEN BEGIN
  primero := 18;
  ultimo := 21;
  IF inst[Y,ultimo] = '' THEN BEGIN
   i:=primero;
   a:=primero;

   REPEAT
    IF inst[Y,a]=' ' THEN i:=i+1;
    a:=a+1;
   UNTIL (inst[y,a-1] <> ' ') OR (a = ultimo);

   WHILE inst[Y,i] <> '' DO BEGIN
    cop:=cop+inst[Y,i];
    i:=i+1;
   END;
   FOR i:=LENGTH(cop) DOWNTO 1 DO
    inst[Y,ultimo-LENGTH(cop)+i]:=cop[i];
   FOR i:=primero TO ultimo - LENGTH(cop) DO
    inst[Y,i]:=' ';
  END;
  Newx:=23;
 END;
 IF (posX > 23) AND ( posX < 29) THEN BEGIN
  primero := 24;
  ultimo := 28;
  IF inst[Y,ultimo] = '' THEN BEGIN
   i:=primero;
   a:=primero;

   REPEAT
    IF inst[Y,a]=' ' THEN i:=i+1;
    a:=a+1;
   UNTIL (inst[y,a-1] <> ' ') OR (a = ultimo);

   WHILE inst[Y,i] <> '' DO BEGIN
    cop:=cop+inst[Y,i];
    i:=i+1;
   END;
   FOR i:=LENGTH(cop) DOWNTO 1 DO
    inst[Y,ultimo-LENGTH(cop)+i]:=cop[i];
   FOR i:=primero TO ultimo - LENGTH(cop) DO
    inst[Y,i]:=' ';
  END;
  Newx:=31;
 END;
 IF (posX > 30) AND ( posX < 34) THEN BEGIN
  primero := 31;
  ultimo := 33;
  IF inst[Y,ultimo] = '' THEN BEGIN
   i:=primero;
   a:=primero;

   REPEAT
    IF inst[Y,a]=' ' THEN i:=i+1;
    a:=a+1;
   UNTIL (inst[y,a-1] <> ' ') OR (a = ultimo);

   WHILE inst[Y,i] <> '' DO BEGIN
```

```
    cop:=cop+inst[Y,i];
    i:=i+1;
   END;
   FOR i:=LENGTH(cop) DOWNTO 1 DO
    inst[Y,ultimo-LENGTH(cop)+i]:=cop[i];
   FOR i:=primero TO ultimo - LENGTH(cop) DO
    inst[Y,i]:=' ';
  END;
  Newx:=34;
 END;
 posX:=newX;
 IF (inst[Y,posX] = '*') THEN BEGIN
  movder;
  IF (posX = 6) AND (Y = oldY) THEN posX:=34;
 END;

 IF posX = 34 THEN BEGIN
  IF Y=maxY THEN maxY := maxY + 1;
  IF Y=Yultima THEN BEGIN
   suena(50,100);
   PosX:=8;maxY:=Yultima;
  END
  ELSE BEGIN
   posX:= 6;
   movaba;
  END;
 END;
END;

PROCEDURE Borra;
VAR
 i,primero,ultimo : INTEGER;
BEGIN
 Edito:=TRUE;
 {IF posX = 1 THEN inst[Y,1]:=' ';
 IF (posX > 1) AND (posX < 5) THEN BEGIN
  primero:=2;ultimo:=4;
  FOR i:=posX to ultimo DO
   inst[Y,i]:=inst[Y,i+1];
 END;}

 IF (posX = 6) {AND (inst[Y,posX]=' ')} THEN inst[Y,6]:=' ';
 IF (posX > 6) AND (posX < 9) THEN BEGIN
  primero:=7;ultimo:=8;
  FOR i:=posX to ultimo DO
   inst[Y,i]:=inst[Y,i+1];
 END;
 IF (posX = 10) AND (inst[Y,posX]='-') THEN inst[Y,10]:=' ';
 IF (posX > 10) AND (posX < 16) THEN BEGIN
  primero:=11;ultimo:=15;
  FOR i:=posX to ultimo DO
   inst[Y,i]:=inst[Y,i+1];
 END;
 IF (posX = 17) AND (inst[Y,posX]='-') THEN inst[Y,17]:=' ';
 IF (posX > 17) AND (posX < 22) THEN BEGIN
  primero:=18;ultimo:=21;
  FOR i:=posX to ultimo DO
   inst[Y,i]:=inst[Y,i+1];
 END;
 IF (posX = 23) AND (inst[Y,posX]='-') THEN inst[Y,23]:=' ';
 IF (posX > 23) AND (posX < 29) THEN BEGIN
  primero:=24;ultimo:=28;
  FOR i:=posX to ultimo DO
   inst[Y,i]:=inst[Y,i+1];
 END;
 IF (posX > 30) AND (posX < 34) THEN BEGIN
  primero:=31;ultimo:=33;
  FOR i:=posX to ultimo DO
```

```
    inst[Y,i]:=inst[Y,i+1];
  END;
  {IF (posX > 34) AND (posX < 38)  THEN BEGIN
   primero:=35;ultimo:=37;
   FOR i:=posX to ultimo DO
    inst[Y,i]:=inst[Y,i+1];
  END;}
END;

PROCEDURE Bor_st;
VAR
 i,primero,ultimo  : INTEGER;
BEGIN
   IF (posX = 31) OR (posX = 23) OR (posX = 17) OR (posX = 10) THEN BEGIN
    REPEAT
     IF posX = 10 THEN posX:=6;
     IF posX = 17 THEN posX:=10;
     IF posX = 23 THEN posX:=17;
     IF posX = 31 THEN posX:=23;
    UNTIL inst[Y,posX] <> '*';
   END
   ELSE BEGIN
    Edita:=TRUE;
   END;
  IF (posX > 7) AND (posX < 9)  THEN BEGIN
   primero:=7;ultimo:=8;
   FOR i:=posX-1 to ultimo DO
    inst[Y,i]:=inst[Y,i+1];
   Moviza;
  END;
  IF (posX > 11) AND (posX < 16)  THEN BEGIN
   primero:=11;ultimo:=15;
   FOR i:=posX to ultimo DO
    inst[Y,i]:=inst[Y,i+1];
   MOviza;
  END;
  IF (posX > 18) AND (posX < 22)  THEN BEGIN
   primero:=18;ultimo:=21;
   FOR i:=posX-1 to ultimo DO
    inst[Y,i]:=inst[Y,i+1];
   moviza;
  END;
  IF (posX > 24) AND (posX < 29)  THEN BEGIN
   primero:=24;ultimo:=28;
   FOR i:=posX-1 to ultimo DO
    inst[Y,i]:=inst[Y,i+1];
   moviza;
  END;
  IF (posX > 31) AND (posX < 34)  THEN BEGIN
   primero:=31;ultimo:=33;
   FOR i:=posX-1 to ultimo DO
    inst[Y,i]:=inst[Y,i+1];
   moviza;
  END;
  {IF (posX > 35) AND (posX < 38)  THEN BEGIN
   primero:=35;ultimo:=37;
   FOR i:=posX-1 to ultimo DO
    inst[Y,i]:=inst[Y,i+1];
   moviza;
  END;}
END;


PROCEDURE Escribe;
VAR
 newX,checa,antX   : INTEGER;
 flag,no_comp,signo :  BOOLEAN;
```

```
BEGIN
   Signo:=FALSE;
   IF opc = 'm' THEN opc:='M';
   escribio:=false;
   newX:=posX;AntX:=PosX;
   flag:=false;
   IF opc IN coman THEN BEGIN
     If (posX = 6) AND (opc = 'M') THEN flag:=true;
     IF ((posX>6) AND (posX<9)) THEN BEGIN
      Flag:=TRUE;posX:=6;
     END;
   END;
   IF (opc IN sig)  THEN BEGIN
     IF opc='+' THEN opc:=' ';
     IF ((posX = 10) OR (posX = 17) OR (posX = 23)) AND
     ((inst[y,posX]=' ') OR (inst[y,posX]='-')) THEN BEGIN
      flag:=true;
      Edito:=TRUE;
     END;
     IF ((posX>10) AND (posX<16)) AND ((inst[Y,10] = ' ') OR ( inst[y,10]='-')) THEN BEGIN
      Signo:=TRUE;
      posX:=10;
      Flag:=TRUE;
      edito:=TRUE;
     END;
     IF ((posX>17) AND (posX<22)) AND ((inst[Y,17] = ' ') OR ( inst[y,17]='-')) THEN BEGIN
      Signo:=TRUE;
      posX:=17;
      Flag:=TRUE;
      Edito:=TRUE;
     END;
     IF ((posX>23) AND (posX<29)) AND ((inst[Y,23] = ' ') OR ( inst[y,23]='-')) THEN BEGIN
      Signo:=TRUE;
      posX:=23;
      Flag:=TRUE;
      Edito:=TRUE;
     END;
   END;
   IF opc IN enteros THEN BEGIN
     IF (posX = 7 ) OR (posX = 8 ) OR (posX = 11) THEN flag:=true;
     IF (posX = 12) OR (posX = 13) OR (posX = 14) THEN flag:=true;
     IF (posX = 15) OR (posX = 18) OR (posX = 19) THEN flag:=true;
     IF (posX = 20) OR (posX = 21) OR (posX = 24) THEN flag:=true;
     IF (posX = 25) OR (posX = 26) OR (posX = 27) THEN flag:=true;
     IF (posX = 28) OR (posX = 31) OR (posX = 32) THEN flag:=true;
     IF (posX = 33) THEN flag:=true;
     IF ((posX = 6 ) OR (posX = 10) OR (posX = 17) OR (posX = 23))(AND(inst[y,posX]=' '))
      THEN BEGIN
      flag:=true;posX:=posX+1;
     END;
   END;
   IF flag THEN BEGIN
     IF NOT Edito THEN BEGIN
      Limpia;Edito:=TRUE;
     END;
     IF ((posX = 6 ) OR (posX = 10) OR (posX = 17) OR (posX = 23))AND(opc IN ENTEROS)
      THEN BEGIN
      posX:=posX+1;
     END;
     inst[Y,posX]:=opc;
     GOTOXY(posX,posY);WRITE(opc);
     posX:=posX + 1;
   END;
   IF NOT flag THEN BEGIN
     suena(50,100);
     Mensaje('e',1,'e');
     Mensaje('m',1,'e');
```

```
  END;
  CASE posX OF
   9 : BEGIN
       no_comp:=false;
       posX:=posX-1;
       checa:=Busca(inst[Y,6]+inst[Y,7]+inst[y,8]);
       {gotoxy(70,1);write(checa);readln;}
       IF Checa <> 0 THEN
        limita(checa,inst[y])
       ELSE BEGIN
        suena(50,100);
        no_comp:=true;
        Mensaje('e',6,'e');
        Mensaje('m',1,'e');
        posX:=6;
       END;
       IF (no_comp=false) AND (checa <> 31) THEN completa;
       escribio:=true;
      END;
   16 : BEGIN
       posX:=posX-1;completa;escribio:=true;
      END;
   22 : BEGIN
       posX:=posX-1;completa;escribio:=true;
      END;
   29 : BEGIN
       posX:=posX-1;completa;escribio:=true;
      END;
   34 : BEGIN
       IF Y < maxY THEN BEGIN
        posX:=6;
        movaba;
       END
       ELSE completa;escribio:=true;
      END;
  END;
  IF signo THEN posX:=antX;
  GOTOXY(posX,posY);
END;

PROCEDURE Ins_lin;
 VAR
  i,j,primero : INTEGER;
  Numero      : STRING [3];

 PROCEDURE Pinta(Gene : Arre_12);
  VAR
   j,k : INTEGER;
  BEGIN
   {CLRSCR;}
   Pantalla('e',CORx,Cory);
   GOTOXY(1,1);
   FOR j:=1 to 20 DO
    FOR k:=1 to 37 DO BEGIN
     GOTOXY(k,j);
     IF k<6 THEN BEGIN
      TEXTCOLOR(LIGHTGRAY)
     END
     ELSE TEXTCOLOR(YELLOW);
     WRITE(gene[j,k]);
    END;
   {FOR j:=1 to 6 DO
    FOR k:=34 TO 78 DO BEGIN
     GOTOXY(k,j);WRITE(' ');
    END;}
  END;

BEGIN
```

```
 IF maxY < Yultima THEN BEGIN
  FOR i:=maxY DOWNTO Y DO BEGIN
    STR(i+1,numero);
   primero:=4;
   FOR b:=LENGTH(numero) DOWNTO 1 DO BEGIN
    inst[i+1,primero]:=numero[b];primero:=primero-1;
   END;
   FOR j:=6 TO 33 DO
    inst[i+1,j]:=inst[i,j];
  END;
  FOR i:=6 TO 33 DO
   inst[Y,i]:=' ';
  Copia;
  Pinta(gemelo);
  posX:=6;
  maxY:=MaxY+1;
  Edito:=FALSE;
 END
 ELSE BEGIN
  Suena(50,100);
  Mensaje('e',2,'m');
  Mensaje('m',1,'m');
 END;
END;

PROCEDURE Bo_lin;

VAR
 i,j   : INTEGER;

 PROCEDURE Pinta(Geme : Arre_12);
  VAR
   j,k :  INTEGER;
  BEGIN
   {CLRSCR;}
   Pantalla('e',CORx,Cory);
   GOTOXY(1,1);
   FOR j:=1 to 20 DO
    FOR k:=1 to 37 DO BEGIN
     GOTOXY(k,j);
     IF k<6 THEN BEGIN
      TEXTCOLOR(LIGHTGRAY)
     END
     ELSE TEXTCOLOR(YELLOW);
     WRITE(geme[j,k]);
    END;
   {FOR j:=1 to 6 DO
    FOR k:=34 TO 78 DO BEGIN
     GOTOXY(k,j);WRITE(' ');
    END;}
  END;
BEGIN
 FOR i:=6 TO 33 DO
  inst[Y,i]:=' ';
 FOR i:=Y TO MaxY DO
  FOR j:=6 TO 33 DO BEGIN
   inst[i,j]:=inst[i+1,j];
  END;
 FOR i:=1 TO 5 DO
  inst[maxY,i]:=' ';
 IF maxY = Y THEN movar;
 IF maxY>1 THEN maxY:=maxY-1;
 posX:=6;
  Copia;
 pinta(gemelo);
 Edito:=FALSE;
END;
PROCEDURE Pag_arr;
```

```
VAR
 Yini : INTEGER;
BEGIN
 Yini:= Y + 1 - posY;
 IF (Yini - 20) >= 1 THEN BEGIN
  Scroll(Yini - 20);
  IF (Y -20) >=1 THEN
   Y:=Y-20
  ELSE BEGIN
   Y:=Yini - 20;
   posY:=1;
  END;
 END
 ELSE BEGIN
  Scroll(1);
  POsY:=1;
  Y:=1;
 END;
END;
PROCEDURE Pag_aba;
VAR
 Yini : INTEGER;
BEGIN
 Yini:= Y + 1 - PosY;
 IF (Yini + 20) <= MaxY THEN BEGIN
  Scroll(yini+20);
  IF (Y+20)< maxY THEN
   Y:=Y+20
  ELSE BEGIN
   Y:=maxY;
   posY:=maxY + 1 - (Yini + 20);
  END;
 END
 ELSE BEGIN
  Scroll(Yini);
  posY := MaxY +1-Yini;
  Y:=maxY;
 END;

END;

PROCEDURE Imprime;
VAR
 Lin_imp : Anystr;
 IoCode  : INTEGER;
BEGIN
 { $I- }
 WRITE(LST,#10);
 IoCode:=IoResult;
 { $I+ }
 IF IoCode<>0 THEN Pro_imp
 ELSE BEGIN
  WRITELN(LST,' Archivo : '+ Archivo);
  WRITELN(LST,' ');
  WRITELN(LST,' Dimensiones : ');
  WRITE(LST,' Largo  = ');WRITELN(LST,Largo);
  WRITE(LST,' Ancho  = ');WRITELN(LST,Ancho);
  WRITE(LST,' Altura = ');WRITELN(LST,Alto);
  WRITELN(LST,' ');
  WRITELN(LST,'    N G/M   X   Y   Z   F');
  FOR a:=1 TO MaxY DO BEGIN
   Lin_imp:='';
   FOR b:=1 TO 33 DO BEGIN
    IF inst[a,b] <> '' THEN Lin_imp:=Lin_imp+inst[a,b]
    ELSE Lin_imp:=Lin_imp+' ';
   END;
   WRITELN(LST,' '+Lin_imp);
  END;
```

```
 END;
END;

PROCEDURE Inicializa;

VAR
 Numero    : STRING [3];
 i,primero,j,k,
 ultimo    : INTEGER;
 cop       : STRING [10];
BEGIN
 Inf;
 FOR a:=1 TO 250 DO
  FOR b:=1 TO 40 DO
   inst[a,b]:=' ';



 Pantalla('m',posX,Y);CLRSCR;
 Pantalla('i',posX,Y);
 Pantalla('e',0,0);CLRSCR;
 {Mensaje('e',1,'e');}
 posX:=6;posY:=1;   {inicializaciones}
 Y:=1;maxY:=1;
 Archivo:='CUALQ';
 Largo:=0;Alto:=0;Ancho:=0;Diam:=0;
 Xus:=0;Yus:=0;Zus:=0;
 FOR j:=1 to 20 DO
  FOR k:=1 to 37 DO
   gemelo[j,k]:=' ';
 Dirgrf:=Dir_ini;
END;

{FUNCTION Lee (Jgo : JgoChar) : CHAR;
 VAR
  C,C1,Aux : CHAR;

 BEGIN
  Aux:=#144;
   C:=READKEY;
   IF KEYPRESSED
    THEN
     BEGIN
      C1:=READKEY;
      CASE C1 OF
       #59 : Aux:=F1;
       #60 : Aux:=F2;
       #61 : Aux:=F3;
       #62 : Aux:=F4;
       #63 : Aux:=F5;
       #67 : Aux:=F9;
       #68 : Aux:=F10;
       #71 : Aux:=Home;
       #72 : Aux:=Ar;
       #73 : Aux:=Pgup;
       #75 : Aux:=Izq;
       #77 : Aux:=Der;
       #79 : Aux:=Fin;
       #80 : Aux:=Ab;
       #81 : Aux:=Pgdn;
       #82 : Aux:=Ins;
       #83 : Aux:=Bo;
      END;
     END
    ELSE
     Aux:=C;
   IF (NOT (Aux IN Jgo)) THEN BEGIN
    Suena(50,100);
```

```
      Mensaje('e',1,'e');
      IF Men THEN Mensaje('m',2,'m');
      IF editor THEN Mensaje('m',1,'e');
    END;
  Lee:=Aux;
 END;}

PROCEDURE Checa_AL;
VAR
 NL,Al : INTEGER;

BEGIN
 Checa(NL,Al,inst,MaxY);
 IF AL <> 0 THEN BEGIN
      mensaje('e',AL + 6,'e');
      mensaje('m',1,'e');
      scroll(NL);
      Y:=NL;
      posY:=1;
      posX:=6;
 END;
END;

PROCEDURE Edita;

BEGIN
 Mensaje('m',1,'e');
 Editor:=true;
 escribio:=false;
 Edito:=FALSE;
 REPEAT
  pantalla(Y,posX,Y);
  Pantalla('e',posX,Y);
  GOTOXY(posX,posY);
  inst[1,4]:='1';
  iluminas;
  opc:=lee( Sig + enteros + comun + term + mueve);
  CASE opc OF
   Der    : movder;
   Izq    : movizq;
   Ab     : movaba;
   Ar     : movar;
   Enter  : completa;
   Home   : posX:=6;
   Fin    : BEGIN
              posX:=33;
              IF inst[Y,posX] = '*' THEN movizq;
            END;
   Bo     : Borra;
   Backspc : Bor_at;
   F9     : Checa_Al;
   F5     : Ins_lin;
   F4     : Bo_lin;
   Pgup   : Pag_arr;
   PgDn   : Pag_aba;
  END;
  IF opc IN (enteros + comun + sig) THEN  Escribe;

 UNTIL (opc=F10);
 CASE opc OF
  F10 : sel:=23;
 END;
 copia;CorX:=posX;CorY:=Y;
 pinta(gemelo);
 IF sel <> 21 THEN sel :=23;
 Editor:=false;
END;
```

```
BEGIN
 {FOR a:=1 TO 1010 DO STR(a,inst[a]); { prueba}
 Inicializa;
 Drv:=3;
 GETDIR(Drv,Dir_ini);
 Direc:=Dir_ini;
 Dirgrf:=Dir_ini;
 Archivo:='CUALQ';
 copia;CorX:=posX;CorY:=Y;
 pinta(gemelo);
 Sel:=23;Uno:=true;
 REPEAT
  Menu_opc(sel);
   CASE sel OF

     11 : BEGIN
          Mensaje('m',3,'e');
          Opc:=Lee(SINO);
          IF UPCASE(Opc) = 'S' THEN BEGIN
           Cargar(inst);
           VAL(inst[223],maxY,a);
           VAL(inst[224],Largo,a);
           VAL(inst[225],Ancho,a);
           VAL(inst[226],Alto,a);
           VAL(inst[227],Xus,a);
           VAL(inst[228],Yus,a);
           VAL(inst[229],Zus,a);
           {write(MaxY);readln;}
           PosX:=6;
           PosY:=1;
           Y:=1;
           Pantalla('e',6,1);
           Scroll(1);
          END;
          Edita;
         END;
     12 : BEGIN
          Mensaje('m',3,'e');
          Opc:=Lee(SINO);
          IF UPCASE(Opc) = 'S' THEN BEGIN
           Inicializa;
           Sel:=23;Uno:=true;
           Archivo:='CUALQ';
           Menu_opc(Sel);
           dim;
          END;

          Edita;
         END;
     13 : BEGIN

          STR(maxY,inst[223]);
          STR(Largo,inst[224]);
          STR(Ancho,inst[225]);
          STR(ALto,inst[226]);
          STR(Xus,inst[227]);
          STR(Yus,inst[228]);
          STR(Zus,inst[229]);
          Salvar(inst);
          Edita;
         END;
     14 : BEGIN
          Directorio;
          Edita;
         END;
     15 : BEGIN
          Imprime;
```

```
    Edita;
   END;
16 : BEGIN
    Mensaje('m',3,'e');
    Opc:=Lee(SINO);
    IF UPCASE(Opc) = 'S' THEN BEGIN
     Sel:=16;
    END
    ELSE
    Edita;
   END;
21 : BEGIN
    Ins_lin;
    Edita;
   END;
22 : BEGIN
    Bo_lin;
    Edita;
   END;
23 : BEGIN
    IF uno THEN Presenta;
    Dim;
    Edita;
      {cargar(inst);
      VAL(inst[223],maxY,e);
      VAL(inst[224],Largo,e);
      VAL(inst[225],Ancho,e);
      VAL(inst[226],Alto,e);
      VAL(inst[227],Xus,e);
      VAL(inst[228],Yus,e);
      VAL(inst[229],Zus,e);
      Checa_Al;
      inicia;
      Paso:=TRUE;
      Simula(inst);
      edita;}
   END;
24 : Edita;
31 : BEGIN
    Checa_Al;
    Edita;
   END;
32 : BEGIN
    Checa_al;
    Checa(a,b,inst,MaxY);
    IF b = 0 THEN
     TRANS(inst,maxY);
    Edita;
   END;
41 : BEGIN
    b:=0;
    Checa(a,b,inst,MaxY);
    IF b = 0 THEN BEGIN
     Paso:=FALSE;
     inicia;
     Paso:=FALSE;
     Simula(inst);
     Pantalla('m',posX,Y);CLRSCR;
     titula(0);
     Pantalla(Y,posX,Y);
     Pantalla('w',0,0);CLRSCR;
     Scroll(1);
     PosX:=6;
     PosY:=1;
     Y:=1;
    END
    ELSE Checa_Al;
    edita;
```

```
      END;
    42 : BEGIN
         h:=0;
         Checa(a,b,inst,MaxY);
         IF b = 0 THEN BEGIN
          Paso:=TRUE;
          inicia;
          Simula(inst);
          Pantalla('m',posX,Y);CLRSCR;
          titula(0);
          Pantalla('i',posX,Y);
          Pantalla('e',0,0);CLRSCR;
          Scroll(1);
          PosX:=6;PosY:=1;
          Y:=1;
         END
         ELSE Checa_al;
         edita;
        END;
    43 : BEGIN
          Presenta;
          Edita;
         END;
   END;
  Uno:=false;
  IF sel<> 16 THEN sel:=24;
 UNTIL Sel = 16;
 WINDOW(1,1,80,24);TEXTCOLOR(WHITE);
 TEXTBACKGROUND(BLACK);CLRSCR;
END.
```

Archivo COMP_1.PAS

```
UNIT Comp_1;

INTERFACE

USES CRT;
CONST
  Pgdn   = #257;
  Pgup   = #256;
  Izq    = #255;
  Der    = #254;
  Ar     = #253;
  Ab     = #252;
  Ins    = #251;
  Do     = #250;
  Home   = #249;
  Fin    = #248;
  F1     = #247;
  F2     = #246;
  F3     = #245;
  F4     = #244;
  F5     = #243;
  F6     = #242;
  F7     = #241;
  F8     = #240;
  F9     = #239;
  F10    = #238;
  BoLin  = #25;
  Enter  = #13;
  Esc    = #27;
  Backspc = #8;
  Spc    = #32;

  Reales : SET OF CHAR = [#46,#48..#57,'+','E','+'];
  Enteros: SET OF CHAR = [#48..#57];
  Strings: SET OF CHAR = [#32..#126];
  Term   : SET OF CHAR = [Ar,Ab,Enter,Esc,F1,F2,F3,F4,F5,F9,F10];
  Edicion: SET OF CHAR = [Izq,Der,Backspc,Home,Fin,Ins,Do,BoLin];
  SiNo   : SET OF CHAR = ['N','n','S','s'];
  FS     : SET OF CHAR = ['L','O'];
  Sig    : SET OF CHAR = ['-','+'];
  mueve  : SET OF CHAR = [ Bo,Backspc,Home,Fin,izq,der,Ar,ab,Enter,F1,Pgup,Pgdn];
  Coman  : SET OF CHAR = ['M','m'];
  Mov_men: SET OF CHAR = [Ar,Ab,der,izq,enter,esc,F10,F1,F5,F4,F9];
  Signo  : SET OF CHAR = ['+','-','E'];
  Letras : SET OF CHAR = ['A'..'Z','a'..'z','_','0'..'9'];
  Grafs  : SET OF CHAR = [spc,enter,esc,F10,F1,F2,F3,F4,F5,F6,F7,F8,F9];
TYPE
  JgoChar  = SET OF CHAR;
  AnyStr   = STRING[40];
  Arreglo  = Array[1..250] OF AnyStr;
  Arre_12  = ARRAY[1..22] OF Anystr;

VAR
  Gemelo          : Arre_12;
  CorX,CorY,maxY      : INTEGER;
  Ancho,Alto,Largo,Diam : INTEGER;
  Xus,Yus,Zus         : INTEGER;
  Men,editor      : BOOLEAN;
  Archivo,Hus_stat    : Anystr;
  inst            : Arreglo;
  Stat_lst,Stat_fron  : INTEGER;
  Char_num        : CHAR;
```

```
PROCEDURE Suena   (Frec,Dur : INTEGER);
PROCEDURE pinta   (Gene : Arre_12);
PROCEDURE Pantalla (pant : char; px,py : INTEGER);
FUNCTION  Leo     (Jgo : JgoChar) : CHAR;



IMPLEMENTATION

PROCEDURE Suena (Frec,Dur : INTEGER);
 BEGIN
   SOUND (Frec);
   DELAY (Dur);
   NOSOUND;
 END;

PROCEDURE Pantalla(pant : char; px,py : INTEGER);
 VAR
  i : INTEGER;
 BEGIN
  CASE pant OF
   'a' : BEGIN
      TEXTBACKGROUND(BLUE);
      TEXTCOLOR(LIGHTGRAY);
      WINDOW(1,1,80,25);
      FOR i:=2 TO 79 DO BEGIN
       GOTOXY(i,24);WRITE(#205);
       IF (i>2)AND(i<24) THEN BEGIN
        GOTOXY(1,i);WRITE(#186);
        GOTOXY(80,i);WRITE(#186);
        END;
      END;
      GOTOXY(80,24);WRITE(#188);
      GOTOXY(1,24);WRITE(#200);
      TEXTCOLOR(YELLOW);
      WINDOW(2,4,79,23);
    END;
    'm' : BEGIN
      WINDOW(1,1,80,1);
      TEXTBACKGROUND(GREEN);
      TEXTCOLOR(WHITE);
    END;
    'f' : BEGIN
      WINDOW(1,1,80,24);
      TEXTCOLOR(LIGHTGRAY);
      TEXTBACKGROUND(BLUE);
      GOTOXY(80,2);WRITE('x');
      GOTOXY(80,3);WRITE('');
      WINDOW(1,2,79,3);
      CLRSCR;
      FOR i:=2 TO 79 DO BEGIN
       GOTOXY(i,1);WRITE('I');
      END;
      GOTOXY(1,1);WRITE('É');
      GOTOXY(1,2);WRITE('');
      GOTOXY(4,2);WRITE('N');GOTOXY(7,2);WRITE('O/M');
      GOTOXY(14,2);WRITE('X');GOTOXY(20,2);WRITE('Y');
      GOTOXY(27,2);WRITE('Z');GOTOXY(32,2);WRITE('F');
      GOTOXY(44,2);WRITE('Linea ');GOTOXY(56,2);WRITE('Columna ');
      GOTOXY(51,2);WRITE('  ');
      GOTOXY(51,2);WRITE(py);
      GOTOXY(65,2);WRITE('  ');
      GOTOXY(65,2);WRITE(px);
    END;
   END;
```

```
END;
PROCEDURE Pinta(Gene : Arre_12);
VAR
 j,k : INTEGER;
BEGIN
 {CLRSCR;}
 Pantalla('e',CORx,Cory);
 GOTOXY(1,1);
 FOR j:=1 to 7 DO
  FOR k:=1 to 37 DO BEGIN
   GOTOXY(k,j);
   IF k<6 THEN BEGIN
    TEXTCOLOR(LIGHTGRAY)
   END
   ELSE TEXTCOLOR(YELLOW);
   WRITE(gene[j,k]);
  END;
 FOR j:=1 to 6 DO
  FOR k:=34 TO 78 DO BEGIN
   GOTOXY(k,j);WRITE(' ');
  END;
END;
FUNCTION Lee (Jgo : JgoChar) : CHAR;
VAR
 C,C1,Aux : CHAR;
BEGIN
 Aux:=#144;
  C:=READKEY;
  IF KEYPRESSED
   THEN
    BEGIN
     C1:=READKEY;
     CASE C1 OF
      #59 : Aux:=F1;
      #60 : Aux:=F2;
      #61 : Aux:=F3;
      #62 : Aux:=F4;
      #63 : Aux:=F5;
      #64 : Aux:=F6;
      #65 : Aux:=F7;
      #66 : Aux:=F8;
      #67 : Aux:=F9;
      #68 : Aux:=F10;
      #71 : Aux:=Home;
      #72 : Aux:=Ar;
      #73 : Aux:=Pgup;
      #75 : Aux:=Izq;
      #77 : Aux:=Der;
      #79 : Aux:=Fin;
      #80 : Aux:=Ab;
      #81 : Aux:=Pgdn;
      #82 : Aux:=Ins;
      #83 : Aux:=Bo;
     END;
    END
   ELSE
    Aux:=C;
  IF (NOT (Aux IN Jgo)) THEN BEGIN
   Suena(50,100);
  END;
 Lee:=Aux;
END;
END.
```

Archivo MENUS_UT.PAS

```
UNIT Menus_ut;

INTERFACE
USES
 CRT,comp_1;

CONST

 Total = 4;
 Max  = 10;

TYPE
 Seleccion = ARRAY [1..9] OF STRING [15];
 Opciones  = RECORD
  Titulo   : STRING [15];
  Long     : INTEGER;
  Total    : INTEGER;
  Desplegado : Seleccion;
 END;
 Menus = ARRAY [1..8] OF Opciones;

VAR
 Menu     : Menus;
 Opcion   : Opciones;
 Despliega  : seleccion;
 Ancho,a,b,
 Mx,My    : INTEGER;
 Tecla    : CHAR;
 Abajo,Uno  : BOOLEAN;


 PROCEDURE Menu_opc(VAR Escogio : INTEGER);
 PROCEDURE Titula(i : INTEGER);
 PROCEDURE Mensaje(tipo : CHAR; Numero : INTEGER; Pnt : CHAR);

IMPLEMENTATION

PROCEDURE Titula(i : INTEGER);
VAR
 j : INTEGER;
BEGIN
 Pantalla('m',0,0);CLRSCR;
 FOR j:=1 TO Total DO BEGIN

  IF i<>j THEN BEGIN
   TEXTCOLOR(WHITE);TEXTBACKGROUND(GREEN);
  END
  ELSE BEGIN
   TEXTCOLOR(WHITE);TEXTBACKGROUND(CYAN);
  END;
  GOTOXY((Ancho*(j-1))+1,1);
  opcion:=menu[j];
  WRITE(opcion.titulo);
 END;

END;

PROCEDURE Muestra(i : INTEGER);
VAR
 j : INTEGER;
BEGIN
 Opcion:=Menu[i];
 WINDOW((ancho*(i-1))+2,3,(ancho*(i-1))+opcion.long+2,4+opcion.total);

 TEXTBACKGROUND(BLACK);
```

```
CLRSCR;
WINDOW((ancho*(i-1))+1,2,(ancho*(i-1))+opcion.long+1,3+opcion.total);
TEXTCOLOR(WHITE);TEXTBACKGROUND(LIGHTGRAY);
CLRSCR;


WINDOW((ancho*(i-1))+1,2,(ancho*(i-1))+opcion.long+1,4+opcion.total);
TEXTCOLOR(GREEN);
GOTOXY(1,1);WRITE(#201);
GOTOXY(opcion.long+1,1);WRITE(#187);
GOTOXY(1,2+opcion.total);WRITE(#200);
GOTOXY(opcion.long+1,2+opcion.total);WRITE(#188);
FOR j:=2 to opcion.long DO BEGIN
 GOTOXY(j,1);WRITE(#205);
 GOTOXY(j,2+opcion.total);WRITE(#205);
END;
FOR j:=2 to opcion.total+1 DO BEGIN
 GOTOXY(1,j);WRITE(#186);
 GOTOXY(opcion.long+1,j);WRITE(#186);
END;
TEXTCOLOR(WHITE);
WINDOW((ancho*(i-1))+1,3,(ancho*(i-1))+opcion.long+1,2+opcion.total);
FOR j:=1 TO Opcion.total DO BEGIN
 IF j = 1 then BEGIN
  TEXTCOLOR(WHITE),TEXTBACKGROUND(CYAN);
 END
 ELSE BEGIN
  TEXTCOLOR(WHITE);TEXTBACKGROUND(LIGHTGRAY);
 END;
GOTOXY(2,j);
WRITE(opcion.despliegado[j]);
 END;
END;

PROCEDURE Sombra(x1,y1,x2,y2:INTEGER; ctxt,ctbk:BYTE);

BEGIN
 TEXTBACKGROUND(Black);
 WINDOW(x2+1,y1+1,x2+1,y2+1);CLRSCR;
 WINDOW(x1+1,y2+1,x2+1,y2+1);CLRSCR;
 WINDOW(42,12,71,19);
 TEXTCOLOR(CTXT);
 TEXTBACKGROUND(ctbk);

END;
PROCEDURE Sombra0(x1,y1,x2,y2:INTEGER; ctxt,ctbk:BYTE);

BEGIN
 TEXTBACKGROUND(Blue);
 WINDOW(x2+1,y1+1,x2+1,y2+1);CLRSCR;
 WINDOW(x1+1,y2+1,x2+1,y2+1);CLRSCR;
 WINDOW(42,12,71,19);
 TEXTCOLOR(CTXT);
 TEXTBACKGROUND(ctbk);

END;

PROCEDURE Mensaje(tipo : CHAR ; Numero : integer;Prt : CHAR);
TYPE
 mt = ARRAY [1..8] OF STRING[29];
 mtx= ARRAY [1..30] OF mt;
VAR
 i,ant   : INTEGER;
 Mensajes : mtx;
 Errores  : mtx;
 texto    : mt;
 tecla    : CHAR;
```

```
BEGIN
 texto[1]:='          EDITOR          ';
 texto[2]:=' ARCHIVO: '+ ARCHIVO;
 texto[3]:=' F10  MENU                ';
 texto[4]:=' F9   Compila             ';
 texto[5]:=' F5   Ins Lin.            ';
 texto[6]:=' F4   Borra Lin.          ';
 texto[7]:='                          ';
 Mensajes[1]:=texto;
 texto[1]:='          MENU            ';
 texto[2]:=' ESC    Salir             ';
 texto[3]:=' Para seleccionar:        ';
 texto[4]:=' ENTER  Muestra opciones / ';
 texto[5]:='        Selecciona        ';
 texto[6]:=' '+#24+' '+#25+' Cambia Opc. Vert.';
 texto[7]:=' '+#27+' '+#26+' Cambia Opc. Hrz.';
 Mensajes[2]:=texto;
 texto[1]:='          CUIDADO         ';
 texto[2]:='                          ';
 texto[3]:='  El programa en memoria  ';
 texto[4]:='          SERA            ';
 texto[5]:='         BORRADO          ';
 texto[6]:='  Deseas continuar S/N    ';
 texto[7]:='                          ';
 Mensajes[3]:=texto;

 texto[1]:='          ERROR           ';
 texto[2]:='                          ';
 texto[3]:='                          ';
 texto[4]:='    CARACTER INVALIDO     ';
 texto[5]:='                          ';
 texto[6]:='                          ';
 texto[7]:='              < ESC >     ';
 Errores[1]:=texto;
 texto[1]:='          ERROR           ';
 texto[2]:='                          ';
 texto[3]:=' A00:                     ';
 texto[4]:='    INSTRUCCION G/M       ';
 texto[5]:='      INCORRECTA          ';
 texto[6]:='                          ';
 texto[7]:='              < ESC >     ';
 Errores[6]:=texto;
 texto[1]:='          ERROR           ';
 texto[2]:='                          ';
 texto[3]:=' A01:                     ';
 texto[4]:='     RADIO / M99          ';
 texto[5]:='     INCORRECTO           ';
 texto[6]:='                          ';
 texto[7]:='              < ESC >     ';
 Errores[7]:=texto;
 texto[1]:='          ERROR           ';
 texto[2]:='                          ';
 texto[3]:=' A02:                     ';
 texto[4]:='        VALOR X           ';
 texto[5]:='      INCORRECTO          ';
 texto[6]:='                          ';
 texto[7]:='              < ESC >     ';
 Errores[8]:=texto;
 texto[1]:='          ERROR           ';
 texto[2]:='                          ';
 texto[3]:=' A03:                     ';
 texto[4]:='        VALOR F           ';
 texto[5]:='      INCORRECTO          ';
 texto[6]:='                          ';
 texto[7]:='              < ESC >     ';
 Errores[9]:=texto;
 texto[1]:='          ERROR           ';
 texto[2]:='                          ';
```

```
texto[3]:=' A04:                    ';
texto[4]:='       VALOR Z           ';
texto[5]:='     INCORRECTO          ';
texto[6]:='                         ';
texto[7]:='               < ESC >';
Errores[10]:=texto;

texto[1]:='          ERROR          ';
texto[2]:='                         ';
texto[3]:=' A05:                    ';
texto[4]:='    FALTA INSTRUCCION    ';
texto[5]:='         M30             ';
texto[6]:='                         ';
texto[7]:='               < ESC >';
Errores[11]:=texto;
texto[1]:='          ERROR          ';
texto[2]:='                         ';
texto[3]:=' A06:                    ';
texto[4]:='    FALTA INSTRUCCION    ';
texto[5]:='         M03             ';
texto[6]:='                         ';
texto[7]:='               < ESC >';
Errores[12]:=texto;
texto[1]:='          ERROR          ';
texto[2]:='                         ';
texto[3]:=' A15:                    ';
texto[4]:='       VALOR Y           ';
texto[5]:='     INCORRECTO          ';
texto[6]:='                         ';
texto[7]:='               < ESC >';
Errores[21]:=texto;
texto[1]:='          ERROR          ';
texto[2]:='                         ';
texto[3]:=' A16:                    ';
texto[4]:='    FALTA INSTRUCCION    ';
texto[5]:='         M06             ';
texto[6]:=' (INDICACION RADIO FRESA) ';
texto[7]:='               < ESC >';
Errores[22]:=texto;
texto[1]:='          ERROR          ';
texto[2]:='                         ';
texto[3]:=' A17:                    ';
texto[4]:='  SUBPROGRAMA INCORRECTO  ';
texto[5]:='                         ';
texto[6]:='                         ';
texto[7]:='               < ESC >';
Errores[23]:=texto;
texto[1]:='          ERROR          ';
texto[2]:='                         ';
texto[3]:=' A18:                    ';
texto[4]:=' RECORRIDO DE COMPENSACION ';
texto[5]:='      DE FRESA           ';
texto[6]:='     MENOR A CERO        ';
texto[7]:='               < ESC >';
Errores[24]:=texto;
texto[1]:='          ERROR          ';
texto[2]:='                         ';
texto[3]:=' A19:                    ';
texto[4]:=' INSTRUCCION G92         ';
texto[5]:='       DUPLICADA         ';
texto[6]:='                         ';
texto[7]:='               < ESC >';
Errores[25]:=texto;
texto[1]:='          ERROR          ';
texto[2]:='                         ';
texto[3]:=' A20:                    ';
texto[4]:=' FALTA INSTRUCCION       ';
texto[5]:='         M05             ';
```

```
texto[6]:='                    ';
texto[7]:='              < ESC >';
Errores[26]:=texto;
WINDOW(41,11,72,20);
TEXTBACKGROUND(white);
CLRSCR;
WINDOW(41,11,72,21);
TEXTCOLOR(BLUE);
GOTOXY(1,1);WRITE('É');
GOTOXY(1,10);WRITE('È');
GOTOXY(32,10);WRITE('¼');
GOTOXY(32,1);WRITE('»');
FOR i:=2 TO 31 DO BEGIN
 GOTOXY(i,1);WRITE(#205);
 GOTOXY(i,10);WRITE(#205);
END;
FOR i:=2 TO 9 DO BEGIN
 GOTOXY(1,i);WRITE(#186);
 GOTOXY(32,i);WRITE(#186);
END;
WINDOW(42,12,71,19);
clrscr;
GOTOXY(1,1);
FOR i:=1 TO 7 DO BEGIN
 IF tipo = 'e' THEN BEGIN

  TEXTCOLOR(red);
  WRITELN(Errores[numero,i]);
 END
 ELSE BEGIN
  TEXTCOLOR(blue);
  WRITELN(Mensajes[numero,i]);
  ant.:=numero;
 END;
END;
IF tipo = 'a' THEN BEGIN
 Sombra(41,11,72,20,red,white);
 REPEAT
  tecla:=lee(term);
 UNTIL tecla=ESC;
 Sombra0(41,11,72,20,blue,white);
END;
Pantalla(pant,CorX,CorY);

END;

PROCEDURE Menu_ope(VAR Escogio : INTEGER);
VAR
 inicial : INTEGER;
BEGIN  {global}
 {FOR a:=1 TO 22 do
  FOR b:=1 to 40 DO
   GEMElo[a,b]:='x';}

 inicial:=escogio;
 Pantalla('e',0,0);
 Pinta(gemelo);
 Pantalla('i',Corz,Cory);
 Ancho:=ROUND(80/(total));
 Despliega[1]:='Cargar     ';
 Despliega[2]:='Nuevo      ';
 Despliega[3]:='Salvar     ';
 Despliega[4]:='Directorio ';
 Despliega[5]:='Imprime    ';
 Despliega[6]:='Salir      ';
 Opcion.Titulo:=' ARCHIVO';
 Opcion.long  :=12;
 Opcion.total :=6;
```

```
Opcion.desplegado:=despliega;
Menu[1]:=opcion;

Despliega[1]:='Ins Lin  F5';
Despliega[2]:='Borra Lin F4';
Despliega[3]:='Dimensiones ';
Despliega[4]:='Edita      ';
Opcion.Titulo:='EDITOR';
Opcion.long  :=13;
Opcion.total :=4;
Opcion.desplegado:=despliega;
Menu[2]:=opcion;
Despliega[1]:='Compilar';
Despliega[2]:='Transmitir';
Opcion.Titulo:='COMPILADOR';
Opcion.long  :=11;
Opcion.total :=2;
Opcion.desplegado:=despliega;
Menu[3]:=opcion;
Despliega[1]:='Sim. Cont';
Despliega[2]:='Sim. Paus';
Despliega[3]:='Acerca de';
Opcion.Titulo:='SIMULADOR';
Opcion.long  :=11;
Opcion.total :=3;
Opcion.desplegado:=despliega;
Menu[4]:=opcion;
titula(1);

Abajo:=False;
Mx:=1;
My:=1;
Mensaje('m',2,'m');
REPEAT
 {Pantalla('m',0,0);gotoxy(75,1);write(Mx,' ',My);
 Muestra(Mx);}
 opcion:=menu[Mx];
 despliega:=opcion.desplegado;
 IF ((inicial = 23) AND Uno) OR (inicial=31) THEN tecla:=ESC
 ELSE
  Tecla:=Lee(Mov_men);

 CASE Tecla OF
  Esc  : BEGIN
         IF abajo THEN BEGIN
          Titula(Mx);
          Pantalla('i',CorX,CorY);
          Pantalla('e',0,0);
          Pinta(gemelo);
          Titula(Mx);
          Tecla:='x';
          Abajo:=FALSE;
         END
         ELSE BEGIN
          Pinta(gemelo);
          Titula(0);
          Pantalla('i',CorX,CorY);
          Pantalla('e',0,0);
          Escogio:=inicial;
         END;
        END;
  Enter : BEGIN
         IF Abajo THEN BEGIN
          Abajo:=FALSE;
          Escogio:=(Mx*10)+My;
          Titula(0);
          Pantalla('i',CorX,CorY);
          Pantalla('e',0,0);
```

```
        Pinta(gemelo);
        tecla:=Esc;
       END
       ELSE BEGIN
        Abajo:=true;
        Pantalla('i',CorX,CorY);
        Pantalla('e',0,0);
        Pinta(gemelo);
        titula(Mx);Muestra(Mx);My:=1;
       END;
      END;
Der   : BEGIN
       Mx:=Mx+1;
       IF MX = Total+1 THEN MX:=1;
       IF NOT abajo THEN BEGIN
        Titula(Mx);
       END
       ELSE BEGIN
        Pantalla('i',CorX,CorY);
        Pantalla('e',0,0);
        Pinta(gemelo);
        titula(Mx);Muestra(Mx);My:=1;
       END;
      END;
Izq   : BEGIN
       Mx:=Mx-1;
       IF MX = 0 THEN MX:=total;
       IF NOT abajo THEN BEGIN
        Titula(Mx);
       END
       ELSE BEGIN
        Pantalla('i',CorX,CorY);
        Pantalla('e',0,0);
        Pinta(gemelo);
        titula(Mx);Muestra(Mx);My:=1;
       END;
      END;
Ab    : BEGIN
       IF Abajo THEN BEGIN
        GOTOXY(2,My);
        TEXTCOLOR(WHITE),TEXTBACKGROUND(LIGHTGRAY);
        WRITE(despliega[My]);
        TEXTCOLOR(WHITE);TEXTBACKGROUND(CYAN);
        My:=My+1;
        IF My = Opcion.total+1 THEN My:=1;
        GOTOXY(2,My);
        WRITE(despliega[My]);
        TEXTCOLOR(WHITE),TEXTBACKGROUND(LIGHTGRAY);
       END;
      END;
Ar    : BEGIN
       IF Abajo THEN BEGIN
        GOTOXY(2,My);

        TEXTCOLOR(WHITE),TEXTBACKGROUND(LIGHTGRAY);
        WRITE(despliega[My]);
        TEXTCOLOR(WHITE),TEXTBACKGROUND(CYAN);
        My:=My-1;
        IF My = 0 THEN My:=opcion.total;
        GOTOXY(2,My);
        WRITE(despliega[My]);
        TEXTCOLOR(WHITE),TEXTBACKGROUND(LIGHTGRAY);
       END;
      END;
F5    : BEGIN
       tecla:=ESC;
       Escogio:=21;
       Abajo:=FALSE;
```

```
        Titula(0);
        Pantalla('i',CorX,CorY);
        Pantalla('e',0,0);
        Pinta(gemelo);
        tecla:=Esc;
      END;
  F4  : BEGIN
        tecla:=ESC;
        Escogio:=22;
        Abajo:=FALSE;
        Titula(0);
        Pantalla('i',CorX CorY);
        Pantalla('e',0,0);
        Pinta(gemelo);
      END;
 END;
UNTIL tecla = esc;
END; {global}


END. {IMPLEMENTATION}
```

Archivo ARCHIVOS.PAS

```
UNIT Archivos;

INTERFACE
USES
  Menus_ut,DOS,CRT,Comp_1;
VAR
  Direc, Dirgrf;
  Dir_ini      : Anystr;
  Drv          : BYTE;

PROCEDURE Cargar(VAR Arre_fin : Arreglo);
PROCEDURE Directorio;
PROCEDURE Salvar(Arre_fin : Arreglo);
PROCEDURE TRANS(TRM : Arreglo;Long : INTEGER);
PROCEDURE Dim;
PROCEDURE Problemas(VAR Termino : CHAR);
PROCEDURE Presenta;
PROCEDURE Pro_Imp;

IMPLEMENTATION
 TYPE
 Anystr  = STRING [255];


 FUNCTION Completa (C:CHAR; A:INTEGER):AnyStr;
  VAR
    I  : INTEGER;
    Aux : AnyStr;
  BEGIN
    Aux:='';
    FOR I:=1 TO A DO Aux:=Aux+C;
    Completa:=Aux;
  END;

 PROCEDURE Captura (x,y,Long:INTEGER; Mensaje:Anystr; Jgo:JgoChar;
           VAR S : Anystr; VAR T :CHAR);
  VAR
    Cont,pcon,Li     : INTEGER;
    Respaldo,Aux     : AnyStr;
    Insrt, Pri,Acceso  : BOOLEAN;
    Car              : CHAR;
    Pto,expon,sig,sige  : BOOLEAN;

  BEGIN
    L:=long;
    GOTOXY (X,Y);
    textbackground(WHITE);textcolor(BLUE);
    Pto:=false;sig:=false;sige:=false;expon:=false;
    Insrt:=FALSE;Pri:=TRUE;
    Respaldo:=S; Cont:=LENGTH(S);
    if cont = 0 then pcon:=L else
      for i:=1 to cont do
      if s[i]='E' then pcon:=i;
    WRITE (Mensaje);
    TEXTBACKGROUND (BLUE); TEXTCOLOR (WHITE);
    WRITE (S,Completa(' ',Long-LENGTH(S)));
    {GOTOXY(72,2);WRITE (Ins Off);}
    REPEAT
      GOTOXY (X+LENGTH(Mensaje)+Cont,Y);
      Car:=Lee (Jgo+Edicion+Term);
      IF Car IN Jgo
```

```
THEN
  BEGIN
    IF Pri THEN
      BEGIN
        S:=';Cont:=LENGTH(S);
        GOTOXY (X+LENGTH(Mensaje),Y);WRITE (Completa (' ',Long));
        GOTOXY (X+LENGTH(Mensaje),Y);
        Pri:=FALSE;
      END;
    IF Cont+1<=Long
      THEN
        BEGIN
          IF Cont+1<=LENGTH(S)
            THEN
              BEGIN
                IF NOT(Insrt)
                  THEN
                    BEGIN
                      CASE Car of
                        '.' : BEGIN
                          (if (pto=true)or(cont+1 >= pcon) then begin
                          suena(50,100);cont:=cont-1;
                          end;)
                          if (not pto)and(cont+1<=pcon) then begin
                          CASE s[cont+1] of
                          '.' : pto:=false;
                          '+' : begin
                           if cont+1=1 then sig:=false;
                           if cont+1<>1 then sige:=false;
                          end;
                          '-' : begin
                           if cont+1=1 then sig:=false;
                           if cont+1<>1 then sige:=false;
                          end;
                          'E' :  begin
                              expon:= false;pcon:=L;
                               end;
                          end;
                          S[Cont+1]:=Car;
                          WRITE(Car);
                          pto:=true;
                          end else begin
                          suena(50,100);cont:=cont-1;
                          end;
                          END;
                        '+' :begin
                          if (cont=0)or(S[cont]='E') then begin
                          CASE s[cont+1] of
                          '.' : pto:=false;
                          '+' : begin
                           if cont+1=1 then sig:=false;
                           if cont+1<>1 then sige:=false;
                          end;
                          '-' : begin
                           if cont+1=1 then sig:=false;
                           if cont+1<>1 then sige:=false;
                          end;
                          'E' :  begin
                              expon:= false;pcon:=L;
                               end;
                          end;
                          S[Cont+1]:=Car;
                          WRITE(Car);
                          if cont = 0 then
                          sig:=true else sige:=true;
                          end else begin
                          suena(50,100);cont:=cont-1;
                          end;
```

```
        end;
    '-' :begin
        if (cont=0)or(S[cont]='E') then begin
        CASE s[cont+1] of
        ' ' : pto:=false;
        '+' : begin
        if cont+1=1 then sig:=false;
        if cont+1<>1 then sige:=false;
        end;
        '-' : begin
        if cont+1=1 then sig:=false;
        if cont+1<>1 then sige:=false;
        end;
        'E' :  begin
            expon:= false;pcon:=L;
            end;
        end;
        S[Cont+1]:=Car;
        WRITE(Car);
        if cont = 0 then
        sig:=true else sige:=true;
        end else begin
        suena(50,100);cont:=cont-1;
        end;
        end;
    'E' : if (not expon) and (cont<>0) then begin
            CASE s[cont+1] of
              ' ' : pto:=false;
              '+' : begin
              if cont+1=1 then sig:=false;
              if cont+1<>1 then sige:=false;
              end;
              '-' : begin
              if cont+1=1 then sig:=false;
              if cont+1<>1 then sige:=false;
              end;
              'E' :  begin
                  expon:= false;pcon:=L;
                  end;
            end;
          expon:=true;pcon:=cont+1;
           S[Cont+1]:=Car;
           WRITE(Car);
          end else begin
           suena(50,100);cont:=cont-1;
          end;
  end;
  if not(car in sigre) then
  begin
   CASE s[cont+1] of
   ' ' : pto:=false;
   '+' : begin
         if cont+1=1 then sig:=false;
         if cont+1<>1 then sige:=false;
         end;
   '-' : begin
         if cont+1=1 then sig:=false;
         if cont+1<>1 then sige:=false;
         end;
   'E' :  begin
          expon:= false;pcon:=L;
         end;
  end;
  S[Cont+1]:=Car;
  WRITE(Car);
  end;
 END
ELSE
```

```
BEGIN
 CASE Car of
  ' ': if (not pto)and (cont+1<=pcon) then begin
       Aux:=COPY(S,1,Cont)+Car+COPY(S,Cont+1,LENGTH(S)-Cont);
       IF LENGTH(Aux)>Long
         THEN S:=COPY(Aux,1,Long)
         ELSE S:=Aux;
       GOTOXY (X+LENGTH(Mensaje),Y);
       WRITE (S,Completa('_',long-LENGTH(S)));
       pto:=true;
       end else begin
       suena(50,100),cont:=cont-1;
       end;
  '+' :begin
       if (cont=0)or(S[cont]='E') then begin
       if (cont=0) and (not sig) then begin
        Aux:=COPY(S,1,Cont)+Car+COPY(S,Cont+1,LENGTH(S)-Cont);
        IF LENGTH(Aux)>Long
          THEN S:=COPY(Aux,1,Long)
          ELSE S:=Aux;
        GOTOXY (X+LENGTH(Mensaje),Y);
        WRITE (S,Completa('_',long-LENGTH(S)));
        sig:=true;
       end else begin
         suena(50,100);cont:=cont-1;
       end;
       if (not sige) and (S[cont]='E') then begin
         Aux:=COPY(S,1,Cont)+Car+COPY(S,Cont+1,LENGTH(S)-Cont);
         IF LENGTH(Aux)>Long
           THEN S:=COPY(Aux,1,Long)
           ELSE S:=Aux;
         GOTOXY (X+LENGTH(Mensaje),Y);
         WRITE (S,Completa('_',long-LENGTH(S)));
         sige:=true;
       end else begin
         suena(50,100);
       end;
       end else begin
        suena(50,100);cont:=cont-1;
       end;
       end;
  '-' :begin
       if (cont=0)or(S[cont]='E') then begin
       if (cont=0) and (not sig) then begin
        Aux:=COPY(S,1,Cont)+Car+COPY(S,Cont+1,LENGTH(S)-Cont);
        IF LENGTH(Aux)>Long
          THEN S:=COPY(Aux,1,Long)
          ELSE S:=Aux;
        GOTOXY (X+LENGTH(Mensaje),Y);
        WRITE (S,Completa('_',long-LENGTH(S)));
        sig:=true;
       end else begin
         suena(50,100);cont:=cont-1;
       end;
       if (not sige) and (S[cont]='E') then begin
         Aux:=COPY(S,1,Cont)+Car+COPY(S,Cont+1,LENGTH(S)-Cont);
         IF LENGTH(Aux)>Long
           THEN S:=COPY(Aux,1,Long)
           ELSE S:=Aux;
         GOTOXY (X+LENGTH(Mensaje),Y);
         WRITE (S,Completa('_',long-LENGTH(S)));
         sige:=true;
       end else begin
         suena(50,100);
       end;
       end else begin
        suena(50,100);cont:=cont-1;
       end;
```

```
                end;
              'E' : if (not expon) and (cont<>0) then begin
                    expon:=true;pcon:=cont+1;
                    Aux:=COPY(S,1,Cont)+Car+COPY(S,Cont+1,LENGTH(S)-Cont);
                    IF LENGTH(Aux)>Long
                      THEN S:=COPY(Aux,1,Long)
                      ELSE S:=Aux;
                    GOTOXY (X+LENGTH(Mensaje),Y);
                    WRITE (S,Completa('_',long-LENGTH(S)));
                  end else begin
                    suena(50,100);cont:=cont-1;
                    end;
              end;
              if not(car in sigre) then begin
               Aux:=COPY(S,1,Cont)+Car+COPY(S,Cont+1,LENGTH(S)-Cont);
               IF LENGTH(Aux)>Long
                 THEN S:=COPY(Aux,1,Long)
                 ELSE S:=Aux;
               GOTOXY (X+LENGTH(Mensaje),Y);
               WRITE (S,Completa('_',long-LENGTH(S)));
              end;
            END;
      END
    ELSE
      BEGIN
        CASE Car of
              '' : BEGIN

                  if (not pto)and(cont+1<=pcon) then begin
                  S:=S+Car;WRITE (Car);
                  pto:=true;
                  end else begin
                    suena(50,100);cont:=cont-1;
                  end;
                  END;
              '+' :begin
                  if (cont=0)or(S[cont]='E') then begin
                  S:=S+Car;
                  WRITE(Car);
                  if cont = 0 then
                  sig:=true else sige:=true;
                  end else begin
                  suena(50,100);cont:=cont-1;
                  end;
                 end;
              '-' :begin
                  if (cont=0)or(S[cont]='E') then begin
                  S:=S+Car;
                  WRITE(Car);
                  if cont = 0 then
                  sig:=true else sige:=true;
                  end else begin
                  suena(50,100);cont:=cont-1;
                  end;
                  end;
              'E' : if (not expon) and (cont<>0) then begin
                    expon:=true;pcon:=cont+1;
                    S:=S+Car;WRITE (Car);
                  end else begin
                    suena(50,100);cont:=cont-1;
                  end;
        end;
        if not(car in sigre) then Begin
        S:=S+Car;WRITE (Car);
        end;
      END;
  Cont:=Cont+1;
END;
```

```
  END
 ELSE
  BEGIN
    Pri:=FALSE;
    CASE Car OF
      Backspc : IF Cont>0 THEN
                BEGIN
                 CASE s[cont] of
                       ' ' : pto:=false;
                       '+' : begin
                             if cont=1 then sig:=false;
                             if cont<>1 then sige:=false;
                             end;
                       '-' : begin
                             if cont=1 then sig:=false;
                             if cont<>1 then sige:=false;
                             end;
                       'E' : begin
                             expon:=false;pcon:=1;
                             end;
                 end;
                 if s[cont]<>'E' then pcon:=pcon-1;
                 DELETE(S,Cont,1);
                 GOTOXY(X+LENGTH(Mensaje),Y);
                 WRITE (S,Completa('_',Long-LENGTH(S)));
                 Cont:=Cont-1;
                END;
      Bo      : IF Cont<=LENGTH(S) THEN
                BEGIN
                 CASE s[cont+1] of
                       ' ' : pto:=false;
                       '+' : begin
                             if cont+1=1 then sig:=false;
                             if cont+1<>1 then sige:=false;
                             end;
                       '-' : begin
                             if cont+1=1 then sig:=false;
                             if cont+1<>1 then sige:=false;
                             end;
                       'E' : begin
                             expon:=false;pcon:=1;
                             end;
                 end;
                 if s[cont+1]<>'E' then pcon:=pcon-1;
                 DELETE(S,CONT+1,1);
                 GOTOXY (X+LENGTH(Mensaje),Y);
                 WRITE (S,Completa('_',Long-LENGTH(S)));
                END;
      Der     : IF Cont+1<=LENGTH(S) THEN Cont:=Cont+1;
      Izq     : IF Cont>0 THEN Cont:=Cont-1;
      Home    : Cont:=0;
      Fin     : Cont:=LENGTH(S);
      BoLin   : BEGIN
                pto:=false;sig:=false;sige:=false;expon:=false;
                pcon:=1;
                Cont:=0;
                S:='';
                GOTOXY(X+LENGTH(Mensaje),Y);
                WRITE (Completa('_',Long));
               END;
      (Ins    : BEGIN
                Insrt:=NOT(Insrt) FALSE;
                GOTOXY(72,2);
                IF Insrt THEN WRITE ('Ins On ')
                         ELSE WRITE ('Ins Off');
               END;)
    END;
  END;
```

```
 UNTIL Car IN Term;
 T:=Car;
 IF Car=Esc THEN S:=Respaldo;
 TEXTBACKGROUND (WHITE);TEXTCOLOR(BLUE);
 GOTOXY (X+LENGTH(Mensaje),Y);
 WRITE (S,':Long-LENGTH(S));
END;
PROCEDURE Sombra(x1,y1,x2,y2:INTEGER; ctxt,ctbk:BYTE);

BEGIN
 TEXTBACKGROUND(Black);
 WINDOW(x2+1,y1+1,x2+1,y2+1);CLRSCR;
 WINDOW(x1+1,y2+1,x2+1,y2+1);CLRSCR;
 WINDOW(42,12,71,19);
 TEXTCOLOR(CTXT);
 TEXTBACKGROUND(ctbk);
END;

PROCEDURE Sombra0(x1,y1,x2,y2:INTEGER; ctxt,ctbk:BYTE);

BEGIN
 TEXTBACKGROUND(Blue);
 WINDOW(x2+1,y1+1,x2+1,y2+1);CLRSCR;
 WINDOW(x1+1,y2+1,x2+1,y2+1),CLRSCR;
 WINDOW(42,12,71,19);
 TEXTCOLOR(CTXT);
 TEXTBACKGROUND(ctbk);
END;

PROCEDURE Ventana;
VAR
 i : INTEGER;

BEGIN
 WINDOW(41,11,72,20);
 TEXTBACKGROUND(white);
 CLRSCR;
 WINDOW(41,11,72,21);
 TEXTCOLOR(BLUE);
 GOTOXY(1,1);WRITE('É');
 GOTOXY(1,10);WRITE('È');
 GOTOXY(32,10);WRITE('¼');
 GOTOXY(32,1);WRITE('»');
 FOR i:=2 TO 31 DO BEGIN
  GOTOXY(i,1);WRITE(#205);
  GOTOXY(i,10);WRITE(#205);
 END;
 FOR i:=2 TO 9 DO BEGIN
  GOTOXY(1,i);WRITE(#186);
  GOTOXY(32,i);WRITE(#186);
 END;
 WINDOW(42,12,71,19);
 clrscr;
 GOTOXY(1,1);

 Sombra(41,11,72,20,red,white);
 {
 Pantalla(pri,CorX,CorY);}
END;


PROCEDURE Cargar(VAR Arre_fin : Arreglo);
VAR
 Nombre      : Anystr;
 Term        : CHAR;
 Var_Arch    : FILE OF Arreglo;
 F           : SEARCHREC;
```

```
i,Ecode     : INTEGER;

BEGIN
 term:=' ';
 Ventana;
 Nombre:='';
 TEXTCOLOR(BLUE);
 REPEAT
  Captura(1,2,8,'ARCHIVO : ',Letras,NOMBRE,Term);
 UNTIL (Term = Esc) OR ((Term = Enter) AND (LENGTH(Nombre) <> 0));

 {nombre:='fresa_05';}

 IF Term <> Esc THEN BEGIN
  FOR i:=1 TO LENGTH(Nombre) DO
  Nombre[i]:=UPCASE(Nombre[i]);
  Nombre:=Nombre+'.FRS';
  FINDFIRST(Nombre,ARCHIVE,F);
  Ecode:=DosError;
  {IF Ecode = 2 THEN BEGIN
   WHILE (Ecode <> 18) AND (Ecode <> 0)  DO BEGIN
    FINDNEXT(F),Ecode:=Doserror;write(ecode);readln;
    IF Ecode = 0 THEN BEGIN
     ASSIGN(Var_Arch,Nombre),
     RESET(Var_Arch);
     READ(Var_Arch,Arre_Fin);
     CLOSE(Var_Arch);
     Archivo:='';
     FOR i:=1 TO LENGTH(Nombre)-4 DO BEGIN
      Archivo:=Archivo+Nombre[i];
     END;
    END;
   END;
   write(Ecode);readln;


   END;}
   IF Ecode = 18 THEN BEGIN
    TEXTCOLOR(RED);
    GOTOXY(2,4),WRITE('Archivo NO encontrado');
    TEXTCOLOR(BLUE);
    REPEAT
    UNTIL KEYPRESSED;
   END;
   IF Ecode = 0  THEN BEGIN
    ASSIGN(Var_Arch,Nombre);
    RESET(Var_Arch);
    READ(Var_Arch,Arre_Fin);
    CLOSE(Var_Arch);
    Archivo:='';
    FOR i:= 1 TO LENGTH(Nombre)-4 DO BEGIN
     Archivo:=Archivo+Nombre[i];
    END;
   END;
  END;
  Sombra0(41,11,72,20,BLUE,white);;
END;

PROCEDURE Directorio;
VAR
 Nombre       : Anystr;
 i,Ocode,Ecode  : INTEGER;
 Term,termina   : CHAR;
 F            : SEARCHREC;
 Existe       : BOOLEAN;
 X,Y,columna    : INTEGER;
 Nom_arch     : STRING;
BEGIN
```

```
Existe:=TRUE;
Ventana;
Nombre:=Direc;
TEXTCOLOR(BLUE);
REPEAT
  Capture(1,1,20,'DIR : ',STRINGS,NOMBRE,Term);
UNTIL (Term = Esc) OR ((Term = Enter));
IF Term <> Esc THEN BEGIN
 FOR i:=1 TO LENGTH(Nombre) DO
 Nombre[i]:=UPCASE(Nombre[i]);

 { I$- }
  Mkdir(Nombre[1]+':\'+'\'+'LPC3070');
  Iocode:=Ioresult;
  IF (IOCode <> 0) THEN EXISTE:=FALSE
  ELSE RmDir(Nombre[1]+':\'+'\'+'LPC3070');
 { I$+ }
 { $I- }
 IF NOT Existe THEN BEGIN
   GOTOXY(1,3);TEXTCOLOR(RED);
   WRITE('ERROR EN LA UNIDAD O');
   GOTOXY(1,5);
   WRITE('DISCO PROTEGIDO');
   REPEAT
   UNTIL KEYPRESSED;
   TEXTCOLOR(BLUE);
 END
 ELSE BEGIN
  CHDIR(Nombre);
  IF Doserror = 3 THEN BEGIN
   GOTOXY(1,3);TEXTCOLOR(RED);
   WRITE('Directorio no encontrado');
   TEXTCOLOR(BLUE);
   REPEAT
   UNTIL KEYPRESSED;
  END
  ELSE BEGIN
   Direc:=Nombre;
   CLRSCR;
   TEXTCOLOR(BLUE);
   GOTOXY(1,1);WRITE('DIR : ',Nombre);
   FINDFIRST('*.FRS',ARCHIVE,F);
   Ecode:=DosError;
   Columna:=1;
   IF (Ecode = 0) THEN BEGIN
    X:=1;Y:=2;
    REPEAT
     Nom_arch:='';
     FOR i:=1 TO LENGTH(F.NAME) - 4 DO
      Nom_arch:=Nom_arch + F.Name[i];
     GOTOXY(X,Y);WRITE(Nom_arch);
     Y:=Y+1;
     IF Y = 9 THEN BEGIN
      Columna:=Columna+1;
      IF Columna = 4 THEN BEGIN
       Columna:=1;
       REPEAT
        Termina:=Lee(mueve);
       UNTIL termina=ENTER;
       termina:=Esc;
       CLRSCR;
       GOTOXY(1,1);WRITE('DIR : ',Nombre);
      END;
      X:=1 + ((Columna-1) * 9);
      Y:=2;
     END;
     FINDNEXT(F);
     Ecode:=DosError;
```

```
    UNTIL Ecode = 18;
    REPEAT
     Termina:=Lee(mueve);
    UNTIL Termina = ENTER;
    Termina:=Esc;
   END
   ELSE BEGIN
    GOTOXY(1,3);TEXTCOLOR(RED);
    WRITE('No hay Archivos');
    REPEAT
    UNTIL KEYPRESSED;
    TEXTCOLOR(BLUE);
   END;
  END;


 END;
 Sombra0(41,11,72,20,BLUE,white);;
END;
PROCEDURE Salvar(Arre_fin : Arreglo);
VAR
 Nombre      : Anystr;
 Term        : CHAR;
 F           : SEARCHREC;
 Existe,Acceso,
 Sal         : BOOLEAN;
 Llocode     : INTEGER;
 Mem         : LONGINT;
 Var_Arch    : FILE OF ARREGLO;

BEGIN
 Existe:=false;
 Acceso:=TRUE;Sal:=TRUE;
 drv:=ORD(Direc[1])-64;{write(drv);readln;}
 {drv:=2;}
 GETDIR(Drv,Direc);
 CHDIR(Direc);
 Ventana;
 TEXTCOLOR(BLUE);
 GOTOXY(1,1);WRITE('DIR ',Direc);
 Nombre:='Archivo';
 REPEAT
  REPEAT
   Captura(1,2,8,'Nombre del archivo : ',Letras,NOMBRE,Term);
  UNTIL (Term = Esc) OR ((Term = Enter) AND (LENGTH(Nombre) <> 0));
  IF term <> Esc THEN BEGIN
   FOR i:=1 TO LENGTH(Nombre) DO
   Nombre[i]:=UPCASE(Nombre[i]);

   Sal:=TRUE;
   Acceso:=TRUE;
   {$I-}
   Mem:=DISKSIZE(Drv);
   Mkdir('LPC3070');
   Iocode:=Ioresult;
   IF (IOCode <> 0) OR (Mem < 10250) THEN Acceso:=FALSE
   ELSE RmDir('LPC3070');
   {write(iocode,' ',mem);readln;}
   {$I+}
   IF Acceso THEN BEGIN
    FINDFIRST('Direc+')*.FRS',ARCHIVE,F);
    Nombre:=Nombre+'.FRS';
    {writeln(Nombre);
    write(F.name,Doserror);readln;}
    IF F.NAME = Nombre THEN Existe := TRUE;
```

```
    IF NOT Existe THEN
    WHILE (Doserror <> 18) AND (NOT Existe) DO BEGIN
     FINDNEXT(F);
     {WRITELn(Nombre);
     Write(F.name,doserror);readln;}
     IF F.NAME = Nombre THEN Existe:=TRUE;
    END;
    IF Existe THEN BEGIN
     TEXTCOLOR(RED);
     GOTOXY(1,4);WRITE('EL ARCHIVO YA EXISTE, QUIERES SALVAR ? (S/N) ');
     TEXTCOLOR(BLUE);
     Term:=Lee(SiNO);
     IF (Term = 'S') OR (term = 's') THEN Existe:=FALSE;
      TEXTCOLOR(BLUE);
    END;
    IF NOT Existe THEN BEGIN
     {write('usig ',nombre);readln;}
     ASSIGN(Var_Arch,{Direc+}Nombre);
     REWRITE(Var_Arch);
     WRITE(Var_Arch,Arre_fin);
     CLOSE(Var_Arch);
     Archivo:='';
     {writeln(nombre);}
     FOR i:= 1 TO LENGTH(Nombre)-4 DO BEGIN
      Archivo:=Archivo+Nombre[i];
      {write(archivo[i],';readln;}
     END;
     {write(archivo);readln;}
    END;
   END;
   IF NOT Acceso THEN BEGIN
    TEXTCOLOR(RED);
    GOTOXY(1,3);
    WRITELN('Error en la unidad o');
    GOTOXY(1,5);
    WRITELN('Disco protegido');
    TEXTCOLOR(BLUE);
    WRITELN('Deseas reintentar ? (S/N)');
    Term:=Lee(SINO);
    Term:=UPCASE(Term);
    IF Term = 'N' THEN Sal:=TRUE;
    IF Term = 'S' THEN BEGIN
     Sal:=FALSE;
     GOTOXY(1,3);
     WRITELN('                ');
     WRITELN('                ');
    END;
   END;
  END;
 UNTIL Sal;
 Sombra0(41,11,72,20,BLUE,white);
END;

PROCEDURE TRANS(TRM : Arreglo;Long : INTEGER);
CONST
 SP = #32;
 CR = #13;
 LF = #10;
 AP = #96;
 IOFort = 1;
VAR
 i,j    : INTEGER;
 Linea  : STRING [32];
 Statport: BYTE;
 {Arr_Tr : ARRAY [0..225] OF STRING[32];}
 Regs   : Registers;

FUNCTION Bits(Siete,Seis,Cinco,Cuatro,Tres,Dos,Uno,Cero:Byte):Byte;
```

```
BEGIN
        Bits:=Cero + Uno shl 1 + Dos shl Dos + Tres shl Tres + Cuatro shl Cuatro +
                Cinco shl 5 + Seis shl 6 + Siete shl 7
END; {Bits}

PROCEDURE InitCom(NumPort,InitParam:Byte);

BEGIN
        WITH Regs DO BEGIN
        AH:=$00;
    AL:=InitParam;
        DX:=PRED(NumPort);
    Intr($14,Regs);
    StatPort:=AH;
  END;
END; {InitCom}

PROCEDURE WriteCarCOM(NumPort:Byte;Car:Char);

BEGIN
        WITH Regs DO BEGIN
        AH:=$01;
    AL:=ORD(Car);
        DX:=PRED(NumPort);
    Intr($14,Regs);
    StatPort:=AH;
  END;
END;{WriteCar}

PROCEDURE WriteStrCOM(NumPort:Byte;Linea:String);
VAR i:Byte;
BEGIN
  FOR i:=1 TO Length(Linea) DO
        WriteCarCOM(NumPort,Linea[i])
END; {WriteStrCOM}

BEGIN
 Ventana;
 TEXTCOLOR(BLUE);
 GOTOXY(8,1);WRITE('TRANSMISION');
 GOTOXY(3,3);WRITE('Antes de iniciar, teclee');
 GOTOXY(3,4);WRITE('O66 INP INI');
 GOTOXY(3,5);WRITE('Presione cualquier tecla');
 REPEAT
 UNTIL KEYPRESSED;
 InitCOM(IOPort,Bits(0,1,0,1,0,1,1,0));
 {writeln('STatus',Statport),readln;}
 IF Statport <> 97 THEN BEGIN
  TEXTCOLOR(RED);
  GOTOXY(3,6);WRITE('ERROR DE TRANSMISION');
  REPEAT
  UNTIL KEYPRESSED;
  TEXTCOLOR(BLUE);
 END
 ELSE BEGIN
  {Pantalla('e',Corx,cory);
  GOTOXY(1,10);}
  Linea:='%'+CR+LF;
  {write(linea,length(linea));readln;}
  WriteSTRCOM(IOport,Linea);
  Linea:='  '+'N'+AP+SP+'G'+AP+'  '+'2'+'  '+AP+'  '+'F'+''+AP+
       '  '+'Z'+SP+AP+SP+'F+'  '+CR+LF;
  {write(linea,length(linea));readln;}
  WriteSTRCOM(IOport,Linea);
  j:=1;
  REPEAT
   i:=1;
   Linea:=' ';
```

```
  REPEAT
   CASE i OF
    5 : i:=i+1;
    9 : i:=i+1;
    16 : i:=i+1;
    23 : i:=i+1;
    29 : i:=i+1;
   END;
   IF trm[j,i]<>'*' THEN
    Linea:=Linea+trm[j,i]
   ELSE Linea:=Linea+' ';
   i:=i+1;
  UNTIL i = 34;
  Linea:=Linea+CR+LF;
  {writeln(linea,length(linea));readln;}
  WriteSTRCOM(IOport,Linea);
  j:=j+1;
 UNTIL (j = Long+1) OR ((Statport) AND ($80)=$80);
 linea:='  '+'M'+T;

 WriteSTRCOM(IOport,Linea);
 {writeln(Linea,Length(linea));}
 IF j=Long+1 THEN
  WRITE('Transmision completa')
 ELSE WRITE('ERROR Transmision incompleta');
 REPEAT
 UNTIL KEYPRESSED;
END;
sombra0(41,11,72,20,BLUE,white);
END;


PROCEDURE Dim;

VAR
Straux,Strlargo,Strdiam,
StrXo,StrYo,StrZo,
Strancho,strAlto : Anystr;
Ps,err          : INTEGER;
Term            : CHAR;
BEGIN
Ventana;
STR(Largo,Strlargo);
STR(Alto,Stralto);
STR(Ancho,Strancho);
STR(Xus,StrXo);
STR(Yus,StrYo);
STR(Zus,StrZo);
TEXTCOLOR(BLUE);
GOTOXY(3,1);WRITE('Longitud : ');
GOTOXY(3,2);WRITE('Ancho   : ');
GOTOXY(3,3);WRITE('Altura  : ');
GOTOXY(3,4);WRITE('Xo      : ');
GOTOXY(3,5);WRITE('Yo      : ');
GOTOXY(3,6);WRITE('Zo      : ');
{GOTOXY(14,1);WRITE(Strlargo);
GOTOXY(14,3);WRITE(Strancho);
GOTOXY(14,5);WRITE(StrAlto);
GOTOXY(14,7);WRITE(Strdiam);}
GOTOXY(7,8);WRITE(' F1 Para Terminar');
Ps:=1;
term:=' ';
REPEAT
 CASE Ps OF
  1 : BEGIN
      Straux:=Strlargo;
      Capturn(3,1,5,'Longitud : ',Enteros,straux,Term);
      Strlargo:=Straux;
```

```
      IF Term = Ar THEN ps:=6;
      IF (Term = Ab) OR (Term = ENTER) THEN ps:=2;
     END;
   2 : BEGIN
      Straux:=Strancho;
      Captura(3,2,5,'Ancho   : ',Enteros,straux,Term);
      Strancho:=Straux;
      IF Term = Ar THEN ps:=1;
      IF (Term = Ab) OR (Term = ENTER) THEN ps:=3;
     END;
   3 : BEGIN
      Straux:=Stralto;
      Captura(3,3,5,'Altura  : ',Enteros,straux,Term);
      Stralto:=Straux;
      IF Term = Ar THEN ps:=2;
      IF (Term = Ab) OR (Term = ENTER) THEN ps:=4;

     END;
   4 : BEGIN
      Straux:=StrXo;
      Captura(3,4,5,'Xo      : ',Enteros,straux,Term);
      StrXo:=Straux;
      IF Term = Ar THEN ps:=3;
      IF (Term = Ab) OR (Term = ENTER) THEN ps:=5;
     END;
   5 : BEGIN
      Straux:=StrYo;
      Captura(3,5,5,'Yo      : ',Enteros,straux,Term);
      StrYo:=Straux;
      IF Term = Ar THEN ps:=4;
      IF (Term = Ab) OR (Term = ENTER) THEN ps:=6;
     END;
   6 : BEGIN
      Straux:=StrZo;
      Captura(3,6,5,'Zo      : ',Enteros,straux,Term);
      StrZo:=Straux;
      IF Term = Ar THEN ps:=5;
      IF (Term = Ab) THEN ps:=1;
      IF Term = ENTER THEN Term:=F1;
      END ;


  END;
 IF Term = F1 THEN BEGIN
  Ps:=0;
  VAL(Strlargo,Largo,err);
  VAL(StrAncho,Ancho,err),
  VAL(StrAlto,Alto,err);
  VAL(StrXo,Xus,err);
  VAL(StrYo,Yus,err);
  VAL(StrZo,Zus,err);
  IF (Largo   > 20000) OR (Largo<=0) THEN ps:=1;
  IF (Ancho   > 10000) OR (Ancho<=0) THEN ps:=2;
  IF (Alto    > 20000) OR (Ancho<=0) THEN ps:=3;
  IF ((Xus)   > 20000) OR (Xus<0)    THEN ps:=4;
  IF ((Yus)   > 10000) OR (Yus<0)    THEN ps:=5;
  IF ((Zus)   > 20000) OR (Zus<0)    THEN ps:=6;
  IF ps<>0 THEN BEGIN
   Term:=' ';
   TEXTCOLOR(RED);
   GOTOXY(7,8),WRITE('Valor  Incorrecto ');
   TEXTCOLOR(BLUE);
   REPEAT
   UNTIL KEYPRESSED;
   GOTOXY(7,8),WRITE('F1 Para terminar ');
  END;
 END;
 UNTIL Term = F1;
```

```
SOMBRA0(41,11,72,20,BLUE,white);
END;

PROCEDURE Problemas(VAR Termino : CHAR) ;
VAR
 Term    : CHAR;
 Dir     : Anystr;
BEGIN
 Ventana;
 Dir:=Dirgrf;
 GOTOXY(3,1);WRITE('El Directorio :');
 GOTOXY(3,2);WRITE(Dir_ini);
 GOTOXY(3,3);WRITE('NO contiene los drivers');
 GOTOXY(3,4);WRITE('graficos. Favor de ');
 GOTOXY(3,5);WRITE('copiarlos, o indicar el');
 GOTOXY(3,6);WRITE('directorio adecuado');
 REPEAT
  Captura(3,7,15,'Dir : ',Strings,dir,term);
 UNTIL (Term = ENTER) OR (Term = ESC);
 Termino:=Term;
 Dirgrf:=Dir;
 SOMBRA0(41,11,72,20,BLUE,white);
END;
PROCEDURE Presenta;
 VAR
  Termina : CHAR;
BEGIN
 Ventana;
 TEXTCOLOR(BLUE);
 GOTOXY(3,1);WRITE(' Universidad Panamericana');
 GOTOXY(3,3);WRITE(' Programa realizado por: ');
 GOTOXY(3,4);WRITE(' Leonel Pernudi Contreras');
 GOTOXY(3,6);WRITE(' Asesorado por:');
 GOTOXY(3,7);WRITE(' Lic. Alejandro Gonzalez');
 GOTOXY(3,8);WRITE('                < ESC >');
 REPEAT
  Termina:=Lee(Term);
 UNTIL Termina= ESC;
 SOMBRA0(41,11,72,20,BLUE,white);
END;
PROCEDURE Pro_Imp;
VAR
 Termina   : CHAR,
BEGIN
 Ventana;
 TEXTCOLOR(RED);
 GOTOXY(3,1);WRITE('Problemas para imprimir');
 GOTOXY(3,3);WRITE('   La impresora');
 GOTOXY(3,5);WRITE('   NO responde');
 GOTOXY(3,7);WRITE('                < ESC >');
 REPEAT
  Termina:=Lee(term);
 UNTIL (Termina = ESC);
 SOMBRA0(41,11,72,20,BLUE,white);
 TEXTCOLOR(BLUE);
END;
END.
```

Anexo 2

Archivo CMDS.PAS

```
UNIT CMDS;

INTERFACE

USES
 Vectores,CRT,Comp_1;
  -
TYPE
  Comandos = RECORD
   Nom : STRING [3];
   Lim : STRING [4];
   Dis : STRING [4];
   Vx  : INTEGER;
   Vy  : INTEGER;
   Vz  : INTEGER;
   Vf  : INTEGER;
  END;
  Com_DAT = ARRAY [1..40] OF Comandos;
  Nombre  = STRING [3];
VAR
  DAT    : Com_DAT;
  Valores : Comandos;

  PROCEDURE inf;
  FUNCTION Busca    (Orden : nombre): INTEGER;
  PROCEDURE Limita  (a : INTEGER;VAR lmte : Anystr);
  PROCEDURE Checa   (VAR Numero : INTEGER;
                     VAR Alarma : INTEGER; Arre_fin : Arreglo;
                     Tot : INTEGER);

IMPLEMENTATION
  PROCEDURE Inf;

    PROCEDURE Borra;
     BEGIN
      Valores.lim:='    ';
      Valores.dis:='    ';
     END;
   BEGIN
   Borra;
   Valores.Nom :=' 00';
   Valores.Lim[4]:='F';
   DAT[1]:=valores;
   Borra;
   Valores.Nom :=' 01';
   DAT[2]:=valores;
   Borra;
   Valores.Nom :=' 02';
   DAT[3]:=valores;
   Borra;
   Valores.Nom :=' 03';
   DAT[4]:=valores;
   Borra;
   Valores.Nom :=' 04';
   Valores.lim[2]:='Y';
   Valores.lim[3]:='Z';
   Valores.lim[4]:='F';
   DAT[5]:=valores;
   Borra;
   Valores.Nom :=' 21';
```

```
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[6]:=valores;
Borra;
Valores.Nom :=' 25';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.dis[4]:='L';
DAT[7]:=valores;
Borra;
Valores.Nom :=' 27';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.dis[4]:='L';
DAT[8]:=valores;
Borra;
Valores.Nom :=' 40';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[9]:=valores;
Borra;
Valores.Nom :=' 45';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[10]:=valores;
Borra;
Valores.Nom :=' 46';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[11]:=valores;
Borra;
Valores.Nom :=' 47';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[12]:=valores;
Borra;
Valores.Nom :=' 48';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[13]:=valores;
Borra;
Valores.Nom :=' 64';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[14]:=valores;
Borra;
Valores.Nom :=' 72';
DAT[15]:=valores;
Borra;
Valores.Nom :=' 73';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
```

```
DAT[35]:=valores;
Borra;
Valores.Nom :=' 74';
Valores.lin[1]:='X';
Valores.dia[2]:='X';
DAT[16]:=valores;
Borra;
Valores.Nom :=' 81';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
DAT[17]:=valores;
Borra;
Valores.Nom :=' 82';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
DAT[18]:=valores;
Borra;
Valores.Nom :=' 83';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
DAT[19]:=valores;
Borra;
Valores.Nom :=' 84';
Valores.lin[1]:='X';
Valores.dia[2]:='X';
DAT[20]:=valores;
Borra;
Valores.Nom :=' 85';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
DAT[21]:=valores;
Borra;
Valores.Nom :=' 89';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
DAT[22]:=valores;
Borra;
Valores.Nom :=' 90';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
Valores.lin[3]:='Z';
Valores.lin[4]:='F';
DAT[23]:=valores;
Borra;
Valores.Nom :=' 91';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
Valores.lin[3]:='Z';
Valores.lin[4]:='F';
DAT[24]:=valores;
Borra;
Valores.Nom :=' 92';
Valores.lin[4]:='F';
DAT[25]:=valores;
Borra;
Valores.Nom :='M00';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
Valores.lin[3]:='Z';
Valores.lin[4]:='F';
DAT[26]:=valores;
Borra;
Valores.Nom :='M03';
Valores.lin[1]:='X';
Valores.lin[2]:='Y';
Valores.lin[3]:='Z';
Valores.lin[4]:='F';
DAT[27]:=valores;
```

```
Borra;
Valores.Nom :='M05';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[28]:=valores;
Borra;
Valores.Nom :='M06';
Valores.dia[1]:='D';
Valores.dia[2]:='S';
Valores.dia[4]:='T';
DAT[29]:=valores;
Borra;
Valores.Nom :='M17';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[30]:=valores;
Borra;
Valores.Nom :='M30';
Valores.lim[1]:='X';
Valores.lim[2]:='Y';
Valores.lim[3]:='Z';
Valores.lim[4]:='F';
DAT[31]:=valores;
Borra;
Valores.Nom :='M98';
Valores.lim[4]:='F';
DAT[34]:=valores;
Borra;
Valores.Nom :='M99';
Valores.dia[1]:='T';
Valores.dia[2]:='F';
Valores.dia[3]:='S';
Valores.lim[4]:='F';
DAT[32]:=valores;
Borra;
Valores.Nom :='  ';
DAT[33]:=valores;
END;
FUNCTION Busca (Orden : nombre): INTEGER;
 VAR
   i      : INTEGER;
   Encontre :  BOOLEAN;
BEGIN
 i:=1;Encontre:=false;Busca:=0;
 REPEAT
  valores:=DAT[i];
  IF valores.nom = Orden THEN Encontre:=true;
  i:=i+1;
 UNTIL (Orden = valores.Nom) OR (i = 36);
 IF encontre THEN Busca:=i-1;
END;
PROCEDURE Limita (a : INTEGER;VAR lmte : Anystr);
VAR
 i : INTEGER;

BEGIN
 valores:=DAT[a];
 IF valores.lim[1]='X' THEN
   FOR i:=10 TO 15 DO lmte[i]:='*'
 ELSE IF (lmte[15]='*') THEN FOR i:=10 TO 15 DO lmte[i]:=' ';
 IF valores.lim[2]='Y' THEN
   FOR i:=17 TO 21 DO lmte[i]:='*'
 ELSE IF lmte[21]='*' THEN FOR i:=17 TO 21 DO lmte[i]:=' ';
 IF valores.lim[3]='Z' THEN
```

```
  FOR i:=23 TO 28 DO lmte[i]:='*'
 ELSE IF lmte[28]='*' THEN FOR i:=23 TO 28 DO lmte[i]:=' ';
 IF valores.lim[4]='F' THEN
   FOR i:=30 TO 33 DO lmte[i]:='*'
 ELSE IF lmte[33]='*' THEN FOR i:=30 TO 33 DO lmte[i]:=' ';
 IF valores.dis[1]<>'' THEN
   lmte[10]:=valores.dis[1];
 IF valores.dis[2]<>'' THEN
   lmte[17]:=valores.dis[2];
 IF valores.dis[3]<>'' THEN
   lmte[23]:=valores.dis[3];
 IF valores.dis[4]<>'' THEN
   lmte[30]:=valores.dis[4];
END;

PROCEDURE Checa(VAR Numero : INTEGER; VAR Alarma : INTEGER; Arre_fin : Arreglo; tot : INTEGER);
VAR
 Linea,valor,error,
 L46,L48,lin_sub,
 Lin_al,Lin_bnd,X,
 Y,Diam,regresa,
 Z,Xant,Yant,Zant,
 Xcent,Ycent,
 Xabs,Yabs,Zabs,
 Lin_bnd2,Lin_al2  : INTEGER;
 Valorstr          : STRING[5];
 Campo             : CHAR;
 M06,M03,G81,G45,
 M05,Cor_mode,
 G46,G40,G48,G92,
 ABSOL,grad,M30    : BOOLEAN;
 Comando,Com_sub,
 Linstr            : STRING[3];
 V1,V2,V3,Vcen     : Vector;
 Diac              : REAL;
BEGIN
 Alarma:=0;Linea:=1;
 G81:=FALSE;G45:=FALSE;Absol:=FALSE;Cor_mode:=FALSE;
 G46:=FALSE;G48:=FALSE;M30:=FALSE;M05:=FALSE;
 M03:=FALSE;M06:=FALSE;G92:=FALSE;
 Lin_al:=230;Lin_bnd:=0;Lin_al2:=230;Lin_bnd2:=0;
 X:=0;Y:=0;Z:=0;Xant:=0;Yant:=0;Zant:=0;Diam:=0;
 Xabs:=0;Yabs:=0;Zabs:=0;

 REPEAT

  Comando:='';
  Comando:=Arre_fin[linea,6]+Arre_fin[linea,7]+Arre_fin[linea,8];
  IF (Comando <> '  ') AND (Comando <> 'M98') THEN BEGIN
   {write(3 'linea,',comando);readln;}
   IF Comando = '91' THEN BEGIN
    Absol:=False;
    Xabs:=X;Yabs:=Y;Zabs:=Z;
    X:=0;Y:=0;Z:=0;

   END;
   IF Comando = '90' THEN BEGIN
    Absol:=TRUE;
    X:=Xabs;Y:=Yabs;Z:=Zabs;
    Xant:=X;Yant:=Y;Zant:=Y;
   END;

   { Alarma 2 Valores X}

   Valor:=0;Campo:=Arre_fin[Linea,10];
   Valorstr:=Arre_fin[linea,11]+Arre_fin[linea,12]+Arre_fin[linea,13]+
        Arre_fin[linea,14]+Arre_fin[linea,15];
   error:=1;
```

```
VAL(Valorstr,Valor,error);
IF error = 0 THEN BEGIN
 CASE Campo OF
  ' ' : IF valor > 19999 THEN Alarma:=2;
  'J' : IF valor > 19999 THEN Alarma:=2;
  'D' : IF valor > 9999 THEN Alarma:=2;
  'T' : IF valor > 9999 THEN Alarma:=2;
 END;
END
ELSE IF Campo <>'*' THEN Alarma:=2;
IF ((alarma = 0) AND ((campo = ' ') OR (campo = 'J') )) AND ( Comando <>'  ') THEN BEGIN
 Xant:=X;X:=Valor;
 IF campo = 'J' THEN BEGIN
  X:=-X;
 END;
END;
IF (campo = 'D') AND (Alarma = 0) THEN BEGIN
   Diam:= Valor;
 END;
{ Alarma 3 Valor F Incorrecto}

Valor:=0;Campo:=Arre_fin[Linea,30];
Valorstr:=Arre_fin[linea,31]+Arre_fin[linea,32]+Arre_fin[linea,33];
error:=1;
VAL(Valorstr,Valor,error);
IF error = 0 THEN BEGIN
 CASE Campo OF
  ' ' : IF (valor > 499) OR (Valor < 2) THEN Alarma:=3;
  'L' : IF (valor > 221) OR (valor < 1) THEN Alarma:=3;
  'T' : IF (valor > 499) OR (valor < 0) THEN Alarma:=3;
 END;
END
ELSE IF Campo <>'*' THEN Alarma:=3;


{ Alarma 15 Valores Y}
Valor:=0;Campo:=Arre_fin[Linea,17];
Valorstr:=Arre_fin[linea,18]+Arre_fin[linea,19]+Arre_fin[linea,20]+
      Arre_fin[linea,21];
error:=1;
VAL(Valorstr,Valor,error);
IF error = 0 THEN BEGIN
 CASE Campo OF
  ' ' : IF valor > 9999 THEN Alarma:=15;
  'J' : IF valor > 9999 THEN Alarma:=15;
  'S' : IF valor > 9999 THEN Alarma:=15;
  'P' : IF valor > 9999 THEN Alarma:=15;
 END;
END
ELSE IF Campo <>'*' THEN Alarma:=15;
IF (alarma = 0) AND ((campo = ' ') OR (campo = 'J')) THEN BEGIN
 Yant:=Y;Y:=Valor;
 IF campo = 'J' THEN BEGIN
  Y:=-Y;
 END;
END;
{ Alarma 4,1 Valores Z}
Valor:=0;Campo:=Arre_fin[Linea,23];
Valorstr:=Arre_fin[linea,24]+Arre_fin[linea,25]+Arre_fin[linea,26]+
      Arre_fin[linea,27]+Arre_fin[linea,28];
error:=1;
VAL(Valorstr,Valor,error);
IF error = 0 THEN BEGIN
 CASE Campo OF
  ' ' : IF valor > 19999 THEN Alarma:=4;
  'J' : IF valor > 19999 THEN Alarma:=4;
  {'D' : IF valor > 9999 THEN Alarma:=4;}
  'K' : IF valor <> 0 THEN Alarma:=4;
```

```
  END;
 END
 ELSE IF Campo <>'*' THEN Alarma:=4;
 IF (alarma = 0) AND ((campo = '') OR (campo = '-')) THEN BEGIN
  IF Comando <> 'M06' THEN BEGIN
   Zant:=Z;Z:=Valor;IF campo = '-' THEN Z:=-Z;
  END;
 END;
 IF comando = '04' THEN X:=Xant;
 IF (comando = '92') THEN BEGIN
   IF G92 = FALSE THEN BEGIN
    G92:=TRUE;
    IF absol THEN BEGIN
    Xant:=X;Yant:=Y;Zant:=Z;
    END;
    IF NOT Absol THEN BEGIN
    Xabs:=X;Yabs:=Y;Zabs:=Z;
    X:=0;Z:=0;Y:=0;
    Xabs:=X
    END;
   END
   ELSE Alarma:=19;
 END;
 IF Comando = 'M98' THEN BEGIN
  X:=Xant;Y:=Yant;Z:=Zant;
 END;
 IF Comando = '72' THEN BEGIN
  G45:=TRUE;
  IF linea < lin_al THEN lin_al:=linea;
  IF Absol THEN BEGIN
   IF ABS(X-Xant) < ROUND(Diam * 1.1) THEN Alarma:=18;
  END
  ELSE BEGIN
   IF ABS(X) < ROUND(Diam * 1.1) THEN Alarma:=18;
  END;
  {IF Absol THEN BEGIN
   IF Z<>Zant THEN Alarma:=4;
  END;
  IF Absol = FALSE THEN BEGIN
   IF Z <> 0 THEN Alarma:=4;
  END;}
  IF Alarma = 0 THEN BEGIN
   IF absol THEN BEGIN
    X:=Xant;Y:=Yant;Z:=Zant;
   END;
   IF Absol = FALSE THEN BEGIN
    X:=0;Y:=0;Z:=0;
   END;
  END;
 END;
 IF Comando = '73' THEN BEGIN
  IF absol THEN
   Z:=Zant;
  IF Absol = FALSE THEN
   Z:=0;
 END;
 IF Comando = '81' THEN BEGIN
  IF absol THEN
   Z:=Zant;
  IF Absol = FALSE THEN
   Z:=0;
 END;
 IF Comando = '82' THEN BEGIN
  IF absol THEN
   Z:=Zant;
  IF Absol = FALSE THEN
   Z:=0;
 END.
```

```
IF Comando = '83' THEN BEGIN
 IF absol THEN
   Z:=Zant;
 IF Absol = FALSE THEN
   Z:=0;
END;
IF Comando = '85' THEN BEGIN
 IF absol THEN
   Z:=Zant;
 IF Absol = FALSE THEN
   Z:=0;
END;
IF Comando = '89' THEN BEGIN
 IF absol THEN
   Z:=Zant;
 IF Absol = FALSE THEN
   Z:=0;
END;
{ Alarma 4.2 Trayectoria tridimensional}
IF Absol THEN BEGIN
 {writeln(Xant,' ',Yant,' ',Zant);
 write(X,' ',Y,' ',Z,' ',linea);readln;}
 IF (X <> Xant) AND (Y <> Yant) AND (Z <> Zant) THEN Alarma:=4

END
ELSE BEGIN
 IF (X <> 0) AND (Y <> 0) AND (Z <> 0) THEN Alarma:=4;
END;

{ Alarma 5}
{IF (Linea = Tot) AND (Arre_fin[Linea,6]+Arre_fin[Linea,7]+Arre_fin[Linea,8]<>'M30') THEN
 Alarma:=5;
IF (Linea <> Tot) AND (Arre_fin[Linea,6]+Arre_fin[Linea,7]+Arre_fin[Linea,8]='M30') THEN
 Alarma:=5;}
IF Comando = 'M30' THEN BEGIN
 M30:=TRUE;
END;
{ Alarma 1 radio equivocado}


Grad:=FALSE;
IF (Arre_fin[Linea+1,6]+Arre_fin[Linea+1,7]+Arre_fin[Linea+1,8]='M99') THEN BEGIN
 Grad:=TRUE;
 Valor:=0;
 Valorstr:=Arre_fin[linea+1,18]+Arre_fin[linea+1,19]+Arre_fin[linea+1,20]+
       Arre_fin[linea+1,21];
 error:=1;
 VAL(Valorstr,Valor,error);
 IF error = 0 THEN BEGIN
  IF valor > 9999 THEN Alarma:=15
  ELSE Ycent:=valor;
 END
 ELSE  Alarma:=15;
 Valor:=0;
 Valorstr:=Arre_fin[linea+1,11]+Arre_fin[linea+1,12]+Arre_fin[linea+1,13]+
       Arre_fin[linea+1,14]+Arre_fin[linea+1,15];
 error:=1;
 VAL(Valorstr,Valor,error);
 IF error = 0 THEN BEGIN
  IF valor > 9999 THEN Alarma:=2
  ELSE Xcent:=valor;
 END
 ELSE  Alarma:=2;
END;
{ G03 Sentido Horario}
IF (Arre_fin[Linea,6]+Arre_fin[Linea,7]+Arre_fin[Linea,8]='03') THEN BEGIN
 IF Absol THEN BEGIN
   IF Z <> Zant THEN Alarma:=1;
```

```
  IF NOT Grad THEN
    IF ABS (X-Xant)<> ABS (Y-Yant) THEN Alarma:=1;
  IF Grad THEN BEGIN
   arrp(X,Y,V2);arrp(Xant,Yant,V1);arrp(Xcent,Ycent,V3);
   Resta(V2,V1,V1);V1[1]:=V1[1]/2;v1[2]:=V1[2]/2;
   rec_pol(V1,V1);
   rec_pol(v3,v3);Disc:=SQR(V3[1])-SQR(V1[1]);
   IF ABS(disc) = Disc THEN Vcen[1]:=SQRT(Disc)
   ELSE Alarma:=1;
   Vcen[2]:=V1[2]-pi/2;
   pol_rec(v1,v1);pol_rec(Vcen,Vcen);
   { asigna el cuadrante en el que se trazara el seg.}
   IF (Signo(Vcen[1]) = '+') AND (Signo(Vcen[2]) = '-') THEN BEGIN
    inst[Linea,36]:='1';inst[Linea,37]:='+';inst[Linea,38]:='-';
   END;
   IF (Signo(Vcen[1]) = '-') AND (Signo(Vcen[2]) = '-') THEN BEGIN
    inst[Linea,36]:='2';inst[Linea,37]:='-';inst[Linea,38]:='+';
   END;
   IF (Signo(Vcen[1]) = '-') AND (Signo(Vcen[2]) = '+') THEN BEGIN
    inst[Linea,36]:='3';inst[Linea,37]:='-';inst[Linea,38]:='-';
   END;
   IF (Signo(Vcen[1]) = '+') AND (Signo(Vcen[2]) = '+') THEN BEGIN
    inst[Linea,36]:='4';inst[Linea,37]:='+';inst[Linea,38]:='+';
   END;
   Suma(v1,Vcen,V2);V1[1]:=V1[1]*2;V1[2]:=V1[2]*2;
   Resta(V1,V2,V3);V2[1]:=ROUND(-V2[1]);V2[2]:=ROUND(-V2[2]);
   V3[1]:=ROUND(V3[1]);V3[2]:=ROUND(V3[2]);
   IF (ABS(V2[1]) <= 5) OR (ABS(V3[1]) <= 5) THEN BEGIN
    V2[1]:=ABS(V2[1]);
    V3[1]:=ABS(V3[1]);
   END;
   IF (ABS(V2[2]) <= 5) OR (ABS(V3[2]) <= 5) THEN BEGIN
    V3[2]:=ABS(V2[2]);
    V3[2]:=ABS(V3[2]);
   END;

   {Gotoxy(50,3);write(linea);
   gotoxy(50,4);write(v2[1]:6:2,',',v3[1]:6:2);
   gotoxy(50,5);write(v2[2]:6:2,',',v3[2]:6:2),readln;}

   IF (signo(V2[1])<>signo(V3[1])) OR (signo(V2[2])<>signo(V3[2])) THEN Alarma:=1;
   V2[2]:=ROUND(V2[2]);V2[1]:=ROUND(V2[1]);
   IF (ABS(ABS(V2[1]) - Xcent)>5) OR (ABS(ABS(V2[2]) -Ycent)>5) THEN Alarma:=1;
  END;
END
ELSE BEGIN    { Incrementales }
 IF Z <> 0 THEN Alarma:=1;
  IF NOT Grad  THEN
    IF ABS (X) <> ABS (Y) THEN Alarma:=1;
  IF Grad THEN BEGIN
   arrp(X, Y, V1),arrp(Xcent, Ycent, V3);
   V1[1]:=V1[1]/2;v1[2]:=V1[2]/2;
   rec_pol(V1,V1);
   rec_pol(v3,v3);Disc:=SQR(V3[1])-SQR(V1[1]);
   IF ABS(disc) = Disc THEN Vcen[1]:=SQRT(Disc)
   ELSE Alarma:=1;
   Vcen[2]:=V1[2]-pi/2;
   pol_rec(v1,v1);pol_rec(Vcen,Vcen);
   { asigna el cuadrante en el que se trazara el seg.}
   IF (Signo(Vcen[1]) = '+') AND (Signo(Vcen[2]) = '-') THEN BEGIN
    inst[Linea,36]:='1';inst[Linea,37]:='+';inst[Linea,38]:='-';
   END;
   IF (Signo(Vcen[1]) = '-') AND (Signo(Vcen[2]) = '-') THEN BEGIN
    inst[Linea,36]:='2';inst[Linea,37]:='-';inst[Linea,38]:='+';
   END;
   IF (Signo(Vcen[1]) = '-') AND (Signo(Vcen[2]) = '+') THEN BEGIN
    inst[Linea,36]:='3';inst[Linea,37]:='-';inst[Linea,38]:='-';
   END;
```

```
      IF (Signo(Vcen[1]) = '+') AND (Signo(Vcen[2]) = '+') THEN BEGIN
        inst[Linea,36]:='4';inst[Linea,37]:='+';inst[Linea,38]:='+';
      END;
      Suma(v1,Vcen,V2);V1[1]:=V1[1]*2;V1[2]:=V1[2]*2;
      Resta(V1,V2,V3);V2[1]:=ROUND(-V2[1]);V2[2]:=ROUND(-V2[2]);
      V3[1]:=ROUND(V3[1]);V3[2]:=ROUND(V3[2]);
      IF (ABS(V2[1]) <= 5) OR (ABS(V3[1]) <= 5) THEN BEGIN
        V2[1]:=ABS(V2[1]);
        V3[1]:=ABS(V3[1]);
      END;
      IF (ABS(V2[2]) <= 5) OR (ABS(V3[2]) <= 5) THEN BEGIN
        V2[2]:=ABS(V2[2]);
        V3[2]:=ABS(V3[2]);
      END;
      IF (signo(V2[1])<>signo(V3[1])) OR (signo(V2[2])<>signo(V3[2])) THEN Alarma:=1;
      V2[2]:=ROUND(V2[2]);V2[1]:=ROUND(V2[1]);
      IF (ABS(ABS(V2[1]) - Xcent)>5) OR (ABS(ABS(V2[2]) -Ycent)>5) THEN Alarma:=1;

    END;
  END;
END;
{ Sentido Antihorario 002}
IF (Arre_fin[Linea,6]+Arre_fin[Linea,7]+Arre_fin[Linea,8]='002') THEN BEGIN
  IF Absol THEN BEGIN
    IF Z <> Zant THEN Alarma:=1;
    IF NOT Grad THEN
     IF ABS (X-Xant)<> ABS (Y-Yant) THEN Alarma:=1;
    IF Grad THEN BEGIN
     arrp(X,Y,V2);arrp(Xant,Yant,V1);arrp(Xcent,Ycent,V3);
     Resta(V2,V1,V1);V1[1]:=V1[1]/2;v1[2]:=V1[2]/2;
     rec_pol(V1,V1);
     rec_pol(v3,v3);Disc:=SQR(V3[1])-SQR(V1[1]);
     IF ABS(disc) = Disc THEN Vcen[1]:=SQRT(Disc)
     ELSE Alarma:=1;
     Vcen[2]:=V1[2]+pi/2;
     pol_rec(v1,v1);pol_rec(Vcen,Vcen);
     { asigna el cuadrante en el que se trazara el arc. }
     IF (Signo(Vcen[1]) = '-') AND (Signo(Vcen[2]) = '+') THEN BEGIN
       inst[Linea,36]:='1';inst[Linea,37]:='-';inst[Linea,38]:='+';
     END;
     IF (Signo(Vcen[1]) = '+') AND (Signo(Vcen[2]) = '+') THEN BEGIN
       inst[Linea,36]:='2';inst[Linea,37]:='+';inst[Linea,38]:='-';
     END;
     IF (Signo(Vcen[1]) = '+') AND (Signo(Vcen[2]) = '-') THEN BEGIN
       inst[Linea,36]:='3';inst[Linea,37]:='+';inst[Linea,38]:='+';
     END;
     IF (Signo(Vcen[1]) = '-') AND (Signo(Vcen[2]) = '-') THEN BEGIN
       inst[Linea,36]:='4';inst[Linea,37]:='-';inst[Linea,38]:='-';
     END;
     Suma(v1,Vcen,V2);V1[1]:=V1[1]*2;V1[2]:=V1[2]*2;
     Resta(V1,V2,V3);V2[1]:=ROUND(-V2[1]);V2[2]:=ROUND(-V2[2]);
     V3[1]:=ROUND(V3[1]);V3[2]:=ROUND(V3[2]);
     IF (ABS(V2[1]) <= 5) OR (ABS(V3[1]) <= 5) THEN BEGIN
       V2[1]:=ABS(V2[1]);
       V3[1]:=ABS(V3[1]);
     END;
     IF (ABS(V2[2]) <= 5) OR (ABS(V3[2]) <= 5) THEN BEGIN
       V2[2]:=ABS(V2[2]);
       V3[2]:=ABS(V3[2]);
     END;
     IF (signo(V2[1])<>signo(V3[1])) OR (signo(V2[2])<>signo(V3[2])) THEN Alarma:=1;
     V2[2]:=ROUND(V2[2]);V2[1]:=ROUND(V2[1]);
     IF (ABS(ABS(V2[1]) - Xcent)>5) OR (ABS(ABS(V2[2]) -Ycent)>5) THEN Alarma:=1;
    END;
  END
  ELSE BEGIN    { Incrementales }
   IF Z <> 0 THEN Alarma:=1;
    IF NOT Grad THEN
```

```
    IF ABS (X) <> ABS (Y) THEN Alarma:=1;
   IF Grad THEN BEGIN
    arrp(X,Y,V1);arrp(Xcent,Ycent,V3);
    V1[1]:=V1[1]/2;v1[2]:=V1[2]/2;
    rec_pol(V1,V1);
    rec_pol(v3,v3);Disc:=SQR(V3[1])-SQR(V1[1]);
    IF ABS(disc) = Disc THEN Vcen[1]:=SQRT(Disc)
    ELSE Alarma:=1;
    Vcen[2]:=V1[2]-pi/2;
    pol_rec(v1,v1);pol_rec(Vcen,Vcen);
    Suma(v1,Vcen,V2);V1[1]:=V1[1]*2;V1[2]:=V1[2]*2;
    Resta(V1,V2,V3);V2[1]:=ROUND(-V2[1]);V2[2]:=ROUND(-V2[2]);
    V3[1]:=ROUND(V3[1]);V3[2]:=ROUND(V3[2]);
    IF (ABS(V2[1]) <= 5) OR (ABS(V3[1]) <= 5) THEN BEGIN
     V2[1]:=ABS(V2[1]);
     V3[1]:=ABS(V3[1]);
    END;
    IF (ABS(V2[2]) <= 5) OR (ABS(V3[2]) <= 5) THEN BEGIN
     V2[2]:=ABS(V2[2]);
     V3[2]:=ABS(V3[2]);
    END;
    IF (signo(V2[1])<>signo(V3[1])) OR (signo(V2[2])<>signo(V3[2])) THEN Alarma:=1;
    V2[2]:=ROUND(V2[2]);V2[1]:=ROUND(V2[1]);
    IF (ABS(ABS(V2[1]) - Xcent)>5) OR (ABS(ABS(V2[2]) -Ycent)>5) THEN Alarma:=1;
   END;
  END;
END;


{ Alarma 6 y 16}
IF Abso1 THEN BEGIN
 IF (X<>Xant) OR (Y<>Yant) OR (Z<>Zant) THEN BEGIN
  IF NOT M06 THEN BEGIN
   IF linea < lin_a1 THEN lin_a1:=linea;
   Alarma:=16;
  END;
 END;
END;
IF Abso1=FALSE THEN BEGIN
 IF (X<>0) OR (Y<>0) OR (Z<>0) THEN BEGIN
  IF NOT M06 THEN BEGIN
   IF linea < lin_a1 THEN lin_a1:=linea;
   Alarma:=16;
  END;
 END;
END;

IF (Comando = '81') OR (Comando = '84') THEN BEGIN
  G81:=TRUE;
  IF Linea < Lin_a12 THEN Lin_a12:=linea;
 END;
IF Comando = '40' THEN BEGIN
 G46:=FALSE;G48:=FALSE;
END;
IF Comando = '45' THEN BEGIN
 G45:=TRUE;
 IF linea < lin_a1 THEN lin_a1:=linea;
END;
IF Comando = '46' THEN BEGIN
 L46:=Linea;
 G46:=TRUE;
 G45:=TRUE;
 IF linea < lin_a1 THEN lin_a1:=linea;
END;
IF Comando = '47' THEN BEGIN
 G45:=TRUE;
 IF linea < lin_a1 THEN lin_a1:=linea;
```

```
  END;
  IF Comando = '48' THEN BEGIN
   L48:=linea;
   G48:=TRUE;
   G45:=TRUE;
   IF linea < lin_al THEN lin_al:=linea;
  END;

  IF (Comando= '81') OR (Comando= '82') OR (Comando= '83') OR
     (Comando= '85') OR (Comando= '89') THEN BEGIN
   IF Absol THEN BEGIN
    Z:=Zant;
   END;

  END;
  IF COMANDO = 'M03'THEN BEGIN
   IF NOT M03 THEN Lin_bnd2:=linea;
   M03:=TRUE;
  END;
  IF Comando = 'M05' THEN M05:=TRUE;
  IF COMANDO = 'M06'THEN BEGIN
   IF NOT M06 THEN Lin_bnd:=linea;
   M06:=TRUE;
  END;
  IF Comando = 'M98' THEN BEGIN
   X:=Xant;Y:=Yant;Z:=Zant;
  END;
  IF (G46) AND (Linea > L46) THEN
   IF Absol THEN BEGIN
    IF (X-Xant=0) AND (Y-Yant=0)  THEN Alarma:=18;
    IF (X-Xant<>0) AND (ABS(X-Xant) < Diam ) THEN Alarma:=18;
    IF (Y-Yant<>0) AND (ABS(Y-Yant) < Diam ) THEN Alarma:=18
   END
   ELSE BEGIN
    IF (X=0) AND (Y=0)  THEN Alarma:=18;
    IF (X <> 0) AND (ABS(X) < Diam ) THEN Alarma:=18;
    {ELSE}
    IF (Y <> 0) AND (ABS(Y) < Diam ) THEN Alarma:=18;
   END;
  IF (G48) AND (Linea > L48) THEN
   IF Absol THEN BEGIN
    IF (X-Xant=0) AND (Y-Yant=0)  THEN Alarma:=18;
    IF (X-Xant<>0) AND (ABS(X-Xant) < Diam * 2) THEN Alarma:=18;
    {ELSE}
    IF (Y-Yant<>0) AND (ABS(Y-Yant) < Diam * 2) THEN Alarma:=18
   END
   ELSE BEGIN
    IF (X=0) AND (Y=0) THEN Alarma:=18;
    IF (X <> 0) AND (ABS(X) < Diam * 2) THEN Alarma:=18;
    {ELSE}
    IF (Y <> 0) AND (ABS(Y) < Diam * 2) THEN Alarma:=18
   END;
  END;

  { Subprogramas }
   IF Comando = '25' THEN BEGIN
    Cor_mode:=Absol;
    Regresa:=Linea+1;
    Valor:=0;Campo:=Arre_fin[Linea,30];
    Valorstr:=Arre_fin[linea,31]+Arre_fin[linea,32]+Arre_fin[linea,33];
    error:=1;
    VAL(Valorstr,Valor,error);
    IF error = 0 THEN BEGIN
     CASE Campo OF
      ' ' : IF (valor > 499) OR (Valor < 2) THEN Alarma:=3;
      'L' : IF (valor > 221) OR (valor < 1) THEN Alarma:=3
          ELSE BEGIN
          IF Valor < Tot THEN BEGIN
```

```
          Linea:=Valor;
          Lin_sub:=Linea-1;
         END
         ELSE Alarma:=17;
          {Write(linea);readln;}
        END;
    'T' : IF (valor > 499) OR (valor < 0) THEN Alarma:=3;
   END;
  END
  ELSE IF Campo <>'*' THEN Alarma:=3;

  IF Alarma = 0 THEN BEGIN
   REPEAT
    Lin_sub:=Lin_sub+1;
    Com_sub:=Arre_fin[lin_sub,6]+Arre_fin[lin_sub,7]+Arre_fin[lin_sub,8];
   UNTIL (Com_sub='M17') OR (Lin_sub = Tot);
   IF (Lin_sub = Tot) AND (Com_sub <> 'M17') THEN BEGIN
    Alarma:=23;
    Linea:=tot;
   END
   ELSE BEGIN
    Linstr:=' ';

    {gotoxy(60,1);Write('       ');
    gotoxy(60,1);write('re ',regresa);}
    STR(Regresa,Linstr);
    VAL(linstr,linea,error);
    {gotoxy(50,2);Write('       ');
    gotoxy(50,2);write(Linstr[0]+Linstr[1]+Linstr[2]+linstr[3]);readln;}
    Char_num:=linstr[0];

    inst[Lin_sub,36]:='*';
    inst[Lin_sub,37]:='*';
    inst[Lin_sub,38]:='*';
    IF Linstr[1] IN Enteros THEN
     inst[Lin_sub,36]:=Linstr[1];
    IF Linstr[2] IN Enteros THEN
     inst[Lin_sub,37]:=Linstr[2];
    IF Linstr[3] IN Enteros THEN
     inst[Lin_sub,38]:=Linstr[3];
    inst[Linea-1,36]:=Linstr[1];
    inst[Linea-1,37]:=Linstr[2];
    inst[Linea-1,38]:=Linstr[3];
    Linea:=Valor-1;
    {gotoxy(50,1),Write('       ');
    gotoxy(50,1);write('l ',inst[lin_sub,36]+inst[lin_sub,37]
    +inst[Lin_sub,38]),readln;}
   END;
  END;
 END;
 IF Comando = '27' THEN BEGIN

  Valor:=0,Campo:=Arre_fin[Linea,30];
  Valorstr:=Arre_fin[linea,31]+Arre_fin[linea,32]+Arre_fin[linea,33];
  error:=1;
  VAL(Valorstr,Valor,error);
  IF error = 0 THEN BEGIN
   CASE Campo OF
    ' ': IF (valor > 499) OR (Valor < 2) THEN Alarma:=3;
    'L' : IF (valor > tot) OR (valor < 1) THEN Alarma:=3
        ELSE BEGIN
         IF Valor < Tot THEN BEGIN
          Linea:=Valor-1;
         END
         ELSE Alarma:=17;
          {Write(linea);readln;}
        END;
    'T' : IF (valor > 499) OR (valor < 0) THEN Alarma:=3;
```

```
    END;
   END

  END;
  IF Comando = 'M17' THEN BEGIN
   {Linstr[1]:=inst[Linea,36];
   Linstr[2]:=inst[Linea,37];
   Linstr[3]:=inst[Linea,38];}
   linstr:='';
   Linstr[0]:=Char_num;
   Linstr[1]:=inst[Linea,36];
   IF inst[Linea,37] IN enteros THEN linstr[2]:='('+Linstr+')inst[Linea,37]
   ELSE linstr[2]:='';
   IF inst[Linea,38] IN enteros THEN linstr[3]:='('+Linstr+')inst[Linea,38]
   ELSE linstr[3]:='';

   Linea:=0;
   VAL(Linstr,Linea,error);
   {gotoxy(50,2);Write('          ');
   gotoxy(50,2);write('regresa ',Linea);readln;}
   Linea:=Linea-1;
   {write('2 ',Linea);readln;}
   IF (Absol=FALSE) AND (Cor_mode=TRUE) THEN BEGIN
    X:=Xabs;Y:=Yabs;Z:=Zabs;
    Xant:=X;Yant:=Y;Zant:=Y;
   END;
   IF (Absol=TRUE) AND (Cor_mode=FALSE) THEN BEGIN
    Xabs:=X;Yabs:=Y;Zabs:=Z;
    X:=0;Y:=0;Z:=0;
   END;
   Absol:=Cor_mode;
  END;
 { Posicion real }
 IF (Comando = '00') OR (Comando = '01') OR
   (Comando = '02') OR (Comando = '03') THEN BEGIN
  IF Absol = FALSE THEN BEGIN
   Xabs:=Xabs + X;
   Yabs:=Yabs + Y;
   Zabs:=Zabs + Z;
  END;
 END;

 {gotoxy(50,3);Write('                    ');
 gotoxy(50,4);Write('                    ');
 gotoxy(50,3);Writeln(Xabs:',':Yabs:',':Zabs);
 gotoxy(50,5);Write('                    ');
 gotoxy(50,6);Write('                    ');
 gotoxy(50,5);Writeln(Xant:',':Yant:',':Zant);
 gotoxy(50,6);Writeln(X:',':Y:',':Z:',':linea);readln;}

 Numero:=linea;
 Linea:=Linea + 1;

UNTIL (Linea - 1 = tot) OR (Comando = 'M30') OR (Alarma <> 0);
IF linea = Tot+1 THEN BEGIN
 IF M30 = FALSE THEN Alarma:=5;
 IF G81 AND (NOT M03) THEN Alarma:=6;
 IF G81 AND (Lin_bnd2 > Lin_al2) THEN Alarma:=6;
 IF NOT M06 THEN Alarma:=16;
 IF G45 AND (NOT M06)  THEN Alarma:=16;
 IF G45 AND (Lin_bnd > Lin_al) THEN Alarma:=16;
 IF (Alarma<>0) AND (Lin_Al <> 230) THEN Numero:=Lin_Al;
 IF (Alarma<>0) AND (Lin_Al2 <> 230) THEN Numero:=Lin_Al2;
 IF M05 = FALSE THEN Alarma:=20;
END;
{write('alrma ',Alarma);readln;}
END;
```

```
END.
```

Archivo VECTORES.PAS

```
unit vectores;
interface
 uses crt;
 type
 vector=array[1..2] of real;
const
 pi=3.14159265359;
 function signo(x:real):char;
 procedure arrp(x1,x2:real;var vec:vector);
 procedure arre(x1,x2:real;var vec:vector);
 procedure rec_pol(x1:vector;var vec:vector);
 procedure pol_rec(x1:vector;var vec:vector);
 procedure suma(x1,x2:vector;var vec:vector);
 procedure resta(x1,x2:vector;var vec:vector);
 procedure mult(x1,x2:vector;var vec:vector);
 procedure divi(x1,x2:vector;var vec:vector);
implementation

function signo(x:real):char;
 begin
  signo:='+';
  if x<>0 then
   if abs(x)/x<>1 then signo:='-';
 end;
procedure arrp(x1,x2:real;var vec:vector);
 begin
  vec[1]:=x1;
  vec[2]:=x2;
 end;
procedure arre(x1,x2:real;var vec:vector);
 begin
  vec[1]:=x1;
  vec[2]:=x2;
 end;
procedure rec_pol(x1:vector;var vec:vector);
 var
  x,y   :char;
 begin
  vec[1]:=sqrt(sqr(x1[1])+sqr(x1[2]));
  x:=signo(x1[1]);y:=signo(x1[2]);
  case x of
   '+' : begin
         if (y='+')and(x1[1]<>0) then vec[2]:=arctan(x1[2]/x1[1])
         else vec[2]:=pi/2;
         if y='-' then vec[2]:=2*pi-abs(arctan(x1[2]/x1[1]));
         end;
   '-' : begin
         if y='-' then vec[2]:=pi+abs(arctan(x1[2]/x1[1]));
         if y='+' then vec[2]:=pi-abs(arctan(x1[2]/x1[1]));
         end;
  end;

 end;
procedure pol_rec(x1:vector;var vec:vector);
 begin
  vec[1]:=x1[1]*cos(x1[2]);
  vec[2]:=x1[1]*sin(x1[2]);
 end;
procedure suma(x1,x2:vector;var vec:vector);
 begin
  vec[1]:=x1[1]+x2[1];
  vec[2]:=x1[2]+x2[2];
 end;
```

```
procedure resta(x1,x2:vector;var vec:vector);
  begin
    vec[1]:=x1[1]-x2[1];
    vec[2]:=x1[2]-x2[2];
  end;
procedure mult(x1,x2:vector;var vec:vector);
var
 p1,p2,p     :vector;
 begin
   rec_pol(x1,p1);
   rec_pol(x2,p2);
   p[1]:=p1[1]*p2[1];
   p[2]:=p1[2]+p2[2];
   pol_rec(p,vec);
 end;
procedure divi(x1,x2:vector;var vec:vector);
var
 p1,p2,p     :vector;
 begin
   rec_pol(x1,p1);
   rec_pol(x2,p2);
   p[1]:=p1[1]/p2[1];
   p[2]:=p1[2]-p2[2];
   pol_rec(p,vec);
 end;
end.□
```

Anexo 3

Unidad HTAS.PAS

```
UNIT Htas;

INTERFACE

USES
 GRAPH,CRT;

TYPE
 Location = OBJECT
  X,Y : INTEGER;
  PROCEDURE Init(InitX,InitY : INTEGER);
  FUNCTION GetX : INTEGER;
  FUNCTION GetY : INTEGER;
 END;

 PointPtr = ^Point;

 Point = OBJECT (Location)
  Visible : BOOLEAN;
  CONSTRUCTOR Init(InitX,InitY : INTEGER);
  DESTRUCTOR  Done; VIRTUAL;
  PROCEDURE Show; VIRTUAL;
  PROCEDURE Hide; VIRTUAL;
  FUNCTION Invisible : BOOLEAN;
  PROCEDURE MoveTo(NewX,NewY : INTEGER);
  PROCEDURE Drag(Dragby : INTEGER); VIRTUAL;
 END;

 CircPtr = ^ Circ;

 Circ = OBJECT (Point)
  Radius : INTEGER;
  CONSTRUCTOR Init(InitX,InitY,InitRadius : INTEGER);
  PROCEDURE Show; VIRTUAL;
  PROCEDURE Hide; VIRTUAL;
  PROCEDURE movcir(Xi,Yi,colr,trazo :INTEGER);VIRTUAL;
  PROCEDURE Expand(Expandby : INTEGER);VIRTUAL;
  PROCEDURE Contract(Contractby : INTEGER);
 END;
 Rec_lat = OBJECT (Point)
  Radius : INTEGER;
  CONSTRUCTOR Init(InitX,InitY,InitRadius : INTEGER);
  PROCEDURE Show; VIRTUAL;
  PROCEDURE Oculta(Trazorl : INTEGER); VIRTUAL;
  PROCEDURE Mov_rec(Xi,Yi,trazo : INTEGER);
 END;
 Rec_fin = OBJECT (Point)
  Radius : INTEGER;
  CONSTRUCTOR Init(InitX,InitY,InitRadius : INTEGER);
  PROCEDURE Show; VIRTUAL;
  PROCEDURE Oculta(Trazorl : INTEGER); VIRTUAL;
  PROCEDURE Mov_rec(Xi,Yi,trazo : INTEGER);
 END;


 VAR
  Xmax,Ymax,Xos,Yos,i,Base,Altura,anc,erp,
  Xof,Yof,Xol,Yol,clr     : INTEGER;
  Vert,Down,Dere,Izqi     : BOOLEAN;
```

```
  Camder,Camizq          : BOOLEAN;
  Horz,Rigth,Arriba,Abajo  : BOOLEAN;
  Camarr,CamAb,Diagonal    : BOOLEAN;
  Subeder,subeizq        : BOOLEAN;
  Dercarr,Dercaba        : BOOLEAN;
  Sub_dif,Sub_dif        : BOOLEAN;
{PROCEDURE ven(i:INTEGER),}

IMPLEMENTATION
  PROCEDURE ven(i:INTEGER);

  BEGIN
   CASE i OF
    0 : SETVIEWPORT(0,0,Xmax,Ymax,Clipon);
    1 : SETVIEWPORT(Xos-(Base DIV 2),Yos-(Anc DIV 2),Xos+(Base DIV 2),Yos+(Anc DIV 2),Clipon);
    2 : SETVIEWPORT(Xof-(Altura DIV 2),Yof-(Anc DIV 2),Xof+(Altura DIV 2),Yof+(Anc DIV 2),clipon);
    3 : SETVIEWPORT(Xol-(Base DIV 2),Yol-(Altura DIV 2),Xol+(Base DIV 2),Yol+(Altura DIV 2),clipon);
   END;
  END;

  PROCEDURE Location.Init(InitX,InitY : INTEGER);
  BEGIN
   X:=InitX;
   Y:=InitY;
  END;
  FUNCTION Location.GetX: INTEGER;
  BEGIN
   GetX:=X;
  END;
  FUNCTION Location.GetY: INTEGER;
  BEGIN
   GetY:=Y;
  END;

  CONSTRUCTOR Point.Init(InitX,initY: INTEGER);
  BEGIN
   Location.Init(InitX,InitY);
   Visible:=FALSE;
  END;
  DESTRUCTOR Point.Done;
  BEGIN
   Hide;
  END;
  PROCEDURE point.Show;
  BEGIN
   Visible:=TRUE;
   PUTPIXEL(X,Y,GETCOLOR);
  END;
  PROCEDURE Point.Hide;
  BEGIN
   Visible:=FALSE;
   PUTPIXEL(X,Y,GETBKCOLOR);
  END;
  FUNCTION Point.Isvisible : BOOLEAN;
  BEGIN
   Isvisible:=Visible;
  END;
  PROCEDURE Point.Moveto (NewX,NewY : INTEGER);
  BEGIN
   Hide;
   Y:=NewY;
   X:=NewX;
   Show;
  END;
  FUNCTION GetDelta(VAR DeltaX:INTEGER; VAR deltaY:INTEGER):BOOLEAN;
  VAR
   KeyChar : CHAR;
   Quit    : BOOLEAN;
```

```
BEGIN
  DeltaX:=0;DeltaY:=0;
  GetDelta:=TRUE;
  REPEAT
    KeyChar:=READKEY;
    Quit:=TRUE;
    CASE ORD(KeyChar) OF
     0 : BEGIN
         Keychar:=READKEY;
         CASE ORD(Keychar) OF
          72 : DeltaY:=-1;
          80 : DeltaY:= 1;
          75 : DeltaX:=-1;
          77 : DeltaX:= 1;
          13 : SETCOLOR(GETCOLOR+1);
          ELSE Quit:=FALSE;
         END;
      END;
      13 : GetDelta:=FALSE;
      ELSE Quit:=False;
    END;
  UNTIL Quit;
END;
PROCEDURE Point.Drag(Dragby : INTEGER);
VAR
  DeltaX,DeltaY  : INTEGER;
  FigureX,FigureY : INTEGER;
BEGIN
  Show;
  FigureX:=GetX;
  FigureY:=GetY;
  WHILE GetDelta(DeltaX,DeltaY) DO
  BEGIN
    FigureX:=FigureX + (DeltaX * Dragby);
    FigureY:=FigureY + (DeltaY * Dragby);
    Hide;
    Y:=FigureY;
    X:=FigureX;
    Show;
  END;
END;

CONSTRUCTOR Circ.Init(InitX,InitY,InitRadius : INTEGER);
BEGIN
  Point.Init(InitX,InitY);

  Radius:=InitRadius;
END;
PROCEDURE Circ.Show;
VAR
  i,Temp  : INTEGER;
BEGIN
  Visible:=TRUE;
  Temp:=GETCOLOR;
  SETCOLOR(BLUE);
  GRAPH.CIRCLE(X,Y,Radius-1);
  SETCOLOR(temp);
END;
PROCEDURE Circ.Hide;
VAR
  Tempcolor,i : WORD;
BEGIN
  Visible:=FALSE;
  Tempcolor:=GRAPH.GETCOLOR;
  GRAPH.SETCOLOR(tempcolor);
  FOR i:=1 To Radius DO
    GRAPH.CIRCLE(X,Y,i);
  GRAPH.SETCOLOR(Tempcolor);
```

```
END;
PROCEDURE Circ.Movcir(Xf,Yf,colr,trazo :INTEGER);
VAR
 DeltaX,DeltaY   : INTEGER;
 FigureX,FigureY : INTEGER;
 Xant,Yant       : INTEGER;
BEGIN
 Xant:=GetX;
 Yant:=GetY;
 FigureX:=GetX;
 FigureY:=GetY;
 {WHILE GetDelta(DeltaX,DeltaY) DO
 BEGIN
  FigureX:=FigureX + (DeltaX * Dragby);
  FigureY:=FigureY + (DeltaY * Dragby);
  x:=figurex;y:=figurey;
  vent(1);}
  IF Trazo = 1 THEN BEGIN
   SETCOLOR(colr);
   Hide;
  END;
  x:=xi;y:=yi;
  CASE Trazo OF
   1 : BEGIN
       SETCOLOR(colr);
       Hide;
       Show;
       SETFILLSTYLE(Solidfill,colr);
       FLOODFILL(Getx,GetY,colr);
      END;
   2 : BEGIN
       SETCOLOR(BLACK);
       LINE(Xant,Yant,X,Y);
       SETCOLOR(colr);
      END;
  END;

  Xant:=x;
  Yant:=y;
 {END;}
END;
PROCEDURE Circ.Expand(Expandby : INTEGER);
BEGIN
 Hide;
 Radius:=Radius + Expandby;
 If Radius<0 THEN Radius:=0;
 Show;
END;
PROCEDURE Circ.Contract(Contractby : INTEGER);
BEGIN
 Hide;
 Radius:=Radius - Contractby;
 If Radius<0 THEN Radius:=0;
 Show;
END;

CONSTRUCTOR Rec_Ist.Init(InitX,InitY,InitRadius : INTEGER);
BEGIN
 Point.Init(InitX,InitY);
 Radius:=InitRadius;
END;
PROCEDURE Rec_Ist.Show;
VAR
 Temp : WORD;
BEGIN
 Temp:=GETCOLOR;
 Visible:=TRUE;
 SETCOLOR(YELLOW);
```

```
 LINE(X-Radius+1,Y,X+Radius-1,Y);{3}
 LINE(X-Radius,Y-altura{3*Radius+1},X-Radius,Y-1);{2}
 LINE(X+Radius,Y-altura{3*Radius+1},X+Radius,Y-1);{1}
 {LINE(X-Radius+1,Y-3*Radius,X+Radius-1,Y-3*Radius);0}
 SETCOLOR(Temp);
END;
PROCEDURE Rec_Lat.Oculta(Trazo1 : INTEGER);
 VAR
  Temp : WORD;
BEGIN
 Temp:=GETCOLOR;
 SETCOLOR(GREEN);
 CASE Trazo1 OF
  0 : BEGIN
      LINE(X-Radius+1,Y,X+Radius-1,Y);{3}
      LINE(X-Radius,Y-altura,X-Radius,Y-1);{2}
      LINE(X+Radius,Y-altura,X+Radius,Y-1);{1}
      LINE(X-Radius+1,Y-1,X+Radius-1,Y-1);{0}
     END;
  1 : BEGIN
      LINE(X-Radius+1,Y-3*Radius,X+Radius-1,Y-3*Radius);{0}
      LINE(X+Radius,Y-altura{3*Radius+1},X+Radius,Y-1);{1}
     END;
  2 : BEGIN
      LINE(X-Radius+1,Y-3*Radius,X+Radius-1,Y-3*Radius);{0}
      LINE(X-Radius,Y-altura{3*Radius+1},X-Radius,Y-1);{2}

     END;
  3 : BEGIN
      LINE(X-Radius+1,Y-1,X+Radius-1,Y-1);{0}
      LINE(X-Radius+1,Y,X+Radius-1,Y);{3}
     END;
  4 : LINE(X-Radius+1,Y-1,X+Radius-1,Y-1);{0}
 END;
 SETCOLOR(temp);
END;
PROCEDURE Rec_Lat.Mov_rec(Xi,Yi,Trazo : INTEGER);
VAR
 Xa,Ya : INTEGER;
BEGIN
 SETCOLOR(YELLOW);
 {IF Y <> 0 THEN Vert:=TRUE;
 IF X <> 0 THEN Vert:=FALSE;}
 Xa:=GETX;Ya:=GetY;
 CASE Trazo OF
  0 : BEGIN
      Oculta(0);
      X:=Xi;Y:=Yi;
      Show;
     END;
  1 : BEGIN
      IF NOT (((Xi<>Xa) AND (Yi<>Ya)))Diagonal THEN BEGIN
       IF Yi<>Ya THEN BEGIN
        IF DOWN THEN oculta(4);
        IF Dere THEN BEGIN
         Oculta(2);
         IF NOT DOWN THEN Oculta(3);
        END;
        IF Izqi THEN BEGIN
         Oculta(1);
         IF NOT DOWN THEN Oculta(3);
        END;
        IF (NOT Down) AND (NOT Dere) AND (NOT Izqi) THEN Oculta(3);
        IF subedor THEN Oculta(2);
        IF subeizq THEN Oculta(1);
        X:=Xi;Y:=Yi;
        Show;
       END;
```

```
      IF Xi > Xa THEN BEGIN
       IF (Ver) OR (Camizq) THEN Oculta(1)
       ELSE BEGIN
        Oculta(1);Oculta(2);
       END;
      X:=Xi;Y:=Yi;
      Show;
      END;
      IF Xi < Xa THEN BEGIN
       IF (Ver) OR (Camder) THEN Oculta(2)
       ELSE BEGIN
        Oculta(1);Oculta(2);
       END;
      X:=Xi;Y:=Yi;
      Show;
      END;
     END;
     IF ((Xi<>Xa) AND (Yi<>Ya)) Diagonal THEN BEGIN
      IF Down THEN BEGIN
       Oculta(1);Oculta(2);
      END;
      IF (Xi = Xa) AND (NOT Sub_dil) THEN oculta(0);

      IF Dere THEN BEGIN
       IF NOT Sub_dil THEN Oculta(0);
       IF Yi > Ya THEN
        LINE(Xa-Radius,Ya,Xi-Radius,Yi);
       IF Yi < Ya THEN
        LINE(Xa+Radius,Ya,Xi+Radius,Yi);
      END;
      IF Izqj THEN BEGIN
       IF NOT Sub_dil THEN Oculta(0);
       IF Yi > Ya THEN
        LINE(Xa+Radius,Ya,Xi+Radius,Yi);
       IF Yi < Ya THEN
        LINE(Xa-Radius,Ya,Xi-Radius,Yi);
      END;
      IF Sub_dil THEN BEGIN
       Oculta(1);Oculta(2);
      END;
      X:=Xi;Y:=Yi;
      Show;
     END;
    END;
  END;
END;

CONSTRUCTOR Rec_fm.Init(InitX,InitY,InitRadius : INTEGER);
BEGIN
 Point.Init(InitX,InitY);
 Radius:=InitRadius;
END;
PROCEDURE Rec_fm.Show;
VAR
 Temp : WORD;
BEGIN
 Temp:=GETCOLOR;
 Visible:=TRUE;
 SETCOLOR(YELLOW);
 LINE(X,Y-Radius+1,X,Y+Radius-1);{3}
 LINE(X-altura+(3*Radius+1),Y+Radius,X-1,Y+Radius);{2}
 LINE(X-altura+(3*Radius+1),Y-Radius,X-1,Y-Radius);{1}
 {LINE(X-3*Radius,Y-Radius+1,X-3*Radius,Y+Radius-1);0}
 SETCOLOR(Temp);
END;
PROCEDURE Rec_fm.Oculta(Trazo1 : INTEGER);
 VAR
```

```
  Temp : WORD;
BEGIN
 Temp:=GETCOLOR;
 SETCOLOR(GREEN);
 CASE Trazori OF
  0 : BEGIN
       LINE(X,Y-Radius+1,X,Y+Radius-1);{3}
       LINE(X-altura,Y+Radius,X-1,Y+Radius);{2}
       LINE(X-altura,Y-Radius,X-1,Y-Radius);{1}
       LINE(X-1,Y-Radius+1,X-1,Y+Radius-1);{0}
      END;
  1 : BEGIN
       LINE(X-altura,Y-Radius,X-1,Y-Radius);{1}
       LINE(X-3*Radius,Y-Radius+1,X-3*Radius,Y+Radius-1);{0}
      END;
  2 : BEGIN
       LINE(X-altura,Y+Radius,X-1,Y+Radius);{2}
       LINE(X-3*Radius,Y-Radius+1,X-3*Radius,Y+Radius-1);{0}
      END;
  3 : BEGIN
       LINE(X-1,Y-Radius+1,X-1,Y+Radius-1);{0}
       LINE(X,Y-Radius+1,X,Y+Radius-1);{3}
      END;
  4 : LINE(X-1,Y-Radius+1,X-1,Y+Radius-1);{0}
 END;
 SETCOLOR(temp);
END;
PROCEDURE Rec_frn.Mov_rec(Xi,Yi,Trazo : INTEGER);
VAR
 Xa,Ya : INTEGER;
BEGIN
 SETCOLOR(YELLOW);
 Xa:=GETX;Ya:=GetY;
 CASE Trazo OF
  0 : BEGIN
       Oculta(0);
       X:=Xi;Y:=Yi;
       Show;
      END;
  1 : BEGIN
       IF NOT(((Xi<>Xa) AND (Yi<>Ya))) Diagonal THEN BEGIN
        IF Xi<>Xa THEN BEGIN
         IF Rigth THEN oculta(4);
         IF Arriba THEN BEGIN
          Oculta(2);
          IF NOT Rigth THEN Oculta(3);
         END;
         IF Abajo THEN BEGIN
          Oculta(1);
          IF NOT Rigth THEN Oculta(3);
         END;
         IF (NOT Rigth) AND (NOT Arriba) AND (NOT Abajo) THEN Oculta(3);
         IF derearr THEN Oculta(2);
         IF derebaj THEN Oculta(1);
         X:=Xi;Y:=Yi;
         Show;
        END;

        IF Yi < Ya THEN BEGIN
         IF (Horz) OR (Carnab) THEN Oculta(1)
         ELSE BEGIN
          Oculta(1),Oculta(2);
         END;
         X:=Xi;Y:=Yi;
         Show;
        END;
        IF Yi > Ya THEN BEGIN
         IF (Horz) OR (Camarr) THEN Oculta(2)
```

```
        ELSE BEGIN
          Oculta(1);Oculta(2);
        END;
        X:=Xi;Y:=Yi;
        Show;
      END;
    END;
    {IF Diagonal THEN BEGIN
     Oculta(0);
     IF Abajo THEN
       LINE(Xa,Ya-Radius,Xi,Yi-Radius);
     IF Arriba THEN
       LINE(Xa,Ya+Radius,Xi,Yi+Radius);
     Y:=Yi;X:=Xi;
     Show;
    END;}
    IF ((Xi<>Xa) AND (Yi<>Ya)) Diagonal THEN  BEGIN
     IF Rigth THEN BEGIN
       Oculta(1);Oculta(2);
     END;
     IF (Yi = Ya) AND (NOT  Sub_dif) THEN oculta(0);

     IF Arriba THEN BEGIN
       IF NOT Sub_dif THEN Oculta(0);
       IF Xi > Xa THEN
         LINE(Xa,Ya+Radius,Xi,Yi+Radius);
       IF Xi < Xa THEN
         LINE(Xa,Ya-Radius,Xi,Yi-Radius);
     END;
     IF Abajo THEN BEGIN
       IF NOT Sub_dif THEN Oculta(0);
       IF Xi > Xa THEN
         LINE(Xa,Ya-Radius,Xi,Yi-Radius);
       IF Xi < Xa THEN
         LINE(Xa,Ya+Radius,Xi,Yi+Radius);
     END;
     IF Sub_dif THEN BEGIN
       Oculta(1);Oculta(2);
     END;
     X:=Xi;Y:=Yi;
     Show;
    END;
  END;
 END;
END.
```

Unidad GRAFPRB.PAS

```
UNIT GRAFPRB;

INTERFACE

USES
  Htas,GRAPH,CRT,Comp_1,Archivos;

CONST
  C0 = BLACK;       {C1 = BLUE;}
  C2 = GREEN;       C3 = CYAN;
  C4 = RED;         C5 = MAGENTA;
  C6 = BROWN;       C7 = DARKGRAY;
  C8 = LIGHTBLUE;   C9 = LIGHTGREEN;
  C10 = LIGHTCYAN;  C11 = LIGHTRED;
  C12 = LIGHTMAGENTA; C13 = YELLOW;
  C14 = WHITE;      C15 = LIGHTGRAY;


VAR
  Xmax,Ymax,Xos,Yos,i,Xdesp,Ydesp,Zdesp,Xu,Yu,Zu,
  Xa,Za,Ya,trz_rf,Zo,Xo,Xorigen,Yorigen,
  Yo,trz_rf,Zorigen,
  Xof,Yof,Xol,Yol,clr  : INTEGER;
  X,Y,Z,Xi,Yi,Zi,Trz   : INTEGER;
  num,num1,num2,num3   : REAL;
  Scale                : REAL;
  circulo              : Circ;
  drive                : STRING [20];
  Rectan_I             : Rec_Lat;
  Rectan_f             : Rec_frn;
  Fron,Lat,Sup         : BOOLEAN;
  Detec                : CHAR;
  Tex_cor              : STRING;
  Xcor,Xcor_a,Xcorus   : REAL;
  Ycor,Ycor_a,Ycorus   : REAL;
  Zcor,Zcor_a,Zcorus   : REAL;
  Scl                  : REAL;
  Errgrf,Salir         : BOOLEAN;

PROCEDURE Limpia(i : INTEGER);
PROCEDURE Instruccion(renglon,tipo : Byte;instruc : Anystr);
PROCEDURE Vent(i : INTEGER);
PROCEDURE inicia;
PROCEDURE Init_Hta(xin,yin,zin,d : INTEGER);
PROCEDURE M03(si_no : BOOLEAN);
PROCEDURE Traza_lin(Xprg,Yprg,Zprg,Color,Diametro,tipo : INTEGER);
PROCEDURE Traza_cir(Xprg,Yprg: INTEGER; radio : REAL;
                    Color,Diametro,tipo : INTEGER;
                    Xc,Yc               : REAL);
```

```
IMPLEMENTATION

FUNCTION Escala(Xesc : INTEGER):INTEGER;
BEGIN
 Escala:=TRUNC(Xesc/Scale);

END;

PROCEDURE inigraph;

 VAR
  gd,gm   : INTEGER;
  error   : BOOLEAN;
  Ultimo  : CHAR;
 BEGIN
  {Dirgrf:=Dir_ini;}
  Errgrf:=FALSE;
  REPEAT
   error:=TRUE;
   gd:=detect;INITGRAPH(gd,gm,dirgrf);
   if GraphResult <> grOk then BEGIN
    Problemas(ultimo);error:=FALSE;
    If Ultimo = Esc THEN BEGIN
     error:=TRUE;
     Errgrf:=TRUE;
    END;
   END;

  UNTIL error;
 END;



PROCEDURE vent(i:INTEGER);

BEGIN
 CASE i OF
  0 : SETVIEWPORT(0,0,Xmax,Ymax,Clipon);
  1 : SETVIEWPORT(Xos-(Base DIV 2),Yos-(Anc DIV 2),Xos+(Base DIV 2),Yos+(Anc DIV 2),Clipon);
  2 : SETVIEWPORT(Xof-(Altura DIV 2),Yof-(Anc DIV 2),Xof+(Altura DIV 2),Yof+(Anc DIV 2),clipon);
  3 : SETVIEWPORT(Xol-(Base DIV 2),Yol-(Altura DIV 2),Xol+(Base DIV 2),Yol+(Altura DIV 2),clipon);
  4 : SETVIEWPORT(Xmax DIV 3 * 2+1,Ymax DIV 2 +1,Xmax - 10, Ymax DIV 2 + 3 * esp,CLIPON);
 END;
END;

PROCEDURE Pieza;

VAR
 Tex              : STRING;
 Ion,Xesc,i,Divisiones,
 Espacio          : INTEGER;
```

```
 Base_alt,Base_anc,
 Lar_anc,Lar_alt       : REAL;


BEGIN
 clr:=c2;
 { escalamiento }
 Lon:=TRUNC(Largo);
 Esp:=Xmax DIV 20;
 Base:=(Xmax DIV 3 * 2) - Esp * 2;
 Altura:=(Xmax DIV 3) - Esp * 2;
 Anc:=(Ymax DIV 2) - Esp * 2;
 Base_alt:=Base/Altura;
 Base_anc:=Base/Anc;
 Lar_anc:=Largo/Ancho;
 Lar_alt:=Largo/Alto;
 { Da proporciones }
 IF (Lar_alt > Base_alt) AND (Lar_anc > Base_anc) THEN BEGIN
  num1:=Alto;num2:=Base;num3:=Largo;
  num:=(num1*num2)/num3;
  Scale:=num2/num3;
  Altura:=TRUNC(num);
  num1:=ancho;
  num:=(num1*num2)/num3;
  Anc:=TRUNC(num);
 END
 ELSE BEGIN
  {IF (Lar_alt > 1) AND (Lar_anc > 1) THEN BEGIN}
   IF (Base_alt/Lar_alt) < (Base_anc/Lar_anc) THEN BEGIN
    num1:=largo;num2:=anc;
    num3:=Ancho;
    num:=(num1 * num2)/num3 ;
    Scale:=num2/num3;
    Base:=TRUNC(num);
    num1:=Alto;
    num:=(num1*num2)/num3;
    Altura:=TRUNC(num);
   END;
   IF (Base_alt/Lar_alt) > (Base_anc/Lar_anc) THEN BEGIN
    num1:=Largo;num2:=Altura;num3:=Alto;
    num:=(num1*num2)/num3;
    Scale:=num2/num3;
    Base:=TRUNC(num);
    num1:=Ancho;
    num:=(num1*num2)/num3;
    Anc:=TRUNC(num);
   END;
  {END;}
 END;
 { dibuja la pieza}
 SETCOLOR(GREEN);
```

```
SETFILLSTYLE(Solidfill,GREEN);
RECTANGLE(Xos-(Base DIV 2),Yos-(Anc DIV 2),Xos+(Base DIV 2),Yos+(Anc DIV 2));
FLOODFILL(Xos,Yos,clr);
RECTANGLE(Xof-(Altura DIV 2),Yof-(Anc DIV 2),Xof+(Altura DIV 2),Yof+(Anc DIV 2));
FLOODFILL(Xof,Yof,clr);
RECTANGLE(Xol-(Base DIV 2),Yol-(Altura DIV 2),Xol+(Base DIV 2),Yol+(Altura DIV 2));
FLOODFILL(Xol,Yol,clr);
{ escalas}
SETCOLOR(LIGHTRED);
SETTEXTSTYLE(2,0,2);
LINE(Xos-BASE DIV 2,Yos + Anc DIV 2 + esp DIV 2,
   Xos+Base DIV 2,Yos + Anc DIV 2 + esp DIV 2);
LINE(Xos+Base DIV 2,Yos + Anc DIV 2 + esp DIV 10 * 4,
   Xos+Base DIV 2,Yos + Anc DIV 2 + esp DIV 10 * 6);
STR(Largo,tex);

Divisiones:=Base DIV (esp DIV 2);
IF Divisiones > 0 THEN
 espacio:=Base DIV Divisiones + ROUND((Base MOD Divisiones)/Divisiones);
FOR i:=0 TO Divisiones-1 DO BEGIN
 Xesc:=Xos - Base DIV 2 + espacio * i;
 LINE(Xesc,Anc DIV 2 + Yos + TRUNC(esp/2.5),Xesc,Yos + Anc DIV 2 + esp DIV 10 * 6);
 IF i = 0 THEN
   OUTTEXTXY(Xesc,Yos + Anc DIV 2 + esp DIV 4 *3,'0');
 IF i = Divisiones - 1 THEN
   OUTTEXTXY(Xos+Base DIV 2,Yos + Anc DIV 2 + esp DIV 4 *3,Tex);
END;
LINE(Xof+Altura DIV 2 + esp DIV 2,Yos - Anc DIV 2,
   Xof+Altura DIV 2 + esp DIV 2,Yos + Anc DIV 2);
LINE(Xof+altura DIV 2 + esp DIV 10 *4,Yof - Anc DIV 2,
   Xof+altura DIV 2 + esp DIV 10 *6,Yof - Anc DIV 2);
STR(Ancho,Tex);
Divisiones:=Anc DIV (ROUND(esp DIV 2 *1.5));
IF Divisiones > 0 THEN
 espacio:=Anc DIV Divisiones + ROUND((Anc MOD Divisiones)/Divisiones);
FOR i:=0 TO Divisiones -1 DO BEGIN
 Xesc:=Yos + Anc DIV 2 - espacio * i;
 LINE(Xof + Altura DIV 2 + esp * 4 DIV 10,Xesc,
    Xof + Altura DIV 2 + esp * 6 DIV 10,Xesc);
 IF i = 0 THEN
   OUTTEXTXY(Xof + Altura DIV 2 + esp DIV 2,Yof+Anc DIV 2+esp DIV 6,'0');
 IF i = Divisiones - 1 THEN
   OUTTEXTXY(Xof + Altura DIV 2 + esp DIV 10 * 4,Yof-Anc DIV 2 - esp DIV 3,Tex);
END;

LINE(Xol+Base DIV 2 + esp DIV 2,Yol - Altura DIV 2,
   Xol+Base DIV 2 + esp DIV 2,Yol + Altura DIV 2);
LINE(Xol+Base DIV 2 + esp DIV 10 *4,Yol + Altura DIV 2,
   Xol+Base DIV 2 + esp DIV 10 *6,Yol + Altura DIV 2);
STR(Alto,Tex);
Divisiones:=Altura DIV (ROUND(esp DIV 2 * 1.5));
```

```
 IF Divisiones > 0 THEN
   espacio:=Altura DIV Divisiones + ROUND((Altura MOD Divisiones)/Divisiones);
 FOR i:=0 TO Divisiones -1 DO BEGIN
   Xesc:=Yol - Altura DIV 2 + espacio * i;
   LINE(Xol + Base DIV 2 + esp * 4 DIV 10,Xesc,
      Xol + Base DIV 2 + esp * 6 DIV 10,Xesc);
   IF i = 0 THEN
     OUTTEXTXY(Xol + Base DIV 2 + TRUNC(esp/2.3),Yol-Altura DIV 2- TRUNC(esp/2.5),'0');
   IF i = Divisiones - 1 THEN
     OUTTEXTXY(Xol + Base DIV 2 + esp DIV 10 * 3,Yol+Altura DIV 2 + esp DIV 6,Tex);
 END;
 SETTEXTSTYLE(0,0,0);

END;{Pieza}
PROCEDURE Limpia(i : INTEGER);
VAR
 Temp : WORD;

BEGIN
 Temp:=GETCOLOR;
 SETCOLOR(C2);
 SETFILLSTYLE(Solidfill,C2);

 Vent(0);
 CASE i OF
  2 : BEGIN
    Vent(2);
    CLEARVIEWPORT;
    Vent(0);
    RECTANGLE(Xof-(Altura DIV 2),Yof-(Anc DIV 2),Xof+(Altura DIV 2),Yof+(Anc DIV 2));
    FLOODFILL(Xof,Yof,GREEN);

  END;
  3 : BEGIN
    Vent(3);
    CLEARVIEWPORT;
    Vent(0);
    RECTANGLE(Xol-(Base DIV 2),Yol-(Altura DIV 2),Xol+(Base DIV 2),Yol+(Altura DIV 2));
    FLOODFILL(Xol,Yol,GREEN);
  END;
  4 : BEGIN
    Vent(4);
    CLEARVIEWPORT;
    Vent(0);
    SETCOLOR(LIGHTGRAY);
    SETFILLSTYLE(SolidFill,LigHtGRAY);
    RECTANGLE(Xmax DIV 3 * 2+1,Ymax DIV 2 +1,Xmax - 10, Ymax DIV 2 + 3 * esp);
    FLOODFILL(Xmax DIV 6 * 5,Ymax DIV 2 + esp,LIGHTGRAY);

  END;
 END;
```

```
 SETCOLOR(Temp);
END;

PROCEDURE Pon_cor;
VAR
 textaux,texX,texY,TexZ : STRING [40];
 Temp                   : WORD;
 i,Xpan,Ypan,Zpan       : INTEGER;
BEGIN
 Vent(0);
 Temp:=GETCOLOR;
 SETCOLOR(LIGHTGRAY);
 Textaux:='';
 Tex_cor:='';
 Xpan:=Escala(Xa-Base DIV 2 - Xu);
 Ypan:=Escala(-Ya+Anc DIV 2 - Yu);
 Zpan:=Escala(-Za-Zu);
 STR(Xcor_a:6:0,TexX);
 STR(Ycor_a:5:0,TexY);
 STR(Zcor_a:6:0,TexZ);
 Tex_cor:='('+TexX+','+TexY+','+TexZ+')';
 OUTTEXTXY(Xmax * 2 DIV 3 + esp DIV 2,Ymax DIV 2 + esp DIV 2,Tex_cor);
 SETCOLOR(BLUE);
 Textaux:='';
 Tex_cor:='';
 Xpan:=Escala(X-Base DIV 2 - Xu);
 Ypan:=Escala(-Y+Anc DIV 2 - Yu);
 Zpan:=Escala(-Z-Zu);
 STR(Xcor:6:0,TexX);
 STR(Ycor:5:0,TexY);
 STR(Zcor:6:0,TexZ);
 Tex_cor:='('+TexX+','+TexY+','+TexZ+')';
 OUTTEXTXY(Xmax * 2 DIV 3 + esp DIV 2,Ymax DIV 2 + esp DIV 2,Tex_cor);
 SETCOLOR(Temp);
END;

PROCEDURE Get_cor( VAR Xg,Yg,Zg : INTEGER; VAR Sale : CHAR);
VAR
 Coor       : CHAR;
BEGIN

 Xg:=0;Yg:=0;Zg:=0;
 Coor:=Lee(Grafs);
 CASE Coor OF
  Ab  : IF (Sup) OR (Fron) THEN Yg:=1
        ELSE Zg:=1;
  Ar  : IF (Sup) OR (Fron) THEN Yg:=-1
        ELSE Zg:=-1;
  Der : IF Fron THEN Zg:=1
        ELSE Xg:=1;
  Izq : IF Fron THEN Zg:=-1
```

```
      ELSE Xg:=-1;
  F1  : BEGIN
        Fron:=FALSE;
        lat:=FALSE;
        Sup:=TRUE;
       END;
  F2  : BEGIN
        Fron:=TRUE;
        lat:=FALSE;
        Sup:=false;
       END;
  F3  : BEGIN
        Fron:=FALSE;
        lat:=TRUE;
        Sup:=false;
       END;
  ELSE Sale:=coor;
 END;{Case}

END;{Get_cor}

PROCEDURE Miras(i: INTEGER);
BEGIN
 VENT(0);
 CASE i OF
  1 : BEGIN
      SETCOLOR(BLUE);
      { superior }
      IF ( Ya < (Anc + TRUNC(esp*0.9))) AND (Ya > -TRUNC(esp*0.9)) THEN BEGIN
       LINE(Xos-Base DIV 2-TRUNC(esp * 0.75),Yos-Anc DIV 2+Ya,
            Xos-Base DIV 2-TRUNC(esp * 0.25),Yos-Anc DIV 2+Ya);
      END;
      IF (Xa < Base + TRUNC(esp * 0.9))  AND (Xa > -TRUNC(esp * 0.9)) THEN BEGIN
       LINE(Xos-Base DIV 2+Xa,Yos-Anc DIV 2-TRUNC(esp * 0.75),
            Xos-Base DIV 2+Xa,Yos-Anc DIV 2-TRUNC(esp * 0.25));
      END;
      { lateral }
      IF (Za < Altura + TRUNC(esp * 0.9))  AND (Za > -TRUNC(esp * 0.9)) THEN BEGIN
       LINE(Xol-Base DIV 2-TRUNC(esp * 0.75),Yol-Altura DIV 2 + Za,
            Xol-Base DIV 2-TRUNC(esp * 0.25),Yol-Altura DIV 2 + Za);
      END;
      IF (Xa < Base + TRUNC(esp * 0.9))  AND (Xa > -TRUNC(esp * 0.9)) THEN BEGIN
       LINE(Xol-Base DIV 2+Xa,Yol-Altura DIV 2-TRUNC(esp * 0.75),
            Xol-Base DIV 2+Xa,Yol-Altura DIV 2-TRUNC(esp * 0.25));
      END;
      { frontal }
      IF ( Ya < (Anc + TRUNC(esp*0.9))) AND (Ya > -TRUNC(esp*0.9)) THEN BEGIN
       LINE(Xof-Altura DIV 2-TRUNC(esp * 0.75),Yof-Anc DIV 2+Ya,
            Xof-Altura DIV 2-TRUNC(esp * 0.25),Yof-Anc DIV 2+Ya);
      END;
      IF (Za < Altura + TRUNC(esp * 0.9))  AND (Za > -TRUNC(esp * 0.9)) THEN BEGIN
```

```
      LINE(Xof - Altura DIV 2 + Za,Yof-Anc DIV 2-TRUNC(esp * 0.75),
         Xof - Altura DIV 2 + Za,Yof-Anc DIV 2-TRUNC(esp * 0.25));
     END;
     SETCOLOR(LIGHTRED);
     { superior }
     IF (X < Base + TRUNC(esp * 0.9))  AND (X > -TRUNC(esp * 0.9)) THEN BEGIN
      LINE(Xos-Base DIV 2+X,Yos-Anc DIV 2-TRUNC(esp * 0.75),
         Xos-Base DIV 2+X,Yos-Anc DIV 2-TRUNC(esp * 0.25));
     END;
     IF ( Y < (Anc + TRUNC(esp*0.9))) AND (Y > - TRUNC(esp*0.9) ) THEN BEGIN
      LINE(Xos-Base DIV 2-TRUNC(esp * 0.75),Yos-Anc DIV 2+Y,
         Xos-Base DIV 2-TRUNC(esp * 0.25),Yos-Anc DIV 2+Y);
     END;
     { lateral }
     IF (X < Base + TRUNC(esp * 0.9))  AND (X > -TRUNC(esp * 0.9)) THEN BEGIN
      LINE(Xol-Base DIV 2+X,Yol-Altura DIV 2-TRUNC(esp * 0.75),
         Xol-Base DIV 2+X,Yol-Altura DIV 2-TRUNC(esp * 0.25));
     END;
     IF (Z < Altura + TRUNC(esp * 0.9))  AND (Z > -TRUNC(esp * 0.9)) THEN BEGIN
      LINE(Xol-Base DIV 2-TRUNC(esp * 0.75),Yol-Altura DIV 2 +Z,
         Xol-Base DIV 2-TRUNC(esp * 0.25),Yol-Altura DIV 2 +Z);
     END;
     { frontal }
     IF ( Y < (Anc + TRUNC(esp*0.9))) AND (Y > -TRUNC(esp*0.9)) THEN BEGIN
      LINE(Xof-Altura DIV 2-TRUNC(esp * 0.75),Yof-Anc DIV 2+Y,
         Xof-Altura DIV 2-TRUNC(esp * 0.25),Yof-Anc DIV 2+Y);
     END;
     IF (Z < Altura + TRUNC(esp * 0.9))  AND (Z > -TRUNC(esp * 0.9)) THEN BEGIN
      LINE(Xof - Altura DIV 2 + Z,Yof-Anc DIV 2-TRUNC(esp * 0.75),
         Xof - Altura DIV 2 + Z,Yof-Anc DIV 2-TRUNC(esp * 0.25));
     END;

     SETCOLOR(clr);
    END;
 END;

END;{miras}

PROCEDURE Instruccion(renglon,tipo : BYTE;instrucc : Anystr);
VAR
 Temp  : WORD;

 BEGIN
   Temp:=GETCOLOR;
   Vent(0);
   SETTEXTSTYLE(2,0,4);
   CASE  tipo OF
     0 : SETCOLOR(LIGHTGRAY);
     1 : SETCOLOR(BLUE);
     2 : SETCOLOR(RED);
   END;
```

```
  OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/3),Ymax DIV 2 + TRUNC(esp*(1.5 + (renglon*0.5))),
       instrucc);
  SETCOLOR(temp);
  SETTEXTSTYLE(0,0,0);
END;

PROCEDURE Inicia;

BEGIN
 {Largo:=3600;
 Alto:=1500;
 Ancho:=1750;}

 detec:='o';
 {write('Drive ');readln(drive);drive:='C:\turbo6\BGI';}
 inigraph;
 Xmax:=GETMAXX;Ymax:=GETMAXY;
 Xos:=(Xmax * 2) DIV 6;Yos:=Ymax DIV 4;
 Xof:=(Xmax * 5) DIV 6;Yof:=Ymax DIV 4;
 Xol:=(Xmax * 2) DIV 6;Yol:=(Ymax * 3) DIV 4;
 SETVIEWPORT(0,0,Xmax,Ymax,CLIPON);
 clr:=BLUE;
 SETBKCOLOR(clr);
 SETCOLOR(LIGHTGRAY);
 CLEARVIEWPORT;
 { Margenes }

 SETLINESTYLE(0,0,3);
 RECTANGLE(1,1,Xmax-1,Ymax-1);
 SETLINESTYLE(0,0,1);
 RECTANGLE(2,2,(Xmax DIV 3)*2,Ymax DIV 2);
 RECTANGLE(2,Ymax DIV 2,(Xmax DIV 3)*2,Ymax);
 RECTANGLE((Xmax DIV 3)*2,Ymax DIV 2,Xmax,Ymax);
 SETFILLSTYLE(SOLIDFILL,LIGHTGRAY);
 FLOODFILL(Xof,Yol,LIGHTGRAY);


 { Pieza }
 Pieza;
 SETCOLOR(BLUE);
 SETTEXTSTYLE(2,0,4);
 OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/3),Ymax DIV 2 + TRUNC(esp*1),
      ' N G/M  X   Y   Z   F');
 OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/2),Ymax DIV 2 + TRUNC(esp*3),
      'SPC BAR  Continua');
 OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/2),Ymax DIV 2 + TRUNC(esp*3.5),
      'ENTER   Pausa');
 OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/2),Ymax DIV 2 + TRUNC(esp*4),
      'F1      Limpia V. Lateral');
 OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/2),Ymax DIV 2 + TRUNC(esp*4.5),
      'F2      Trazo  V. Lateral');
```

```
OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/2),Ymax DIV 2 + TRUNC(esp*5),
     'F3      Limpia V. Frontal');
OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/2),Ymax DIV 2 + TRUNC(esp*5.5),
     'F4      Trazo  V. Frontal');
OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/2),Ymax DIV 2 + TRUNC(esp*6),
     'ESC     Salir');
OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp/2),Ymax DIV 2 + TRUNC(esp*6.5),
     'Husillo');Hus_stat:='Apagado';
OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp*2.23),Ymax DIV 2 + TRUNC(esp*6.5),
     'Apagado');
SETTEXTSTYLE(0,0,0);


{ Ejes}
SETCOLOR(Lightred);
LINE(Xos,Ymax DIV 2,Xos + Base DIV 2,Ymax DIV 2);
LINE(Xos + Base DIV 2,Ymax DIV 2,Xos + Base DIV 2 - esp DIV 4,Ymax DIV 2 - esp DIV 4);
LINE(Xos + Base DIV 2,Ymax DIV 2,Xos + Base DIV 2 - esp DIV 4,Ymax DIV 2 + esp DIV 4);
OUTTEXTXY(Xos + Base Div 2 + esp DIV 2,Ymax DIV 2 + esp DIV 2,'X');
LINE(Xmax DIV 3 * 2,Yos,Xmax DIV 3 * 2,Yos - Anc DIV 2);
LINE(Xmax DIV 3 * 2,Yos - Anc DIV 2,Xmax DIV 3 * 2 - esp DIV 4,Yos - Anc DIV 2 + esp DIV 4);
LINE(Xmax DIV 3 * 2,Yos - Anc DIV 2,Xmax DIV 3 * 2 + esp DIV 4,Yos - Anc DIV 2 + esp DIV 4);
OUTTEXTXY(Xmax * 2 DIV 3 - esp DIV 2,Yos - Anc  DIV 2 - esp DIV 2,'y');
LINE(Xof-Altura DIV 2,Ymax DIV 2,Xof + Altura DIV 2,Ymax DIV 2);
LINE(Xof + Altura DIV 2,Ymax DIV 2,Xof + Altura DIV 2 - esp DIV 4,Ymax DIV 2- esp DIV 4);
OUTTEXTXY(Xof + Altura DIV 2 + esp DIV 4,Ymax DIV 2 - esp DIV 2,'-Z');
LINE(Xmax DIV 3 * 2,Yol - Altura DIV 2,Xmax DIV 3 * 2,Yol + Altura DIV 2);
LINE(Xmax DIV 3 * 2,Yol + Altura DIV 2,Xmax DIV 3 * 2 - esp DIV 4,Yol + Altura DIV 2 - esp DIV 4);
OUTTEXTXY(Xmax * 2 DIV 3 - TRUNC(esp/1.55) ,Yol + altura DIV 2 + esp DIV 2,'-Z');



Stat_fron:=1;Stat_lat:=1;
trz_rl:=1;trz_rf:=1;
trz:=1;
clr:=C4;
SETCOLOR(clr);

scl:=Base/largo;
Zdesp:=Zus;
Xu:=ROUND(Xus  * scl);
Yu:=-ROUND(Yus * scl);
Zu:=-ROUND(Zus * scl);
Xorigen:=Xu;
Yorigen:=Anc + Yu;
Zorigen:=0 + Zu;

X:=Xorigen;
Y:=Yorigen;
Z:=Zorigen;
Xa:=X;
```

```
 Ya:=Y;
 Za:=Z;
 Xi:=0;Yi:=0;Zi:=0;
 Xcorus:=0;Ycorus:=0;Zcorus:=0;
 Xcor:=0+Xcorus;Ycor:=0+Ycorus;Zcor:=0+Zcorus;

 Sup:=TRUE;
 Fron:=FALSE;
 Lat:=false;
 Subeder:=FALSE;
 Subelzq:=FALSE;
 Dercarr:=FALSE;
 Dereaba:=FALSE;
 Salir:=FALSE;
 Circulo.Init(X,Y,5);
 Vent(2);
 Rectan_f.Init(Z,Y,5);
 Vent(3);
 Rectan_l.Init(X,Z,5);
END;

PROCEDURE Init_Hta(xin,yin,zin,d : INTEGER);

BEGIN
 scl:=Base/Largo;
 X:=Xorigen+ROUND(Xin * Scl);
 Y:=Yorigen-ROUND(Yin * Scl);
 Z:=Zorigen-ROUND(Zin * Scl);

 d:=ROUND(d*Base/Largo);
 Circulo.Init(X,Y,d);
 Vent(2);
 Rectan_f.Init(Z,Y,D);
 Vent(3);
 Rectan_l.Init(X,Z,D);
 Xcor:=Xin;Ycor:=Yin;Zcor:=Zin;
 Miras(1);
 Pon_cor;
END;

PROCEDURE M03(si_no : BOOLEAN);
BEGIN
 Vent(0);
 SETCOLOR(LIGHTGRAY);
 SETTEXTSTYLE(2,0,4);
 OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp*2.23),Ymax DIV 2 + TRUNC(esp*6.5),
     Hus_stat);
 IF si_no THEN BEGIN
  Hus_stat:='Encendido';
  SETCOLOR(RED);
 END
```

```
 ELSE BEGIN
  SETCOLOR(BLUE);
  Hus_stat:='Apagado';
 END;
 OUTTEXTXY(Xmax * 2 DIV 3 + ROUND(esp*2.23),Ymax DIV 2 + TRUNC(esp*6.5),
     Hus_stat);
 SETTEXTSTYLE(0,0,0);

END;


PROCEDURE Traza_lin(Xprg,Yprg,Zprg,Color,Diametro,tipo : INTEGER);
VAR
 Xaprg,Yaprg,Entero,Entero_a : INTEGER;
 Entdep,Entdep_a,ciclos      : INTEGER;
 Mlinea,Xlinea,Ylinea        : REAL;
 IndX,IndY,IndZ,Cambio,ciclo : BOOLEAN;
 CambioX,CambioY,CambioZ,Cam2: BOOLEAN;
 DepX,DepY,DepZ,Pausa,Cam,Int: BOOLEAN;
 Final,Sign,Zaprg,cont       : INTEGER;
 Xinicial,Yinicial,Zinicial  : INTEGER;
 Xcorini,Cor,Ycorini         : REAL;
 Zcorini                     : REAL;
BEGIN



 Cambio:=FALSE;CambioX:=FALSE;CambioY:=FALSE;CambioZ:=FALSE;
 Down:=FALSE;Rigth:=FALSE;
 Vert:=False;Horz:=FALSE;
 Dere:=FALSE;Arriba:=FALSE;
 Izqi:=FALSE;Abajo:=FALSE;
 Diagonal:=FALSE;
 Trz:=Tipo;
 Clr:=Color;
 Scl:=Base/Largo;
 Diametro:=ROUND(Diametro*scl);
 Xaprg:=Xprg;Yaprg:=Yprg;Zaprg:=Zprg;
 Xcorini:=Xcor;Ycorini:=Ycor;Zcorini:=Zcor;
 Xprg:=ROUND(Xprg);
 Yprg:=ROUND(-Yprg);
 Zprg:=ROUND(-Zprg);
 Xa:=X;Ya:=Y;Za:=Z;
 Xo:=X;Yo:=Y;Zo:=Z;
 Mlinea:=0;
 XLinea:=0;
 Ylinea:=0;
 IndX:=FALSE;DepX:=FALSE;
 IndY:=FALSE;DepY:=FALSE;
 IndZ:=FALSE;DepZ:=FALSE;
 Int:=TRUE;
```

```
Vent(1);
Circulo.Init(X,Y,Diametro);

Vent(2);
Rectan_f.Init(Z,Y,Diametro);
Vent(3);
Rectan_l.Init(X,Z,Diametro);
IF Xprg <> 0 THEN BEGIN
  IF Yprg <> 0 THEN BEGIN
    Mlinea:=Yprg/Xprg;
    DepY:=TRUE;
    IF ABS(Mlinea) > 1 THEN BEGIN
      Mlinea:=1/Mlinea;
      DepY:=FALSE;
      IndY:=TRUE;
      DepX:=TRUE;
      IndX:=FALSE;
      Final:=Yaprg;
      Sign:=ROUND(ABS(Yprg)/Yprg);
    END;
  END;
  IF Zprg <> 0 THEN BEGIN
    Mlinea:=Zprg/Xprg;
    DepZ:=TRUE;
    IndX:=TRUE;
    Diagonal:=TRUE;
  END;
  IF IndY = FALSE THEN BEGIN
    IndX:=TRUE;
    Sign:=ROUND(ABS(Xprg)/Xprg);
    Final:=Xaprg;
  END;
END;

IF (Yprg <> 0) AND (Xprg = 0)  THEN BEGIN
  IF Zprg <> 0 THEN BEGIN
    Mlinea:=Zprg/Yprg;
    DepZ:=TRUE;
    Diagonal:=TRUE;
  END;
  IF Zprg = 0 THEN BEGIN
    DepZ:=TRUE;
    Mlinea:=0;
  END;
  IndY:=TRUE;
  Sign:=ROUND(ABS(Yprg)/Yprg);
  Final:=Yaprg;
END
ELSE
IF (Yprg = 0) AND (Xprg = 0) THEN BEGIN
  IF Zprg <> 0 THEN BEGIN
```

```
    IndZ:=TRUE;
    Sign:=ROUND(ABS(Zprg)/Zprg);
    Final:=Zaprg;
    DepZ:=FALSE;
   END;
   IF Zprg = 0 THEN
   BEGIN
    IndZ:=TRUE;
    Final:=Zaprg;
    DepZ:=FALSE;
   END;
END;
Ciclos:=TRUNC (ABS(Final*scl));
IF ciclos > 1 THEN ciclo:=TRUE
ELSE ciclo:=FALSE;

{STR(Final,tex_cor);
TEXTCOLOR(BLUE);
Vent(0);
OUTTEXTXY(Xmax-150,Ymax-30,Tex_cor);
STR(Xaprg,tex_cor);
TEXTCOLOR(BLUE);
Vent(0);
OUTTEXTXY(Xmax-100,Ymax-30,Tex_cor);
readln;}

Xinicial:=X;Yinicial:=Y;Zinicial:=Z;
Camlzq:=FALSE;
Camder:=FALSE;
Camarr:=FALSE;
Cam:=TRUE;Cam2:=TRUE;
Camab:=FALSE;
Cont:=1;
 REPEAT

    Vent(0);

    Pon_cor;

    {IF CambioX THEN BEGIN
     Camlzq:=FALSE;
     Camder:=FALSE;
    END;
    IF CambioY THEN BEGIN
     Camarr:=FALSE;
     CAmab:=FALSE;
    END;}


    Vent(1);
```

```
IF Cambio THEN BEGIN
 IF CambioZ THEN BEGIN
  IF Za <> 0 THEN Zo:=Za;
  Za:=Z;
 END;
 IF CambioX THEN BEGIN
   IF Xa <> 0 THEN Xo:=Xa;
   Xa:=X;
 END;
 IF CambioY THEN BEGIN
   IF Ya <> 0 THEN Yo:=Ya;
   Ya:=Y;
 END;

END;

Xcor_a:=Xcor;Ycor_a:=Ycor;Zcor_a:=Zcor;

{Get_Cor(Xi,Yi,Zi,Detec);}

Entero_a:=TRUNC(Xlinea*scl);
Entdep_a:=TRUNC(Ylinea*scl);

Xlinea:=Xlinea+1*sign;
IF ciclo THEN BEGIN
 Xlinea:=Xlinea+1/scl*sign;
 IF Trz = 1 THEN DELAY(60)
 ELSE DELAY(30);

END;

Ylinea:=Mlinea*Xlinea;

Entdep:=TRUNC(Ylinea*scl);
Entero:=TRUNC(Xlinea*scl);



Xi:=0;Yi:=0;Zi:=0;
IF IndX THEN BEGIN
 Xi:=ROUND(Xlinea);
 Xcor:=Xcorini+Xlinea;
 Cor:=Xlinea;
END;
IF IndY THEN BEGIN
 Yi:=ROUND(Xlinea);
 Ycor:=Ycorini-Xlinea;
 Cor:=-Xlinea;
END;
IF IndZ THEN  BEGIN
```

```
  Zi:=ROUND(Xlinea);
  Zcor:=Zcorini-Xlinea;
  Cor:=-Xlinea;
 END;
 IF DepX THEN BEGIN
  Xi:=ROUND(Ylinea);
  Xcor:=Xcorini+Ylinea;
 END;
 IF DepY THEN BEGIN
  Yi:=ROUND(Ylinea);
  Ycor:=Ycorini-Ylinea;
 END;
 IF DepZ THEN BEGIN
  Zi:=ROUND(Ylinea);
  Zcor:=Zcorini-Ylinea;
 END;
 Xi:=ROUND(scl*Xi);Yi:=ROUND(scl*Yi);Zi:=ROUND(scl*Zi);
 {IF ciclo THEN BEGIN
  Xi:=ROUND(Xi);Yi:=ROUND(Yi);Zi:=ROUND(Zi);
 END;}
 Cambio:=FALSE;CambioX:=FALSE;CambioY:=FALSE;CambioZ:=FALSE;
 IF {(Xi<>X-Xinicial) OR (Yi<>Y-Yinicial) OR (Zi<>Z-Zinicial)} entero<> entero_a THEN BEGIN
  IF indX{Xi <> X-Xinicial} THEN BEGIN
   CambioX:=TRUE;
   {Xo:=Xa;Xa:=X;}
   X:=Xinicial+Xi;
  END;
  IF indY{Yi<> 0} THEN BEGIN
   {Yo:=Ya;Ya:=Y;}
   Y:=Yinicial+Yi;
   CambioY:=TRUE;
  END;
  IF IndZ THEN BEGIN
   {Zo:=Za;Za:=Z;}
   Z:=Zinicial+Zi;
   CambioZ:=TRUE;
  END;
  IF DepY THEN BEGIN
   IF entdep <> entdep_a THEN BEGIN
    {Yo:=Ya;Ya:=Y;}
    Y:=Yinicial+Yi;
    CambioY:=TRUE;
   END;

  END;
  IF DepX THEN BEGIN
   IF entdep <> entdep_a THEN BEGIN
    X:=Xinicial+Xi;
    CambioX:=TRUE;
   END;
  END;
```

```
 IF Depz THEN BEGIN
  IF entdep <> entdep_a THEN BEGIN
   Z:=Zinicial+Zi;
   CambioZ:=TRUE;
  END;

 END;
 CAmbio:=TRUE;

END;

IF (X-Diametro > Base) OR (X+Diametro < 0) OR
(Y-Diametro > Anc)  OR (Y+Diametro < 0) THEN BEGIN
 Trz_rl:=0;
 Trz_rf:=0;
END;

 IF (X-Diametro <= Base) AND (X+Diametro >= 0) AND
 (Y-Diametro <= Anc)  AND (Y+Diametro >= 0) THEN BEGIN
  IF Stat_lat = 0 THEN Trz_rl:=Stat_lat
  ELSE Trz_rl:=1;
  IF Stat_fron = 0 THEN Trz_rf:=Stat_fron
  ELSE Trz_rf:=1;
 END;


IF Zcor > Zdesp THEN Trz:=0;
Circulo.Movcir(X,Y,clr,trz);
Miras(1);

Vent(2);

IF ((Zo = Z) AND ( Z < Za)) OR ((Z < Za) AND (Y <> Yo)) THEN Rigth:=TRUE
EISE Rigth:=False;
IF trz = 0 THEN BEGIN
 Arriba:=FALSE;Abajo:=FALSE;
END;
IF Z > Za THEN BEGIN
 derearr:=FALSE;
 Dereaba:=FALSE;
END;

IF Zo <> Z THEN BEGIN
 Horz:= TRUE;
END
ELSE Horz:=FALSE;
IF (Zo = Z) AND (Z < Za) AND (Xa = X) THEN BEGIN
 Derearr:=FALSE;
 Dereaba:=FALSE;
END;
```

```
IF (Zo = Za) AND (Y=Ya) THEN BEGIN
  Rigth:=TRUE;
  Arriba:=FALSE;
  Abajo:=FALSE;
END;

IF Y < Ya THEN BEGIN
  Arriba:=TRUE;Abajo:=FALSE;
  Derearr:=TRUE;Dereaba:=FALSE;
END;

IF Y > Ya THEN BEGIN
  Abajo:=TRUE;Arriba:=FALSE;
  Dereaba:=TRUE;Derearr:=FALSE;
END;

{IF ((Y = Yo) OR (Yo = Ya)) AND (Y<Ya) THEN BEGIN
  Camab:=TRUE;
  Abajo:=FALSE;
  Arriba:=FALSE;
END;
IF ((Y = Yo) OR (Yo = Ya)) AND (Y>Ya) THEN BEGIN
  Camarr:=TRUE;
  Abajo:=FALSE;
  Arriba:=FALSE;
END;}
IF (Camarr OR Camab) {AND (Y <> Ya )} THEN BEGIN
  Camarr:=FALSE;
  Camab:=FALSE;
  Cam:=FALSE;
END;
IF Cam=TRUE  THEN BEGIN
  IF ((Y = Yo) OR (Yo = Ya)) {AND (Y<Ya)} THEN BEGIN
    IF (Y < Ya) AND (Camab = FALSE) THEN BEGIN
      Camab:=TRUE;
      Camarr:=FALSE;
      Abajo:=FALSE;
      Arriba:=FALSE;
    END;
  END;
  IF (((Y = Yo) OR) (Yo = Ya)) {AND (Y>Ya)} THEN BEGIN
    IF (Y > Ya) AND (Camarr = FALSE) THEN BEGIN
      Camarr:=TRUE;
      Camab:=FALSE;
      Abajo:=FALSE;
      Arriba:=FALSE;
    END;
  END;
END;
```

```
Rectan_F.Mov_rec(Z,Y,trz_rf);

Vent(3);

IF ((Zo = Z) AND ( Z < Za)) OR ((Z < Za) AND (X <> Xo)) THEN Down:=true
EISE Down:=False;
IF trz = 0 THEN BEGIN
  Dere:=FALSE;Izqi:=FALSE;
END;
IF Z > Za THEN BEGIN
  Subeder:=FALSE;
  Subeizq:=FALSE;

END;
IF Zo <> Z THEN BEGIN
  Vert:= TRUE;
END
ELSE Vert:=FAlse;
IF (Zo = Z) AND (Z < Za) AND (Xa = X) THEN BEGIN
  subeder:=FALSE;
  subeizq:=FALSE;
END;
IF (Zo = Za) AND (X=Xa) THEN BEGIN
  Down:=TRUE;
  Dere:=FALSE;
  Izqi:=FALSE;
END;
IF X > Xa THEN BEGIN
  Dere:=TRUE;Izqi:=FALSE;
  Subeder:=TRUE;Subeizq:=FALSE;
END;
IF dere THEN Subeder:=TRUE;
IF izqi THEN Subeizq:=TRUE;
IF X < Xa THEN BEGIN
  Izqi:=TRUE;Dere:=FALSE;
  Subeizq:=TRUE;Subeder:=FALSE;
END;
IF (diagonal) AND ( Zo <> Za) THEN BEGIN
  Sub_dil:=FALSE;
  Sub_dif:=FALSE;
END;
IF ((Z = Zo) ) AND (Diagonal) THEN BEGIN
  IF X<>Xa THEN Sub_dil:=TRUE;
  IF Y<>Ya THEN Sub_dif:=TRUE;
END;
IF (diagonal) AND ( Zo <> Za) THEN BEGIN
  Sub_dil:=FALSE;
  Sub_dif:=FALSE;
END;
```

```
{IF ((X = Xo) OR (Xa = Xo)) AND (X>Xa) THEN Camizq:=TRUE;
IF ((X = Xo) OR (Xa = Xo)) AND (X<Xa) THEN Camder:=TRUE;}
IF (Camder OR Camizq) {AND (Y <> Ya )} THEN BEGIN
 Camder:=FALSE;
 Camizq:=FALSE;
 Cam2:=FALSE;
END;
IF Cam2=TRUE  THEN BEGIN
 IF ((X = Xo) OR (Xo = Xa)) {AND (X<Xa)} THEN BEGIN
  IF (X > Xa) AND (Camizq = FALSE) THEN BEGIN
   Camizq:=TRUE;
   Camder:=FALSE;
   Dere:=FALSE;
   Izqi:=FALSE;
  END;
 END;
 IF (((X = Xo) OR) (Xo = Xa)) {AND (X>Xa)} THEN BEGIN
  IF (X < Xa) AND (Camder = FALSE) THEN BEGIN
   Camder:=TRUE;
   Camizq:=FALSE;
   Dere:=FALSE;
   Izqi:=FALSE;
  END;
 END;
END;

Rectan_I.Mov_rec(X,Z,trz_rl);


IF KEYPRESSED THEN BEGIN

 Pausa:=TRUE;
 Detec:=Lee(grafs);
 IF Detec = ESC THEN Salir:=TRUE;
 IF Detec = ENTER THEN BEGIN
  Pausa:=FALSE;

  REPEAT


  {IF Detec = ENTER THEN BEGIN
    Pausa:=FALSE;}
    Detec:=Lee(grafs);
    CASE Detec OF
      {F4 : BEGIN
           clr:=clr+1;
           IF clr > C15 THEN clr:=c3;
           detec:='o';
          END;
       F5 : BEGIN
           trz:=trz+1;
```

```
            IF trz = 4 THEN trz:=1;
            detec:='o';
           END;}
        F2 : BEGIN
            IF trz_rl = 0 THEN trz_rl:=1
            ELSE trz_rl:=0;
            Stat_lat:=Trz_rl;
            detec:='o';
           END;
        F4 : BEGIN
            IF trz_rf = 0 THEN trz_rf:=1
            ELSE trz_rf:=0;
            Stat_fron:=Trz_rf;
            detec:='o';
           END;
        F3 : BEGIN
            Limpia(2);
            detec:='o';
            Vent(2);
            Rectan_f.Show;
           END;
        F1 : BEGIN
            Limpia(3);
            detec:='o';
            vent(3);
            Rectan_l.Show;
           END;
       ENTER : BEGIN
            {IF Pausa THEN} Pausa:=FALSE;
            {ELSE Pausa:=TRUE;}
            Detec:='o';
           END;
       ESC   : BEGIN
            Salir:=TRUE;
           END;
       Spc   : BEGIN
            Pausa:=TRUE;
            Detec:='o';
           END;
     END;{CASE}
    {END {Pausa
    ELSE Pausa:=TRUE;}
    UNTIL (Pausa = TRUE) OR (Salir);
  END;{ IF Pausa}
 END;{ IF Keypressed}
 IF cont >= ciclos-1 THEN ciclo:=FALSE;
 cont:=cont+1;
UNTIL (Detec = ESC) OR (TRUNC(ABS(Cor)) >= ABS(Final) ) OR (Salir);
{Xcor_a:=Xcor;Ycor_a:=Ycor;Zcor_a:=Zcor;}
Xcor:=Xcorini+Xaprg;Ycor:=Ycorini+Yaprg;Zcor:=Zcorini+Zaprg;
Pon_cor;
```

```
 IF trz = 1 THEN BEGIN
   Vent(1);
   SETCOLOR(color);
   Circulo.Hide;
 END;
END;

PROCEDURE Traza_cir(Xprg,Yprg: INTEGER; radio : REAL;
                    Color,Diametro,tipo : INTEGER;
                    Xc,Yc               : REAL);
VAR
 Xaprg,Yaprg,Entero,Entero_a : INTEGER;
 Entdep,Entdep_a,ciclos,Zprg : INTEGER;
 Mlinea,Xlinea,Ylinea        : REAL;
 IndX,IndY,IndZ,Cambio,ciclo : BOOLEAN;
 CambioX,CambioY,CambioZ     : BOOLEAN;
 DepX,DepY,DepZ,Pausa,Cam    : BOOLEAN;
 Final,Sign,Zaprg,cont,signY : INTEGER;
 Xinicial,Yinicial,Zinicial  : INTEGER;
 Scl,Xcorini,Cor,Ycorini     : REAL;
 Zcorini,radioint            : REAL;
BEGIN
 Cambio:=FALSE;CambioX:=FALSE;CambioY:=FALSE;CambioZ:=FALSE;
 Down:=FALSE;Rigth:=FALSE;
 Vert:=False;Horz:=FALSE;
 Dere:=FALSE;Arriba:=FALSE;
 Izqi:=FALSE;Abajo:=FALSE;
 Diagonal:=FALSE;
 Zprg:=0;
 Trz:=1;
 Clr:=Color;
 Scl:=Base/Largo;
 Diametro:=ROUND(Diametro*scl);
 Xaprg:=Xprg;Yaprg:=Yprg;Zaprg:=Zprg;
 Xcorini:=Xcor;Ycorini:=Ycor;Zcorini:=Zcor;
 Xprg:=(Xprg);
 Yprg:=(-Yprg);
 Zprg:=(-Zprg);
 Xa:=X;Ya:=Y;Za:=Z;
 Xo:=X;Yo:=Y;Zo:=Z;
 {Mlinea:=0;}
 XLinea:=0;
 Ylinea:=0;
 radioint:=SQR(radio);


 IndX:=TRUE;DepX:=FALSE;
 DepY:=TRUE;IndY:=FALSE;
 IndZ:=FALSE;DepZ:=FALSE;
 Sign:=ROUND(ABS(Xprg)/Xprg);
 SignY:=ROUND(ABS(Yprg)/Yprg);
```

```
Final:=Xaprg;
Vent(1);
Circulo.Init(X,Y,Diametro);

Vent(2);
Rectan_f.Init(Z,Y,Diametro);
Vent(3);
Rectan_l.Init(X,Z,Diametro);



{STR(Final,tex_cor);
TEXTCOLOR(BLUE);
Vent(0);
OUTTEXTXY(Xmax-150,Ymax-30,Tex_cor);
STR(radioint:8:2,tex_cor);
TEXTCOLOR(BLUE);
Vent(0);
OUTTEXTXY(Xmax-100,Ymax-30,Tex_cor);}

Xinicial:=X;Yinicial:=Y;Zinicial:=Z;
Camlzq:=FALSE;
Camder:=FALSE;
Camarr:=FALSE;
CAmab:=FALSE;
Cam:=TRUE;
{Cont:=1;}
 REPEAT

   Vent(0);

   Pon_cor;{repeat until keypressed;}
   IF CambioX THEN BEGIN
     Camlzq:=FALSE;
     Camder:=FALSE;
   END;
   {IF CambioY THEN BEGIN
     Camarr:=FALSE;
     CAmab:=FALSE;
   END;}

   Vent(1);

   IF Cambio THEN BEGIN
     {IF CambioZ THEN BEGIN}
     IF Za <> 0 THEN Zo:=Za;
     Za:=Z;
     {END;}
     IF CambioX THEN BEGIN
       IF Xa <> 0 THEN Xo:=Xa;
```

```
  Xa:=X;
 END;
 IF CambioY THEN BEGIN
  IF Ya <> 0 THEN Yo:=Ya;
  Ya:=Y;
 END;

END;

Xcor_a:=Xcor;Ycor_a:=Ycor;Zcor_a:=Zcor;

{Get_Cor(Xi,Yi,Zi,Detec);}

Entero_a:=TRUNC(Xlinea*scl);
Entdep_a:=TRUNC(Ylinea*scl);

Xlinea:=Xlinea+1*sign;
{IF ciclo THEN BEGIN
 Xlinea:=Xlinea+1/scl*sign;
 IF Trz = 1 THEN DELAY(80)
 ELSE DELAY(40);
END;}



Ylinea:=signY*(Yc+ tipo * SQRT(ABS(Radioint - SQR(Xlinea-Xc))));

Entdep:=TRUNC(Ylinea*scl);
Entero:=TRUNC(Xlinea*scl);



Xi:=0;Yi:=0;Zi:=0;
IF IndX THEN BEGIN
 Xi:=ROUND(Xlinea);
 Xcor:=Xcorini+Xlinea;
 Cor:=Xlinea;
END;
{IF IndY THEN BEGIN
 Yi:=ROUND(Xlinea);
 Ycor:=Ycorini-Xlinea;
 Cor:=-Xlinea;
END;}
{IF IndZ THEN BEGIN
 Zi:=ROUND(Xlinea);
 Zcor:=Zcorini-Xlinea;
 Cor:=-Xlinea;
END;
IF DepX THEN BEGIN
 Xi:=ROUND(Ylinea);
```

```
  Xcor:=Xcorini+Ylinea;
END;}
IF DepY THEN BEGIN
  Yi:=ROUND(Ylinea);
  Ycor:=Ycorini-Ylinea;
END;
{IF DepZ THEN BEGIN
  Zi:=ROUND(Ylinea);
  Zcor:=Zcorini-Ylinea;
 END;}
Xi:=ROUND(scl*Xi);Yi:=ROUND(scl*Yi);Zi:=ROUND(scl*Zi);
{IF ciclo THEN BEGIN
  Xi:=ROUND(Xi);Yi:=ROUND(Yi);Zi:=ROUND(Zi);
END;}
Cambio:=FALSE;CambioX:=FALSE;CambioY:=FALSE;CambioZ:=FALSE;

IF  entero<> entero_a THEN BEGIN
  IF indX THEN BEGIN
    CambioX:=TRUE;
    X:=Xinicial+Xi;
  END;

  CAmbio:=TRUE;
END;
IF  entdep<> entdep_a THEN BEGIN
  {IF indX THEN BEGIN
    CambioX:=TRUE;
    X:=Xinicial+Xi;
  END;}


  IF DepY THEN BEGIN
     Y:=Yinicial+Yi;
     CambioY:=TRUE;
  END;
  CAmbio:=TRUE;
END;

IF (X-Diametro > Base) OR (X+Diametro < 0) OR
(Y-Diametro > Anc)  OR (Y+Diametro < 0) THEN BEGIN
  Trz_rl:=0;
  Trz_rf:=0;
 END;

 IF (X-Diametro <= Base) AND (X+Diametro >= 0) AND
(Y-Diametro <= Anc)  AND (Y+Diametro >= 0) THEN BEGIN
  IF Stat_lat = 0 THEN Trz_rl:=Stat_lat
  ELSE Trz_rl:=1;
  IF Stat_fron = 0 THEN Trz_rf:=Stat_fron
  ELSE Trz_rf:=1;
 END;
```

```
Circulo.Movcir(X,Y,clr,trz);
Miras(1);

Vent(2);

IF ((Zo = Z) AND ( Z < Za)) OR ((Z < Za) AND (Y <> Yo)) THEN Rigth:=TRUE
ELSE Rigth:=False;
IF trz = 0 THEN BEGIN
 Arriba:=FALSE;Abajo:=FALSE;
END;
IF Z > Za THEN BEGIN
 dercarr:=FALSE;
 Dereaba:=FALSE;
END;

IF Zo <> Z THEN BEGIN
 Horz:= TRUE;
END
ELSE Horz:=FALSE;
IF (Zo = Z) AND (Z < Za) AND (Xa = X) THEN BEGIN
 Derearr:=FALSE;
 Dereaba:=FALSE;
END;


IF (Zo = Za) AND (Y=Ya) THEN BEGIN
 Rigth:=TRUE;
 Arriba:=FALSE;
 Abajo:=FALSE;
END;

IF Y < Ya THEN BEGIN
 Arriba:=TRUE;Abajo:=FALSE;
 Derearr:=TRUE;Dereaba:=FALSE;
END;

IF Y > Ya THEN BEGIN
 Abajo:=TRUE;Arriba:=FALSE;
 Dereaba:=TRUE;Derearr:=FALSE;
END;
IF (Camarr OR Camab) {AND (Y <> Ya )} THEN BEGIN
 Camarr:=FALSE;
 Camab:=FALSE;
 Cam:=FALSE;
END;
IF Cam=TRUE  THEN BEGIN
 IF ((Y = Yo) OR (Yo = Ya)) {AND (Y<Ya)} THEN BEGIN
  IF (Y < Ya) AND (Camab = FALSE) THEN BEGIN
   Camab:=TRUE;
```

```
    Camarr:=FALSE;
    Abajo:=FALSE;
    Arriba:=FALSE;
   END;
  END;
  IF (((Y = Yo) OR) (Yo = Ya)) (AND (Y>Ya)) THEN BEGIN
   IF (Y > Ya) AND (Camarr = FALSE) THEN BEGIN
    Camarr:=TRUE;
    Camab:=FALSE;
    Abajo:=FALSE;
    Arriba:=FALSE;
   END;
  END;
 END;

Rectan_F.Mov_rec(Z,Y,trz_rf);

Vent(3);

IF ((Zo = Z) AND ( Z < Za)) OR ((Z < Za) AND (X <> Xo)) THEN Down:=true
ElSE Down:=False;
IF trz = 0 THEN BEGIN
 Dere:=FALSE;Izqi:=FALSE;
END;
IF Z > Za THEN BEGIN
 Subeder:=FALSE;
 Subeizq:=FALSE;
END;
IF Zo <> Z THEN BEGIN
 Vert:= TRUE;
END
ELSE Vert:=FAlse;
IF (Zo = Z) AND (Z < Za) AND (Xa = X) THEN BEGIN
 subeder:=FALSE;
 subeizq:=FALSE;
END;
IF (Zo = Za) AND (X=Xa) THEN BEGIN
 Down:=TRUE;
 Dere:=FALSE;
 Izqi:=FALSE;
END;
IF X > Xa THEN BEGIN
 Dere:=TRUE;Izqi:=FALSE;
 Subeder:=TRUE;Subeizq:=FALSE;
END;
IF dere THEN Subeder:=TRUE;
IF izqi THEN Subeizq:=TRUE;
IF X < Xa THEN BEGIN
 Izqi:=TRUE;Dere:=FALSE;
 Subeizq:=TRUE;Subeder:=FALSE;
END;
```

```
IF ((X = Xo) OR (Xa = Xo)) AND (X>Xa) THEN Camizq:=TRUE;
IF ((X = Xo) OR (Xa = Xo)) AND (X<Xa) THEN Camder:=TRUE;

Rectan_1.Mov_rec(X,Z,trz_rl);


IF KEYPRESSED THEN BEGIN

Pausa:=TRUE;
Detec:=Lec(grafs);
IF Detec = ESC THEN Salir:=TRUE;
IF Detec = ENTER THEN BEGIN
 Pausa:=FALSE;

 REPEAT


 {IF Detec = ENTER THEN BEGIN
   Pausa:=FALSE;}
   Detec:=Lec(grafs);
   CASE Detec OF
     {F4 : BEGIN
         clr:=clr+1;
         IF clr > C15 THEN clr:=c3;
         detec:='o';
        END;
      F5 : BEGIN
         trz:=trz+1;
         IF trz = 4 THEN trz:=1;
         detec:='o';
        END;}
      F2 : BEGIN
         IF trz_rl = 0 THEN trz_rl:=1
         ELSE trz_rl:=0;
         Stat_lat:=Trz_rl;
         detec:='o';
        END;
      F4 : BEGIN
         IF trz_rf = 0 THEN trz_rf:=1
         ELSE trz_rf:=0;
         Stat_fron:=Trz_rf;
         detec:='o';
        END;
      F3 : BEGIN
         Limpia(2);
         detec:='o';
         Vent(2);
         Rectan_f.Show;
        END;
      F1 : BEGIN
```

```
                Limpia(3);
                detec:='o';
                vent(3);
                Rectan_1.Show;
              END;
          ENTER : BEGIN
                {IF Pausa THEN} Pausa:=FALSE;
                {ELSE Pausa:=TRUE;}
                Detec:='o';
              END;
          ESC   : BEGIN
                Salir:=TRUE;
              END;
          Spc   : BEGIN
                Pausa:=TRUE;
                Detec:='o';
              END;
        END;{CASE}
      {END {Pausa
      ELSE Pausa:=TRUE;}
      UNTIL (Pausa = TRUE) OR (Salir);
    END;{ IF Pausa}
  END;{ IF Keypressed}
  IF cont = ciclos-1 THEN ciclo:=FALSE;
  cont:=cont+1;
UNTIL (Detec = ESC) OR (TRUNC(ABS(Cor)) >= ABS(Final) ) OR (Salir);
Xcor:=Xcorini+Xaprg; Ycor:=Ycorini+Yaprg;Zcor:=Zcorini+Zaprg;
Pon_cor;
IF trz = 1 THEN BEGIN
 Vent(1);
 SETCOLOR(color);
 Circulo.Hide;
END;
END;


END.
```

Unidad CMDS_GRA.PAS

```
UNIT Cmds_gra;

INTERFACE

USES
 CRT,Grafprb,Htas,GRAPH,Comp_1;
VAR
 Paso,Pausa    : BOOLEAN;
 Colores       : ARRAY [1..2,1..16] OF INTEGER;
 Zaltura       : INTEGER;
 Detec         : CHAR;
PROCEDURE Simula(Arre_fin : Arreglo);


IMPLEMENTATION

FUNCTION Asig_Color(Z : INTEGER):INTEGER;
VAR
 Color  : INTEGER;

BEGIN
 Colores[2,1]:=0;
 color:=0;
 FOR i:=1 TO 16 DO BEGIN
  IF Colores[2,i]=Z THEN color:=Colores[1,i];
 END;
 IF Z > -Zdesp THEN color:=-1;

 IF color = 0 THEN BEGIN
  i:=1;
  REPEAT
   i:=i+1;
  UNTIL Colores[2,i] = -1;
  color:=Colores[1,i];
  Colores[2,i]:=Z;
 END;

 Asig_color:=color;
END;



PROCEDURE Simula(Arre_fin : Arreglo);
VAR
 Comando,Linstr,Com_sub : STRING[3];
 Vx,Vy,Vz,Vxa,Vya,
 Vza,Tiempo,Linea,err,
 clrant,Borrador,Xabs,
```

```
 Yabs,Zabs,Regresa,
 Lin_sub,
 clr,trazo,i,siY,Valor : INTEGER;
 Absoluta,Cor_mod      : BOOLEAN;
 G40,G45,G46,G47,G48   : BOOLEAN;
 StrVx,StrVy,StrVz     : STRING[6];
 Valorstr              : STRING[6];
 Campx,Campy,
 Campz,caso            : CHAR;
 Inst_text             : Anystr;
 Rad,Xcen,Ycen         : REAL;
 error,Laux,Clrtemp    : INTEGER;

PROCEDURE Pon_inst;
VAR
 i          : INTEGER;

BEGIN
 Inst_text:='';
 IF Linea > 2 THEN BEGIN
  FOR i:=1 TO 33 DO BEGIN
   IF Arre_fin[linea-2,i] <> '*' THEN
    Inst_text:=Inst_text+Arre_fin[linea-2,i]
   ELSE
    Inst_text:=Inst_text+' ';
  END;
  Instruccion(0,0,Inst_text);
 END;
 IF Linea>1 THEN BEGIN
  Inst_text:='';
  FOR i:=1 TO 33 DO BEGIN
     IF Arre_fin[linea-1,i] <> '*' THEN
      Inst_text:=Inst_text+Arre_fin[linea-1,i]
     ELSE
      Inst_text:=Inst_text+' ';
  END;
  Instruccion(1,0,Inst_text);
  Inst_text:='';
  FOR i:=1 TO 33 DO BEGIN
     IF Arre_fin[linea,i] <> '*' THEN
      Inst_text:=Inst_text+Arre_fin[linea,i]
     ELSE
      Inst_text:=Inst_text+' ';
  END;
  Instruccion(2,0,Inst_text);
 END;
 Inst_text:='';
 IF Linea > 1 THEN BEGIN
  FOR i:=1 TO 33 DO BEGIN
   IF Arre_fin[linea-1,i] <> '*' THEN
    Inst_text:=Inst_text+Arre_fin[linea-1,i]
```

```
    ELSE
      Inst_text:=Inst_text+' ';
   END;
   Instruccion(0,1,Inst_text);
  END;
  Inst_text:='';
  FOR i:=1 TO 33 DO BEGIN
    IF Arre_fin[linea,i] <> '*' THEN
      Inst_text:=Inst_text+Arre_fin[linea,i]
    ELSE
      Inst_text:=Inst_text+' ';
  END;
  Instruccion(1,2,Inst_text);

  Inst_text:='';
  IF Linea < maxY THEN BEGIN
   FOR i:=1 TO 33 DO BEGIN
     IF Arre_fin[linea+1,i] <> '*' THEN
       Inst_text:=Inst_text+Arre_fin[linea+1,i]
     ELSE
       Inst_text:=Inst_text+' ';
   END;
   Instruccion(2,1,Inst_text);
  END;
END;

PROCEDURE Borra_ins(L : INTEGER);
VAR
 i          : INTEGER;

BEGIN
 Inst_text:='';
 IF L > 2 THEN BEGIN
  FOR i:=1 TO 33 DO BEGIN
   IF Arre_fin[l-2,i] <> '*' THEN
     Inst_text:=Inst_text+Arre_fin[l-2,i]
   ELSE
     Inst_text:=Inst_text+' ';
  END;
  Instruccion(0,0,Inst_text);
 END;
 IF Linea>1 THEN BEGIN
  Inst_text:='';
  FOR i:=1 TO 33 DO BEGIN
    IF Arre_fin[l-1,i] <> '*' THEN
      Inst_text:=Inst_text+Arre_fin[l-1,i]
    ELSE
      Inst_text:=Inst_text+' ';
  END;
  Instruccion(1,0,Inst_text);
  Inst_text:='';
```

```
  FOR i:=1 TO 33 DO BEGIN
    IF Arre_fin[1,i] <> '*' THEN
      Inst_text:=Inst_text+Arre_fin[1,i]
    ELSE
      Inst_text:=Inst_text+' ';
  END;
  Instruccion(2,0,Inst_text);
 END;
END;

PROCEDURE Com_72(Xcav,Ycav : INTEGER);
VAR
 SignX,SignY   : REAL;
 Yavan,FinalX,
 FinalY        : INTEGER;

BEGIN
 signX:=ABS(Xcav)/Xcav;
 SignY:=ABS(Ycav)/Ycav;
 FinalX:=(ABS(Xcav)-Diam)*ROUND(SignX);
 FinalY:=(ABS(Ycav)-Diam)*ROUND(SignY);
 Ycav:=0;Yavan:=0;
 REPEAT
  Traza_lin(FinalX,0,0,clr,Diam,1);
  Yavan:=Ycav;
  Ycav:=Ycav + (diam*2*ROUND(SignY));
  IF ABS(Ycav) > ABS(FinalY) THEN BEGIN
   Ycav:=FinalY;
  END;

  Yavan:=Ycav-Yavan;
  Traza_lin(0,Yavan,0,clr,Diam,1);
  Traza_lin(-FinalX,0,0,clr,Diam,1);
  Yavan:=Ycav;
  Ycav:=Ycav + (diam*2*ROUND(SignY));
  IF ABS(Ycav) > ABS(FinalY) THEN BEGIN
   Ycav:=FinalY;
  END;
  Yavan:=Ycav-Yavan;
  Traza_lin(0,Yavan,0,clr,Diam,1);
 UNTIL ABS(Ycav) = ABS(FinalY);
 Traza_lin(finalX,0,0,clr,Diam,1);
 Traza_lin(0,-finalY,0,clr,Diam,1);
 Traza_lin(-finalX,0,0,clr,Diam,1);
 Traza_lin(0,finalY,0,clr,Diam,1);
 Traza_lin(0,-finalY,0,clr,Diam,1);
END;

PROCEDURE Com_73 (Zprof : INTEGER);
VAR
 SignZ       : INTEGER;
```

```
 FinalZ,Zavan  : INTEGER;
BEGIN
 SignZ  := ABS(zprof) DIV Zprof;
 finalZ := Zprof;
 Zavan  := 0;
 Zprof := 200*ROUND(SignZ);
 IF ABS(Zprof) > ABS(finalZ) THEN BEGIN
  Zprof:=finalZ;
 END;
 Traza_lin(0,0,Zprof,clr,Diam,trazo);
 {IF ABS(Zprof) <> ABS(FinalZ) THEN BEGIN
  Zprof:=Zprof - 20 * ROUND(SignZ);
  Traza_lin(0,0,-(20) * ROUND(signZ),clr,Diam,trazo);
 END;}
 IF ABS(Zprof) < ABS(finalZ) THEN BEGIN
  REPEAT
  Zavan:=Zprof;
  Zprof:=Zprof+(200) * ROUND(signZ);
  IF ABS(Zprof) > ABS(FinalZ) THEN BEGIN
   Zprof:=finalZ;
   Zavan:=finalZ-Zavan
  END
  ELSE Zavan:=Zprof-Zavan;
   Traza_lin(0,0,Zavan,clr,Diam,trazo);
   {IF ABS(Zprof) <> ABS(FinalZ) THEN BEGIN
    Zprof:=Zprof-(20) *ROUND(signZ);
    Traza_lin(0,0,-20*ROUND(signZ),clr,Diam,trazo);
   END;}
  UNTIL ABS(FinalZ) = ABS(Zprof);
 END;
 Traza_lin(0,0,-finalZ,clr,Diam,trazo);
END;

PROCEDURE Com_83 (Zprof : INTEGER);
VAR
 SignZ       : REAL;
 FinalZ,Zavan  : INTEGER;
BEGIN
 SignZ  := ABS(zprof)/Zprof;
 finalZ := Zprof;
 Zavan  := 0;
 Zprof := 600*ROUND(SignZ);
 IF ABS(Zprof) > ABS(finalZ) THEN BEGIN
  Zprof:=finalZ;
 END;
 Traza_lin(0,0,Zprof,clr,Diam,trazo);
 Traza_lin(0,0,-Zprof,clr,Diam,trazo);
 IF ABS(Zprof) < ABS(finalZ) THEN BEGIN
  REPEAT
  Zprof:=Zprof+(550) * ROUND(signZ);
  IF ABS(Zprof) > ABS(FinalZ) THEN
```

```
    Zprof:=finalZ;
    Traza_lin(0,0,Zprof,clr,Diam,trazo);
    Traza_lin(0,0,-Zprof,clr,Diam,trazo);
  UNTIL ABS(FinalZ) = ABS(Zprof);
 END;
END;


BEGIN

 {Traza_lin(900,0,0,c3,Diam,1); readln;

 Traza_lin(0,0,-19,c3,Diam,1);    readln;
 Traza_lin(-100,0,0,c3,Diam,1);    readln;
 Traza_lin(110,110,0,c3,Diam,1);    readln;}
 Colores[1,1]:=c2;Colores[1,2]:=c3;Colores[1,3]:=c4;
 Colores[1,4]:=c5;Colores[1,5]:=c6;Colores[1,6]:=c7;
 Colores[1,7]:=c8;Colores[1,8]:=c9;Colores[1,9]:=c10;
 Colores[1,10]:=c11;Colores[1,11]:=c12;Colores[1,12]:=c13;
 Colores[1,13]:=c14;Colores[1,14]:=c15;Colores[1,15]:=c3;
 Colores[1,16]:=c4;
 FOR i:=2 TO 16 DO
  Colores[2,i]:=-1;
 colores[1,1]:=c2;
 Zdesp:=0;Ydesp:=0;Xdesp:=0;Zaltura:=0;
 Linea:=1;
 Pausa:=TRUE;
 Detec:='o';

 Absoluta:=FALSE;Cor_mod:=FALSE;
 G40:=FALSE;G45:=FALSE;G46:=FALSE;
 G47:=FALSE;G48:=FALSE;
 Vx:=0;Vy:=0;Vz:=0;
 Vxa:=0;Vya:=0;Vza:=0;
 Xabs:=0;Yabs:=0;Zabs:=0;
 Clr:=C2;
 REPEAT
  Pon_inst;
  Comando:='';
  Comando:=Arre_fin[Linea,6]+Arre_fin[Linea,7]+Arre_fin[Linea,8];
  StrVx:=Arre_fin[Linea,11]+Arre_fin[Linea,12]+
       Arre_fin[Linea,13]+Arre_fin[Linea,14]+Arre_fin[Linea,15];
  CampX:=Arre_fin[Linea,10];
  StrVy:=Arre_fin[Linea,18]+
       Arre_fin[Linea,19]+Arre_fin[Linea,20]+Arre_fin[Linea,21];
  Campy:=Arre_fin[Linea,17];
  StrVz:=Arre_fin[Linea,24]+Arre_fin[Linea,25]+
       Arre_fin[Linea,26]+Arre_fin[Linea,27]+Arre_fin[Linea,28];
  Campz:=Arre_fin[Linea,23];
  {IF Comando = 'M30' THEN Salir:=TRUE;}
  IF Comando = ' 00' THEN BEGIN
```

```
Trazo:=2;
Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
IF G45 THEN BEGIN
 IF Vx <> 0 THEN Vx:=(ABS(Vx)+Diam) * (Vx DIV ABS(Vx));
 IF Vy <> 0 THEN Vy:=(ABS(Vy)+Diam) * (Vy DIV ABS(Vy));
END;
IF G46 THEN BEGIN
 IF Vx <> 0 THEN Vx:=(ABS(Vx)-Diam) * (Vx DIV ABS(Vx));
 IF Vy <> 0 THEN Vy:=(ABS(Vy)-Diam) * (Vy DIV ABS(Vy));
END;
IF G47 THEN BEGIN
 IF Vx <> 0 THEN Vx:=(ABS(Vx)+(Diam*2)) * (Vx DIV ABS(Vx));
 IF Vy <> 0 THEN Vy:=(ABS(Vy)+(Diam*2)) * (Vy DIV ABS(Vy));
END;
IF G48 THEN BEGIN
 IF Vx <> 0 THEN Vx:=(ABS(Vx)-(Diam*2)) * (Vx DIV ABS(Vx));
 IF Vy <> 0 THEN Vy:=(ABS(Vy)-(Diam*2)) * (Vy DIV ABS(Vy));
END;
IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
IF Campz='-' THEN Vz:=-Vz;

IF Absoluta THEN BEGIN
 Zaltura:=Vz;clrant:=clr;
 clr:=Asig_color(Zaltura);
 IF clr = -1 THEN BEGIN
  clr:=clrant;Trazo:=0;
 END;
 Traza_lin(Vx-Vxa,Vy-Vya,Vz-Vza,clr,Diam,Trazo);
END
ELSE BEGIN
 Zaltura:=Zaltura+Vz;clrant:=clr;
 clr:=Asig_color(Zaltura);
 IF clr = -1 THEN BEGIN
  clr:=clrant;Trazo:=0;
 END;
 Traza_lin(Vx,Vy,Vz,clr,Diam,Trazo);
 Xabs:=Xabs+Vx;Yabs:=Yabs+Vy;Zabs:=Zabs+Vz;
END;
END;
IF Comando = '01' THEN BEGIN
 Trazo:=1;
 Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
 VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
 IF G45 THEN BEGIN
  IF Vx <> 0 THEN Vx:=(ABS(Vx)+Diam) * (Vx DIV ABS(Vx));
  IF Vy <> 0 THEN Vy:=(ABS(Vy)+Diam) * (Vy DIV ABS(Vy));
 END;
 IF G46 THEN BEGIN
  IF Vx <> 0 THEN Vx:=(ABS(Vx)-Diam) * (Vx DIV ABS(Vx));
  IF Vy <> 0 THEN Vy:=(ABS(Vy)-Diam) * (Vy DIV ABS(Vy));
```

```
  END;
  IF G47 THEN BEGIN
   IF Vx <> 0 THEN Vx:=(ABS(Vx)+(Diam*2)) * (Vx DIV ABS(Vx));
   IF Vy <> 0 THEN Vy:=(ABS(Vy)+(Diam*2)) * (Vy DIV ABS(Vy));
  END;
  IF G48 THEN BEGIN
   IF Vx <> 0 THEN Vx:=(ABS(Vx)-(Diam*2)) * (Vx DIV ABS(Vx));
   IF Vy <> 0 THEN Vy:=(ABS(Vy)-(Diam*2)) * (Vy DIV ABS(Vy));
  END;
  IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
  IF Campz='-' THEN Vz:=-Vz;

  IF Absoluta THEN BEGIN
   Zaltura:=Vz;clrant:=clr;
   clr:=Asig_color(Zaltura);
   IF clr = -1 THEN BEGIN
    clr:=clrant;Trazo:=0;
   END;
   ClrTemp:=Clr;
   IF ((Vx-Vxa <> 0) AND (Vz-Vza > 0))
    OR ((Vy-Vya <> 0) AND (Vz-Vza > 0)) THEN BEGIN
     Clr:=clrant;
   END;
   Traza_lin(Vx-Vxa,Vy-Vya,Vz-Vza,clr,Diam,Trazo);
   clr:=clrtemp;
  END
  ELSE BEGIN
   Zaltura:=Zaltura+Vz;clrant:=clr;
   clr:=Asig_color(Zaltura);
   IF clr = -1 THEN BEGIN
    clr:=clrant;Trazo:=0;
   END;
   ClrTemp:=Clr;
   IF ((Vx <> 0) AND (Vz > 0))
    OR ((Vy <> 0) AND (Vz > 0)) THEN BEGIN
     Clr:=clrant;
   END;

   Traza_lin(Vx,Vy,Vz,clr,Diam,Trazo);
   Xabs:=Xabs+Vx;Yabs:=Yabs+Vy;Zabs:=Zabs+Vz;
   clr:=clrtemp;
  END;
 END;
 IF Comando = '03' THEN BEGIN
  Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
  VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
  IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
  IF Campz='-' THEN Vz:=-Vz;
  IF Arre_fin[Linea+1,6]+Arre_fin[Linea+1,7]+Arre_fin[Linea+1,8] <> 'M99' THEN BEGIN
   IF Absoluta THEN BEGIN
    IF (Vx-Vxa > 0) AND (Vy-Vya > 0 ) THEN BEGIN
```

```
    Xcen:=Vx-Vxa;Ycen:=0;Rad:=Xcen;
    SiY:=1;
   END;
   IF (Vx-Vxa > 0) AND (Vy-Vya < 0 ) THEN BEGIN
    Xcen:=0;Ycen:=-(Vy-Vya);Rad:=Ycen;
    SiY:=-1;
   END;
   IF (Vx-Vxa < 0) AND (Vy-Vya < 0 ) THEN BEGIN
    Xcen:=(Vx-Vxa);Ycen:=0;Rad:=-Xcen;
    SiY:=1;
   END;
   IF (Vx-Vxa < 0) AND (Vy-Vya > 0 ) THEN BEGIN
    Xcen:=0;Ycen:=(Vy-Vya);Rad:=Ycen;
    SiY:=-1;
   END;
  END;
  IF Absoluta = FALSE THEN BEGIN
   IF (Vx > 0) AND (Vy > 0 ) THEN BEGIN
    Xcen:=Vx;Ycen:=0;rad:=Xcen;
    SiY:=1;
   END;
   IF (Vx > 0) AND (Vy < 0 ) THEN BEGIN
    Xcen:=0;Ycen:=-(Vy);rad:=-Ycen;
    SiY:=-1;
   END;

   IF (Vx < 0) AND (Vy < 0 ) THEN BEGIN
    Xcen:=(Vx);Ycen:=0;rad:=-Xcen;
    SiY:=1;
   END;
   IF (Vx < 0) AND (Vy > 0 ) THEN BEGIN
    Xcen:=0;Ycen:=(Vy);rad:=Ycen;
    SiY:=-1;
   END;
  END;
 END;
 IF Arre_fin[Linea+1,6]+Arre_fin[Linea+1,7]+Arre_fin[Linea+1,8] = 'M99' THEN BEGIN
  Caso := Arre_fin[Linea,36];
  Valor:=0;
  Valorstr:=Arre_fin[linea+1,18]+Arre_fin[linea+1,19]+Arre_fin[linea+1,20]+
        Arre_fin[linea+1,21];
  VAL(Valorstr,Valor,error);
  Ycen:=valor;
  Valor:=0;
  Valorstr:=Arre_fin[linea+1,11]+Arre_fin[linea+1,12]+Arre_fin[linea+1,13]+
        Arre_fin[linea+1,14]+Arre_fin[linea+1,15];
  VAL(Valorstr,Valor,error);
  Xcen:=valor;
  IF Arre_fin[Linea,37] = '-' THEN Xcen:=-Xcen;
  IF Arre_fin[Linea,38] = '-' THEN Ycen:=-Ycen;
  Rad:=SQRT(SQR(Xcen) + SQR(Ycen));
```

```
  CASE Caso OF
    '1' : SiY:=1;
    '2' : SiY:=-1;
    '3' : Siy:=1;
    '4' : SiY:=-1;
  END;
 END;

 IF Absoluta THEN BEGIN
  {Zaltura:=Vz;clrant:=clr;
  clr:=Asig_color(Zaltura);
  IF clr = -1 THEN BEGIN
   clr:=clrant;Trazo:=0;
  END;}
  Traza_cir(Vx-Vxa,Vy-Vya,rad,clr,Diam,SiY,Xcen,Ycen);
 END
 ELSE BEGIN
  {Zaltura:=Zaltura+Vz;clrant:=clr;
  clr:=Asig_color(Zaltura);
  IF clr = -1 THEN BEGIN
   clr:=clrant;Trazo:=0;
  END;}
  Traza_cir(Vx,Vy,rad,clr,Diam,SiY,Xcen,Ycen);
  Xabs:=Xabs+Vx;Yabs:=Yabs+Vy;
 END;
END;
IF Comando = '02' THEN BEGIN
 Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
 VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
 IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
 IF Campz='-' THEN Vz:=-Vz;
 IF Arre_fin[Linea+1,6]+Arre_fin[Linea+1,7]+Arre_fin[Linea+1,8] <> 'M99' THEN BEGIN
  IF Absoluta THEN BEGIN
   IF (Vx-Vxa > 0) AND (Vy-Vya > 0 ) THEN BEGIN
    Xcen:=0;Ycen:=(Vy-Vya);Rad:=Ycen;
    SiY:=-1;
   END;
   IF (Vx-Vxa > 0) AND (Vy-Vya < 0 ) THEN BEGIN
    Xcen:=Vx-Vxa;Ycen:=0;Rad:=Xcen;
    SiY:=1;
   END;
   IF (Vx-Vxa < 0) AND (Vy-Vya < 0 ) THEN BEGIN
    Xcen:=0;Ycen:=-(Vy-Vya);Rad:=-Ycen;
    SiY:=-1;
   END;
   IF (Vx-Vxa < 0) AND (Vy-Vya > 0 ) THEN BEGIN
    Xcen:=Vx-Vxa;Ycen:=0;Rad:=Xcen;
    SiY:=1;
   END;
  END;
  IF Absoluta = FALSE THEN BEGIN
```

```
    IF (Vx > 0) AND (Vy > 0 ) THEN BEGIN
      Xcen:=0;Ycen:=Vy;rad:=Ycen;
      SiY:=-1;
    END;
    IF (Vx > 0) AND (Vy < 0 ) THEN BEGIN
      Xcen:=Vx;Ycen:=0;rad:=Xcen;
      SiY:=1;
    END;
    IF (Vx < 0) AND (Vy < 0 ) THEN BEGIN
      Xcen:=0;Ycen:=-Vy;rad:=-Ycen;
      SiY:=-1;
    END;
    IF (Vx < 0) AND (Vy > 0 ) THEN BEGIN
      Xcen:=Vx;Ycen:=0;rad:=Xcen;
      SiY:=1;
    END;
  END;
END;
IF Arre_fin[Linea+1,6]+Arre_fin[Linea+1,7]+Arre_fin[Linea+1,8] = 'M99' THEN BEGIN
  Caso := Arre_fin[Linea,36];
  Valor:=0;
  Valorstr:=Arre_fin[linea+1,18]+Arre_fin[linea+1,19]+Arre_fin[linea+1,20]+
        Arre_fin[linea+1,21];
  VAL(Valorstr,Valor,error);
  Ycen:=valor;
  Valor:=0;
  Valorstr:=Arre_fin[linea+1,11]+Arre_fin[linea+1,12]+Arre_fin[linea+1,13]+
        Arre_fin[linea+1,14]+Arre_fin[linea+1,15];
  VAL(Valorstr,Valor,error);
  Xcen:=valor;
  IF Arre_fin[Linea,37] = '-' THEN Xcen:=-Xcen;
  IF Arre_fin[Linea,38] = '-' THEN Ycen:=-Ycen;
  Rad:=SQRT(SQR(Xcen) + SQR(Ycen));
  CASE Caso OF
    '1' : SiY:=-1;
    '2' : SiY:=1;
    '3' : Siy:=-1;
    '4' : SiY:=1;
  END;
END;

IF Absoluta THEN BEGIN
  {Zaltura:=Vz;clrant:=clr;
  clr:=Asig_color(Zaltura);
  IF clr = -1 THEN BEGIN
    clr:=clrant;Trazo:=0;
  END;}
  Traza_cir(Vx-Vxa,Vy-Vya,rad,clr,Diam,SiY,Xcen,Ycen);
END
ELSE BEGIN
  {Zaltura:=Zaltura+Vz;clrant:=clr;
```

```
    clr:=Asig_color(Zaltura);
    IF clr = -1 THEN BEGIN
      clr:=clrant;Trazo:=0;
    END;}
    Traza_cir(Vx,Vy,rad,clr,Diam,SiY,Xcen,Ycen);
    Xabs:=Xabs+Vx;Yabs:=Yabs+Vy;
  END;
END;
IF Comando = '04' THEN BEGIN
  VAL(StrVx,Tiempo,err);
  DELAY(tiempo*10);
END;
IF Comando = '25' THEN BEGIN
  Cor_mod:=Absoluta;
  Borrador:=Linea+1;
  Lin_sub:=Linea;
  Regresa:=Linea+1;
  REPEAT
    Lin_sub:=Lin_sub+1;
    Com_sub:=Arre_fin[lin_sub,6]+Arre_fin[lin_sub,7]+Arre_fin[lin_sub,8];
  UNTIL (Com_sub='M17') ;
  Linstr:='  ';
  STR(Regresa,Linstr);
  Char_num:=linstr[0];

  inst[Lin_sub,36]:='0';
  inst[Lin_sub,37]:='0';
  inst[Lin_sub,38]:='0';

  inst[Lin_sub,36]:=Linstr[1];
  inst[Lin_sub,37]:=Linstr[2];
  inst[Lin_sub,38]:=Linstr[3];

  {Linstr:='';
  Linstr:=inst[20,36]+inst[20,37]+inst[20,38];
  outtext(linstr);readln;}

  Valor:=0;
  Valorstr:='';
  Valorstr:=Arre_fin[linea,31]+Arre_fin[linea,32]+Arre_fin[linea,33];
  error:=1;
  {Str(linea,valorstr);
  outtext('este'+valorstr);readln;}
  VAL(Valorstr,Valor,error);
  Linea:=Valor-1;
END;
IF Comando = '27' THEN BEGIN
  Borrador:=Linea+1;
  Valor:=0;
  Valorstr:=Arre_fin[linea,31]+Arre_fin[linea,32]+Arre_fin[linea,33];
  error:=1;
```

```
  VAL(Valorstr,Valor,error);
  Linea:=Valor-1;
 END;
 IF Comando = '40' THEN BEGIN
  G40:=FALSE;G45:=FALSE;G46:=FALSE;
  G47:=FALSE;G48:=FALSE;
 END;
 IF Comando = '45' THEN BEGIN
  G45:=TRUE;{G46:=FALSE;
  G47:=FALSE;G48:=FALSE;}
 END;
 IF Comando = '46' THEN BEGIN
  {G40:=FALSE;G45:=FALSE;}G46:=TRUE;
  {G47:=FALSE;G48:=FALSE;}
 END;
 IF Comando = '47' THEN BEGIN
  {G40:=FALSE;G45:=FALSE;G46:=FALSE;}
  G47:=TRUE;{G48:=FALSE;}
 END;
 IF Comando = '48' THEN BEGIN
  {G40:=FALSE;G45:=FALSE;G46:=FALSE;}
  {G47:=FALSE;}G48:=TRUE;
 END;
 IF Comando = '72' THEN BEGIN
  Trazo:=1;
  Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
  VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
  IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
  IF Campz='-' THEN Vz:=-Vz;
  IF Absoluta THEN BEGIN
   Zaltura:=Vz;clrant:=clr;
   clr:=Asig_color(Zaltura);
   IF clr = -1 THEN BEGIN
    clr:=clrant;Trazo:=0;
   END;
  END
  ELSE BEGIN
   Zaltura:=Zaltura+Vz;clrant:=clr;
   clr:=Asig_color(Zaltura);
   IF clr = -1 THEN BEGIN
    clr:=clrant;Trazo:=0;
   END;
  END;
  IF Absoluta THEN BEGIN
   Vx:=Vx-Vxa;Vy:=Vy-Vya;Vz:=Vz-Vza;
  END;
  Traza_lin(0,0,Vz,clr,Diam,trazo);
  Com_72(Vx,Vy);
  traza_lin(0,0,-Vz,clr,Diam,trazo);
  IF Absoluta THEN BEGIN
   Vx:=Vxa;Vy:=Vya;Vz:=Vza;
```

```
  END;
 END;
 IF Comando = ' 73' THEN BEGIN
  Trazo:=1;
  Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
  VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
  IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
  IF Campz='-' THEN Vz:=-Vz;
  IF Absoluta THEN BEGIN
    Vx:=Vx-Vxa;Vy:=Vy-Vya;Vz:=Vz-Vza;
  END;
  Zaltura:=Zaltura+Vz;clrant:=clr;
  clr:=Asig_color(Zaltura);
  IF clr = -1 THEN BEGIN
    clr:=clrant;Trazo:=0;
  END;
  Com_73(Vz);
  IF absoluta THEN BEGIN
    Vz:=Vza;Vx:=Vxa;Vy:=Vya;
  END;
 END;
 IF Comando = ' 81' THEN BEGIN
  Trazo:=1;
  Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
  VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
  IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
  IF Campz='-' THEN Vz:=-Vz;
  IF Absoluta THEN BEGIN
    Vx:=Vx-Vxa;Vy:=Vy-Vya;Vz:=Vz-Vza;
  END;
  Zaltura:=Zaltura+Vz;clrant:=clr;
  clr:=Asig_color(Zaltura);
  IF clr = -1 THEN BEGIN
    clr:=clrant;Trazo:=0;
  END;
  Traza_lin(0,0,Vz,clr,Diam,trazo);
  Traza_lin(0,0,-Vz,clr,Diam,2);
  IF absoluta THEN BEGIN
    Vz:=Vza;Vx:=Vxa;Vy:=Vya;
  END;
 END;
 IF Comando = ' 82' THEN BEGIN
  Trazo:=1;
  Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
  VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
  IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
  IF Campz='-' THEN Vz:=-Vz;
  IF Absoluta THEN BEGIN
    Vx:=Vx-Vxa;Vy:=Vy-Vya;Vz:=Vz-Vza;
  END;
  Zaltura:=Zaltura+Vz;clrant:=clr;
```

```
  clr:=Asig_color(Zaltura);
  IF clr = -1 THEN BEGIN
   clr:=clrant;Trazo:=0;
  END;
  Traza_lin(0,0,Vz,clr,Diam,trazo);
  DELAY(2000);
  Traza_lin(0,0,-Vz,clr,Diam,2);
  IF absoluta THEN BEGIN
   Vz:=Vza;Vx:=Vxa;Vy:=Vya;
  END;
 END;
 IF Comando = '83' THEN BEGIN
  Trazo:=1;
  Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
  VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
  IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
  IF Campz='-' THEN Vz:=-Vz;
  IF Absoluta THEN BEGIN
   Vx:=Vx-Vxa;Vy:=Vy-Vya;Vz:=Vz-Vza;
  END;
  Zaltura:=Zaltura+Vz;clrant:=clr;
  clr:=Asig_color(Zaltura);
  IF clr = -1 THEN BEGIN
   clr:=clrant;Trazo:=0;
  END;
  Com_83(Vz);
  IF absoluta THEN BEGIN
   Vz:=Vza;Vx:=Vxa;Vy:=Vya;
  END;
 END;
 IF Comando = '85' THEN BEGIN
  Trazo:=1;
  Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
  VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
  IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
  IF Campz='-' THEN Vz:=-Vz;
  IF Absoluta THEN BEGIN
   Vx:=Vx-Vxa;Vy:=Vy-Vya;Vz:=Vz-Vza;
  END;
  Zaltura:=Zaltura+Vz;clrant:=clr;
  clr:=Asig_color(Zaltura);
  IF clr = -1 THEN BEGIN
   clr:=clrant;Trazo:=0;
  END;
  Traza_lin(0,0,Vz,clr,Diam,trazo);
  Traza_lin(0,0,-Vz,clr,Diam,2);
  IF absoluta THEN BEGIN
   Vz:=Vza;Vx:=Vxa;Vy:=Vya;
  END;
 END;
```

```
IF Comando = ' 89' THEN BEGIN
 Trazo:=1;
 Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
 VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
 IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
 IF Campz='-' THEN Vz:=-Vz;
 IF Absoluta THEN BEGIN
  Vx:=Vx-Vxa;Vy:=Vy-Vya;Vz:=Vz-Vza;
 END;
 Zaltura:=Zaltura+Vz;clrant:=clr;
 clr:=Asig_color(Zaltura);
 IF clr = -1 THEN BEGIN
  clr:=clrant;Trazo:=0;
 END;
 Traza_lin(0,0,Vz,clr,Diam,trazo);
 DELAY(2000);
 Traza_lin(0,0,-Vz,clr,Diam,2);
 IF absoluta THEN BEGIN
  Vz:=Vza;Vx:=Vxa;Vy:=Vya;
 END;
END;
IF Comando = ' 90' THEN BEGIN
 Absoluta:=TRUE;
 Vx:=Xabs;Vy:=Yabs;Vz:=Zabs;
 Vxa:=Vx;Vya:=Vy;Vza:=Vz;
END;
IF Comando = ' 91' THEN BEGIN
 Absoluta:=FALSE;
 Xabs:=Vx;Yabs:=Vy;Zabs:=Vz;
 Vx:=0;Vy:=0;Vz:=0;
END;
IF Comando = ' 92' THEN BEGIN
 Vxa:=Vx;Vya:=Vy;Vza:=Vz;Vx:=0;Vy:=0;Vz:=0;
 Vxa:=0;Vya:=0;Vza:=0;
 VAL(StrVx,Vx,err);VAL(StrVy,Vy,err);VAL(StrVz,Vz,err);
 IF Campx='-' THEN Vx:=-Vx;IF Campy='-' THEN Vy:=-Vy;
 IF Campz='-' THEN Vz:=-Vz;
 Zaltura:=Vz;
 Init_Hta(Vx,Vy,Vz,Diam);
 Xabs:=Vx;Yabs:=Vy;Zabs:=Vz;
END;
IF comando = 'M00' THEN BEGIN
 Pausa:=FALSE;
 Detec:=ENTER;
END;
IF Comando = 'M03' THEN M03(TRUE);
IF Comando = 'M05' THEN M03(FALSE);
IF Comando = 'M06' THEN BEGIN
 Diam:=0;
```

```
  Val(StrVx,Diam,err);
END;

IF Comando = 'M17' THEN BEGIN
    Borrador:=Linea+1;
    {str(linea,linstr);
    outtext('linea'+linstr);readln;}

    linstr:='';
    linstr[0]:=Char_num;
    Linstr[1]:=inst[Linea,36];
    IF inst[Linea,37] IN enteros THEN linstr[2]:={Linstr+}inst[Linea,37];
    IF inst[Linea,38] IN enteros THEN linstr[3]:={Linstr+}inst[Linea,38];
    VAL(Linstr,Laux,error);

    {outtext('este'+linstr);readln;
    STR(Laux,comando);
    Outtext(comando);readln;}

    Linea:=Laux-1;
    IF (Absoluta = FALSE) AND (Cor_mod = TRUE) THEN BEGIN
     Vx:=Xabs;Vy:=Yabs;Vz:=Zabs;
     Vxa:=Vx;Vya:=Vy;Vza:=Vz;
    END;
    IF (Absoluta = TRUE) AND (Cor_mod = FALSE) THEN BEGIN
     Xabs:=Vx;Yabs:=Vy;Zabs:=Vz;
     Vx:=0;Vy:=0;Vz:=0;
    END;
    Absoluta:=Cor_mod;
   END;
Linea:=Linea+1;
IF KEYPRESSED THEN
  Detec:=Lee(grafs);
IF Detec = Esc THEN Salir:=TRUE;

IF ((detec IN grafs) AND (detec <> ESc)) OR (Paso) THEN BEGIN
REPEAT
      Pausa:=FALSE;

      Detec:=Lee(grafs);
      CASE Detec OF
        {F4 : BEGIN
            clr:=clr+1;
            IF clr > C15 THEN clr:=c3;
            detec:='o';
           END;
         F5 : BEGIN
            trz:=trz+1;
            IF trz = 4 THEN trz:=1;
            detec:='o';
           END;}
```

```
        F2 : BEGIN
             IF trz_rl = 0 THEN trz_rl:=1
             ELSE trz_rl:=0;
             Stat_lat:=Trz_rl;
             detec:='o';
            END;
        F4 : BEGIN
             IF trz_rf = 0 THEN trz_rf:=1
             ELSE trz_rf:=0;
             Stat_fron:=Trz_rf;
             {IF Stat_fron = 0 THEN Suena(100,1000);}
             detec:='o';
            END;
        F3 : BEGIN
             Limpia(2);
             detec:='o';
             Vent(2);
             Rectan_f.Show;
            END;
        F1 : BEGIN
             Limpia(3);
             detec:='o';
             vent(3);
             Rectan_l.Show;
            END;
       ENTER : BEGIN
             {IF Pausa THEN} Pausa:=FALSE;.
             {ELSE Pausa:=TRUE;}
             Detec:='o';
            END;
       ESC   : BEGIN
             Salir:=TRUE;
            END;
       Spc   : BEGIN
             Pausa:=TRUE;
             Detec:='o';
            END;
      END;{CASE}
    {END {Pausa
    ELSE Pausa:=TRUE;}
    UNTIL (Pausa = TRUE) OR (salir);
  END;{if detec in grafs}
  IF (Comando = '27') OR (Comando = 'M17') OR (Comando = '25') THEN Borra_ins(Borrador);
 UNTIL (Comando = 'M30') OR (Salir);
 IF (NOT Salir) AND (NOT Paso) THEN
 REPEAT
 UNTIL KEYPRESSED;
 CLOSEGRAPH;
END;
END.
```