INFORMATICA PROGRAMACION

Arquitectura de Computadoras

INFORMATICA BASICA

La informática es manipular la información (Datos) en una maquina.
| Ordenador
Los dos campos |
| Programas → Nos moveremos aquí, utilizando el ordenador.
ORDENADOR → Hardware ← Parte física. (Hard = Duro).
PROGRAMAS → Software ← Parte lógica. (Soft = Blando).
La configuración básica de un ordenador está compuesta por:
Ordenador en sí → Unidad Central de Proceso (U.C.P).
Maquina Aux.(Comunicación) → Periféricos.
Es imprescindible la comunicación con el ordenador.
La U.C.P.- Es donde realmente se va a ordenar y procesar la información.
PARTES DE LA U.C.P:
LA MEMORIA.- Para almacenar. Para manipular la información es necesario almacenar la información.
UNIDAD DE CONTROL.- La información por si sola no vale para nada, necesita un tipo de instrucciones.
UNIDAD DE ARITMETEREOLOGICA Básicamente el ordenador hace
| Cálculos aritméticos.
| Cálculos lógicos.

La Memoria, Unidad de Control y la Unidad Aritmetereologica tienen que estar relacionadas entre sí.
El ordenador representa la información mediante la Electricidad. |+|-> 0
Mediante la carga de Electricidad | | CODIGO BINARIO.
|-|-> 1

En la Unidad de Control se encuentra el Reloj o Sincronizador y es para que el ordenador funcione y pueda enviar la información y determina la velocidad del ordenador.
Velocidad de Giro: Vueltas/Seg=hz ← Hercios.
El primer PC 4 Meghahercios[2] y el ultimo PC 133. Lo normal es estar entre 33 y 100.
Para razonar los números de bits fijos. 8 bits.
Para saber si es una o dos palabras -> SANTITO
No palabras Nº fijo de caracteres|0,1 -> Representan Signos
Letras Chinas | Entre +150 y -500 ->A,a,á | 1 - 2
| 2 - 4
| 3 - 8
| 4 - 16
| 8 → 256 ← 8 bits
|
|8bits → byte ← carácter
| a= 01001101→byte, 8bits

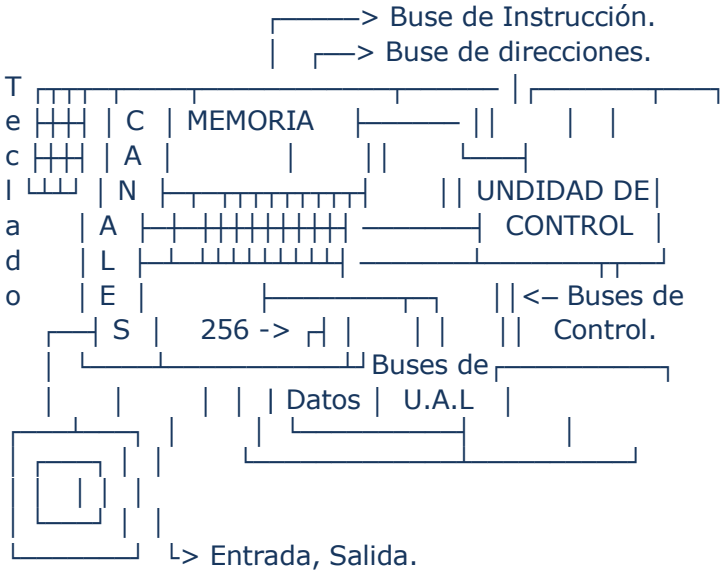
La memoria está distribuida en pequeños bloques llamados bytes adoptan el nombre de,

2Un millón de hercios.

posiciones de memoria. Cada bloque está representada por un número, dirección de memoria.

Es necesario conectar los periféricos con la memoria y para que sea posible intervienen los Canales: Entrada, Salida.

Para que la instrucción de memoria llegue a U.Control necesita un camino puesto que tienes que estar intercomunicados.



Ordenador

Las líneas internas de comunicación se denominan Buses.

En principio necesita 8 pasos para transformar la información pero ahora también interviene la memoria Caché. La memoria de U.C. retiene toda la instrucción. Se envía la suma pero también se envía los números cuando el buse de control envió también la suma también se incluye los números y así se hace que el ordenador sea más rápido.

La información la representa mediante 8 bits → o sea 8 cargas eléctricas y es imposible utilizar el mismo cable. Por eso para enviar la información es necesario 8 cables y por cada uno enviará un bit. Pero el Buse de direcciones no puede estar delimitado a 8 sino que tiene 32 cables → 2^{32} → 4000 Millones.

Con un ordenador de 32 en los Buses de Control podría tener 4 Gb.

Para medir la capacidad del ordenador están creadas unas Unidades de Medida. 1024 bytes → 1 Kb, 1024Kb → 1Mg, 1024 Mg → 1 Gigabyte.

U.Control y Unidad Aritmética están unidos en un chip, llamado microprocesador.

Hay una premisa, que haga un programa en memoria, permanente. No puede depender de que tenga suministro eléctrico o no lo tenga. También necesita una memoria no permanente. En la primera, la memoria de arranque y en la segunda memoria de trabajo.

ROM -> Pequeña -> 64 K.

RAM -> Grande -> Normalmente 4Mb. La memoria por si sola pierde fuerza por eso se realimenta y se llama refresco de memoria. Recarga condensadores.

Microprocesador → Es el corazón del ordenador, es quien determina el tipo del ordenador. Hay diferentes tipos de Familias de Microprocesador.

| INTEL -------- Microprocesador | 1,5 en años 70
PC - IBM | | Abarcó Peq. y
| MICROSOFT ---- Software | medianas Empr.

Intel creo el microprocesador → 8088 ← 8 Bits

INFORMATICA PROGRAMACION

8086 ← 16 Bits mejorado.
80286 SX Paso intermedio entre los dos.
386 DX Era un verdadero 386
486 DX
DX2 Montaron en el mismo microprocesador 2 unidades de control y 2 a.l.u.
Pentium Mejoraba el problema de comunicación exterior. Con fracaso por fallo en el decimal nº 20.
DX4 -> 100 Mhz.
Nuevos Pentium -> 133 Mhz.
Y dentro de poco saldrá el P6.

PERIFERICOS

Son maquinas que permite inicialmente comunicarnos con la U.C.P.
ENTRADA.- Teclado, Ratón, Escáner.

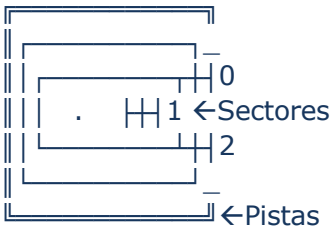
SALIDA.- Monitor: Un conjunto de puntos no de líneas.
Impresora: 1.- Matriciales: económicas y calidad bastante optima|ruidosa, y sufre decoloración. 2.- Inyección: Silenciosa y calidad buena; mantenimiento muy caro. 3.- Láser: Rápida y calidad excelente; precio.
Plotter.- Trazador lineal, se emplea para dibujar planos.

E/S.- Unidad de discos magnéticos.- Discos flexibles, recubrimiento de plástico, para transportar o como copia de seguridad, 360 Kb a 1´44 Mg. Discos duros, metálicos, para el trabajo, 10 Mg a 250 Mg y más.

	DOBLE	**ALTA**
5" 1/4	360 K. 40 p y 9 sect	1´2 Mg. 80 p y 15 sect
3" 1/2	720 K. 80 p y 9 sect	1´44 Mg. 80 p y 18 sect

Unidad CD-ROM.- Las unidades de CD-ROM son unidades de lectura óptica y hoy en día ya hay unidades de lectura y escritura. Son más fiables y con muchas ventajas ya que tienen mucha capacidad y mucho más rápido. Discos ópticos (El futuro).
STEMMER.- Son unidades de cinta para almacenar copias de seguridad, cintas de alta capacidad, la más pequeña 60 Mg.

INFORMATICA PROGRAMACION



La intersección de una pista y un sector es la unidad base de trabajo. En cada pista sector se almacena 512 K. La pista 0 es reservada para almacenar información especial y se almacena el BOOT, donde se encuentran las características de este. Después la FAT, es la tabla de localización de la información. Después el directorio, es una lista de la información que hay en el disco. La información se grava en el disco en bloque y estos bloques son los ficheros o archivos.

El disco duro a parte de tener el número de pistas y sectores muy variable dispone de múltiples platos por lo tanto más de dos caras. Cada cara tiene su cabeza lecturas y se dice cuantas cabezas tiene.

La estructura del disco duro es similar en cuanto al BOOT, FAT y DIR pero se suele guardar más pistas y sectores. También tiene una tabla de particiones porque un disco duro se puede dividir en distintas porciones (Diferentes Discos).

- SOFTWARE -

Existe un software básico que nos resuelve el problema de gestionar el funcionamiento del ordenador. **SISTEMA OPERATIVO** -> Con el que trabajaremos, es el MS-DOS, se encuentra siempre almacenado en un disco pero parte común de programas de gestión están en el programa ROM. Todos los sistemas operativos están compuestas de 2 partes.- 1. Control, que controla los periféricos. 2. Servicio que ayuda al usuario. El MS-DOS es la adaptación de 2 sistemas que había antes:

UNIX

CP/M <- Ya desaparecido.

Aparte de estos dos entran otras categorías:

APLICACIONES.- Son todos los programas específicos para resolver tareas concretas.

LENGUAJES DE PROGRAMACION.- El ordenador inicialmente entiende el binario pero dar las ordenes en binario es muy complejo por eso se opto por crear un simbolismo para controlar ordenes. Estos son los Lenguajes de programas.

El 1º fue el Ensamblador → Códigos de tres letras que simbolizaban instrucciones.

LENGUAJES DE PROGRAMACION

Específicos | - Basic
| - Cobol
| - Pascal
1.- Compilados | - Lisp
2.- Interpretes | - Sql
| - C

La compilación hace una traducción total y con anterioridad a

la ejecución del programa.

Los interpretes hace una traducción instrucción a instrucción y simultanea al momento de la ejecución.

Son dos métodos diferentes y cada una tienen ventajas:

1.- La compilación solo es necesario realizarla una vez, y ya me sirve para ejecutar múltiples veces y tiene el inconveniente de que me veo obligado.

2.- Los lenguajes interpretados tiene la ventaja que se pueden ir traduciendo partes del

programa y tiene el inconveniente que cada vez que se quiere traducir el programa cada vez que se ejecute.
Turbo → Permite la interpretación del trabajo y compila al final.

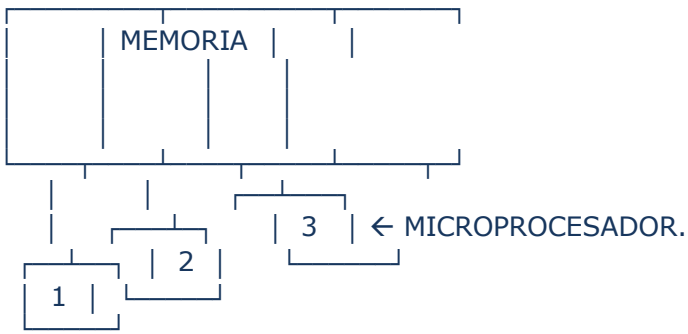
TIPOS DE SOFTWARE

Modos de explotación de un ordenador:

Puestos de trabajo. Tareas.

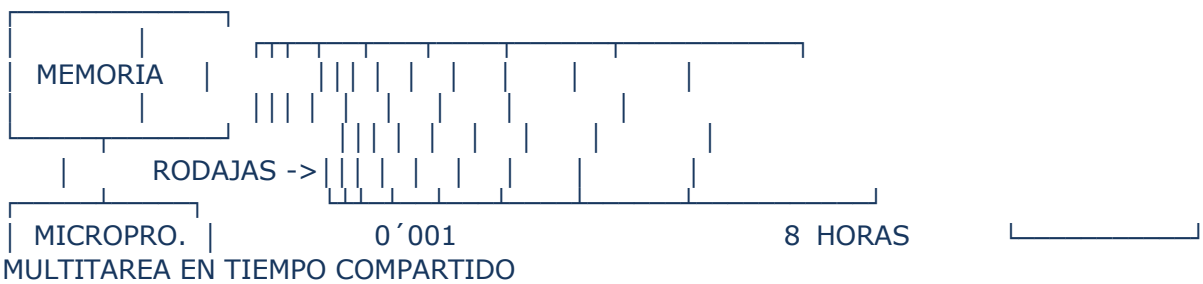
- Monopuesto
- Multipuesto
- Monotaréa
- Multitarea

Se optó por una gran memoria para la Multitarea.



Si la empresa crece y necesita más:

Había que buscar hacer la Multitarea con 1 sola memoria y un microprocesador.



MULTITAREA EN TIEMPO COMPARTIDO

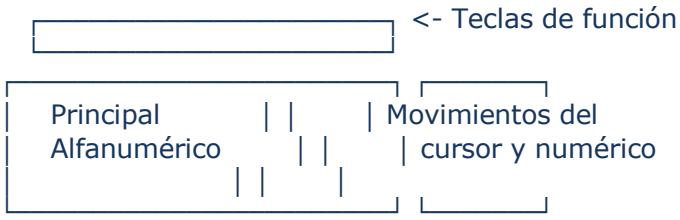
En este tiempo sí es capaz de realizar bastantes tareas.
Hay que crear prioridades, poner en unas tareas más tiempo que en otras.

INFORMATICA PROGRAMACION

Teleproceso.- El teleproceso consiste en conectar un terminal a un ordenador a larga distancia. Para conectar dos ordenadores es necesario unirlos, por un cable, al ser larga distancia se conecta por la red telefónica. El teléfono es analógico y no eléctrico y el ordenador necesita un aparato llamado Módem.

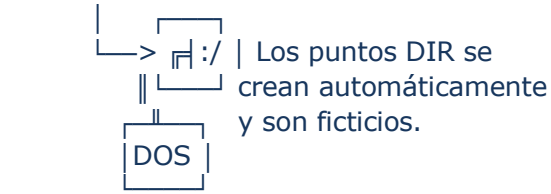
MS-DOS

C:\> <- Petición de orden.
D: C: -> Disco duro. <- Unidad base de trabajo.
A: B: -> Discos flexibles.
\ -> Directorio Principal.
> -> Separador.



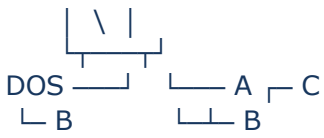
Control+C -> Interrumpe.
Control+una letra -> funciones especiales.
ALT GR -> Para obtener los terceros caracteres de las letras y permite obtener un carácter mediante un número.
Tabla ASCII <- Código binario -> 01000001="65"
Tabla ASCII <- Todos los caracteres con que se pueden trabajar.
La tabla se divide en 128 y 128 la 1ª es la estándar y no se pueden cambiar y en la 2ª los especiales y sí se pueden cambiar para adaptar la tabla a ese idioma.
Bloqueo de desplazamiento -> Se utiliza normalmente para un documento más grande que la pantalla.
Control+Alt+Supr -> Inicializa el ordenador.
Versiones.- El MS-DOS tiene diferentes versiones y nosotros veremos el 6.2.
VER.- Nos muestra la versión del sistema operativo.
CLS.- Borra la pantalla.
Para conocer la información se visualiza la información, y se hace con DIR. La primera columna tiene el nombre y la segunda la extensión, cumple la misión de decirnos que contiene el fichero, pero no siempre es fijo. La tercera que te indica si es directorio, la cuarta con el tamaño en bits, la quinta la fecha la ora de creación o ultima actualización.

. DIR
..DIR ← Enlaces de directorio ← esto es lo que simboliza.



DIRECTORIOS

Crear.- No puede existir, cuando se crea está determinado por su situación.
Creamos PEM[3]



Camino.- Para expresar el camino de un directorio, desde el principio y todos los que voy pasando.

Para crear | MKDIR | → U:Camino
| MD |

Visualización del árbol.- TREE → U:Camino. Con opción /F muestra los ficheros. Añadiendo en cualquier orden de visualización | MORE muestra la orden con pausas.
Y con Control+C interrumpe.

Borrar.- Para borrar este debe de existir y estar vacío
Para borrar | RMDIR | U:Camino
| RD |

Para cambiar de directorios | CHDIR | U:Camino
| CD |

Para borrar un directorio prescindiendo de si hay algún directorio o ficheros dentro. Para borrar una rama entera del árbol DELTREE U:Camino.
Para cambiar el nombre, MOVE U:Camino.
MEM /D → Para ver la colocación de la memoria y capacidad de disco.

Dir /AJ.- Muestra los ficheros ocultos.
Dir /AD.- Muestra los directorios, pero tan solo los directorios.

FICHEROS

Principalmente lo que se suele hacer con los ficheros es copiarlos para la prevención de una perdida. Para trabajar con un fichero es necesario indicar la U:Camino\nombre.ext -> Especifichero.
Copiar.- Para instalar un fichero la orden es COPY Especifichero (origen) Especifichero (destino).

CONFIG.SYS |
| Importantes para el arranque.
AUTOEXEC.BAT |

Dentro de Dos hay ficheros .INI que queremos copiar sin la ayuda del asterisco.
Los de extensión .INI, iniciación para el Windows.
El * -> Permite sustituir parte o todos los caracteres.
*.- Conjunto. ?.- Uno solo.

BORRAR

3Programador Entorno Microordenador

Borrar un fichero consiste en quitar su nombre del directorio y readmitir ese espacio como reutilizable.

| DEL | Especifichero.
| ERASE |

Existe un parámetro donde te pide confirmación y es /P.

VISUALIZAR EL CONTENIDO

Para visualizar un fichero se consigue con TYPE (Especifichero).
Fasthelp dir -> Ayuda rápida. = Dir /?
Help dir -> Una ayuda más extensa.
Los que non tienen tabla ASCII, con type, es intendible. Ejecutables con ASCII.
En ocasiones necesitamos o queremos cambiar el contenido de un fichero ASCII. Para crear o cambiar es necesario un editor.
Un editor es un procesador de texto a pequeña escala para crear programas. El que tiene el MS-DOS es EDIT y se suele acompañar del Especifichero.

CONFIG.SYS | -> Para la configuración.
AUTOEXEC.BAT |

FICHERO CONFIG.SYS

Para configurar Type config.sys y se ven las ordenes que hay gravadas. Varias ordenes empiezan por Device.- Se utiliza para cargar los programas.
Rem -> Poniendo esta orden se utiliza esa línea como comentario y es como si la orden no estuviera.
El MS-DOS directamente solo puede trabajar con los 640K primeros de la memoria, el resto que será expandida y extendida, etc.
HIMEM.sys.- Que se puede trabajar con la memoria superior.
EMM386.- Poder utilizar expandida y extendida al mismo tiempo, indiferentemente. Para configurar ambas.
BUFFERS |
|Importantes, Configura el acceso al trabajo con las FILE | unidades de disco.

FILE.- Determina cuantos ficheros podrán estar abiertos simultáneamente. El número de files debe estar comprendidos entre <30,40 -- 100> nunca sobrepasar los 100.
BUFFERS.- Establece que porción de disco se lee en cada acceso. Este número no definido(podemos indicar el número que queramos), pero no se debe poner números altos porque se ocupa demasiada memoria.
DEVICE.- Carga los programas de configuración.
DOS=HIGH o DOS=UMB.- Al hacer esta operación sitúa el MS-DOS en la memoria alta.
COUNTRY.- Para determinar formato; de fecha, ora.
LASTDRIVE.- Determina cual será la ultima letra de unidad (A,B,C).
SHELL.- Shell=c:\dos\comand.com c:\dos\ /P. Indica el camino de búsqueda del comand.com o interpretes de parámetros. Con /P ejecuta el autoexec.bat.
STACK.- Pilas, espacio para cuantos retornos.
REM.- Líneas de comentario.
FCBS.- Cuando ya a leído un bloque de información ya no va al disco para volver.

INFORMATICA PROGRAMACION

FICHERO AUTOEXEC.BAT

Bat, contiene ordenes del dos.
Se autoejecuta, de echo lo ejecuta el comand.com.
@ECHO OFF.- Que no se presenten las ordenes del que hay en pantalla mientras se ejecuta.
PROMPT.- Configura el inductor.
PATH.- Indica los caminos de ejecución.
MODE.- Selecciona la página de códigos. La 850 permite mayúsculas acentuadas.
Mode con codepage prepare=((437) c:\dos\ega.cpi).
KEYB SP.- El tipo de teclado (SP, Spanisch). Se puede conseguir el teclado nativo Control+Alt+F1. Para volver al otro teclado Control+Alt+F2.
DOSKEY.- Es una utilidad que permite almacenar y ejecutar las ordenes utilizadas.
Estos dos últimos comandos pueden tener un LH delante e indica que se cargue en memoria alta.

CAMBIAR NOMBRE

Renombra el fichero, cambia nombre o extensión.

REN | Especifichero - nombre extensión.
RENAME |
Ej.- REN c:\pem\provin.txt comuni.pro

Existe otra forma de cambiar el nombre además del emplazamiento y es la orden MOVE Especifichero Especifichero.

REESTABLECER

Permite recuperar ficheros borrados. Es posible mientras no se grave nada en el espacio.
UNDELETE Especifichero.
Ej.- Undelete c:\pen*.*

COPIA MASIVA

La copia masiva consiste en poder copiar directorios enteros o parte de los mismos con la peculiaridad de que también copia los directorios.
XCOPY Especifichero Especifichero /S/E.
/S.- Que se copien todos los directorios que hay en esa rama.
/E.- Que copie los directorios aunque se encuentren vacíos.

ATRIBUTOS

Los ficheros disponen de unos atributos internos que caracterizan el trabajo con los mismos. Estos atributos pueden ser:
R.- Acceso de solo lectura.
H.- Atributo de archivo oculto.
S.- Archivo de atributo de sistema.
AR.- Atributo de indicador de lectura y escritura o también de archivo modificado.
Estos atributos pueden ser desactivados o activados.
ATTRIB +,- ATRIBUTO Especifichero.
Attrib con el + y el especifichero nos muestra todos los ficheros con sus atributos.
ATTRIB +R +H C:\PEM\COMUNI.PRO
Si queremos que aparezca los ocultos la opción será /A. Si queremos ver las de solo un

atributo por ejemplo los de lectura -> /AR.

DISCOS

FORMATEARLO

FORMAT U: En caso de los discos flexibles para evitar la perdida de la información, se grava automáticamente en Mirrow.Fil.
/U.- No guarda la información en Mirrow.Fil
/Q.- Formateo rápido, solo a discos ya formateados.
/S.- Transfiere la parte de arranque del sistema operativo a ese disco, con ello creamos un disco de arranque.
/F: tamaño.- Cuando formateamos discos de doble densidad.
Para cambiar el nombre de la etiqueta LABEL U:.

TEST DE DISCO

Permite encontrar, incluso, corregir problemas o errores en el disco, estos problemas pueden ser físicos o lógicos.
SCANDISK U:
Si hay cadenas perdidas se eliminan/sin deshacer.

DESFRAGMENTAR

Para reunir la información que está desfragmentada se utiliza DEFRAG U: iremos a configurar y después a método para optimización plena.

COPIAR DISCOS

Se realiza una duplicidad exacta, una copia exacta en tamaño y densidad.
La orden es DISKCOPY U: U: -> solo acepta copiar discos flexibles.
Al hacer la copia, si el disco de destino tiene alguna información la pierde. Surgieron unos programas llamados Copiones que te permiten copiar juegos y programas.

TRANSFERENCIA DEL SISTEMA

La orden es SYS U: -> Transfiere | IO SYS |
| MSDOS SYS |
| COMMAND COM |
| SYS COM ←|← Para transferir al disco duro.

TABLAS DE PARTICIONES

Solo existen en los discos duros y es donde se especifica la división del disco. El disco duro por su tamaño puedo tenerlo dividido en 2 partes iguales y se realiza para instalar 2 sistemas operativos diferentes (MSDOS,UNIX) o para gestionarlo como dos discos duros (C,D).
El MSDOS solo puede gestionar 500 Mb, no más, pero puedo meter un disco duro de 1 gigabyte y dividirlo. Por ejemplo para 2 personas.
El comando es FDISK.

VAMOS A HACER UN DISCO DE ARRANQUE

INFORMATICA PROGRAMACION

Formatear con sistema -> Format /S
Ficheros que existen que pueden ser necesarios:
SYS.-, KEYB SP, SCANDISK, FORMAT, FDISK, HIMEM, COUNTRY, DOSKEY.
Crearemos un config.sys y un autoexec.bat con:
El primero con Himem, Country, Files, Buffers.
El segundo Prompt, Keyb, Doskey.

COPIAS DE SEGURIDAD

Siempre deben hacerse y los que se deben copiar son los datos. Cuando estos datos son cambiantes las copias han de hacerse con regularidad.
El programa para hacer copias de seguridad es MSBACKUP, desde este programa se podrán hacer las copias, verificarlas y restaurarlas.
La primera vez que se ejecuta el Msbackup realiza una prueba de adaptación al ordenador. Y requiere dos discos.
El proceso de copia comienza seleccionando archivos y marcando los que queramos.
Copiar en.- Permite seleccionar donde vamos a realizar la copia y cual es el formato y unidad.
Opciones.- Verificar datos → Que se comprueben los datos si están bien leídos. Se hace más lento el programa.
Comprimir datos → Indica que se compriman los datos para que ocupen menos discos.
Proteger con contraseña → No se suele utilizar, menos cuando deseas proteger una información.
Aviso de sobre escritura → Es absurdo seleccionar esto puesto que ya sabemos que el disco puede tener información.
Dar formato siempre → Es recomendable.
Usar corrección de errores → Que marquen los errores y que siga imprimiendo. Pero no es recomendable puesto que puede haber una pequeña porción estropeada pero con información y no la grabaría.
Conservar catálogos antiguos → No se hace por que si no tendríamos muchos. Es como un índice con los directorios y ficheros.
Avisos sonoros → Por supuesto.
Salir después de copiar → A gusto del consumidor.
Los discos deberían rotularse con el nombre del programa y la fecha del mismo.

DATE.- Permite consultar y cambiar la fecha del sistema.

TIME.- Permite poner el inductor del sistema. Si se indica así, sola sin parámetros nos muestra el Prompt por defecto. Si lo acompaño de un texto el Prompt por defecto. Si lo acompaño de un texto el Prompt, será ese texto. Para establecer que en el Prompt salgan datos especiales existen unos caracteres que precedidos del signo dólar, simbolizan dicha información.
$P→ C:\.- Simboliza la unidad y el camino. $G→> $L→< $E→La ora del sistema $D→La fecha del sistema.
$H→Borra el carácter anterior. $_→Un cambio de línea.
MSD.- Ofrece información sobre las características del ordenador.

PUERTOS

Son colectores estándar para diferentes periféricos los:
| LPT
| COM
LPT.- Paralelos. La conexión se hace en un conjunto de líneas paralelas.
COM.- Puertos serie o RS232. Una única línea de comunicación de datos. Envían la información bit a bit.

A los puertos paralelos se suelen conectar las impresoras y en los puertos en serie {ratón, módem, u.disco externa, etc}.

IRQ.- Las irqs son las interrupciones del sistema que se emplean para controlar los periféricos. Es necesario de vez en cuando tiene que ir refrescando la pantalla, hay pequeños programitas que se encarga de esto.

KEYB.- Selecciona el idioma que hay en el teclado. Identificado.

Idioma.- Sp→Español. Si quiero información detallada de Gr→Alemania. cualquier comando. Inmediato orden/? y extensa Help comando.

Xcopy.- /E Copia subdirectorios aunque estén vacíos.

/S Copia directorios y subdirectorios excepto, vacíos.

FICHEROS DE PROCESO POR LOTES

También llamados ficheros. Son los ficheros ASCII que se caracterizan porque su extensión a de ser BAT y contienen ordenes del MSDOS. La ejecución estos ficheros consisten en la sucesión de ejecuciones de los comandos del MSDOS que contiene. Hay un fichero BATCH especial que se llama Autoexec.Bat. Dentro de estos ficheros podrá incluirse cualquier orden conocida pero además existen ordenes especificas para este tipo de ficheros y que su ejecución fuera de los mismos carecen de sentido.

Por ejemplo:

Pause.- Lo que hace es esperar a que se pulse una tecla y se utiliza preferentemente para parar la ejecución del fichero BAT y que el ordenador pueda leer un mensaje.

ECHO.- Esta instrucción cumple dos misiones. Si lleva on/off activa o desactiva la presentación de las instrucciones durante la ejecución del fichero de proceso por lotes. Si va acompañado de un texto presenta ese texto en pantalla. Si le pongo una @ Echo off y después cls, para limpiar la pantalla.

El comando IF. IF Condición.- Me permite tomar decisiones dentro de un fichero de proceso por lotes. La condición estará formada si una cadena de caracteres es igual a la otra.

Cadena1==Cadena2

Exist Especifichero→Existe un fichero o si se ha producido un Error Level nº. A continuación irá la orden que quiero que se ejecute.

Instrucción GOTO etiqueta.- Permite saltar de un punto a otro dentro de un fichero BAT. La etiqueta será un nombre estándar y figurará en el punto donde se quiera saltar precedida de dos puntos.

:Etiqueta :FIN

VARIABLES DE ENTORNO O VARIABLES

Cuando mandamos ejecutar un fichero BAT, el nombre puedo acompañarlo de varios parámetros.

LOTE2 B: ←Especifico la unidad.

Estos parámetros que especifico se van a almacenar en unas variables(lugares) que se nombran desde %1...hasta...%9.

EJEM.- @Echo off, cls, If %1Z==Z GOTO FIN, Del %1, Rd %1, :FIN

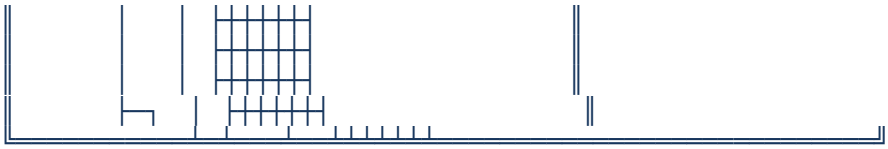
GESTION DE MEMORIA

La memoria es donde almacena la información para procesarla.

El MSDOS divide la memoria en dos partes fundamentales. Memoria convencional, memoria entendida.

0K 640K ↓1024K EXTENDIDA

INFORMATICA PROGRAMACION



MEMO CONV. ↑SUPERIOR ↑64K 2MB

Dentro de la memoria convencional está los 640K que se definen al usuario. El resto de la memoria hasta el MB se llama memoria superior. Esta memoria es destinada para los datos del Hardware. Hay una parte reservada para la tarjeta gráfica y ahí se podrá situar los drivers para los dispositivos periféricos. En la memoria extendida existe un pequeño bloque de 64K llamado memoria alta que se utiliza para cargar la mayor parte del sistema operativo y así liberar memoria convencional de usuario.

Para controlar la memoria existe un comando llamado MEM.
MEM.- Me informa sobre toda la memoria y la que tenemos libre.
La memoria Extendida.- La memoria que está mas allá, surgió del procesador 286, se le conoce como memoria XMS, esta memoria puede ser usada por programas que estén preparados para gestionar dicha memoria pero aún así es necesario instalar un controlador o gestor de dicha memoria este es el HIMEM.SYS.
La memoria Extendida no se puede usar en modo real sino que principalmente se emplea para situar información en ella para incrementar la memoria de trabajo es necesario memoria Expandida. Esta memoria que se conoce como memoria EMS, tiene la ventaja de que funciona como cualquier microprocesador pero el inconveniente de que es algo más lenta. Este tipo de memoria es una emulación de la memoria extendida. La gestión de la memoria expandida se realiza a través de bancos, o sea bloques de 16K cada uno y existen una zona llamada marco de página que se encuentra localizado en la memoria convencional superior donde se podrán situar 4 bancos. Para emular la memoria expandida es necesario cargar un driver. Este es el EMM386.
De todos modos para optimizar la memoria en cuanto su configuración extendida, expandida y la colocación del software en memoria, el MSDOS dispone de un programa de gestión llamado MEMMAKER.

ARJ.- X,E DESCOMPRIME | ORIGEN -- DESTINO
A COMPRIME | DESTINO -- ORIGEN

INFORMATICA PROGRAMACION

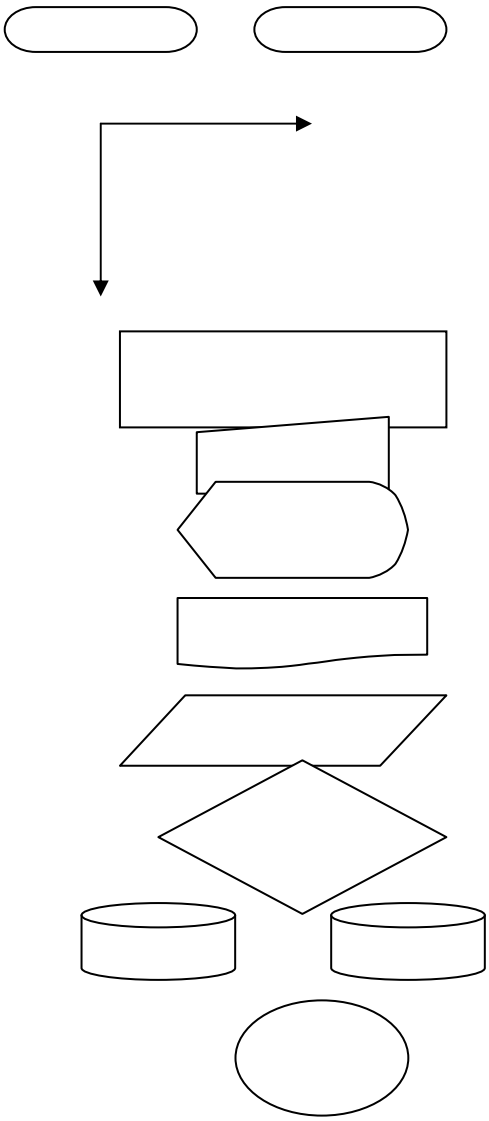
PROGRAMACION

METODOLOGIA DE LA PROGRAMACION

Se encarga de realizar estudios sobre los problemas para obtener los pasos detallados que resuelvan dicho problema. Estudia los problemas paso a paso teniendo en cuenta las salvedades de ese programa. El estudio del programa a de ser standard, es decir, que lo entienda cualquier persona ya que la programación la pueden realizar varias personas.
Para que esto sea standard se ha hecho un simbolismo.

El estudio se divide en dos partes:
- Sistema o Equipo Requerido.- Se representa en un esquema llamados ORGANIGRAMA.
- Ordenes o Pasos para Resolver ese Problema.- ORDINOGRAMA.

Los Organigramas solo son necesarios para grandes aplicaciones en grandes ordenadores, sobre todo cuando se trabaja con multipuesto.
Los Ordinogramas. Para generar los Ordinogramas se emplea el siguiente simbologismo:



Inicio/Fin. Se emplea para indicar el comienzo o la Fin.

FLUJO. Establece la secuencia de ejecución. Partirá una única flecha del inicio y llevará una única flecha al fin, permitiéndose ramificaciones durante el proceso, aunque estas deberán confluir a un solo flujo, sin quedar ninguno perdido.

PROCESO. Se emplea para realizar cálculos aritméticos. Dentro del polígono se empleará la formula matemática.

Indica la petición de un valor al teclado.

SALIDA PANTALLA. Indica que se presenta algo en pantalla.

SALIDA IMPRESORA.

E/S ESTANDAR. Se emplea para el acceso a los discos tanto para leer como para escribir.

Es un condicional. Se emplea para tomar decisiones y permite dividir el flujo para poder realizar dos procesos paralelos dependientes de la condición.

ACCESO A DISCO. Se emplea para abrir y cerrar ficheros.

ENLACE. Sirve para enlazar dos puntos distantes en el Ordinograma.

INFORMATICA PROGRAMACION

Concepto Variable.- No se puede trabajar con valores concretos. Es un nombre que representarán valor en un momento determinado y que podrá variar ese valor a lo largo de ese programa.

COBOL

El Cobol es un lenguaje de alto nivel compilado y orientado hacia la gestión de empresas y de ahí viene su nombre.
Lenguaje común orientado hacia los negocios. Aunque existen diferentes versiones es el más estándar existiendo en leves diferencias entre una y otra versión, nosotros vamos a trabajar con RMCOBOL, que se adapta perfectamente a Cobolands85.

Elementos de Lenguaje

Un programa Cobol se encuentra estructurado en 4 divisiones. Las cuales son:

1.- IDENTIFICATION DIVISION
2.- ENVIRONMENT DIVISION
3.- DATA DIVISION
4.- PROCEDURE DIVISION

A su vez cada división se divide en varias secciones excepto la 1 y cada sección en párrafos.

En la creación de un programa intervienen 3 ficheros, 2 de los cuales son fundamentales.
El 1º llamado, Línea Fuente va a contener el programa que nosotros escribimos y se caracterizará porque la extensión va a ser .CBL, este fichero será de tipo texto y dispone de unas estructuras determinadas.
El 2º fichero es el llamado fichero objeto, su extensión será .COB y tendrá el resultado de la compilación del fichero fuente.
- Para escribir el fichero fuente es necesario tener en cuenta las siguientes normas.

- De la columna 1 a la 6 se empleaban antiguamente para estipular el número de línea. Hoy en día numerar las líneas es opcional. Nosotros la dejaremos en blanco.
- La columna 7 podrá contener un guión o un asterisco o nada, el guión se empleará para indicar que continuará un literal de la línea anterior, el asterisco para indicar que esa línea es un comentario.
- De las columnas 8 a 72 estas columnas se emplean para escribir el programa fuente.

Se dividen en 2 áreas.

El área a) entre columnas 8,11 y área b) 12,72. En el área a) se debe a comenzar a escribir los nombres de divisiones, los nombres de secciones, nombres de párrafos, nombres de procedimiento. El indicador F0 y los números de nivel 01 y 77. En el área b) debe comenzar el resto de declaraciones y las sentencias, instrucciones de programa. Estas son las partes que componen cada línea del programa.

REGLAS DE CODIFICACION

1. Los nombres de división, sección y párrafo. Se escriben solo en una línea y finalización por punto.
2. Cada entrada o sentencia del programa debe finalizar por punto.
3. Si una línea de programa no cabe en una línea física se continuará en una línea o líneas siguientes pero a partir del margen b).
4. Se pueden dejar líneas en blanco en medio del programa.
5. El punto, la coma y el punto y coma, no deben ir precedidos de un espacio en blanco

pero si seguidos.

CARACTERES COBOL

Para formar un programa se podrán emplear los siguientes caracteres tanto desde la A a la Z tanto en Mayúsculas como en minúsculas excepto la letra Ñ.
Los números de 0 a 9 y los signos especiales ,.;`()></+-*=

CONSTANTES

Una constante es un valor que no cambia durante la ejecución del programa Cobol las constantes pueden ser números, constantes de caracteres, o una constante figurativa.
Constantes numéricas.- Se expresaran con o sin signo y un máximo de 18 dígitos. EJ→3483 -317 15.3745
Constantes de caracteres.- Una constante de caracteres o literal es un conjunto de caracteres cerrado entre comillas simples o dobles. No podrá exceder los 2047 caracteres. Y podrá contener cualquier concurrencia de los mismos.
Constantes figurativas.- Es una constante predefinida por el Cobol y representa un valor. Las más importantes son las siguientes:
ZERO→Representa el valor cero y también puede referenciarse con 0 o Z o Cerves.

SPACE→Representa uno o más espacios en blanco.
HIGH-VALUE→Se emplea para literales y representa el valor más grande.
LOW-VALUE→Se emplea para literales y representa el valor más pequeño.
ALL "literal"→Representa la repetición del literal.

PALABRAS RESERVADAS

Son identificadores predefinidos que tienen un significado especial para el Cobol y para el que no podremos usar aparte de este significado. Por ejemplo como nombre de variable.
Identificadores o nombre Cobol creadas por el usuario, para formar un nombre hay que tener las siguientes reglas:

1. Un nombre consta de un máximo de 30 caracteres.
2. Estos caracteres pueden ser letras dígitos y el carácter guión.
3. No deben comenzar o finalizar con un guión.
4. No podrá ser una palabra reservada.
5. Dispondrá forzosamente de al menos una letra excepto en los nombres de párrafo que podrá ser un dígito.

OPERADORES

Los operadores son símbolos que indican la manipulación de datos.
Operadores Aritméticos.- + - * / = () **
Operadores de Relación.- = < > <= >= Serán los empleados en las condiciones.

ESTRUCTURA DE UN PROGRAMA COBOL

Un programa fuente Cobol es un conjunto de entradas y sentencias sintácticamente correctas.
Estará formado por las 4 divisiones:

IDENTIFICATION DIVISION
PROGRAM-ID nombre.prg.
AUTHOR nombre.programador.
INTALATION lugar de instalación.

1ª DIVISION

TODO PROGRAMA COBOL DEBE EMPEZAR ASI,CON

INFORMATICA PROGRAMACION

LA IDENTIFICACION DIVISION.

```
DATE-WRITTEN   fecha de escritura.
DATE-COMPILED  fecha de compilación.
SECURITY       comentario.
REMARKS        comentario.
```

- El program-id, sirve para especificar el nombre del programa, esto lo utilizará el compilador.
- El author, es opcional y se expresa para poner el nombre del programador.
- Intalation, es para poner el lugar de instalación del programa.
- El date-written, sirve para expresar la fecha de escritura.
- El date-compiled, sirve para expresar la fecha de compilación.
- Scurity para delimitaciones de uso.
- El remarks para comentarios adicionales.
- Todos estos párrafos hoy en día se suelen indicar pero están concebidos porque el Cobol lo suelen utilizar varios programadores.

```
ENVIRONMENT DIVISION            2ª DIVISION
 CONFIGURATION SECTION
   SOURCE-COMPUTER      comentario
   OBJECT-COMPUTER      comentario
 SPECIAL-NAMES
      Nombres especiales.
   DECIMAL-POINT IS COMMA
 IMPUT-OUTPUT SECTION
   FILE-CONTROL
      Control de ficheros.
```

- El environment division permite definir las características del equipo que se usarán en el programa.
- La primera section es opcional y se indica para estipular el modelo del ordenador en que se escribe el programa (soure-computer) el modelo del ordenador en el que se ejecutará (object-computer).
- El párrafo se emplea para cambiar configuraciones y definir nombres especiales, la cláusula mas importante es, decimal-point is comma, esta cláusula indica que la marca decimal pasará a ser una coma.
- La sección de entrada y salida es opcional y se empleará para definir los ficheros de datos.

```
DATA DIVISION            3ª DIVISION
 FILE SECTION
    Descripción de fichero
    Descripción de fichero
    ....
 WORKING-STORAGE SECTION
    Descripción de variables
 SCREEN SECTION
    Descripción de pantallas
```

- Data division. Se describirán todas las zonas de datos que vamos a emplear en el programa son variables, registros o zona de trabajo para cada tipo principal de zona existe una sección principal.
- File sección. Se emplea para la refinición de ficheros.
- El working-storage section es la sección de trabajo y se emplea para definir variables y zona de trabajo.
- El screen section se emplea para definir estructura de presentación en pantalla.

INFORMATICA PROGRAMACION

DESCRIPCION DE UNA ESTRUCTURA

nn nombre | Picture | item value valor.
| Pic |

La descripción consta de un número de nivel, un nombre y una serie de cláusulas. Se trata de una estructura Jerarquizada compuesta por una o más inscripciones Jerarquizadas. Estas inscripciones Jerarquizadas estarán formadas por una o más inscripciones Jerarquizadas. Estas otras inscripciones Jerarquizadas podrán repetirse.

NUMEROS DE NIVEL

Sirven para indicar la relación o jerarquía que existe entre un campo y el resto de los campos. Los campos podremos clasificarlos en:
Elementales, tipo de campo que pueden subdividirse.
Compuestos, campo que se encuentra dividido por varios campos.
Independientes, es un campo aislado en la sección trabajo.

Los números de nivel que pueden utilizarse son el 01, del 02 al 49, 77 y 88. El número de nivel superior es 01.

Las normas para utilizar los números de nivel son:

1. Un campo compuesto tendrá un nivel superior que cualquiera de sus campos elementales.
2. El número de nivel 01 solo podrá pertenecer al nombre del registro o estructura.
3. Los números de nivel 02 a 49 pueden pertenecer a cualquier campo sea elemental o compuesto.
4. Los números de nivel utilizados no tienen porque ser correlativos.
5. El número de nivel 77 se utiliza para describir campos independientes.
6. El número de nivel 88 se utiliza para describir nombres de condición.

Por ejemplo para describir la siguiente estructura emplearemos:

DATOS-EMPLEADOS

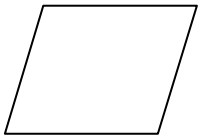
NOMBRE	DIRECCION			SUELDO
40 CARACTERES	CALLE 35 CARACTERES	CIUDAD 20 CARACTERES	PROVINCIA 15 CARACTERES	7 CARACTERES

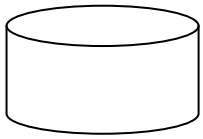
```
01 DATOS-EMPLEADOS
   02 NOMBRE
   02 DIRECCION
      03 CALLE
      03 CIUDAD
      03 PROVINCIA
   02 SUALDO
```

INFORMATICA PROGRAMACION

METODOLOGIA

Para la gestión de ficheros vamos a destapar 2 ficheros:

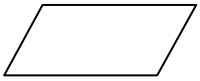




La estructura de ficheros, es un conjunto de información.
1º.- Se encuentra dividido en elementos que almacenarán todos los datos, por ejemplo los de cada alumno. A esto se le llama Registro. Estos registros se van a dividir en unas unidades que van a almacenar un dato concreto a uno de estos se le llamará campo.
Nosotros para manipular utilizaremos campos pero para acceso a disco los registros.

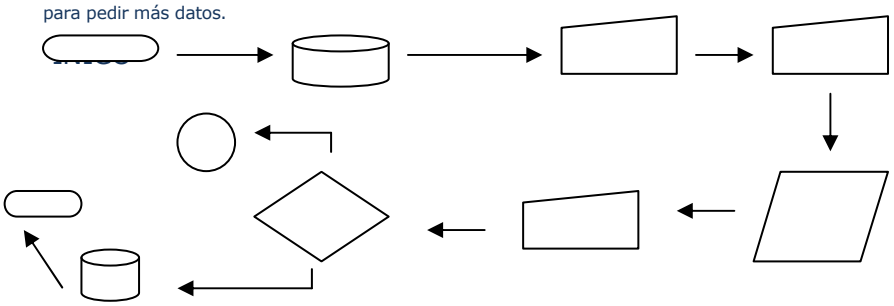
Para trabajar con un fichero primero abrirlo y por ultimo cerrarlo→→→→

En el medio se lee o esc

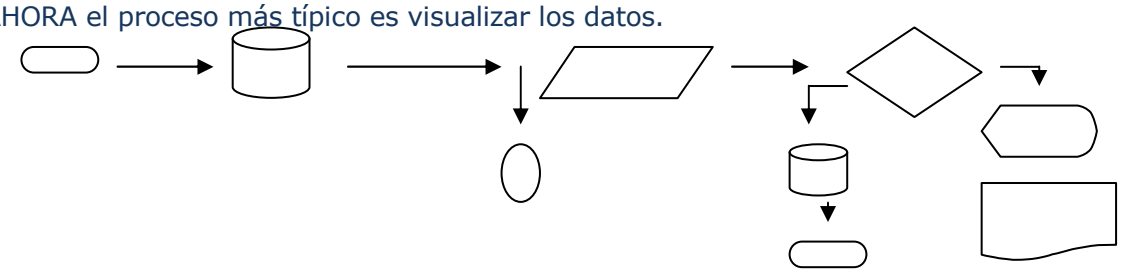


CREAR/AÑADIR DATOS EN UN FICHERO.

Tengo que poder repetir para pedir
los datos y lo que se suele hacer es una entrada
para pedir más datos.



AHORA el proceso más típico es visualizar los datos.

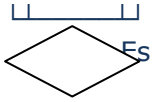


Hay una forma de expresar que sea el fin de un fichero:

/*-Fin de registro
EOF- " " "
FF- " " "

Si no es el final de fichero que los represente o imprima.
Estos esquemas básicos pueden sufrir variaciones sobre todo en pantalla o impresora de la segunda parte.

Llamada a un subprograma, subrutina. Para realizarlo una sola vez.



Es como un indicador (Dentro(sw = 0)).

INFORMATICA PROGRAMACION

DESCRIPCION

nn nombre |picture | item
|pic |

Nombre, será un nombre Cobol y cumplirá sus características si este nombre no va a ser referenciado podrá indicarse el nombre común FILLER.
PIC, PICTURE. ITEM.- Esta practica permite declarar en tipo y tamaño de un campo elemental. El tipo determina que información puede almacenarse en un campo, así como las operaciones que pueden realizarse, con el se podrán clasificar en alfabéticos, alfanuméricos y numéricos.

La longitud indicará el número máximo de información que se podrá almacenar.

Campos Alfabéticos.- Permiten almacenar letras y/o espacios en blanco. Se identificarán mediante una A. Para indicar el tamaño se repetirá este carácter (AAAAA) tantas veces como sea deseado o se representará al lado con paréntesis (5).

Campos Alfanuméricos.- Permiten almacenar cualquier carácter y se identifica con una X.

Campos Numéricos.- Permite almacenar valores, números y el número de cifras, máximo es de 18. Se identifican con un 9, pero además disponen de unos caracteres que realizan otras funciones:

S→Indica la aparición de el signo. S999 (+756, - 6)
V→Indica la marca decimal. 999V99 534.75

Value→Un valor inicial a un campo elemental o independiente. El valor podrá ser un dato concreto y una constante figurativa.
Punto→Todas las descripciones aunque sean de grupo y solo lleven el nombre, deben finalizar este punto.
Procedure Division→Contendrá las operaciones necesarias para resolver el problema dado, será un fiel reflejo del Ordinograma. Estará estructurado en párrafos. Un párrafo comenzará por un nombre y contendrá un conjunto de sentencias, este finalizará por un punto.

LA ESTRUCTURA VA A SER: PROCEDURE DIVISION.
Parrafo1 ————————
————————
————————.
Parrafo2 ————————
————————
————————.
ParrafoN ————————
————————
————.

SENTENCIAS BASICAS

INFORMATICA PROGRAMACION

Visualización de datos, sentencia DISPLAY, Permite representar información en la pantalla. El formato resumido es:

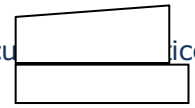
DISPLAY |Campo | Line nº Position nº.
|Literal|

Se corresponde con este símbolo→→→→

Display con un literal se debe emplear siempre antes de cualquier petición de datos antes de especificar el dato requerido antes de cualquier entrada de datos hay que hacer un Display.

Leer datos del teclado, sentencia ACCEPT, realiza una parada para solicitar un dato al teclado y asignárselo a un campo. El formato es:

ACCEPT CAMPO LINE Nº POSITION Nº →→→→



Sentencia de asignación y calculo COMPUTE. Permite realizar cálcu icos COMPUTE expresión aritmética→

Sentencia de finalización de programa. Finaliza el programa.

STOP RUN.

Condicional o sentencia IF→Permite tomar decisiones igual que un condicional y el formato es:

IF Condicion THEN
Sentencias **si**
Else
Sentencias **no**
END-IF

La parte de las sentencias no , podrán omitirse cuando la rama se encuentra vacía.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. AGENCIA.
AUTHOR. JAVI.
*INSTALATION. PROBADURA.
DATE-WRITTEN. 02-02-94.
DATE-COMPILED. 02-02-94.
SECURITY. EXCLUSIVO.
REMARKS. CUALQUIER COSA.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. PC-COMPATIBLE.
SPECIAL-NAMES.
        DECIMAL-POINT IS COMMA.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 DIAS PIC 99 VALUE 0.
77 NPER PIC 9(3) VALUE 0.
77 FLETE 9(7) VALUE 0.
77 PRECPER 9(6) VALUE 0.
77 TEMP PIC A VALUE SPACE.
77 PH PIC 9(4) VALUE 0.
77 GASTHOT PIC 9(8) VALUE 0.
77 DTO PIC 9(7) VALUE ZEROES.
77 INGR PIC 9(8) VALUE Z.
77 GASTOS PIC 9(8) VALUE 0.
77 BENEF PIC S9(7) VALUE 0.
PROCEDURE DIVISION.
ENTRADA.
```

```
        DISPLAY "DIAS DE ESTANCIA" LINE 5 POSITION 10 ERASE
        ACCEPT DIAS LINE 5 POSITION 27
        DISPLAY "NUMEROS DE PERSONAS" LINE 6 POSITION 10
        ACCEPT NPER LINE 6 POSITION 29
        DISPLAY "FLETE" LINE 7 POSITION 10
        ACCEPT FLETE LINE 7 POSITION 16
        DISPLAY "PRECIO PERSONA" LINE 8 POSITION 10
        ACCEPT PRECPER LINE 8 POSITION 29
        DISPLAY "TEMPORADA" LINE 9 POSITION 10
        ACCEPT TEMP LINE 9 POSITION 20
PROCESO.
        IF TEMP = "A" THEN
                MOVE 5000 TO PH
                ELSE IF TEMP = "B"
                        MOVE 3800 TO PH
                        ELSE
                                MOVE 3200 TO PH
                        END-IF
        END-IF
        COMPUTE GASTHOT = PH * NPER * DIAS
        IF NPER > 150 THEN
                COMPUTE DTO = GASTHOT * 0.03
                ELSE
                  MOVE 0 TO DTO
        END-IF
        COMPUTE INGR = NPER * PRECPER
        COMPUTE GASTOS = GASTHOT - DTO + FLETE
        COMPUTE BENEF = INGR - GASTOS.
SALIDA.
        DISPLAY "INGRESOS" LINE 12 POSITION 10
        DISPLAY "BENEFICIOS" LINE 14 POSITION 10
        DISPLAY INGR LINE 12 POSITION 20
        DISPLAY BENEF LINE 14 POSITION 22
        STOP RUN.
```

La orden para compilar el programa es:

RMCOBOL U:nombre | MORE

Hay un tipo de palabras que no se pueden utilizar porque son ordenes del Cobol.

Ejecutar →→→→ **RUNCOBOL U:nombre**

GESTION DE FICHEROS | COBOL

```
INPUT-OUTPUT SECTION.
FILE-CONTROL
        SELECT nombre.int-impresora ASSIGN TO RANDOM "especifich"
    .
        .
        SELECT nombre.int-impresora ASSIGN TO PRINTER "PRINTER"
DATA DIVISION.
FILE SECTION.
FD   nombre.int   |standard|
        LABEL RECORD |omitted |
        DATA RECORD  nombre-registro.
01      nombre-registro
        02  .
```

. Descripción de campo
.

DESCRIPCION DE FICHEROS

Un fichero se empieza a describir en Input-Output section. En el párrafo File control, aquí se asignará a un fichero determinado el tipo de este así como su correspondencia física. Esto se realiza con la cláusula SELECT. De fichero se le adjudicará un nombre interno que será el que empleemos en este programa para aludir al mismo con la cláusula ASSIGN seleccionaremos el tipo de fichero, se especificará en RANDOM para ficheros que se encuentren almacenados en disco.

Cobol permite manipular tres ficheros de datos dependiendo de su organización. Estos son los ficheros secuenciales, Relativos o Directos y Secuenciales Indexados.

Nosotros inicialmente trabajamos con los ficheros Secuenciales. Las características de estos ficheros es que el acceso a sus datos, o sea el registro, es únicamente secuencial. Un acceso secuencial es un acceso lineal, registro a registro desde el primero hasta el ultimo, siendo obligatorio pasar por el registro 1 y 2 si queremos llegar al tres.

El Especifichero será el nombre real junto con su unidad y camino que tiene el fichero de datos. Habrá que indicar una cláusula SELECT para CAD fichero de datos que manipulemos.

La impresora, el Cobol la gestiona como un fichero y por lo tanto se definirá de forma similar pero obviamente cambiando el indicativo de tipo de fichero. Esto lo hacemos en la DATA DIVISION, dentro de la FILE SECTION.

Cada fichero se describe con una cláusula FD→(FILE DESCRIPCION). La cláusula irá acompañada de el nombre interno del fichero (El mismo que indicamos en SELECT).

LABEL RECORD→Determina el tipo de nombre externo, standard se emplea para ficheros de disco cuyo nombre externo se confecciona con un Especifichero.

DATA RECORD→Se empleará para especificar el nombre al que adjudicaremos al registro.

A continuación a nivel 01 se indicará el nombre de registro y la descripción de cada unos de los campos. Tras haber concluido la descripción de los campos será cuando comience la siguiente cláusula FD.

ORDENES DE GESTION DE FICHEROS

Primero apertura de un fichero.- Para ficheros secuenciales. La orden para abrir el fichero es OPEN modo nombre.int.

MODO→Determina el modo de apertura. El modo de apertura variará según las operaciones que vaya a realizar en este fichero.

Los modos existentes son:

OUTPUT→Se emplea para leer registros del fichero.

I-O→Entrada/Salida.

EXTEND→Abre el fichero para añadir registros, gravar a continuación de los que tengo. Si en la cláusula SELECT le indicamos a continuación OPTTIONAL EXTEND→Solo si este existe.

CERRAR → CLOSE nombre.int → Gravar registros. Almacena en el fichero los datos que se encuentran en los campos de registro.
Estos han de ser situados con anterioridad a los campos.

La orden WRITE nombre.reg END-WRITE→Lectura, permite leer un registro del fichero situando los valores leídos en los campos, el formato de la orden será:

READ FICHERO

```
        AT END →→→→ Esto codifica
MOVE "*" TO FIN         .ordenes.
        NOT AT END
      .ordenes.
END-READ
```

En el NOT AT END figurará las ordenes que aparecen en la rama del NO de la condición de fichero. En AT END estableceremos un indicador para determinar cuando se realiza el final del proceso dado que normalmente junto a una lectura de un fichero secuencial siempre aparece un bucle que expresa la repetición de la lectura, por ejemplo, como indicador podemos que cuando esta variable tome un valor "asterisco" que establezca el final del fichero, por ello indicaremos MOVE "*" TO FIN.

BUCLES

Permiten repetir un conjunto de instrucciones siempre están de vuelta atrás en el Ordinograma. El BUCLE comenzará donde retorna la flecha y finaliza donde comienza el retorno. La condición que genera otro camino independiente será la condición del bucle. La instrucción que crea el bucle es: **PERFORM UNTIL** condición. Y se finaliza **END-PERFORM**.

En este caso como el fichero viene marcada por * sería FIN = "*".

UTILIZACION DE LA IMPRESORA

```
FILE-CONTROL.
.
.
SELECT IMPRE ASSIGN TO PRINTER "PRINTER".
FILE-SECTION.
.
.
FD IMPRE
        LABEL RECORD OMITTED
        DATA RECORD pepe
01 pepe PIC X(80)
```

```
X————————————————————————<———>————————————————————————X
30                        80              CABECERA 30
     |    6      |   (20)          |     12       | D
     |10 NOMBRE  24|LISTADO DE EMPLEADOS| SUELDO BRUTO | E
     | X————————X|                 | 9————————9| T 10
     |           |       9         |     7      | A
     |           |                 |            | L
                                               L
                        TOTAL SUELDO 9————————9   E
                                     14
```

80-20=60/2=30✡✡✡✡80-52=28/3=9✡✡✡✡9+40+9+3+7=68

Lineas de cabecera: disponen de solo texto.
Lineas de detalle: disponen de solo datos.

```
WORKING-STORAGE SECTION.
01 LINE1.
        02 FILLER PIC X(30) VALUE SPACE.
        02 F PIC X(20) VALUE "LISTADO DE EMPLEADOS"
```

```
01 LINEA2.
        02 F PIC X(19) VALUE SPACE.
        02 F PIC X(16) VALUE "NOMBRE".
        02 F PIC X(33) VALUE SPACE.
        02 F PIC X(12) VALUE "SUELDO BRUTO".
01 LINEA3.
        02 F PIC X(9) VALUE SPACE.
        02 NOM-I PIC X(40).
        02 F PIC X(12) VALUE SPACE.
        02 SUE-I PIC 9(7).
01 LINEA4.
        02 F PIC X(44) VALUE SPACE.
        02 F PIC X(14) VALUE "SUELDO TOTAL: ".
        02 SUET-I PIC 9(10).
```

Para adaptar el Ordinograma al Cobol, abrir la impresora, cerrar la impresora, cerrar la impresora. Para adaptar todavía mas el fichero de impresora al Cobol debo pasar los campos del fichero a la impresora.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PERSONAL.
AUTHOR. JAVI.
ENVIROMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
        DECIMAL-POINT IS COMMA.
IMPUT-OUTPUT SECTION.
FILE CONTROL.
        SELECT IMPRE ASSIGN TO PRINTER "PRINTER".
        SELECT PERSONAL ASSIGN TO RANDOM "A:\SERGAS.DAT".
FILE SECTION.
FD IMPRE
        LABEL RECORD OMITTED.
        DATA RECORD pepe.
01 pepe PIC X(80).
FD PERSONAL.
        LABEL RECORD STANDARD.
        DATA RECORD DATOS-EMPLEADOS
01 DATOS-EMPLEADOS
        02 NOMBRE PIC X(40).
        02 DIRECCION.
          03 CIUDAD PIC (15)
          03 CALLE PIC (35)
          03 PROVINCIA PIC (20)
        02 SUELDO PIC 9(7)
WORKING-STORAGE SECTION.
01 LINEA1.
        02 FILLER PIC X(30) VALUE SPACE.
        02 F PIC (20) VALUE "LISTADO EMPLEADOS".
01 LINEA2.
        02 F PIC X(19) VALUE SPACE.
        02 F PIC X(6) VALUE "NOMBRE".
        02 F PIC X(33) VALUE SPACE.
        02 F PIC X(12) VALUE "SUELDO BRUTO".
01 LINEA3.
```

```
        02 FILLER PIC X(9) VALUE SPACE.
        02 NOM-I PIC X(40).
        02 FILLER PIC X(12) VALUE SPACE.
        02 SUE-I PIC 9(10).
01 LINEA4.
        02 F PIC X(44) VALUE SPACE.
        02 F PIC X(14) VALUE "SUELDO TOTAL".
        02 SUET-I PIC 9(10).
77 FIN PIC X VALUE SPACE.
77 SUET PIC 9(10) VALUE 0.
PROCEDURE DIVISION.
INICIO.
        OPEN-OUTPUT IMPRE.
        OPEN IMPUT PERSONAL.
        WRITE RIMP FROM LINEA1 END-WRITE
        WRITE RIMP FROM LINEA2 END-WRITE
        PERFORM UNTIL FIN = "*".
        READ EMPLEADOS
                AT END
                MOVE "*" TO FIN
                        NOT AT END
                        COMPUTE SUET = SUE + SUET
                        MOVE SUET TO SUE-I
                        MOVE NOM TO NOM-I
                        WRITE RIMP FROM LINEA3 END-WRITE
        END-READ
        END-PERFORM
        MOVE SUET TO SUET-I
        WRITE RIMP FROM LINEA4 END-WRITE
        CLOSE READ
```

TABLAS

Las tablas son zonas multivaloricas, o sea que podrán almacenarse simultáneamente más de un valor. Es una gran variable que se encuentra dividida en múltiples elementos. Para identificar dispone de un índice que los identifica. Para identificar a un elemento en concreto F(2). F(2) --- variable F(I).

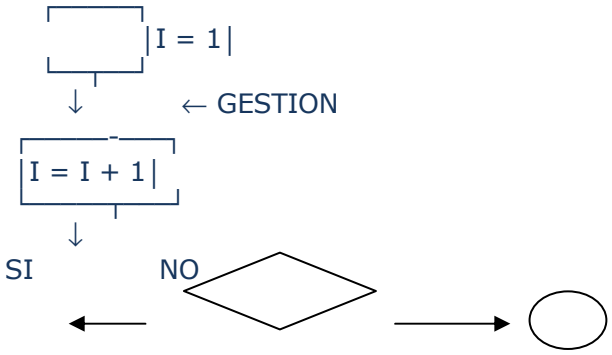
1	1996	01	02
2	1996	01	08
3	1996	01	09
4			
50			
51			

Para definir una tabla inicialmente se utilizan los niveles de 01 a 49 según campos o variables, ya que dispondrá de una estructura a nivel 01 si procede, irá el nombre genérico 01 tabla, en un segundo nivel se indica el nombre del elemento, aunque puede ser descrito con un pic ... lo normal es incluir la cláusula **02 ELE OCCURS Nº TIMES**. El Time es para indicar cuantos elementos lo componen.

```
01 TABLA.
02 ELE OCCURS 52 TIMES.
      03 A PIC 9(4).
      03 M PIC 99.
      03 D PIC 99.
```

La Gestión de las tablas siempre se encuentran asociadas a un bucle. Este bucle dispondrá de una variable que va a recorrer el número de elementos que componen la tabla, por ejemplo como en este caso de 01 a 51 y dentro del bucle se efectuará la Gestión.

En el Ordinograma se representará del siguiente modo.



Y la codificación será **PERFORM VARYING I FROM 1 BY 1 UNTIL I > 51**.

VARYING I.- Expresa cual será la variable que cambie de valor.
FROM 1.- Establece el valor numérico.
BY 1.- Determina el paso, el incremento.
UNTIL I > 52.- Establece cuando finalizará el bucle.

FICHEROS DIRECTOS

Se caracterizan porque el acceso a sus registros realizan directamente sin pasar por los registros anteriores. Existen dos tipos de ficheros directos.

Los relativos y los secuenciales indexados.

Los ficheros relativos se identifican por el número que es asignado automáticamente a cada uno de los registros, en cambio los indexados, los registros son identificados por un campo del mismo que se denominará campo clave. Los relativos para datos numéricos y los indexados para el resto de los datos.

INDEXADOS

Descripción de un fichero indexado.

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
      SELECT FICH ASSIGN TO RANDOM "------"
      ORGANIZATION INDEXED
```

ACCESS MODE | RANDOM
| SEQUENTIAL
| DYNAMIC
RECORD KEY Campo
FILE STATUS VARIEBLE.

La cláusula select es similar salvo que lleve algunas otras especificaciones.
ORGANIZATION INDEXAD.- Ese fichero especificará que es un fichero secuencial indexado.
ACCESS MODE.- Establece el tipo de acceso a los registros, las posibilidades serán: RANDOM.- Indica un acceso directo.
SEQUENTIAL.- Establece un acceso como un fichero secuencial, se emplea para generar listados. DYNAMIC.- Permite tanto un acceso directo como un acceso secuencial, una mezcla entre los dos anteriores.
RECORD KEY.- Esta cláusula se emplea para especificar el campo clave, este campo figurará en la descripción del registro.
FILE STATUS.- Desempeña la función de asignarle una variable de control de acceso al fichero. Esta variable que se definirá en la working con dos dígitos alfanuméricos. Podremos emplearla en la Procedure para averiguar si ha ocurrido algún error en la lectura o escritura del fichero.
La descripción del fichero en la FILE SECTION no varía respecto a los ficheros secuenciales, en cuanto a las ordenes que permitirán el trabajo con los ficheros secuenciales indexados se a de decir que el cambio es debido al diferente acceso.
Siendo las siguientes:

1ª) **Apertura de un Fichero**

OPEN modo nombre-fichero. Los modos de apertura varían respecto a los ficheros secuenciales siendo los siguientes:
- OUTPUT.- Crea el fichero y permite gravar nuevos registros en el.
- INPUT.- Permite leer registros ya sea en acceso secuencial o directo.
- I-O.- Permite leer, gravar, regravar y borrar registros.

Cerrar el fichero

La instrucción es **CLOSE nombre-fichero**.

2ª) **Escritura de un Fichero**

Almacena el registro según la clave indicada, la clave será el valor depositado en el campo clave.
La orden es:

WRITE nombre-fichero
INVALID KEY ordenes
NOT INVALID KEY ordenes
END-WRITE

Antes de dar la orden para almacenar el registro en los campos han de ser únicos en cuanto a la clave. En caso de que esa clave ya existiese no se producirá el almacenamiento y se ejecutarían las ordenes que se especifiquen en la cláusula Invalid Key. Por el contrario si la clave no existe esta será almacenado y se ejecutarán las ordenes que existan en la cláusula Not Invalid Key.
Estas dos cláusulas son opcionales y solo las emplearé cuando tenga que comprobar la existencia del registro en el proceso de escritura.

3ª) **Lectura**

Existen dos modos de lectura en un fichero secuencial indexado. Esta podrá ser directa, de un registro concreto o secuencial, de todos los registros uno a uno desde el principio hasta el final.

Lectura Directa.- Se emplea para localizar un registro concreto y antes de la lectura hemos de situar el valor de la clave que queremos leer empleando posteriormente la orden:

READ nombre-fichero
INVALID KEY ordenes
NOT INVALID KEY ordenes
END-READ

Si esta clave no existiese se ejecutarán automáticamente las ordenes de la cláusula Invalid Key por lo contrario se ejecutarán las ordenes del Not Invalid Key.

Lectura Secuencial.- La lectura sequencial de un fichero secuencial indexado es similar a los ficheros secuenciales siempre y cuando el modo de acceso sea sequential. En cambio si el acceso establecido es dynamic, tendremos que especificar una cláusula NEXT RECORD antes de la cláusula AT END.

4ª) **Reescritura**

Este proceso se empleará para modificar el contenido de un registro que había sido almacenado anteriormente. Por lo tanto solo se podrán regravar los registros que ya haya sido gravado, el formato es idéntico al Write pero la orden se llamará **RE WRITE**.

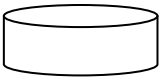
5ª) **Borrar un Registro**

Un registro borrado físicamente con esta orden, es irrecuperable y se han de especificar la clave exacta para el proceso borrado. La orden es:

DELETE nombre-fichero RECORD
INVALID KEY ordenes
NOT INVALID KEY ordenes
END-DELETE

→→→ EXISTE, si o no.

I.K→→ Invalid Key.



→→→→→ Si hay final de fichero RANDOM en el Access Mode sino SEQUENTIAL.

ejerc.- Realizar un programa que permita dar de alta datos en un fichero inexistente que se llamará librow.dat y que constará de los siguientes campos:
ISBN, TITULO, AUTOR, EDITORIAL, AÑO DE PUBLICACION, EDICION, TEMA, PRECIO, STOCK Y STOCK MINIMO. El Campo Clave Será "ISBN"

SUBRUTINAS

Las subrutinas consisten en enlazar diferentes programas a través de un programa menú. Una subrutina se caracteriza que en vez de finalizar por un stop run finaliza por EXIT PROGRAM.
Para llamar a la subrutina y que esta se ejecute utilizaremos la instrucción CALL U:nombre. Cuando finaliza la subrutina EXIT PROGRAM retorna a la instrucción siguiente de la llamada pero conservando la PSP de la subrutina. La PSP es una zona de memoria que

contiene el estado actual de la ejecución así como el valor final de las variables, si volviésemos a ejecutar la subrutina estos valores permanecerían, para evitar esto podremos cancelar la PSP con la instrucción CANCEL U:nombre.

FORMATOS DE PANTALLA

Son estructuras que se definen en la SCREEN SECTION que pertenece a la DATA DIVISION. Estas estructuras me permitirán agilizar en la Procedure el uso de la pantalla y el teclado permitiéndome englobar presentaciones y peticiones de datos así como aumentar la potencia de estos dos aspectos. La estructura de definición de formato de pantalla es del siguiente modo: A nivel 01 indicaremos el nombre genérico de formato de la pantalla y en subniveles inferiores cada una de las partes según la siguiente estructura:

```
          | VALUE "texto"              |
   nn nombre |         | FROM  | CAMPO    ||
          | PIC Item | TO    |          ||
          |          | USING | VARIABLE ||
```

nn →→ Representa el número de nivel.

nombre →→ Se utiliza para identificar cada uno de los apartados del formato de pantalla no podría utilizarse el nombre común Filler, y si no queremos identificar esta parte en concreto podremos omitir dicho nombre.

LINE →→ Se emplea para expresar el número de línea donde vamos a situar el cursor.

COLUMN →→ Expresa el número de columna, realiza la misma función que la de Position pero esta en la Screen.

VALUE →→ Se empleará cuando lo que queremos realizar es presentar en pantalla un texto.

PIC →→ Cláusula que utilizaremos para trabajar con campos o variables.

Item →→ La descripción de un campo.

FROM/TO/USING →→ Determina que tarea voy a realizar con ese campo o variable.

From.- Lo emplearé cuando vaya a visualizar.
TO.- Se usará cuando vayamos a pedir un valor para el campo.
USING.- Cuando vayamos a realizar ambas cosas.

En la Procedure los formatos de pantalla se visualizarán a través de su nombre (Display, Accept).

```
IDENTIFICATION DIVISION.
PROGRAM-ID. NUEVOS-SOCIOS.
AUTHOR. JAVI.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
```

```
SPECIAL-NAMES.
    DECIMAL-POINT IS COMMA.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT SOC ASSIGN TO RANDOM "C:\PELICULA.DAT"
    ORGANIZATION INDEXED
    ACCESS MODE RANDOM
    RECORD KEY NºSOCIO
    FILE STATUS FSS.
DATA DIVISION.
FILE SECTION.
FD SOC
    LABEL RECORD STANDARD
    DATA RECORD SOCIOS.
01 SOCIOS.
    02 NºSOCIO PIC X(5).
    02 NOMBRE.
       03 APEL1 PIC X(15).
       03 APEL2 PIC X(15).
       03 NOMBRE PIC X(20).
    02 NIF.
       03 DNI PIC 9(8).
       03 LETRA PIC A.
    02 DIRECCION PIC X(25).
    02 CP PIC X(5).
    02 LOCALIDAD PIC X(10).
    02 TELEFONO PIC X(6).
    02 PELICULA ALQ OCCURS 10 TIMES.
       03 NP PIC 9(4).
       03 FALQ.
          04 A PIC 9999.
          04 M PIC 99.
          04 D PIC 99.
WORKING-STORAGE SECTION.
01 DATOS PIC X(24) VALUE "TRWAGMYFPDXBNIZSQVHLCKE".
01 TABLA.
    02 ELE OCCURS 24 TIMES REDEFINES DATOS.
       03 LET PIC A.
77 N PIC 9(8) VALUE 0.
77 DN PIC 9(8).
77 NL PIC 99.
77 MAS PIC A VALUE SPACE.
77 FSS PIC XX VALUE SPACE.
SCREEN SECTION.
01 PAN.
    02 BLANK SCREEN
    02 LINE 1 COLUMN 35 VALUE "NUEVOS SOCIOS"
    02 LINE 3 COLUMN 10 VALUE "NOMBRE Y APELLIDOS"
    02 LINE 3 COLUMN 29 PIC X(10) TO NOM
    02 LINE 3 COLUMN 40 PIC X(10) TO APE1
    02 LINE 3 COLUMN 50 PIC X(10) TO APE2
    02 LINE 5 COLUMN 10 VALUE "DIRECCION"
    02 LINE 5 COLUMN 22 PIC X(25) TO DIRECCION
    02 LINE 7 COLUMN 10 VALUE "CODIGO POSTAL"
    02 LINE 7 COLUMN 25 PIC X(5) TO CP
```

```
           02 LINE 7 COLUMN 35 VALUE "TELEFONO"
           02 LINE 7 COLUMN 47 PIC X(6) TO TELEFONO
           02 LINE 11 COLUMN 10 VALUE "NIF"
           02 LINE 11 COLUMN 15 PIC X(8) TO DNI
           02 LINE 11 COLUMN 24 PIC A FROM LETRA

       01 VINAGRE.
           02 BLANK SCREEN
           02 LINE 5 COLUMN 10 VALUE "PEDIR Nº DE SOCIO"
           02 LINE 5 COLUMN 29 PIC 9(5) TO SOCIO

       01 TOMATE.
           02 LINE 7 COLUMN 10 VALUE "DESEA MAS"
           02 LINE 7 COLUMN 21 PIC A TO MAS

       01 CEBOLLA.
           02 LINE 12 COLUMN 10 VALUE "NO SE PUEDE"
       PROCEDURE DIVISION.
       INICIO.
           OPEN I-O SOC
           PERFORM UNTIL = "N"
           DISPLAY VINAGRE
           ACCEPT VINAGRE
       LECTURA.
           READ SOC
           INVALID KEY
           DISPLAY PAN
           ACCEPT PAN
           PERFORM LETRA
           WRITE SOCIOS END-WRITE
           DISPLAY TOMATE
           ACCEPT TOMATE
           NOT INVALID KEY
           DISPLAY CEBOLLA
           DISPLAY TOMATE
           ACCEPT TOMATE
           END-IF
           END-READ
           END-PERFORM
           CLOSE SOC
           EXIT PROGRAM.
       LETRA.
           COMPUTE N = DNI / 23
           COMPUTE DN = N * 23
           COMPUTE N2 = DNI - DN
           COMPUTE LET(NL).

WRITE PEPE FROM LINEA 5 AFTER2 END-WRITE
                ↑ Línea en blanco entre la 4 y la 5.
```

INFORMATICA PROGRAMACION

PASCAL

El Pascal es un lenguaje compilado de orientación general seudocientifica, se trata de un lenguaje reciente de finales de la década de los 70 que surgió como alternativa al Basic y al Fortran.

Al igual que el Cobol se trata de un lenguaje estructurado pero en este caso mucho mas. Para facilitar el trabajo con este lenguaje y mejorar su potencia se genero lo que se conoce como TURBOPASCAL. El Turbopascal es un compilador mucho más potente y rápido, que detecta automáticamente errores y genera un entorno de trabajo.

FORMATO DE UN PROGRAMA EN PASCAL

El lenguaje Pascal está concebido para generar una programación en bloques. Estos bloques se les llaman funciones, procedimientos o bloques.

Un programa requiere inicialmente que se defina y se declare todo lo que va a ser usado en el programa. Comenzará por una cláusula "PROGRAM", a continuación irán las declaraciones y definiciones y después el bloque principal de programa que comenzará por un "BEGIN", y finalizará con un "END". Las declaraciones deben de seguir un orden estricto y este es:

1º) Declaraciones de etiquetas.
2º) Definición de constantes.
3º) Definición de tipo.
4º) Declaración de variable.
5º) Declaración de procedimiento.
6º) Declaración de funciones.

Para formar un programa tan solo es obligatorio incluir la cabecera del programa y el cuerpo, el resto opcional.

SIMBOLOS DEL LENGUAJE

El Pascal podrá utilizar cualquier carácter existente en el primer bloque en el código ASCII aunque muchos de estos signos tienen significados previamente asignados por lo que su uso será reservado: + - * / . ; : = >< () [] {}

Además ciertas combinaciones de estos signos tienen significados especiales: (*Comentario*) ← Delimita comentarios.

Junto con estos símbolos existen palabras reservadas que tienen su significado.

Estas son:
AND, ABRAY, BEGIN, CASE, CONST, DIV, DO, DOWNTO, ELSE, END, FILE, FOR, FORWARD, FUNCTION, GOTO, IF, IN, LABEL, MOD, NIL, NOT, OF, OR, PACKED, PROCEDURE, PROGRAM, RECORD, REPEAT, SET, THEN, TO, TYPE, UNTIL, VAR, WHILE, WITH.

TIPOS DE DATOS

El Pascal gestiona cuatro tipos fundamentales de datos llamados tipos escalares. Estos son Enteros, Reales, Caracteres y Booleanos. Partiendo de estos tipos básicos el usuario podrá definir otros.

La identificación para los tipos es la siguiente:

Para Enteros → Integer
Para Reales → Real
Para Caracteres → Char
Para Boleanos → Boolean

DATOS DE TIPO ENTERO

Se considera datos enteros los números positivos y negativos sin decimales las operadores que se usarán con los datos enteros son: Signo + suma, - resta, * multiplica, DIV división. Truncando el resultado por su parte entera MOD, modulo de la división (el resto de la división).

OPERADORES DE RELACION

Son mayor o igual, menor o igual, igual, menor, mayor.

DATOS DE TIPO REAL

Son los números reales con o sin decimales. Los operadores son: + - * /

DATOS DE TIPO CARACTER

Almacenan un carácter, los caracteres irán siempre entre apóstrofes.

DATOS DE TIPO BOOLEANO

Pueden tener solo 2 valores. Verdadero o Falso. True o False. Los operadores para datos booleanos son: AND ← Y Lógico OR ← O,Lógico NOT ← Negación lógica.

Además de estos tipos de datos escalares o básicos el Turbopascal nos ofrece otros tipos de datos.
El más importante en cuanto a su uso es el tipo STRING que sirve para definir cadenas de caracteres de hasta 255 de tamaño. Las cadenas de caracteres se delimitan por comillas dobles[4].

DECLARACION DE VARIABLES

El apartado o bloque de definición de variables comenzará por el indicador VAR y el formato es: nombre,...:tipo;

```
                              .       .
EJPLO.    VAR                 .       .
          I, I : INTEGER
          OPL : CHAR;
          NOMBRE : STRING;
```

DEFINICION DE CONSTANTES

Una constante será un nombre que no variará de valor durante toda la ejecución. El bloque es identificado por CONST y la definición nombre = valor;

```
EJPLO.    CONST
            PI = 3.14;
```

[4]LOS OTROS TIPOS SE VEN EN EL LIBRO

INFORMATICA PROGRAMACION

EXPRESIONES ARITMETICAS

Algo fundamental es que los tipos de datos y la expresión que se realiza con ellos sean compatibles.

Para esta norma existe una excepción que consiste que a los números enteros puede usarse expresiones reales.

El compilador lo que hace es convertir el Entero en Real.

```
N + 1
  ↑ 01
```

El formato para una expresión es: Variable := expresión

```
p    := N + 1;
```

ENTRADA Y SALIDA ESTANDARD

Se refiere al acceso a la consola. La consola hace referencia a la pantalla y teclado. Normalmente todos los programas realizan accesos a la misma y esto habrá que indicarlo en la cabecera de programa

```
PROGRAM  nombre-int(acceso,...);      |INPUT  | Entrada y
         INPUT,OUTPUT;      |OUTPUT | Salida
```

* INSTRUCCIONES:

READ, READLN → Realizan una parada en la ejecución del programa para solicitar el valor o valores al teclado. El

formato de la instrucción es: |READ |
|READLN | (variable...)

La diferencia entre ambas variantes que read permite una siguiente entrada en la misma línea mientras que readln realiza un cambio de línea tras la lectura.

WRITE y WRITELN → Presenta la pantalla lo especificado.

El formato |WRITE | (| variable |,...)
|WRITELN | (| expresión |,...)
(| tanto |,...)

FORMATO DE SALIDA (DENTRO DEL WRITE)

Sobre todo cuando se visualiza variables puede interesar estipular el tamaño de representación y en el caso de los números reales el número de decimales.

Para limitar el tamaño, a continuación de la variable, indicaremos dos puntos y tamaño.

WRITE (PCI:8:2); → Dos decimales.

ESTRUCTURAS DE CONTROL

Las estructuras de control son las instrucciones que rompe la ejecución secuencial de las instrucciones. Las conforman los bucles y las condicionales.

*Condición, La condición es una instrucción IF y permite tomar decisiones a lo largo del programa el formato será:

IF expresión lógica THEN instrucción/es (si) ELSE instrucción/es (no).

Cuando en el apartado SI o en el NO existan varias instrucciones o sea un bloque de instrucciones este debe comenzar por Begin y finalizar por End. Más de una instrucción.

Para calcular el cuadrado de un número existe la función SQR y entre paréntesis irá el número. SQRT → Para la raíz cuadrada.

INFORMATICA PROGRAMACION

PROGRAMA RAIZ DE 2º GRADO.

```
PROGRAM RAIZ(INPUT, OUTPUT);
VAR
  A, B, C: REAL;
  R, X1, X2: REAL;
BEGIN
  WRITE ("PRIMER PARAMETRO");   ← COMO UN DISPLAY
  READLN (A);    ← COMO UN ACCEPT
  WRITE ("SEGUNDO PARAMETRO");
  READLN (B);
  WRITE ("TERCER PARAMETRO");
  READLN (C);
  R := SQR(B)-4*A*C
  IF R<0 THEN
     WRITE ("NOTIENE SOLUCION")
     ELSE
     BEGIN
       IF A=0 THEN
          WRITE ("NO TIENE SOLUCION")
          ELSE
          BEGIN
            X1 := (-B+SQRT(R))/(2*A);
            X1 := (-B-SQRT(R))/(2*A);
            WRITE ("X1=",X1:5:3);
            WRITE ("X2=",X2:5:3);
          END;
     END;
END.
```

TALLER MECANICO

```
PROGRAM TALLER(INPUT,OUTPUT);
CONST
        IVA:=0.16;
VAR
        MATRIC:STRING[11];
        CATEG,HORAS,MATER,PIVA:INTEGER;     Integer=Entero
        IT,IB:REAL;
        BEGIN
                WRITE('MATRICULA');
                READLN(MATRIC);
                WRITE('CATEGORIA');
                READLN(CATEG);
                WRITE('MATERIAL');
                READLN(MATER);
                WRITE('HORAS');
                READLN(HORAS);
                IF CATEG=1 THEN
                        P:=3600
                ELSE
                BEGIN
```

```
            IF CATEG=2 THEN
            P:=2500
            ELSE
            P:=1500;
            IB:=(HORAS*P)+MATER;
            PIVA:=IVA*IB;
            IT:=IB+PIVA;
            WRITE('IMPORTE BRUTO',IB:10);
            WRITE('IVA= ',PIVA:6);
            WRITE('IMPORTE TOTAL',IT:12);
      END;
END.
```

{:=} → ASIGNACION
{=} → COMPARACION
Para entrar en el programa CD tp y después turbo [enter].

Cuando se utiliza "STRING" es necesario indicar entre corchetes el tamaño de la cadena.

*F2→Gravar *ALT+F9→Compilar *CTRL+F9→Ejecutarlo *ALT+F5→Ver ventana de ejecución

Dentro del Write comillas simples.
85: ";" Expecteed → Falta el punto y coma.
3: Unknown Identifier → Una variable no definida.
Si queremos trabajar con números grandes aunque no tengan decimales debemos utilizar números REALES.

FUNCIONES EXTERNAS

El Turbopascal dispone de series de funciones que se encuentran almacenadas en librerías externas. Para cargar la librería en el programa y hacer uso de estas funciones se emplea la directiva USES librería,...;
Una de las librerías más típicas es la CRT que contiene funciones para gestión de consola.
Función CLRSCR esta función borra la pantalla.
Función READKEY lee un carácter de un teclado se suele utilizar para parar pantalla.
GOTOXY(X,Y) → Sitúa el cursor en una posición de la pantalla.
La esquina superior izquierda es la 1,1. Como X e Y se pueden utilizar números o variables, estas variables han de ser Enteras.

BUCLES

Un bucle es una instrucción que permite repetir una serie de instrucciones. Vamos a tener diferentes instrucciones dependiendo del tipo de bucle.
BUCLES Condicionales o Inanimados → se caracterizan porque la repetición depende de una condición sería algo similar al perform until. Existe dos, WHILE REPEAT, La diferencia entre ambos está en que el while comprueba la condición el comienzo del bucle y repeat al final.

Según esto observamos que el bucle repeat obliga a que las instrucciones que contienen se ejecuten cuando menos una vez aunque inicialmente la condición no sea verdadera.
El while repite mientras se cumpla la condición y el repeat hasta que se cumpla.

*FORMATO DE LA INSTRUCCION.

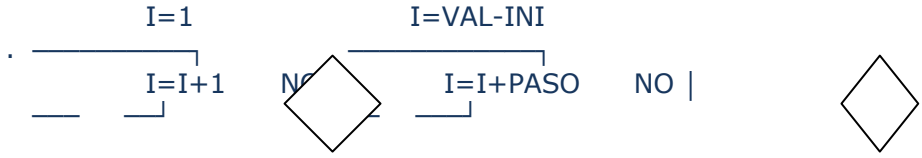
```
| REPEAT          |
| INSTRUCCION;    |
|     .           |
| UNTIL CONDICION |
```

Inicialmente un bloque de instrucciones debe ir rodeado de Begin, End pero esto se podrá omitir cuando la instrucción que va a englobar este conjunto disponga de comienzo y finalización.

*FORMATO DE LA INSTRUCCION WHILE.

```
WHILE CONDICION DO
        INSTRUCCION;
```

BUCLES animados. Permiten repetir un conjunto de instrucciones un número determinado de veces y variando automáticamente el valor de una variable es similar al perform varying. Su expresión metodológica:



Este formato de esta instrucción:

```
FOR variable:=val-ini  TO/DOWNTO   val-fin  DO
                                          INSTRUCCION;
```

INSTRUCCION CASE

Esta instrucción permite una elección múltiple dependiendo del valor de una variable. El formato es:

```
CASE  variable  OF
        Valor1:Instrucción;
        .
        Valorn:Instrucción;
                ELSE
                Instrucción;
                END;
```

El else se ejecutará cuando ninguno de los valores expresados sea igual.
TRUNC() ← Para truncar un número por la coma decimal.

```
A:=TRUNC(F-(M*100));
```

PROCEDIMIENTOS Y FUNCIONES

INFORMATICA PROGRAMACION

Los procedimientos y funciones son apartados del programa que realizan procesos auxiliares al mismo (como si fuese una cabecera).

PROCEDIMIENTOS

Se utiliza para identificar un bloque de instrucciones que formará un subprograma dentro del programa. Su estructura es la misma que la de un PROGRAM salvo que en vez de comenzar por la directiva **program** empieza por la directiva PROCEDURE.
Para invocar a este procedimiento desde el programa principal simplemente expresaremos el nombre del mismo.

FUNCIONES

Es un nombre que se adjudica a un conjunto de instrucciones que realizan una tarea determinada. Cuando se ejecuta una función el resultado de la misma es asignado al nombre de la función.
La diferencia es que Procedimiento no contiene valor y las Funciones sí.

```
procedimiento          función
      ↓                   ↓
   Write              a:=trunc(18/3)
```

Dado que una función va a contener un valor cuando se define habrá que adjudicarle un tipo de dato.

*El formato para definir una función es:

```
FUNCTION nombre(parámetros):tipo;
BEGIN
INSTRUCCIONES;
END;
```

EJEM.

```
FUNCTION MEDIA(N1,N2,N3,N4:REAL):REAL;
BEGIN
MEDIA:=(N1+N2+N3+N4)/4;
END;
```

MATRICES [Tablas]

Una matriz es una variable multidimensional, se define como una variable pero incluyendo la siguiente cláusula.

```
VAR
   nombre:ARRAY [Valor inicial..Valor final] OF TIPO;
      T1:ARRAY [1..10] OF INTEGER;
```

Una de las principales peculiaridades de los ARRAY o matrices en Pascal es que los índices son totalmente libres y no tienen porque partir siempre del valor (1). Se emplearán para almacenar y gestionar múltiples valores.
Una variable multidimensional siempre está asociada a un Bucle.

```
I=1
↓   <──────────┐
ACEPTAR T1[I]  |
↓              |
I=I+1          |
```

```
↓           |
NO (SI I<10) SI <──┘ Sería un FOR.
```

TYPE → Sirve para definir un tipo de variable.

```
TYPE
        PP=ARRAY[1..10] OF REAL;
VAR
        T1:PP;
```

REGISTROS

Un registro es una estructura compuesta por diferentes campos elementales que permiten definir variables con dicha estructura. Además de poder ser usado en ficheros también se emplea para matrices(tablas). Se definirán dentro del bloque TYPE con la siguiente estructura:

```
Type
Nombre =
Record
Nombre:Tipo
        ;
        ;
        ;
        End;
Var
Identificador:nombre;
```

Ejemplo:

```
TYPE
        ALUMNOS=
                RECORD
                NOM: STRING[30];
                ASING: ARRAY [1..5] OF STRING [10];
                NOTA: ARRAY [1..5] OF REAL;
                NM: REAL;
                END;
        VAR
        PEM: ARRAY [1..16] OF ALUMNOS.
```

Para referenciar cualquier dato en concreto, por ejemplo el nombre, habrá que indicar en nombre de la variable y el nombre de ese dato, separándolas por un punto.

```
PEM[I].NOM
PEM[I].NOTA[P]
PEM[I].FALTAS[J][P]
```

GRÁFICOS

Es un modo de trabajo en pantalla que sustituye al modo texto con lo que todas las instrucciones que hasta ahora empleamos ya no son operativas. Otra de las ventajas que nos ofrecen es el trabajo con paletas de colores. Las instrucciones para activar ese color son:

Textcolor(color[+Blind].- Establece el color de los caracteres. +Blind es opcional:

determina su intermitencia.
Textbackground(color).- Establece el color de fondo.

Los colores se seleccionarán a través de su nombre o su número:

CONSTANTE	**VALOR**
BLACK(NEGRO)	1
BLUE(AZUL)	2
GREEN(VERDE)	3
CYAN(CIANO)	4
RED(ROJO)	
MAGENTA(VIOLETA)	5
BROWN(MARRON)	6
LIGHT GRAY(GRIS CLARO)	7
DARK GRAY(GRIS OSCURO)	8
LIGHT BLUE(AZUL CLARO)	9
LIGHT GREEN(VERDE CLARO)	10
LIGHT CYAN(CIANO CLARO)	11
LIGHT RED(ROJO CLARO)	12
LIGHT MAGENTA(VIOLETA CLARO)	13
YELLOW(AMARILLO)	14
WHITE(BLANCO)	15

Para trabajar será necesario activar el monográfico, esto se realiza con la instrucción **INITGRAPH**. Tanto esta instrucción como las demás se encuentra en una librería llamada graph. Será necesario cargarla con antelación.
La instrucción Initgraph requiere los siguientes parámetros:
Tipo de tarjeta gráfica que tenemos **[INITGRAPH(tarjeta,modo,director)]**
Modo gráfico que vamos a utilizar ...
Directorio donde se encuentran las definiciones de tarjeta ...

La tarjeta será un valor comprendido entre 0-10. Según la siguiente tabla:

VALOR	**CONSTANTE**
0	DETECT
1	CGA
2	MCGA
3	EGA
4	EGA 64
5	EGA MONO
6	IBM 8514
7	HERC MONO
8	ATT 400
9	VGA
10	PC 3270

El modo 0 indica que el propio pascal detecte automáticamente la tarjeta instalada.
Los modos determinan la resolución de la tarjeta, las resoluciones para la tarjeta VGA son:

VGA Lo ---- 0 ---- 640x200

VGA Med --- 1 ---- 640x350
VGA H ----- 2 ---- 640x480

El directorio donde se encuentran las utilidades gráficas según la instalación que tenemos actualmente del turbo pascal es: C:\TP\BGI → VGA.BGI ← Se cambia para la mejora de pantalla.

Una vez finalizada el trabajo monográfico será necesario cerrar este con la instrucción **CLOSEGRAPH**.

INSTRUCCIONES DE GESTIÓN GRÁFICA

Visualización de texto.- La instrucción para presentar un texto en la pantalla gráfica son:

OUTTEXT(`TEXTO´)
OUTTEXTXY(X,Y,`TEXTO´) ← Para ordenar el read.

Estilo de Letra

Existe variaciones en la letra que determina el estilo de la misma, para seleccionar esto emplearemos la orden **SETTEXTSTYLE**
Los parámetros de esta orden son:
1.- Fuente: es el tipo de la letra y dependerá de la siguiente tabla.

0	Fuente por Defecto
1	Fuente Triplex
2	Fuentes pequeñas (small)
3	Sans Serif
4	Gothic

2.- Dirección: existe 2 posibles direcciones. 0 ← Horizontal
1 ← Vertical
3.- Tamaño:

settextstyle(fuente,orientación,tamaño)

IMÁGENES GRÁFICAS.- Para determinar el color de las imágenes gráficas se emplea **SETCOLOR(**el color según la tabla anterior**)**

Puntos.- La instrucción es **PUTPIXEL(X,Y,COLOR)** y como parámetros se especifica coordenada x e y, y el color.

Línea.- Traza una línea recta entre 2 puntos determinados con el color establecido por setcolor. Los parámetros serán las coordenadas de ambos puntos. Ocasionalmente nos puede interesar comenzar la línea en el punto de finalización de la línea anterior, entonces emplearé en vez de LINE la instrucción **LINETO** especificando como parámetro la coordenada destino. Comenzará donde terminó la última línea.

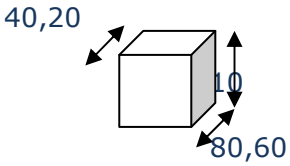
Rectángulo.- Traza un polígono no cuadrangular, estableciendo las coordenadas de los puntos de dos de sus esquinas opuestas.

RECTANGLE(0,0,640,380) (0,0)

(640,380)

Barras.- Una barra es un paralelopípedo relleno. Las especificaciones son las mismas que para un rectángulo pero la instrucción es **BAR**.

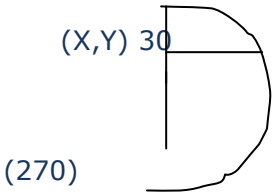
Existe una variante en 3 dimensiones que incluye dos parámetros más el 1º, el fondo que será la profundidad y el 2º, el tope que determina la orientación de la barra. El fondo a de ser boolean **BAR3D(40,20,80,60,10,TRUE)**



Circunferencia.- Para trazar la circunferencia será necesario el radio y el centro como parámetros. **CIRCLE(1º punto,radio)**

Arco.- Es un segmento de una circunferencia, se realiza con la instrucción **ARC** debiendo de indicar los parámetros.
1.- Coordenadas del centro de la circunferencia.
2.- Angulo inicial del arco.
3.- Angulo final del arco.
4.- Radio.

Para trazar una semicircunferencia por el lado superior sería: ARC(X,Y,270,90,30) (90)



Tartas.- **PIESLICE(X,Y,ANGULO INI,ANGULO FIN,RADIO)**
CENTRO

SETFILLSTYLE → CAMBIO DE FORMA DE SECTORES.(EL FONDO)

FLOODFILL(X,Y,LIMITE)→ FONDO DENTRO DE LÍNEAS Y EL LÍMITE ES EL COLOR DE LA LÍNEA POR ESO ANTES SE USA EL **SETCOLOR(COLOR DE LÍNEA)**.

FICHEROS

Un conjunto de datos almacenados en disco el Pascal gestiona ficheros de tipo secuencial de tipo texto o de tipo binario, lo normal será definir un registro en el bloque type. Definiendo el fichero en el bloque VAR del siguiente modo.
FICHERO: FILE OF REGISTRO;

EJEMPLO.

```
TYPE
        REG = RECORD     ← ASSIGN(AGENDA,`A:AGENDA.DAT´);
        NOM: STRING[20];
        TELF: STRING[12];
        END;
VAR
        AGENDA: FILE OF REG;
```

La forma de asignarle un nombre externo al fichero será a través de la instrucción

ASSIGN. El formato de esta instrucción ASSIGN(VARIABLE.FICHERO,`U:NOMBRE.EXT´);

APERTURA DE FICHERO

Existe dos formas de abrir un fichero según vayamos a leer datos o a gravarlos, estas son **RESET** para leer datos, y **REWRITE** para gravar datos.

En ambas instrucciones como parámetro se indicará la variable de tipo fichero RESET(AGENDA)

REWRITE(AGENDA)

Tanto la escritura como la lectura en un fichero de datos se realiza de forma idéntica que a la consola con la única diferencia que se expresará como primer parámetro la variable de fichero. (para escribir) → **WRITE(VARIABLE.FICHERO,...);**

La definición del fichero será el registro definido anteriormente. **REG=RECORD**

La lectura será con la instrucción **READ** operando igual que con la escritura.

Para cerrar el fichero emplearemos la instrucción **CLOSE** indicando como parámetro la variable del fichero.

Al ser un fichero secuencial la lectura será secuencial y nos es imprescindible controlar el final de fichero. Esto lo haremos con una función llamada **EOF**. Cuyo valor será cierto(TRUE). Si alcanza el último registro del fichero cuya variable se especifique como parámetro esta función normalmente se encuentra en una condición de un Bucle lógico. (WHILE, REPEAT) Nos serán válidos.

FICHERO DE TEXTO

Es un fichero de datos que almacena la información en un formato ASCII puro. Su definición será en el bloque de la cláusula VAR pero indicando que sea de tipo TEXT.

```
VAR
Fichero: TEXT;
```

La gestión es idéntica a los anteriores pero:

1.- Existe un nuevo modo de apertura que me permite añadir al final de fichero nuevos datos. La instrucción es APPEND.

2.- El registro normalmente lo conforma la línea y para leer y escribir emplearemos **READLN** y **WRITELN**.

TURBO C++

El camino donde se encuentra el entorno es c:\tc\bin.

Las directivas #include sirven para cargar las librerías.

NOMBRES DE IDENTIFICADORES

Son los nombres usados por el usuario y deben cumplir las siguientes características.

1º.- Han de comenzar por una letra o un guión de subrayado.

2º.- Estarán compuestas por letras, guiones de subrayado o números.

3º.- El tamaño es ilimitado aunque TC++ toma como significativo, los 32 primeros caracteres.

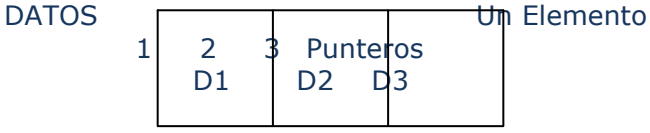
4º.- Diferencia minúsculas y mayúsculas.

PUNTEROS

Los punteros son variables que almacenan direcciones de memoria. Los punteros se emplean para poder almacenar múltiples datos al igual que las tablas pero con dos ventajas fundamentales.

1º.- No dispone de un número concreto y delimitado de elementos.

2º.- Dispone de una estructura dinámica o sea no cerrada.
Los elementos dinámica disponen de dos subelementos pero uno imprescindible.



Un puntero que me sirve para enlazar un elemento con el siguiente.
Para definir una variable de puntero lo haremos de la misma forma que cualquier variable pero delante del tipo, indicaremos un circunflejo(^).

Ej. VAR
Puncar:^String[20];

Aunque lo normal es utilizar punteros en estructuras de tipo registro. Incluyendo en esta estructura un puntero a la propia estructura para indicar la dirección del registro siguiente.
Ej.- Supongamos que quiero realizar una agenda telefónica a través de listas dinámicas para tenerla perfectamente ordenada para incluir en cualquier momento elementos. Necesitaremos una estructura del siguiente tipo.

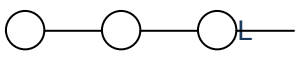
```
TYPE
    AGE=RECORD
        NOM:STRING[20];
        TLF:STRING[12];
    PUNSIGUIENTE:^AGE; ← Dirección de cada elemento
    END;
```

Para manipular una lista, como mínimo, necesito dos variables de puntero. Una que almacenará el comienzo de la lista.

VAR
PUNCAB(PUNTERO CABECERA),PUNTRA(PUNTERO DE TRABAJO):^AGE (DIRECCIÓN DE CADA ELEMENTO).

Y necesito un puntero de trabajo para moverme por la tabla.
El peligro al trabajar con punteros es perder el control de la dirección de los discos. Los punteros hace falta inicializarlos (Un valor neutro), para ello un puntero lo igualaré a la constante NIL.

```
BEGIN
    PUNCAB:=NIL;
```

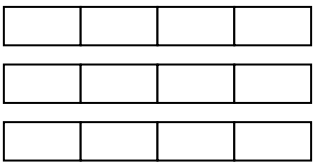


Para poder almacenar un valor en un elemento tendré que reservar memoria para este elemento, esto lo haremos con la instrucción **NEW** indicando como parámetro un puntero de dicha estructura.

```
Ej.-    BEGIN
        PUNCAB:=NIL;
        NEW(PUNTRA);
```

Cuando queremos hacer referencia a un dato del elemento que apunta el puntero que contiene la dirección de ese elemento lo haremos indicando el nombre del puntero seguido de un circunflejo punto y el nombre del dato.

- Para pedir el nombre del dato: READLN(PUNTRA^.NOM);
  Si queremos hacer desaparecer un elemento de la lista he de liberar la memoria que ocupaba ese elemento. Se hace con DISPOSE(PUNTERO); El perder las direcciones de los discos es el error más grave.



Para visualizar **PRINTF**.
Gets(variable de cadena) → En vez de un scanf para una cadena de caracteres. Con espacios sobre todo.

Un puntero va a ser una dirección de memoria. El nombre precedido de *.
Ejemplo.- Un puntero para una zona de datos de tipo entero se definirá → int *pun;
Para reservar espacio en la memoria malloc(tamaño);
Para especificar el tamaño se puede utilizar → sizeof(zona);
Puntra1=(struct ag*) malloc(sizeof(struct ag)); Y lo calcula directamente. Una vez seleccionada la memoria accederemos por medio de un puntero. Por ello habrá dentro de la estructura dinámica un puntero a la propia estructura. Para manipular esta lista será necesario un puntero de cabecera que apunta siempre al primer elemento de la lista, y como mínimo un puntero de trabajo que me servirá para ir recorriendo la lista.

MANIPULACIÓN DE PUNTEROS

Para poder almacenar en una estructura a través de punteros es necesario reservar el espacio en la memoria, que se hará con la instrucción malloc(tamaño); esta función devolverá como valor la dirección de el puntero. Para especificar el tamaño e byte de la estructura podremos calcular el número y especificarlo.

Char → 1 byte.
Int → 2 byte.

```
        Char    [40]
Char     [20]
        Int     [10]
```

80 bytes.

Para especificar el tamaño se puede utilizar la función sizeof(zona);. Una vez utilizada la memoria tendremos que acceder a ella por medio de un puntero, hay que recordar que siempre que se quiera crear una estructura dinámica a través de punteros dentro de la estructura habrá un puntero de trabajo que me servirá para ir recorriendo la lista.

LIBERACIÓN DE LA MEMORIA

Para liberar utilizaremos la función free(puntero); indicando como parámetro, el puntero de la zona a liberar. Cuando un puntero no contiene ninguna dirección de memoria toma el valor NULL.

En C++ todo son funciones, por eso viene previamente definidas. El INCLUDE sirve para cargar librerías de funciones. A continuación los bloques. Para englobar un bloque {}. Y así se manipulan.

TIPOS DE DATOS

Los 5 tipos de datos son:

- Carácter ..... Char ..... 8 bit .... 1 bite
- Entero ..... Int ..... 16 bit ... 2 bite
- Punto Flotante .... Float ..... 32 bit ... 4 bite
- Doble Punto Flo ... Double Float . 64 bit ... 8 bite
- Sin Valor ..... Void ..... 0 bit .... 0 bite

Ejemplo.- void[5] main(void)

MODIFICADORES

Por defecto ya llevan **Signed**(diferencia en signo) pero **Unsigned**(que lo tenga en cuenta). **Short**(Indica una representación corta (por defecto)). **Long** (Duplica el nº de bit con lo que se amplia el margen o rango).

INT ————Lo mismo
SIGNED SHORT INT

Para definir variables.

Tipo -------- Lista de variables
. .
. .

La declaración de variables se realiza en tres sitios:

- Al comienzo del programa antes de cualquier función, se llama variable global y podrá usarse en cualquier parte del programa.
- Dentro de una función, son variables locales y solo para usarse dentro de la función.
- En la definición de la función, son variables de parámetro formal y se empleará para trasladar información.

CONSTANTES

Se definen como una variable pero igualando el nombre a un valor → INT X=18;

OPERADORES

+, -, *, /, % Módulo de división, -- decremento, ++ incremento.
Las prioridades → ++ -- * / % + -

[5] Void: No retorna ningún valor. El segundo no lo recibe.

OPERADORES RELACIONALES

< > <= >= ==(Comparación) i=(no igual)

OPERADORES LÓGICOS

&& ← AND, || ← OR, i ← NOT

PRESENTACION EN PANTALLA

PRINTF("CADENA CONTROL(El Texto que se va a visualizar)", LISTA ARGUMENTOS);

%d ← Visualiza un nº entero en formato decimal.
%f ← Visualiza un nº flotante en formato decimal.
%c ← Visualiza un carácter.
%s ← Visualiza una cadena de caracteres.

\n ← Línea en blanco.
\\ ← Representa barra inversa.
\" ← Representa comillas.
\f ← Alinearía la hoja(impresora).
\t ← Realiza una tabulación.

Para pedir la orden de teclado → SCANF("%",&nombre);

SENTENCIAS DE CONTROL

La instrucción condicional es IF, el formato es el siguiente:

```
If (condición)
        Sentencias si;
        Else
        Sentencias no;
        (Else es opcional).
```

Sentencia SWITCH → Es un multicondicional es idéntica a la sentencia KEYS del Pascal. El formato de esta sentencia es:

```
Switch (variable) {
        Case valor1: --- break;
          "    "      "   "
        default
        sentencias; }
```

Default → Es para cuando el CASE no se cumpla y el break para no encadenar la sentencia de todas las sentencias.

BUCLES

- SENTENCIA FOR(INICIACION[6]; CONDICION[7]; INCREMENTO[8])
  Un bucle podrá romperse en cualquier momento con la instrucción BREAK. Va sumando valores.
- SENTENCIA WHILE(CONDICION); Si la primera no es verdad no se ejecutará ni una

[6] Iniciación: Iniciar Variable.
[7] Condición: Condición Variable.
[8] Incremento: Incremento o Decremento.

sola vez.

- SENTENCIA DO WHILE

```
DO
 {
  SENTENCIAS;
     }
WHILE(CONDICION);
```

Es similar al while con la excepción de que la condición se comprobará al final del bucle por eso como mínimo se ejecuta una vez.

El formato de definición de cualquier ARRAY sería → tipo nombre[tamaño];

Char nombre[tamaño]; → El tamaño uno más de lo normal. Y es necesario porque identifica la cadena al final con \0.

ARRAYS UNIDIMENSIONALES

Para almacenar 10 números enteros → int números[10];

Siempre va relacionado con un FOR.

```
for(x=0;x<10;x+1);
scanf("%d", número[x]);
```

Si queremos que tenga un valor inicial lo igualaremos a una lista de valores.

Los días de los meses → int diasmes[12]={31,28,31,30...}

LAS CADENAS DE CARACTERES.- En sí sería como un array bidimensional de caracteres. Por lo tanto habrá que pedir 2 tamaños.

1.- El nº de elementos que va a tener el array.

2.- El tamaño de la cadena. "ENERO"

Char nomes[12][11[9]]= { "ENERO", "FEBRERO", "MARZO" }

ESTRUCTURAS

Una estructura es una colección de variables que se referencian bajo el mismo nombre. Las Estructuras proporcionan un medio adecuado para mantener asociadas varias variables bajo un mismo nombre cuando los datos están relacionados. Son equivalentes a los record de Pascal.

El formato de definición es:

```
Struct nombre {
        Tipo variable;
          .          .
          .          .
        }Variable global;
```

GESTIÓN DE CADENAS

1.- Asignación de una cadena:

```
if(strcpy(a[10],desde[11])==0);
```

[9] [11]: Uno más.

[10] A: Destino.

(strcpy (xnom,nom)==0);

2.- Comparación. Se iguala a 0.

If(strcmp(xnom,nom)==0);
Hay que incluir la librería <string.h>.

FICHEROS

Todas las funciones requeridas se encuentran en la librería <stdio.h>.
El puntero de archivo, en sí es una variable de tipo FILE.

FILE *fichero;

- Apertura → fichero=fopen("c:\\Tc\\Bin\\datos.dat","wt");
- File*fichero;....<stdin.h>
- Control de fin de fichero → feof(puntero del fichero); Se forma un bucle para determinar que es fin de fichero.
- Cerrar → fclose(puntero de fichero);
- Lectura y escritura → fscan – Lectura, Entrar.
  Fprintf – Escritura, Salir.

- r.- Abre un fichero de texto para lectura.
- A.- Añade en un fichero de texto.
- W.- Crea y abre un fichero de texto para escritura.
- Wb.- Crea un archivo binario para escritura.
- R+.- Abre un archivo para leer/escribir para modificaciones.
- A+.- Añade o crea un archivo para leer/escribir.
- W+b.- Crea un archivo binario para leer/escribir.
- R+.- Igual que r.
- A+.- Igual que a.
- W+t.- Igual que w.
- Rb.- Crea un archivo binario para lectura.
- Ab.- Añade en un archivo binario.
- W+.- Crea un archivo para leer/escribir.
- R+b.- Abre un archivo binario para leer/escribir.
- A+b.- Añade o crea en un archivo binario para leer/escribir.
- Wt.- Igual que w.
- R+t.- Igual que r.
- A+t.- Igual que a.

[11] Desde: Origen.