```
----------------------------//-----------------------------
  ======    ||            //
||      ||  ||           //
||      ||  ||                =====      =====    ||  ==
||      ||  ||          ||  ||     ||  ||     ||  ||//  \\
||   \\ ||  ||          ||  ||     ||  ||=====    ||
||    \\||  ||          ||  ||     ||  ||         ||
  =====\\     ========   || ||=====      =====    ||
         \\                ||
          \\               ||
----------\\----------------||-----------------------------
1994            VERANO                              n. 50
-------------------------------------------------------
```


```
-------------------------------------------------------
                          QLíper
                Redactor: Salvador Merino
                  Tel. +34-(9)5-2475043
                      Cerámicas Mary
            Ctra. Cádiz (Torreblanca del Sol)
                   ES-29640 FUENGIROLA
-------------------------------------------------------
```


## Cuota Socio

La cuota socio es anual (Comienza en Enero y termina en Diciembre).
La salida de QLíper es TRIMESTRAL (4 discos al año).

Durante 1994 se ha decidido editar 5 discos.

ESPAÑA..... 1500 Pesetas.
RESTO MUNDO ..... 2000 Pesetas o US $15 o UK 10

## Editorial

Por fin se puede observar en este número de QLíper que el número de participantes ha aumentado considerablemente. Y como veis, participar es muy fácil. Espero que se animen los demás socios en mandar algún mensaje, porque de algunos solamente se sabe que llevan pagando la cuota varios años, pero ni tenemos su foto, ni sabemos que hacen con el QL actualmente.

Hay buenas noticias. El software de la QXL va a estar totalmente, o casi, terminado en primavera. Y no solamente eso, Miracle Systems ha lanzado al mercado la SUPER GOLD CARD a un precio de 375 (una opción para aquellos que no deseen comprar un PC trasto), la cual consiste en una tarjeta del mismo tamaño que la GOLD CARD, pero con un 68020 a 24 MHz, 4 MBytes de RAM, interface paralelo, port de expansión (para tarjeta gráfica o interface SCSI), y todo lo que tenia una GOLD CARD Standard. ¡La velocidad de la SUPER GOLD CARD es muy similar a la QXL!

Existen rumores de que el sistema operativo SMS2 ya se está vendiendo en forma de cartucho ROM para los ordenadores ATARI ST por unas 135.

A estas alturas del año solamente somos 15 socios, no creo que seamos más. He intentado con una mini campaña publicitaria por correo recordar a 11 antiguos socios que aún estamos al pie del cañón. La respuesta ha sido:

- Ha renovado (Félix Alonso).
- Me he comprado un MAC LC III, y mi QL tiene averiada la unidad de disco (Celestino Alvarez Pérez).

- 9 no han respondido.

La pequeña campaña de publicidad ha sido un fracaso. De todas formas, no me ha cogido por sorpresa, sabia que daria este resultado, pero habia que limpiar nuestra base de datos.

Aunque seamos pocos, he de recordar que este Club en mi manos tiene limitado su número máximo de socios a 24. Mis razones son obvias. El Club no existe como empresa con ánimo de lucro, sino que simplemente es un Club de amigos para intercambiar experiencias, y el tiempo que puedo dedicarle al club tiene sus limites.

Si deseamos darle un futuro a QLíper, habria que ir pensando en reconvertir el Club. En otras palabras, un Club de usuarios (programadores en su mayor parte) de máquinas y sistemas operativos diferentes. El sistema OLIMPO de Pedro Reina por su orientación a la programación multiplataforma podria ser el núcleo de la mayor parte del interés del Club. Sin embargo, es solamente una idea dadas las tendencias del mercado (nuevas familias de microprocesadores y nuevos sistemas operativos, y solamente una cosilla en común, el lenguaje 'C').

En este número he colocado la base de datos multimedia de QLíper en su estado actual. En ella podreis ver las caras de aquellos socios que han enviado sus datos y foto, pero de aquellos que no lo han hecho solamente vais a ver el sello de sin foto.

Salvador Merino, 26/2/1994

EN ESTE DISCO...
==========

| | |
|---|---|
| QLIPER50_SCR | (portada disco QLíper) |
| inventario_txt | (Tesoro de QLíper) |
| oferta_txt | (La programoteca QLíper) |
| carta_miguel_txt | (Miguel Estarellas) |
| carta_joaquin_txt | (Joaquin Gallardo Rodríguez) |
| libro_txt | (Salvador Merino) |
| modems_LANDATA_txt | (Felix Alonso y respuesta de Salvador Merino) |
| APPLE_Macintosh_txt | ( " " y respuesta de Salvador Merino) |
| oferta_ampliación_txt | ( " " ) |
| mi_mayor_equivocación_txt | (Salvador Merino) |
| historia_doc | (Salvador Merino) |
| - DiaSem | (Pedro Reina) |
| DiaSem_txt | |
| DiaSem_c | |
| DiaSem_exe | |
| DiaSem_bas | |
| - Tempus | (Pedro Reina) |
| Tempus_txt | |
| Tempus_c | |
| Tempus_exe | |
| Tempus_bas | |
| - LanzaUnProg | (Pedro Reina) |
| LanzaUnProg_txt | |
| LanzaUnProg_blq | |
| - Periféricos HP | (Pedro Reina) |
| PerifHP_txt | |
| VerSoftFont_bas | |
| Ejemplo_djp | |
| DeskScan_txt | |
| *** CALCULADORA *** | (Felix Alonso) |
| calculadora_exe | |
| calculadora_c | |
| calculadora_txt | |
| *** MERINO TIL v2.02 *** | (Salvador Merino) |
| merino_exe | |

demo_til
boot_til
manual_merino_til_txt
*** Foto-DBase socios QLíper ***  (Salvador Merino)
foto_dbase_til
socios_qliper_fxi
socios_qliper_fxd
socios_qliper_def

```
+------------+
| INVENTARIO |
+------------+
```

==========

CAJA QLIPER

==========

| Fecha | Concepto | ESP | Saldo |
|---|---|---|---|
| 93.01.01 | Saldo anterior ..... | | +12355 |
| 93.12.21 | Franqueo QLíper 46 | -1386 | +10969 |
| 93.12.28 | Envio material sobrante a S. Merino | -2207 | +8762 |
| 94.01.04 | Derechos correos y telegrafos | -204 | +8558 |
| 93.12.30 | Franqueo QLíper 47 | -1505 | +7053 |
| 94.01.05 | 30 discos | -1980 | +5073 |
| 94.01.05 | Pedido atrasados de Francisco Diaz | +4000 | +9073 |
| 94.01.05 | Francisco Diaz-Tendero | +1500 | +10573 |
| 94.01.05 | Pablo Cardenes Cañeque | +1500 | +12073 |
| 94.01.07 | Franqueo pedido Francisco Diaz | -410 | +11663 |
| 94.01.11 | 20 Etiquetas 50x75 KORES Fixo | -100 | +11563 |
| 94.01.11 | Dasio Carballeira Tella | +2000 | +13563 |
| 94.01.11 | Javier Zubieta Aguirre | +1500 | +15063 |
| 94.01.12 | 20 discos | -1320 | +13743 |
| 94.01.12 | Franqueo | -255 | +13488 |
| 94.01.13 | Franqueo | -135 | +13353 |
| 94.01.14 | Sellos | -90 | +12263 |
| 94.01.14 | Marcos Cruz Martín | +1500 | +14763 |
| 94.01.14 | Miguel Estarellas | +1500 | +16263 |
| 94.01.14 | Joaquín Gallardo Rodríguez | +1500 | +17763 |
| 94.01.17 | Pedro Reina | +2000 | +19763 |
| 94.01.18 | Josu Regidor Eguren | +1500 | +21263 |
| 94.01.18 | Sobres | -30 | +21233 |
| 94.01.18 | Franqueo | -45 | +21188 |
| 94.01.20 | Sobres | -145 | +21043 |
| 94.01.20 | Franqueo | -585 | +20458 |
| 94.01.20 | Ian-Charles Coleman | +3200 | +23685 |
| 94.01.21 | Franqueo | -345 | +23340 |
| 94.01.25 | Franqueo intercambio y pedido Ian | -425 | +22915 |
| 94.01.25 | Eduard Losada | +2000 | +24915 |
| 94.01.26 | Franqueo 49 | -705 | +24210 |
| 94.01.26 | 20 discos | -1320 | +22890 |
| 94.01.31 | 5 sobres | -25 | +22865 |
| 94.01.31 | Publicidad ex-socios'91 | -308 | +22557 |
| 94.01.31 | Franqueo | -110 | +22447 |
| 94.01.31 | Rafael Illanes Muñoz | +1500 | +23947 |
| 94.01.31 | 20 discos | -1320 | +22627 |
| 94.02.01 | Felipe Berganza Picón | +2000 | +24627 |
| 94.02.01 | Franqueo | -65 | +24562 |
| 94.02.01 | 17 sobres | -85 | +24477 |
| 94.02.01 | 40 etiquetas Kores Fixo 50x75 | -200 | +24277 |
| 94.02.02 | Franqueo 49 a Sinclair QL World | -135 | +24142 |
| 94.02.02 | 40 etiquetas Kores Fixo 50x75 | -200 | +23942 |

| | | | |
|---|---|---|---|
| 94.02.03 | Pedido de Joaquin | +1000 | +24942 |
| 94.02.04 | Franqueo Joaquin | -105 | +24837 |
| 94.02.04 | Publicidad ex-socios'93 | -180 | +24657 |
| 94.02.10 | Franqueo 49 a Clubes | -740 | +23917 |
| 94.02.14 | Felix Alonso | +2500 | +26417 |
| 94.02.14 | Franqueo | -65 | +26352 |
| 94.02.14 | 20 discos | -1320 | +25032 |
| 94.02.15 | Envio QLíper 49 a Qubbesoft | -135 | +24897 |
| 94.02.23 | Franqueo (Retorno material y un pedido) | -235 | +24662 |

==========

MATERIAL QLIPER

==========

| Fecha | Sobres | Sobres-Acolchados | Discos | DISCOS-ATRASADOS-PD-SHARE |
|---|---|---|---|---|
| 94.01.07 | 6 | 7 | 18 | 172 |
| 94.01.11 | 2 | 6 | 14 | 172 |
| 94.01.12 | 2 | 6 | 34 | 172 |
| 94.01.14 | -2 | 5 | 30 | 172 |
| 94.01.18 | 2 | 5 | 29 | 174 |
| 94.01.20 | 25 | 5 | 25 | 174 |
| 94.01.21 | 23 | 5 | 23 | 175 |
| 94.01.25 | 22 | 5 | 22 | 177 |
| 94.01.26 | 8 | 5 | 31 | 177 |
| 94.01.31 | 4 | 5 | 49 | 177 |
| 94.02.01 | 20 | 5 | 47 | 177 |
| 94.02.02 | 20 | 4 | 46 | 177 |
| 94.02.03 | 20 | 4 | 36 | 187 |
| 94.02.04 | 16 | 3 | 32 | 187 |
| 94.02.10 | 11 | 3 | 27 | 187 |
| 94.02.14 | 10 | 3 | 45 | 187 |
| 94.02.15 | 9 | 3 | 44 | 187 |
| 94.02.23 | 8 | 2 | 39 | 192 |

---

OFERTAS

---

DISCOS QL DE DOMINIO PUBLICO

Descripción
==========

| | | |
|---|---|---|
| CUQ_A1 | Números | 1-2-3-4 |
| CUQ_A2 | " | 5-6-7-8 |
| CUQ_A3 | " | 9-10 |
| CUQ_A4 | " | 11-12 |
| CUQ_A5 | " | 13-14 |
| CUQ_A6 | " | 15-16 |
| CUQ_A7 | " | 17 + Screens |
| CUQ_A8 | " | 18-19 |
| CUQ_A9 | " | 20-21 |
| CUQ_A10 | " | 22-23 |
| CUQ_A11 | " | 24-25 |
| CUQ_A12 | " | 26-27 |
| CUQ_A13 | " | 28-29 |
| CUQ_A14 | " | 30-31 |
| CUQ_A15 | " | 32 |
| CUQ_A16 | " | 33 |
| CUQ_A17 | " | 35 |

```
SCREEN_1    Pantallas digitalizadas
SCREEN_2          "      "
SCREEN_3          "      "
SCREEN_4          "      "
SCREEN_5    Pantallas Spectrum
GRAFICOS PARA ADULTOS 1, 2
COLECCION GRAFICOS COMPRIMIDOS
TRAINING V5.1
QLIPER_A1 números 36-38
QLIPER_A2    "    37-40
QLIPER_A3    "    39
QLIPER_A4    "    41
QLIPER_A5    "    42
QLIPER_A6    "    43
QLIPER_A7    "    44
QLIPER_A8    "    45
QLIPER_A9    "    46
QLIPER_A10   "    47


QLAVE
=====

QLAVE   1-2-3


QUANTA
======
C.A.D_1
COMMS_XFER_1
COMMS_XFER_2
EDUC_1
GMS_STRAT_4
GRAPHICS_1
GRAPHICS_2
KERMIT      1-2-3
LANGUAGES_1
MATHS_1
QDOS_JS_1     (QDOS JS ROM DISASSEMBLY)
UTIL_DRCPY
UTIL_EMACS_1   (UTILs Micro-EMACS editor)
UTIL_EMACS_2   (Run version editor)
UTIL_GEN 1, 2, 3, 4
PAGE DESIGNER
PSION 1, 2, 3, 4
VT 1, 2
PF 1, 2, 3
SP 1

C.G.H. Services
===============

C001-C002-C004-C005-C006-C007
AUSTRALIAN 1
CONECTIONS 1
COMMS 1
IMAGENES GIF 1, 2, 3
SUPERBASIC UTILITIES 1
PROG_LANGUAGE 1

New England QL User Group (NESQLUG)
===================================

A001

Svenska QL Gruppen (SveQL)
==========================

S001-S002-S003-S004-S005-S006
```

QL Contact France
==========

F001-F002-F003-F004-F005-F006-F007-F008-F009-F010-F011-F012-F013

QITALY CLUB
==========

I001-I002-I003-I004-I005-I006-I007-I008-I009-I010-I011-I012-I013
I014-I015

Qitaly Magazine 24-25

QUBBE
=====

Q001-Q002
MOLECULAR GRAPHIC v2.0
ELVIS EDITOR
Text 'N' Graphix (Demo)
QPACer
ZM1+ Spectrum Emu

Scottish QL Users Group (SQLUG)
==========

T001

National Dutch QL-Users Club (sin_QL_air)
==========

H001-H002-H003-H004-H005-H006-H007

QUASAR
======

QUASAR 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Individual Software (Diferentes origenes)
==========

X001-X002-X003-X004-X005
QBOX v1.19b
SPECTATOR v1.35 (Spectrum 128K)
Software Spectrum discos 1, 2, 3
Xtricator v1.10
software ZX81 discos 1, 2
PSION XCHANGE v3.90
GZIP
SPECULATOR (48K)
YACC
QTPI

Compilador C68 v4.12 en versión runtime (3 discos).
Compilador C68 para QDOS (v4.01 completa, y runtime v3.05).- Son 5 discos con el código Fuente. Solamente son necesarios 3 que contienen el runtime, varias utilidades y los manuales.

LIB CPORT
LIB QPTR
C Debugging Tools
C Programming Tools 1
LIB CURSES
GNU Text Utilities 1, 2

Condiciones:
---

- Debe enviar 100 ptas por cada disco a copiar. Esto incluye copia y embalaje/sellos (200 Ptas para Europa y 250 Ptas Resto Mundo).
- Naturalmente todos los discos enviados deben estar formateados a 1440 sectores. En otras palabras, no deben tener sectores defectuosos.
- La forma de pago para estas pequeñas cantidades es preferible en monedas o billetes.
- Para aquellos que no deseen enviarme los discos, puedo ofrecerlos grabados por 200 ptas (300 Ptas a Europa, y 350 Ptas Resto Mundo).
- Existe un fichero conteniendo una lista de todos los programas disponibles en los diferentes discos de nuestra libreria.

CARTA ABIERTA

Hace ya mucho tiempo que me hice el propósito de comunicarme con vosotros y por más de un motivo. Había escrito con anterioridad dando ánimos al pueblo para que colaborara y diera vida a QLIPER. Por otra parte quería contestar a algunas citas de anteriores números. Pero todo ello se fué al garete y han pasado meses sin haber cumplido mi propósito. Pondré la excusa de que nuevamente, en 1993, he tenido unas vacaciones fallidas (no os preocupéis, no os soltaré el rollo de mis penas) y he andado descabalado desde entonces.

Marcos: He tenido la desagradable impresión de haber provocado, o por lo menos haber colaborado fuertemente, una crisis en la editorial de QLIPER. Pero de ver_ dad que yo no pretendía levantar ninguna tormenta. La lectura de tu introducción a Qlíper 44 me dejó muy apenado. He sentido dolor en cuerpo ajeno. Espero, Marcos, que ahora veas tu implicación pasada con mayor optimismo, que has dado mucho de tí a Qlíper y, creo que puedo hablar por todos, te estamos muy agradecidos. Animo, tío, en lo personal y en lo profesional.

Ian Charles: Muchas gracias por tu consejo. De hecho esa acción es la que también me recomendó Bryan Davies (sí, he aparecido en los papeles, en Sinclair QL World) y debe haber sido tomada por algún otro perjudicado o por el mismo Bryan, como vereis en lo que sigue.

Saga TK Computerware - VISA - BS: Cronología subsiguiente

4-3-93: La secuencia de cargos-devolucione-cargos me permite pedir a VISA la devolución del primer cargo que era por un valor muy aproximado al precio del nunca recibido 'Lightning'. Resultado: VISA me devuelve ¡el ultimo cargo de TK!, por superior valor al demandado.
31-5: TK ataca de nuevo, anunciándome VISA un nuevo cargo, esta vez por 102 libras, superior a todos los anteriores, y que, naturalmente, rechazo.
30-6: VISA anuncia que cargará la citada suma el 15-7. Les insisto que el cargo es indebido.
7-7: Recibo una llamada de la policía, Departamento de Delicuencia Internacional: Interpol Londres les ha informado que tienen denuncias de "abusos repetidos de tarjetas de crédito" por parte de TK Computerware a personas de, creo recordar, tres países (me parece recordar que una de ellas tambien había escrito a Bryan Davies) y que les habían dado mis coordenadas. Esto sólamente podía provenir bien de TK (harto improbable), bien de SQLW. Acordé una cita con ellos el día siguiente y realicé una declaración a la que adjunté mi listado (gracias, QL) de hechos BS-VISA-TK.
9-7: Insisto ante el banco y VISA que el cargo es indebido, les informo de lo precedente y les reclamo contestación.
2-11: No se ha realizado el cargo pero tampoco he recibido contestación, que formalmente reclamo.
Epílogo: al día de hoy todo sigue igual. No he tenido más noticias ni de TK, ni de VISA, ni del banco, quienes ya verbalmente me dijeron, más o menos literalmente, que "no podían perder el tiempo escribiendo". Por lo tanto sólo me queda, en lo que a la actuación bancaria se refiere, escribir a su defensor del cliente.

Nacho Enrique: Muchas gracias Enrique por dedicarme un par de horas de tu ocupa-

dísimo tiempo. Estuve jugando unos cuantos días con el emulador del QL en el Amiga 500 y me encontré con una serie de dificultades que fuí incapaz de resolver debido en parte a lo reducido que dejamos el DOS y, sobre todo, a mis escasos conocimientos del mismo y falta de herramientas. Digamos que he olido lo que puede ser el funcionamiento del QL en el Amiga, quedándome con la miel en los labios.

Salvador Merino: Por orden más o menos cronológico.
Encantado de conocer una versión de la historia de los antecesores de Qlíper. Intuyo una posible relación con algún personaje de una "tienda" de productos para QL que, creo, todos hemos conocido.
Tengo la sensación de que te pasas al pensar en BBS, BBC y BBD. Aparte de la ausencia de modem, hay un salto muy grande (not a quantum leap anyways) de configuración entre tu sistema, el mío e incluso diría que el medio. En cualquier caso ahí va mi respuesta a tus seis preguntas básicas: sí - no - no - sí - quizás - hoy por hoy, sí. Pero me pregunto ¿sabemos todos en qué consiste (equipo, logicial, precios) y qué hace una BBS. Te doy caña, pero muy ligera; parte de esto ya lo has comentado.
He mencionado antes la configuración de nuestros sistemas y he pensado en ese momento que ésta podría ser dato de la foto_dbase, til o no til. Si te animas (más caña): QL MGE con Trump Card y FLP/RAM level 2, disquetera doble de 3.5" DD Power Computing, impresora Admate DP-100.
Salvador, creo que sería muy útil una puesta al día de un trabajo anterior, un fichero con los directorios (y quizás un breve comentario) de las últimas adquisiciones de la biblioteca; de otra forma la decisión de pedido de discos es prácticamente a ciegas.
Bromas aparte (por lo de darte caña), que sepas de mi agradecimiento por tu dedicación a Qlíper y de mi admiración por la productividad que consigues.

Pedro Reina: Muchas gracias por tus frecuentes ayudas. En particular quiero citar la resolución de mis problemas con el ATOF de C gracias al C68 V3.05 que me facilitaste. El próximo mes quisiera dedicar algún tiempo a la lectura del manual y probatinas del Olimpo V1.1
Bien por Cifras y, en otro orden, por WSET, que estoy utilizando en mi lanzador. Desde luego eres un productor incansable. Y el número uno en documentación.
Pedro y yo hemos sido los animadores-organizadores de las últimas reuniones de Qlíper. Desde Noviembre pasado tengo la pelota en mi tejado por iniciativa mía, pero voy a tener que abandonar. Ya daré señales de vida más adelante.

Rafael Illanes: Bravo Rafael. Quedo a la espera de la segunda parte.

¡Larga vida al QDOS y que llegue de una puñetera vez el SMSQ (¿es así?)!

25 Enero 1994

Miguel Estarellas
Cruz del Sur, 7
28007 Madrid

## CARTA JOAQUIN

Para los Spectrum-maníacos, ahí van direcciones de interés:

- Microsat.- Consejo de Ciento 345, ptas 6 y 7. Tlfs. 2160013 y 2157496. Fax 2160199, 08007 Barcelona (esta es la que está más cerquita). Sirven programas y periféricos. Venden el +D con o sin unidades de disco y utilidades de Tasman, Hisoft, Datel...

- Datel Electronics ltd. Govan road, Fenton Industrial Estate, Fenton, Stoke-on-Trent, ST4 2RS, England. Tel. 0782 744707. Fax 0782 744292. Como no me gustan los intermediarios, he escrito directamente a Datel. He podido comprobar que al fin y al cabo están cobrandote lo mismo que Microsat, por las variaciones de los tipos de Cambio, comisiones, gastos de envio, etc, asi que, en algunos casos, no merece la pena (en otros como el del ratón Genious + Art Studio si merece la pena).

Tasman Software. Hilton Court, 2 North Hill Road, Leeds LS6 2EN (UK). Son los creadores del famoso Tasword, aunque tienen más utilidades, algunas bastante curiosas.

Kobrahsoft Software. "Pleasant View", Hulme Lane, Hulme, Nr. Longton, Stoke-on-Trent, Staffs, ST3 5BH, England. También tienen algunas cosas bastante interesantes, como agendas, copiadores.....

Joaquin Gallardo, 31-1-94.

## CONCEPCION Y DISEÑO DE BASES DE DATOS (RA-MA)

Se trata de un libro de 989 páginas que proporciona conocimientos teóricos sobre el diseño de bases de datos del tipo Jerárquico, Codasyl, relacional y E/R extendido.

Hace ya 4 meses que tengo el libro, y no he sido capaz de leer más de la mitad, porque se trata de un libro super aburrido. Todo lo que vamos a encontrar en el libro es teórico, no hay nada práctico. Hay que aprenderse el significado de un montón de nombres y siglas para nombrar cosas que yo ya conocia con otros nombres. Hasta tal punto ha llegado el aburrimiento e incompresión (olvido de qué significa DDL, DML, DMCL, SI, SGBD, OLTP, BDD, DRI, IRDS, ISO/IEC, L4G, DEP) que no sabia el significado de lo que estaba leyendo. Además, casi todo el libro es una colección de Autores, Fechas y Documentos a consultar en otros libros.

Si he de decir la verdad, todo el libro es un lio de tal dimensión que un programador con mi experiencia en el mundo práctico que sabe realmente de qué va la cosa, termina creyendo que ha olvidado todo lo que ha aprendido en la práctica.

Si como se dice en el prólogo, este libro es realmente una copia actualizada a noviembre de 1992 de los manuales del curso de Bases de Datos que se imparte en la Facultad de Informática de Madrid, me alegro de no ser uno de sus alumnos, pues no habria sido capaz de terminar el curso, o habria memorizado toda la teoría para aprobar el examen, pero no sabria escribir un programa de Bases de Datos.

El cursillo de Analista Programador de CEAC tenia el gran inconveniente de que era muy caro, pero su contenido 100% práctico es realmente lo que hace que un ser humano sea capaz de comprender y asimilar los conocimientos de una forma fácil, rápida y durarera.

En resumen, no es el primer libro en Español que he comprado y no he sido capaz de leer por aburrimiento e incompresión del texto. Afortunadamente todos los libros en Inglés que he comprado, han sido muy faciles de entender (a pesar de que mis conocimientos de Inglés son muy limitados) y con un contenido muy práctico.

Nuestra Universidad Española es famosa por su teórica y su falta de práctica. Será verdad eso de que hay gente que se sabe el nombre de todos los huesos del cuerpo humano, pero luego no saben distinguir un hueso al natural, porque nunca lo han visto. La mayoria de mis amigos han estudiado la carrera de Empresariales (era una de las pocas que habia en Málaga en aquella época), y ellos mismos reconocen que todo lo que estudiaron alli, no sirve para nada en el mundo real, e incluso han tenido que hacer un cursillo práctico para poder trabajar en su actual empleo.

Salvador Merino, 3-2-94

MODEMS LANDATA
==========
Acaba de llegar a mis manos, sin cargo, un modems LANDATA, sin ningún tipo de información, tabla de características, ni manual de aplicaciones y de uso.
Viene acompañado de un conector modular (creo del tipo telefónica RJ-11), con enchufes machos a ambos extremos.
Esto es todo lo que tengo. Poca cosa para poder darle alguna aplicación.
Voy a indicar a continuación todos sus signos externos, por si alguien del Club puede ayudarme, facilitándome sus características, su conexión a la red de Telefónica y al ordenador, sus aplicaciones y programa a utilizar.
En el Panel frontal dispone de:
Un pulsador en rojo, RM
Un interruptor de palanca, ON
Una base de enchufe hembra, tipo Telefónica (creo RJ-11), LIN
Una base de enchufe hembra, tipo Telefónica (creo RJ-11), AUX
Una base de enchufe Amphenol, tipo DB, de 25 pins (creo para interfaz tipo RS-232-C), C-1
Un cable con enchufe para su conexión a la red eléctrica.
En el Panel posterior dispone de:
Inscripción LANDATA Modems MV-213
Ocho diodos LED con las siguientes señalizaciones:

| | | | | |
|---|---|---|---|---|
| | | H/L | RXD | DCD |
| ON | LIN | AW | TXD | DTR |

En la Placa base inferior dispone de:
Diez microinterruptores
La placa del fabricante: NAFAR ELEKTRONIKA S.A.
Modelo: LANDATA
Código: MV-213
V.eficaz: 220V
Consumo: .2 mA
Fabricado: En O9-89
Frecuencia: 50 HZ
Confío que con estos datos, alguien pueda conocer este modems y me facilite la información que solicito, por medio de la revista QLIPER.

Félix Alonso
Febrero de 1994

RESPUESTA
---

Para utilizar ese MODEMs desde QL solamente necesitas el programa de telecomunicaciones QTPI disponible en nuestra programoteca PD QLíper.
En sus documentos podras encontrar la información para construir un cable
Es recomendable utilizar el mismo de la impresora en SER2 (yo lo utilizo sin modificar, porque he configurado mi MODEMs para trabajar con 3 hilos) y hacer una prolongación con las rectificaciones, si fuesen necesarias, utilizando un conector hembra y uno macho de 25 Pines.

Salvador Merino, 26/2/1994

APPLE-MACINTOSH
==========
He leído en un anuncio de un periódico que Macintosh puede trabajar con documentos Macintosh, MS_DOS , Windows y UNIX. Con sólo añadir la aplicación SoftPC de Insignia Solutions, Macintosh puede trabajar con programas MS_DOS y Windows.
Además, añade, Apple tiene un modelo, el Macintosh Quadra 610DOS Compatible, que combina un procesador Motorola y un procesador Intel, de manera

que puedes pasar del entorno Macintosh al entorno MS_DOS ó Windows con -- sólo pulsar una tecla, ó trabajar con ambos a la vez si tienes dos moni-- tores.
Con relación al modelo Macintosh LC III 4/80, da la siguiente configura- ción:
Disco duro de 80 MB.
Ram de 4 MB ampliables a 36 MB.
Unidad de disco Apple SuperDrive capaz de leer discos con formato Macin- tosh, MS_DOS, OS/2, ProDOS y Windows.
Microprocesador 68030 con bus de datos de 32 bits.
Sonido, vídeo y conexión a red incorporadas.
Teclado, ratón y sistema operativo Sistema 7.1 en castellano incorporado.
Soporte para conectar 7 periféricos SCSI.
Su precio: 129.900,- Ptas.
A la vista de toda esta información, que en principio debemos admitir co- mo cierta, parece una buena oferta ésta del Macintosh LC III 4/80, ya que con la apliación SoftPC de Insignia Solutions, que no sé su precio, dispon- dríamos de un ordenador en el que se podrían ejecutar programas para Macin tosh y programas para MS_DOS y Windows, indistintamente, aunque no en mul-- titarea.
Pero se me ocurre preguntar, en relación con los programas para MS_DOS y Windows, si es compatible 100% sin problemas, y que velocidad se consigue.
Y otra pregunta, muy importante, es si se podrá aplicar la tarjeta QXL, y si será compatible 100% sin problemas, ó si para ésto sería necesario partir del modelo ya referenciado , el Macintosh Quadra 610 DOS Compati- ble, cuya configuración y precio no conozco, con el cual es de suponer que trabajando en entorno MS_DOS, la aplicación de la Tarjeta QXL no tendrá - ninguna dificultad ni dará problemas.
Espero información, por medio de Qliper, de quien esté al corriente de es- tos ordenadores.

Félix Alonso
Burgos, Febrero de 1994

RESPUESTA
---

El programa SoftPC es simplemente un emulador PC que emula el chip 80286 en modo real y protegido; también emula la memoria extendida y expandida y gráficos VGA.

Ficha técnica de SoftPC, según revista PC ACTUAL Junio'93:

DESCRIPCION: Aplicación Macintosh que emula un IBM PC/AT compatible y permite ejecutar aplicaciones MS-DOS y WINDOWS. Se suministra con Windows 3.1 ya instalado. A pesar de que evaluamos una versión Beta, SoftPC se mostró lento y repleto de incompatibilidades.

REQUISITOS: Se recomienda un MAC con MC68040, tipo Quadra. Un buen disco duro para la partición y al menos 8 Megas de RAM.

EQUIPO DE PRUEBAS: Macintosh IIfx con procesador 68030 a 25 MHz, con 80 Mbytes de disco duro y 8 Mbytes de RAM.

PRECIO: 74.500 Ptas.

DOCUMENTACION: El programa se suministra en Inglés. Documentación muy pobre (analizamos una Beta).

FABRICANTE: Insignia Solutions.

DISTRIBUIDOR: Aberón. Corcega, 411, 4 Pta. Barcelona 08037. Tel (93) 4590700

También existen Clonos PC y MAC a la vez. Consiste en meter una tarjeta madre PC y una MAC, dos monitores (o uno compatible para ambos), dos discos duros,... El precio es interesante y cuestan un poco menos de lo que costaria un PC y un MAC por separado.

Salvador Merino, 26/2/1994.

OFERTA
======
En una reciente conversación mantenida con un usuario de QL, me ha comunicado que está interesado en la compra de:
Una ampliación de memoria con controlador de disco y
Una unidad de disco
Po lo tanto, si alguno desea vender estos aparatos, puede dirigirse a mi dirección , para ponerle en contato con el interesado.

Félix Alonso
Pza. Francisco Sarmiento, 2-6"
09005 BURGOS.

MI MAYOR EQUIVOCACION, COMPRAR UN PC NATURALMENTE
==========

Han pasado 4 meses desde que se me ocurrió comprar en mi pueblo un PC 486 y una VIDEO BLASTER. Si me avisan por adelantado de que iba a sufrir todas las penitencias que he sufrido, no habria pedido nunca la QXL, y mucho menos habria comprado un PC.

La casa STRONGER me vendió por mediación de ACADEMIA LOS BOLICHES cobrando por adelantado y demorándose en la entrega 2 semanas en el PC y 3 en la VIDEO BLASTER. Ambos productos me fueron entregados defectuosos, y fueron retornados lo más rápidamente posible al distribuidor. Pues bien, después de 4 devoluciones durante los 4 meses, lo único que funciona ya aceptablemente es el PC que ahora es un 80486DX-40 MHz. La VIDEO BLASTER no ha habido manera de que la repararan o la cambiasen por otra nueva, y no solamente eso, me han han retornado los manuales quemados por alguien que fumaba cigarrillos en el trabajo, y para colmo, algún chorizo que trabaja en STRONGER se ha quedado con mi disco CD-ROM del 'Video For Windows'. Y aún hay más, las 4 devoluciones le han costado al vendedor unas 24.000 Ptas en portes más unas pocas miles de pesetas más en gastos de teléfono, lo que significa que ha perdido todo el beneficio de la venta. Aunque el mayor perdedor de esta historia he sido yo, que he perdido mi dinero y ha faltado muy poco para perder una amistad de 27 años con un amigo por culpa de un distribuidor de informática ESPAÑOL, STRONGER.

La única solución que me quedaba era denunciar a STRONGER, pero conociendo como funciona la justicia en este pais, y como quiebran las empresas últimamente, creo que es perder el tiempo y más dinero. Para que os hagais una idea de como ha salido todo, os diré que el vendedor ha cambiado de distribuidor después de todo este lio.

He enviado mi VIDEO BLASTER a Singaphur, pues según la garantia del fabricante, tiene un año de garantia incluyendo mano de obra si se envia directamente a la fábrica. Lo que no está incluido son los gastos de envio y retorno, pero son muy inferiores a lo que cobra SEUR por enviar un paquete de Málaga a Madrid. Ya lo único que me queda por esperar es: ¿Será Creative Labs una casa seria y responsable de sus productos?, porque lo que se dice STRONGER, deja mucho que desear (me recuerda a INVESTRONICA).

Tenia pensado comprar una SOUNDBLASTER y un lector CD-ROM en Navidad'93, pero las cosas han salido tan torcidas que se me han quitado las ganas de ampliar el PC, e incluso de comprar programas PC para que mi PC 486 pueda hacer algo útil.

He enviado mi QXL 2M a actualizar a QXL 5M, y hoy he recibido la revista Sinclair QL World conteniendo la noticia de que la SUPER GOLD CARD ya está a la venta por 375.

La SUPER GOLD CARD consiste en una tarjeta del mismo tamaño que la GOLD CARD, pero con un MOTOROLA 68020 a 24 MHz, 4 MBytes RAM, interface de disco hasta 4 unidades, interface paralelo y ranura de expansión (es posible ampliar con tarjeta gráfica e interface SCSI). La velocidad de la SUPER GOLD CARD es muy similar a la QXL (no se sabe cuál de la dos será más rápida, pero la diferencia será de muy poco).

Después de todo lo que he pasado con la compra de mi PC, si yo tuviese que recomendar a alguien con un QL que necesitase más memoria y velocidad, le recomendaria la SUPER GOLD CARD.

Sobre el software de la QXL hay buenas noticias. La compatibilidad con el slot de 8-bit y el bug en el formateo de discos QL, han sido resueltos. Hay una nueva versión del sistema operativo SMSQ que no ha sido enviado a los usuarios, porque tiene algunas cosillas sin terminar. El nuevo BASIC ya está disponible, pero no para enviarlo a los usuarios.

Sobre que estoy haciendo con mi PC 486. Pues os vais a reir. Por lo pronto mi programa MERINO TIL tiene problemas debido a que el compilador TURBO C++ trabaja con varios tipos de memorias (small, compact, large, huge) debido a los lios del MS-DOS y el Intel 8088 (manejo de la memoria por segmentos de 64 KBytes). Bueno el problema es más bien de algunas rutinas de la librería OLIMPO (de Pedro Reina) que funcionan perfectamente en memoria SMALL, pero que tienen problemas con los tipos de memorias grandes (large y huge). Esas rutinas son el objeto 'BASEDATO' y la función Cad_Une(). Esta última es fácil de solucionar sustituyendo con malloc(), strcpy() y strcat(), pero conseguir que el objeto 'BASEDATO' corra en memoria 'huge', eso es otra historia (lo que en un QL es muy fácil, en el PC se complica una barbaridad).

El WINDOWS es muy bonito, pero no me ha hecho mucha gracia, y no sé cómo programar bajo WINDOWS. Prefiero el QDOS y el QPAC II.

El PC 486 lo estoy usando principalmente para algo que no hago casi nunca con el QL, correr software de JUEGOS. Cada vez que pienso que me he gastado casi 300.000 Ptas para eso, me pongo a toser y a lagrimear los ojos.

Salvador Merino, 25/2/1994

## DiaSem

Hace muchos, muchos, pero muchos... años encontré en algún sitio (de verdad que no recuerdo cuál) un algoritmo para calcular el día de la semana de una fecha.

Lo implementé en una calculadora programable, la HP 11C, y me sirvió para... conocer el día de la semana de las fechas de nacimiento de mis compañeros de Bachillerato (¿veis como no exageraba respecto a los años?).

También se puede adaptar para calcular el número de días entre dos fechas, lo que me permitió calcular biorritmos con mi "sencilla" HP.

Volví a implementar hace un par de años el algoritmo, esta vez en SB para un programa que imprime una agenda mensual. Este programa está en proceso de adaptación, lo veréis en algún número próximo de QLíper.

Y llegó el momento de añadir el algoritmo a Olimpo, de modo que he reescrito otra vez (espero que ya sea la última) el algoritmo en C. Podéis ver el fuente en el fichero DiaSem_c. El programa se puede compilar tanto en un QL como en un

PC, basta cambiar el #define que hay al principio.

Podéis poner a prueba el programa con la orden

EXEC FLP1_DiaSem_exe

o cualquiera equivalente, o simplemente ejecutando el programa SB llamado DiaSem_bas.

Francamente, no sé si el algoritmo es absolutamente fiable, ya sabéis que con eso del paso del calendario juliano al gregoriano se organizó una jarana grande (hubo que adelantar el calendario 10 fechas), de modo que si pensáis usarlo, haced algunas pruebas primero. Me gustaría que me contestárais cómo os va si lo hacéis.

Pedro Reina, V.11.2.1994

## Tempus

En verano de 1989 conocí París. En el Centro Pompidou había una exposición llamada "Magos de la Tierra". Magnífica, estaban represntadas muchas formas distintas de expresión plástica, me pareció llena de novedades.

Una de las cosas que más me llamó la atención fue un montaje cuyo autor no recuerdo: consistía en una habitación cuadrada de unos tres metros de lado, totalmente a oscuras. A dos metros de altura había una serie de contadores de dos dígitos rojos, a lo largo de toda la habitación, de modo que rodeaban a los asistentes. Cada contador llevaba su propio ritmo de cambio. Unos iban rapidísimamente, otros muy despacio.

Me quedé embobado, estuve bastante tiempo llenándome de sensaciones. Bueno, puede parecer una tontada, pero os aseguro que me inspiró muchas cosas el montaje.

Así que quise remedar aquella instalación en mi ordenador. Hace algún tiempo que lo hice en SB, para lo que necesité unas extensiones que permiten medir el tiempo con precisión de décimas de segundo, más que la función DATE. Después lo escribí en C para PC, y por fin hace unos días lo he vuelto a escribir para el QL, pero en C, usando un "truqui", nuevo para mí, que me permite medir el tiempo en décimas de segundo, pero sin la necesidad de las extensiones.

Este es el truqui: el QDOS aumenta en una unidad el valor de la palabra de dirección 163886, de modo que la expresión

PEEK_W(163886)/50

da una referencia temporal con bastante precisión. Lo he leido en QUANTA, volumen 2, número 1, página 6, y se debe a Richard Kingslake.

Para ver el programa en acción hay que teclear

LRUN FLP1_Tempus_bas

El efecto que pretendo conseguir se aprecia mejor si tenéis un monitor en color, tenéis la luz apagada, hay silencio a vuestro alrededor y (esto sólo es recomendable, no obligatorio) tenéis un QL "pata negra" (es decir, con una Gold Card dentro). No os prometo nada, pero intentadlo con buena voluntad. Cuando os canséis, pulsáis cualquier tecla y el efecto se pierde.

Pedro Reina, V.11.2.1994

## LanzaUnProg

Necesito nuy a menudo lanzar un programa ejecutable desde SB, pero me apetece preparar antes la pantalla y ofrecer al usuario la posibilidad de lanzar o no el programa, o de lanzarlo varias veces sucesivamente.

Por tanto, tenía un programa que escribe un mensaje en la pantalla y pregunta al usuario si se quiere lanzar el ejecutable. Pero el programa lo tenía que retocar para cada nuevo nombre que quería poner.

Preparando los programas Tempus y DiaSem para QLíper, me di cuenta de la necesidad de tener automatizado el proceso, de escribir un programa que sirva para todas las situaciones. Para eso queremos los ordenadores, ¿no?

Por tanto, he escrito LanzaUnProg, que es un programa en SB que presenta un nombre en pantalla bien centradito, pregunta al usuario si quiere ejecutar el programa binario y por fin lo lanza. Cuando hay que cambiar los nombres, se cambian las variables globales y ya está. El programa es muy corto y bastante sencillo, pero me he decidido a mandarlo a la revista por si hay algún novato al que le pueda venir bien.

Podéis consultar el fichero LanzaUnProg_blq. La extensión "blq" la utilizo como abreviatura de "bloque". Un bloque es para mí un programa en SB que no necesita números de línea. Escribiendo bloques tengo más libertad para recombinar programas antiguos para escribir los nuevos (reutilización de código). Cuando voy a ejecutar el programa en el intérprete, añado las líneas, cosa que hago automáticamente con el editor que uso (ArcED).

Pedro Reina, V.11.2.1994

## Periféricos Hewlett-Packard

Los periféricos de HP suelen marcar la pauta del mercado. Las LaserJet dominan el mercado de las impresoras láser, las DeskJet son un éxito de ventas y los digitalizadores DeskScan suelen ser los mejores de su gama.

Pero yo tenía un problema con ellos: la falta de documentación. Los manuales suelen ir dirigidos al usuario medio de PC, es decir, una persona que no sabe mucho de informática y que no lee los manuales. Una persona que enchufa, pincha en un botón de Windows donde pone "LaserJet IV" y al menor fallo, protesta.

Los que hemos nacido a la informática de la mano del tío Sinclair vamos de otra manera, nos gusta saber lo que tenemos entre mano. Los que quedamos en el mundo QL, por lo general, somos artesanos informáticos. No tenemos suficiente con unos manuales tan escuálidos como los normales de HP.

Mi tortura empezó con la documentación de la impresora DeskJet 500. Yo quería, entre otras cosas, conocer el formato de las fuentes de letra en disco, que se pueden cargar en la impresora cuando a ésta se le enchufa un cartucho de memoria. En el manual, nada; compré el manual "técnico", nada; compré el libro "DeskJet Unlimited" a una empresa de U.S.A., nada. Parecía que nadie tenía ni idea de cómo está definida una fuente Deskjet.

Con el cabreo consiguiente me planté en el chiringuito HP del SIMO. Incordiando, incordiando, me mandaron a Jordi Riat, responsable de periféricos de HP España. ¡Menos mal! A la semana siguiente me mandó desde su despacho de Barcelona, totalmente gratis (supongo que por mi cabreo) el "Software Developer's Kit" (conocido mundialmente como SDK) de la familia de impresoras DeskJet, que en aquel momento comprendía la HP DeskJet, la Plus, la 500 y la 500C. El kit contenía unos "apuntes" (fotocopiados) con la documentación "de verdad" y un disquete con las "métricas" de las fuentes de la impresora (es decir, el ancho

de cada carácter).

En el paquete también me mandó la dirección a la que hay que pedir los SDK que se deseen. Es ésta:

Hewlett-Packard
Int. Business Reply Service
C.C.R.I. Numéro 353
1100 VC Amsterdam-Zuidoost
THE NETHERLANDS

Tienen bastantes disponibles, son muy baratos, los mandan con cierta rapidez y sólo admiten pago mediante VISA.

Gracias a esa documentación pude enterarme del formato de las fuentes en disco ("soft fonts") y por tanto, manipularlas para adaptarlas a mi gusto.

Para uso personal escribí un pequeño programa SB que visualiza cada carácter de una fuente. El programa es sencillo y lento, pero me sirvió para entender bien el formato. Lo podéis probar tecleando

LRUN FLP1_VerSoftFont_bas

He incluido la fuente Ejemplo_djp para que podáis probar el programa. La fuente está generada con el programa "More Fonts 3.0" de Micrologic Software, y por tanto tiene "copyright".

Para salir el programa tenéis que introducir el cero "0" cuando os pregunte el carácter que queréis visualizar.

Animado con el SDK de la DeskJet, me animé a pedir el de los digitalizadores ScanJet, ya que planeaba comprarme uno y quería saber de antemano y de verdad sus características. Los vendedores, usualmente, no tienen ni idea de cómo funciona un bicho de estos.

Me cobraron unos 1000 duros, pero esta vez me sorprendí; la cantidad de documentación era tremenda:

* Scanner Control Language and C Language Library for HP Scanners
  Unas 150 páginas explicando las órdenes para manejar los ScanJet y la librería en C provista en disquete.

* TIFF Revision 6.0
  Más de cien páginas con la descripción oficial del formato TIFF (es un formato para almacenar imágenes "bit-map" reconocido universalmente).

* A Guide to the Tag Image File Format. Version 5.0
  Un tutorial sobre el formato TIFF.

* TIFF C Language Library for HP Scanners
  Documentación de la librería de HP para manejar ficheros TIFF.

* Un disquete de 1.44M.

Como véis, no se puede pedir más por 1000 duros.

En el fichero ScanJet_txt podéis ver el contenido del disquete.

Ya sabéis, si queréis desarrollar software para algún periférico HP, pedís la documentación a Holanda y a las dos semanas podéis estar trabajando.

Y termino con unas preguntas: ¿cómo diablos se pueden manejar desde una QXL los periféricos enganchados al PC? ¿Documentará Miracle Systems el acceso a los puertos del PC? ¿Podremos usar un digitalizador desde el adaptador SCSI que está desarrollando Miracle? Como véis, en informática hay algo más (sí, más) importante que los productos: su documentación.

Pedro Reina, L.14.2.1994

CALCULADORA DOBLE PRECISION
==========

Este programa, Calculadora_exe, es en realidad la primera versión de una calculadora científica que trabaja aprovechando la doble precisión del -- compilador C-68.
Su utilización es tan sencilla que no necesita ninguna explicación. Símplemente seguir las indicaciones que se dán.
Como primera versión adolece de algunos errores, ya comprobados, que es preciso eliminar, y además puede ser mejorado, por ejemplo con entradas no válidas, cazafallos, etc,. Igualmente se puede complementar añadiendo nuevos bloques ó menus de funciones, como conversiones de coordenadas polares a cartesianas, funciones estadísticas, etc.

Félix Alonso
Burgos, Febrero de 1994

---

MANUAL MERINO TIL

---

CUOTA USUARIO REGISTRADO
---

MERINO TIL es un programa ShareWare. Su código ejecutable puede copiarse sin ninguna limitación.

Si te gusta MERINO TIL, hazte usuario registrado. Vas a recibir la última versión del programa, el código fuente en 'C' y vas a estar informado de la realización de las nuevas versiones.

La cuota de usuario registrado es de 1.000 ptas.

Salvador Merino
Ctra Cádiz, Cerámicas Mary
29640 Torreblanca del Sol
Fuengirola (Málaga)
TEL. 2475043

PROLOGO
---

Después de terminar el cursillo de Analista Programador, me habia propuesto como objetivo crear mi propio lenguaje de Bases de Datos, y unos pocos meses más tarde nació MERINO TIL (un lenguaje todo-terreno especializado en Bases de Datos).

A pesar de que soy el autor de MERINO TIL, debo reconocer que MERINO TIL toma código prestado del libro "WRITE YOUR OWN PROGRAMMING LANGUAGE using C++" (suministrado por FORTH INTEREST GROUP de California) y la libreria 'C' OLIMPO de Pedro Reina. Sin la ayuda de ambos MERINO TIL no existiria en su forma actual. Gracias a ambos por su ayuda.

¿Qué es MERINO TIL?
---

MERINO TIL es un interprete/compilador basado principalmente en:

- Un THREADED INTERPRETER LANGUAGE (TIL). Traducido al Castellano debería sonar como "lenguaje intérprete cosido/entrelazado".

- Un compilador ANP (Anotación Polaca Inversa).

- Una librería 'C' llamada OLIMPO, y escrita por Pedro Reina, que aporta el bloque principal de palabras primitivas que sirven para realizar algo útil.

- Una colección de comandos para la concepción y diseño de Bases de datos Multimedia (Gráficos y texto).

Las características más importantes de este lenguaje son:

- Es un lenguaje bastante rápido, casi tan rápido como su propio código 'C'.

- Puede ser compilado e intérpretado.

- Una diferencia muy importante con otros lenguajes, es poder crear nuestro propio diccionario de palabras, porque en MERINO TIL cada palabra hace referencia a otras palabras que ejecutan instrucciones especificas, ya que cuando definimos una rutina para producir el cuadrado de un número, estamos definiendo una palabra (el ordenador coge la definición y la compila generando una lista de llamadas a otras rutinas que ejecuta con gran rapidez).

- La diferencia más importante frente a otros lenguajes es el 'stack' y su modo de operación notación polaca inversa.

La traducción del término 'stack' es pila. En los ordenadores un stack o pila es una zona de la memoria donde se van depositando y sacando los datos con un sistema conocido por LIFO o "Last In, First Out" (último en entrar, primero en salir).

## ¿Para qué sirve MERINO TIL?

Un lenguaje de aplicación es un lenguaje dentro de una aplicación que facilita al usuario la capacidad de escribir pequeños programas, a menudo llamados macros, para trabajos automáticos repetitivos o extender la utilidad de una aplicación. Muchos programas comerciales incluyen lenguajes de aplicación. Lotus 1-2-3 tiene un lenguaje de macros. Wordstar tiene macros. dBASE tiene un lenguaje. AutoCAD tiene AutoLisp.

La portabilidad es hoy en día muy importante. MERINO TIL ha sido escrito en un lenguaje portable, el 'C'. Normalmente un lenguaje TIL se escribe en lenguaje assembler, y se obtiene una velocidad excelente, pero nada de compatibilidad con otros sistemas. Un TIL escrito en 'C' no tiene la velocidad del assembler, pero es muy portable.

TILs son muy simples comparados con compiladores convencionales. Utiliza un 'parsing' muy simple y no requiere tablas de simbolos. Genera direcciones de funciones 'C' para ejecutar.

TILs son un interesante cruce entre compiladores e intérprete. Mi primer contacto con los TILs fue el FORTH en 1984 con el Sinclair SPECTRUM. Luego he utilizado FORTH con el Sinclair QL, y he escrito versiones FORTH en assembler Z80 para el Cambridge Z88 y en assembler 68000 para el Sinclair QL.

Como mis nuevos proyectos de desarrollo de software pasan por el 'C', y he usado FORTH durante los últimos años, he creido que seria muy interesante continuar utilizando la naturaleza interactiva de un TIL en mis nuevos programas, incluso cuando el 'C' no es un lenguaje interactivo.

## EL MANEJO DE NUMEROS

Todos los números son tratados como enteros de doble longitud (32 bit) con un valor en el rango -2147483648 a 2147483647 con signo.

## EL STACK

Todas las operaciones aritméticas usan números sobre el stack. Puede ser visto como una pila de números, siendo el que está en lo alto llamado TOS y el segundo 2OS y asi sucesivamente. Si introducimos dos números 123 130, entonces 130 es el TOS y 123 el 2OS. Si queremos sumar los dos números, entonces escribimos + , el resultado es escrito en el nuevo TOS.

```
123 130  + . <ENTER>  la respuesta es   253
```

## OPERACIONES ARITMETICAS

```
+      { n1 n2 -- n3 } Esto suma el TOS n1 al NOS n2, y da la suma en n3.
       100 23 +    deja 123 en el stack.

-      { n1 n2 -- n3 } Resta n2 a n1, y deja la diferencia en n3.
       100 23 -    deja 77 en el stack

*      { n1 n2 -- n3 } Multiplica n1 por n2, y deja el producto en n3.
        123 3 *    deja 369 en el stack

/      { n1 n2 -- n3 } Divide n1 por n2, y deja el cociente en n3.
       120 30 /    deja 4 en el stack

mod    { n1 n2 -- n3 } Divide n1 por n2, y deja el resto en n3.
       136 30 mod  deja 16 en el stack
```

## OPERACIONES LOGICAS

```
and    { n1 n2 -- n3 } AND lógico.
       69 7  and   deja 1 en el stack

or     { n1 n2 -- n3 } OR lógico.
       10 19 or    deja 1 en el stack

not    { n1 -- n2 } NO lógico.
       -1 not      deja 0 en el stack
```

## OPERACIONES A NIVEL DE BITS

```
&      { n1 n2 -- n3 } AND a nivel de bits.

|      { n1 n2 -- n3 } OR a nivel de bits.

^      { n1 n2 -- n3 } XOR a nivel de bits.
>>     { n1 n2 -- n3 } Desplazamiento a la derecha de n2 posiciones de bits
       en n1.

<<     { n1 n2 -- n3 } Desplazamiento a la izquierda de n2 posiciones de
       bits en n1.
```

## LAS PALABRAS QUE MANIPULAN EL STACK

```
drop   { n1 n2 n3 -- n1 n2 } Quita el número que está en lo alto del stack.

dup    { n1 n2 - n1 n2 n2 } Duplica el número que está en lo alto del stack.

rot    { n1 n2 n3 - n2 n3 n1 } Rota  3OS a TOS.

swap   { n1 n2 - n2 n1 } Simplemente cambia los dos números que están en lo
       alto del stack.

depth  { n1 n2 n3 -- n1 n2 n3 3 } Deja la cantidad de números que hay en el
       stack.
```

COMPARACIONES CONDICIONALES
---

Hay palabras que comparan números en el stack y dejan falso (TOS = 0) o verdadero (TOS = 1). Son:

```
<      { n1 n2 -- flag } Verdadero si n1 < n2
=      { n1 n2 -- flag } Verdadero si n1 = n2
>      { n1 n2 -- flag } Verdadero si n1 > n2
>=     { n1 n2 -- flag } Verdadero si n1 >= n2
<>     { n1 n2 -- flag } Verdadero si n1 no es igual a n2
```

VARIABLES Y CONSTANTES
---

Son 'variable' y 'constant'. Ejemplos:

```
variable JOSE    crea una variable llamada JOSE

123 constant SALVADOR  crea una constante llamada SALVADOR a la que se
                       asigna el valor 123
```

'constant' asigna el número en lo alto del stack al nombre que le sigue. 'variable' no asigna un valor al nombre que le sigue. Cuando esos nombres son ejecutados, una constante deja su valor en el stack y una variable deja su dirección en el stack.

```
SALVADOR .    imprime 123
JOSE .        imprime una dirección dependiendo de la posición de JOSE en
              el diccionario.
```

USANDO VARIABLES
---

Las palabras que escriben y leen valores de una variable son ! y @, y sus equivalentes en byte son c! y c@.

```
!      { n ad -- } Carga el valor n en la dirección ad.
       234 FRED !  carga el valor 234  en la variable FRED

@      { ad -- n } Deja el valor n contenido en la dirección ad.
       FRED @    deja 234 en el stack (si ha usado el ejemplo anterior).

c!     { n ad -- } Carga el byte n en la dirección ad.
       99 FRED c!  carga el valor 99 en la variable FRED

c@     { ad -- n } Deja el byte contenido en la dirección ad.
       FRED c@     deja  99 (si ha usado el ejemplo anterior).

?      { ad -- } Imprime en la pantalla el valor contenido en la
       dirección 'ad'.
       FRED ?    imprime en la pantalla 99.
```

ARRAYS Y CADENAS
---

```
."  Texto"     Solamente se puede usar en modo compilación. Cuando se
               imprime el texto en la pantalla.

" Texto"       { -- ad } Solamente se puede usar en modo compilación.
               Cuando se ejecuta deja la dirección de la cadena constante.

$variable      { n -- } Crea una cadena de longitud n. Cuando se ejecuta la
               cadena, deja su dirección en el stack.

                     20 $variable JUAN

read" Texto"   { ad -- } Compila el texto en la dirección 'ad'.

                    JUAN read" Juan"

$array         { n2 n1 -- } Crea una array de cadena de 'n2' elementos de
               longitud 'n1'.

                      20 10 $array PEDRO


array          (n -- } Crea un array unidimensional de'n' números.

                      10 array NUMERO

2array         { n2 n1 -- } Crea un array bidimensional de números de 'n1'
                por 'n2'.

                     3 4 2array 2NUMERO

carray         { n -- } Crea un array unidimensional de 'n' bytes.

                      20 carray PUNTOS


RETORNO DE STACK
----------------

r>           Pasa un número del stack al retorno de stack.

>r           Pasa un número del retorno de stack al stack.


DEFINIENDO PALABRAS
-------------------

Una definición comienza con la palabra ':' y termina con ';'.
Ejemplo:

: cuadrado dup * ;

Compila una palabra llamada 'cuadrado' en el diccionario. Para ejecutarla
se necesita un número en el stack para 'cuadrado'.

3 cuadrado  . <ENTER>   imprime  9

Ahora 'cuadrado' puede ser utilizado en otras definiciones :

: a_la_cuarta cuadrado cuadrado ;

2 a_la_cuarta . <ENTER>  imprime 16


CONTROL DE ESTRUCTURAS
----------------------

Para poder controlar el curso de un programa existe los bucles, que son
unas estructuras que nos permite repetir un conjunto de instrucciones unas
```

determinadas veces. En el bucle se distiguen dos partes principales, los delimitadores de control que se encuentran en los extremos señalando el principio y el fin del bucle y el bloque central, que son las instrucciones que queremos ejecutar repetitivamente.

Los bucles se dividen en dos tipos distintos: bucles fijos y bucles condicionales. Los fijos son aquellos en los que el número de veces que se repite el bloque central es fijo y se conoce al empezar el bucle (es una condición fija  y externa al bucle).

Los bucles condicionales tienen la misma estructura que los fijos, pero se diferencian en que el número de veces que se repiten las operaciones no se da fijo al empezar el bucle, sino que es una condición variable (puede, incluso repetirse hasta el infinito) e interna del bucle. Esta condición se determina por el resultado de una operación dentro del bloque principal.

La estructuras de control solamente pueden ser usadas en modo compilación.

Las estructuras son:

```
if...else...endif { flag ---} Si el flag es verdadero, las palabras entre
                  'if' y 'else' son ejecutadas. Si es falso, las palabras
                  entre 'else' y 'endif' son ejecutadas.

          : test if ." Verdadero" else ." Falso" endif ;

          0 test    imprime Falso
          1 test    imprime Verdadero


begin.......while { flag --- } 'while' examina el flag, si es verdadero
                  ejecuta el bucle de nuevo hasta que sea falso.
```

¿cómo se controla cuando se termina el bucle? Parece no existir, ya que no hay ningún bloque destinado a ese fin. En realidad, se hace desde dentro del bloque de palabras, ahí realizamos todas las operaciones, comparaciones y entradas de datos que deseamos. Y cuando la ejecución llega a 'while', se comprueba si el número en lo alto del stack es VERDADERO, si lo es, se ejecuta el bucle de nuevo.

```
    : test 1 begin  dup . 1 + dup 10 < while drop ;
```

Esto imprime los números de 1 al 9.

```
(.......)    {n --}  El bucle se repite n veces.

         : repite 10 ( ." QL" cr ) ;


for.....loop {n1 n2 -- } n1 es el limite y n2 el inicio. Se repite n1-n2.

         : repite 10 0 for ." QL" cr loop ;  repite QL 10 veces
```

Otras palabras asociadas con for....loop y (.....) son:

```
i        Deja el valor del contador (n2 o n) en el stack.
leave    Hace terminar prematuramente un 'for...loop' o '(....)'.

         : ejemplo 10 0 for i . i 4 > if leave endif loop ;

         imprime los números 0 1 2 3 4 5
```

CASE
---

El uso de 'case' es para evitar el uso de muchos 'if', cuando hace falta una tomar una decisión multi-camino utilizando el número que hay en lo alto

del stack.

```
            : test case
                  1 =of ." uno" endof
                  2 =of ." dos" endof
                  4 <of ." tres" endof
                  3 8 rng_of ." entre 3 y ocho" endof
                  100 >of ." mayor de 100" endof
                  not_of ." error" endof
                  endcase
                          ;
```

case..endcase {n --} Marca el comienzo y fin de las decisiones alternativas.

=of {n n1 -- flag} Compara n y n1. Si igual, las palabras entre '=of' y 'endof' son ejecutadas. En caso contrario, el control pasa más alla del 'endof'.

>of { n n1 -- flag} Compara n y n1. Si n es mayor que n1, se ejecutan las palabras entre '>of' y 'endof'.

<of { n n1 -- flag} Compara n y n1. Si n es menor que n1, se ejecutan las palabras entre '<of' y 'endof'.

rng_of { n n1 n2 -- flag} Compara si n está entre n1 y n2. Si lo está, se ejecutan las palabras entre 'rng_of' y 'endof'.

not_of { n -- } Cuando ninguna de las comparaciones anteriores ha sido verdadera, se ejecuta las palabras entre 'not_of' y 'endof'.

endof Marca el fin de una secuencia XXX_of....endof.

## COMPILANDO DESDE UN FICHERO

load_file Compila un programa escrito con un editor de textos y guardado en un fichero.

```
load_file demo_til
```

## Descripción del sistema

El sistema está distribuido por objetos. Cada uno se ocupa de una parte de la programación, y uno de los criterios de desarrollo es que sean lo más independientes entre sí que se pueda. Los objetos de alto nivel deben usar los de bajo nivel, pero nunca al revés.

Cada objeto tiene un identificador de tres letras y todas las palabras y variables globales de ese objeto comienzan con ese identificador.

A continuación describo cada objeto:

### Pantalla ( pan )

Este objeto permite controlar la apariencia completa de la pantalla.

La palabra 'pan_define' permite iniciar los valores por defecto necesarios. En esta versión se utiliza un parámetro, que puede ser PAN_TEXTO o PAN_GRAFICO, para indicar al PC si se desea tener la pantalla en modo texto o en modo gráfico. Esto permite incluir gráficos en el PC. Este debe tener tarjeta VGA. Si no se dispone de ella, el sistema arranca en modo texto.

La palabra 'pan_cierra' se utiliza normalmente al final de la aplicación para

restaurar todo y volver al sistema operativo.

Es perfectamente posible usar en una parte central del programa 'pan_cierra' y luego otra vez 'pan_define' para cambiar de modo, aunque el contenido de la pantalla se perderá.

En cualquier momento se puede consultar el modo en que se encuentra la pantalla mediante la palabra 'pan_modo'.

La pantalla es de 24 filas numeradas de 0 a 23 comenzando por arriba y 80 columnas numeradas de 0 a 79 empezando por la izquierda.

Se dispone de 4 colores, definidos con las constantes BLANCO, NEGRO, ROJO y VERDE. Se manejan con las palabras 'pan_papel', 'pan_tinta' y 'pan_color'. Estas palabras conviene utilizarlas antes de escribir algo, ya que los colores de papel y de tinta se cambian muy a menudo al llamar a otros objetos, de modo que no hay ninguna garantía de que permanezcan como los determine el programador.

La palabra 'pan_cursor' controla la posición del cursor.

La palabra 'pan_cursorvisible' determina si el cursor se ve en la pantalla o no. En el PC en modo gráfico no está disponible el cursor.

Las palabras 'pan_texto', 'pan_entero' y 'pan_caracter' permiten escribir en la pantalla texto (cadenas), números y caracteres.
Las palabras 'pan_pontexto', 'pan_ponentero' y 'pan_poncar' colocan el cursor en la posición indicada y luego escriben el texto, número o carácter.

Se puede activar o desactivar el modo de escritura resaltada con la palabra 'pan_resalta'. La situación normal es la de escritura resaltada desactivada, así que es importante conectarla cuando se desee usar e, inmediatamente, desconectarla. En el PC en modo gráfico no se dispone de esta posibilidad.

La palabra 'pan_limpia' borra la pantalla completa del color actual del papel.

La palabra 'pan_borralinea' borra una línea con el color indicado.

La palabra 'pan_borra' borra una zona rectangular con el color indicado.

## Zona ( zon )

Este objeto es el encargado de los gráficos. Sólo se puede utilizar cuando la pantalla está en modo gráfico. Por lo tanto, en un PC que no disponga de tarjeta gráfica VGA no se puede usar este objeto.

Hay un tipo de datos llamado "zona" que permite manejar las posibilidades del objeto.

Lo primero que hay que hacer es crear una zona mediante la palabra 'zon_crea'. Si no se puede crear, por falta de memoria o porque la pantalla no está en modo gráfico, esta función devuelve NIL.

A partir de ese momento, podemos acceder a cada pixel de la zona.

Las palabras 'zon_alto' y zon_ancho' dan las dimensiones de una zona en pixels. Obviamente, la misma zona no tendrá las mismas dimensiones en ordenadores distintos, de ahí la importancia de disponer de estas palabras.

Los pixels de la zona se numeran empezando en 0 de arriba hacia abajo y de izquierda a derecha, como es lo habitual.

Se puede dibujar cada pixel de la zona mediante dos palabras, 'zon_pixel' y 'zon_pixelseguro'. La diferencia entre ambas es que la primera comprueba que el pixel pedido realmente existe en esa zona (si no existe, no lo dibuja) y la segunda no. Por tanto, la primera es más lenta; pero para usar la segunda hay que estar seguro de que el pixel pedido pertenece a la zona.

Manejar una pantalla pixel a pixel es bastante lento, de modo que he buscado una

manera más rápida de manejar físicamente las zonas. El resultado de la búsqueda ha sido el manejo de cada "oxel" de la zona. Tanto la tarjeta VGA como la memoria gráfica del QL manejan los pixels agrupados de ocho en ocho. Pues bien, cada grupo de 8 pixels manejado globalmente por el hardware es lo que he denominado oxel. Siempre agrupa 8 pixels en línea, nunca en columnas.

Se puede saber cuántos oxels de ancho tiene una zona se usa la palabra 'zon_anchoenoxel'. La altura de una zona no se puede medir en oxels.

Para manejar cada oxel de una zona se usan las palabras 'zon_oxel' y 'zon_oxelseguro'. La diferencia entre ambas es la misma que para los pixels. Estas dos palabras necesitan conocer no sólo el color que hay que usar, sino también cuáles de los pixels del oxel se desean poner de ese color. Esto se indica mediante las "máscaras". Una máscara es simplemente un octeto, ocho bits. Cuando el bit es un 1, el pixel correspondiente cambia al color pedido; cuando es 0, no cambia. Por ejemplo, si deseamos cambiar de color el pixel más a la izquierda y el más a la derecha de un oxel, la máscara debe ser 10000001, que en decimal es 129, y en hexadecimal 0x81.

Cuando una zona ya no sea necesaria, hay que liberar la memoria que usa con la palabra 'zon_destruye'.

## Azar ( azr )

Se pueden obtener números enteros aleatorios en el margen que se desee. Basta iniciar el generador al principio con 'azr_inicia' y luego ir pidiendo números con 'azr_entero'.

## Tiempo ( tim )

La palabra 'tim_crono' devuelve una referencia temporal de precisión de tipo "tiempo". Si se restan dos valores devueltos por 'tim_crono', se obtiene el tiempo real transcurrido entre las dos llamadas.

La palabra 'tim_espera' obliga al programa a suspender su ejecución un periodo de tiempo determinado.

Se puede consultar el reloj del sistema, viendo el año, mes y día en curso.

## Cadena ( cad )

En esta implementación me he decidido por utilizar cadenas de asignación dinámica, por ser una aproximación más flexible que las cadenas de asignación estática.

Si declaramos una variable para guardar la dirección de una cadena, no se reserva automáticamente memoria para ella, sino que hay que pedirla explícitamente y luego liberarla. Muchas de las funciones definidas en este objeto y otros que lo utilizan realizan la reserva de memoria sin que el programador se tenga que ocupar.

Esto permite trabajar de dos formas distintas con las cadenas:

Por ser punteros, se pueden asignar a cadenas constantes. Por ejemplo, es válido:

```
variable CAD
: ejemplo    " Esto es una cadena" CAD !
                 { Aquí tu código }
             " Y esto es otra cadena" CAD !
                                          ;
```

Y por otro lado, se pueden crear cadenas de la longitud precisa en cada momento:

```
variable CAD
10 cad_crea CAD !
: ejemplo " 012345789" CAD @ cad_copia
            { Aquí tu código }
            CAD @ cad_destruye
                                ;
```

Las palabras más importantes son las que reservan memoria y la liberan.

'cad_crea' reserva memoria para una cadena de la longitud indicada (reservando un octeto más para el carácter NULL), le da el valor inicial de cadena vacía y devuelve la dirección de memoria reservada (que es un puntero a char). La memoria así reservada no se libera automáticamente al salir de la palabra en la que se reserva, de modo que es responsabilidad del programador destruirla cuando no sea necesaria.

'cad_destruye' es la encargada de liberar la memoria reservada mediante 'cad_crea'.

## Lista ( lis )

Las listas encadenadas son una de las estructuras más importantes en informática. MERINO TIL proporciona una implementación muy sencilla de las listas.

Esta es la estructura general de una lista:

```
  -------------          -------------                 -------------
  | Contenido |          | Contenido |                 | Contenido |
  |-----------|    /-->  |-----------|    /--> ... --> |-----------|
  | Siguiente | --/      | Siguiente | --/             | Siguiente | --> NIL
  -------------          -------------                 -------------
```

El tipo "lista" corresponde a estructuras como la de arriba. Con 'lis_crea' se obtiene una lista con un solo nodo que no tiene contenido y que tiene como siguiente nodo NIL, es decir: ninguno (o "tierra", como se dice a veces).

El "contenido" puede ser cualquier cosa, ya que lo que realmente se almacena es un puntero a una dirección de memoria; por ejemplo, se puede almacenar una cadena creada por 'cad_crea'. Para anotar en el nodo recién creado el contenido que se desee está la palabra 'lis_poncontenido'.

A continuación hay que ir añadiendo más nodos a la lista. Esto es más sencillo, porque las palabras 'lis_agrega' y 'lis_agregafin' sólo necesitan recibir la lista a la que hay que añadir el nodo y el contenido del nuevo nodo; ellas se encargar de crear el nuevo nodo, poner el contenido, añadir el nuevo nodo y devuelven la lista resultante.

Para usar la lista hay que ir viendo su contenido con 'lis_contenido' y navegar por la lista usando 'lis_siguiente'.

Cuando la lista no sea necesaria, se libera su memoria con 'lis_destruye', a la que basta darle el primer nodo, ya que ella va recorriendo toda la lista liberando la memoria reservada para todos los nodos.

## Sonido ( son )

Este objeto se encarga de las señales acústicas. Se pueden conectar con 'son_enciende', desconectar con 'son_apaga' o cambiar de estado con 'son_cambia'.

Las llamadas a las palabras de este objeto no provocan inmediatamente la generación del sonido, eso dependerá de si en ese momento está conectado o no.

## Tecla ( tec )

---

He definido 'tec_pulsada' de tal manera que podemos olvidarnos de las grandes diferencias en el tratamiento de las teclas en los diversos sistemas operativos. En sólo hay que saber que 'tec_pulsada' devuelve la tecla que haya pulsado el usuario. Junto con eso, hay toda una colección de constantes que definen cada tecla; por ejemplo, la tecla F1 está definida con la constante TEC_F1, y el programador no necesita conocer qué codigo (o códigos) asigna el sistema operativo a F1 (cada uno lo hace de una forma distinta). En esta versión no dispongo de macros para la ñ ni las vocales acentuadas.

Se puede consultar en cualquier momento cuál fue la última tecla pulsada por el usuario con 'tec_ultima', lo que resulta muy cómodo para obtener información en palabras muy alejadas de aquellas en las que el programa interactúa con el usuario.

Se puede pedir una tecla que pertenezca a un determinado rango con 'tec_validada' e incluso hacerlo con vuelta inmediata, si no hay tecla correcta disponible, con 'tec_validadarapido'.

Hay palabras para pasar a mayúsculas o minúsculas, pero en esta versión sólo para caracteres ASCII (hasta 127).

No he definido palabras para todas las teclas que son posibles en cada ordenador, sino sólo para aquellas que son perfectamente compatibles. En el PC existen algunas teclas que no están disponibles en el QL, pero con funciones que normalmente en el QL se asignan a determinadas combinaciones de tecla.

Esta es la lista de los macros que corresponden a distintas pulsaciones en PC y en QL:

| Constante | Tecla PC | Tecla QL |
|---|---|---|
| TEC_INICIO | Inicio | ALT/Cursor izquierda |
| TEC_FIN | Fin | ALT/Cursor derecha |
| TEC_AVPAG | AvPág | ALT/Cursor abajo |
| TEC_REPAG | RePág | ALT/Cursor arriba |
| TEC_CTRL_INICIO | CTRL/Inicio | CTRL/ALT/Cursor izquierda |
| TEC_CTRL_FIN | CTRL/Fin | CTRL/ALT/Cursor derecha |
| TEC_CTRL_AVPAG | CTRL/AvPág | CTRL/ALT/Cursor abajo |
| TEC_CTRL_REPAG | CTRL/RePág | CTRL/ALT/Cursor arriba |
| TEC_INSERTAR | Insert | MAY/CAPSLOCK |
| TEC_SUPRIMIR | Supr | CTRL/Cursor derecha |
| TEC_RETROCESO | Retroceso | CTRL/Cursor izquierda |

Y esta es la de teclas comunes:

TEC_ESC  TEC_ENTER  TEC_ESPACIO
TEC_ARRIBA  TEC_ABAJO  TEC_IZQUIERDA  TEC_DERECHA
TEC_TAB  TEC_MAY_TAB
TEC_A ... TEC_Z
TEC_MAY_A ... TEC_MAY_Z
TEC_0  TEC_1 ... TEC_9
TEC_CTRL_A ... TEC_CTRL_Z
TEC_ALT_A ... TEC_ALT_Z
TEC_ALT_0  TEC_ALT_1 ... TEC_ALT_9
TEC_F1  TEC_F2  TEC_F3  TEC_F4  TEC_F5
TEC_MAY_F1  TEC_MAY_F2  TEC_MAY_F3  TEC_MAY_F4  TEC_MAY_F5
TEC_CTRL_F1  TEC_CTRL_F2  TEC_CTRL_F3  TEC_CTRL_F4
TEC_ALT_F1  TEC_ALT_F2  TEC_ALT_F3  TEC_ALT_F4  TEC_ALT_F5

## Cuadro ( cdr )

Se pueden representar cuadros, cajas y líneas de cualquier color disponible y en dos grosores distintos.

El color se indica con 'cdr_color' y para representar los grosores de las líneas

se tienen las constantes CDR_SIMPLE y CDR_DOBLE.

## Usuario ( usr )

Las dos últimas filas de la pantalla están reservadas para interactuar con el usuario. Hay palabras para darle indicaciones, informarle de acciones y también para que edite números y texto (cadenas).

## Fichero ( fch )

La palabra 'fch_abreleer' permite abrir un fichero para leer datos de él, la palabra 'fch_abregrabar' abrirlo para grabar datos y la 'fch_abreactualizar' abrirlo para leer sus datos y modificarlos. Ambas cosas se pueden hacer en modo texto y en modo binario, y para indicarlo están las constantes FCH_TEXTO y FCH_BINARIO. Naturalmente, esta distinción es necesaria por culpa del MS-DOS; de hecho, en el QL no hay ninguna diferencia entre los dos modos.

Una vez abierto un fichero, se pueden leer y escribir líneas con 'fch_leelinea' y 'fch_escribelinea' y grupos de octetos con 'fch_leeocteto' y 'fch_escribeocteto'.

Se puede mover la posición de lectura y escritura en el fichero con las palabras 'fch_coloca' y 'fch_colocafinal'.

Las palabras se encargan de gestionar los errores que se puedan producir. Por ejemplo, al llamar a 'fch_borra', primero se pide confirmación al usuario, luego se comprueba la existencia del fichero y por último se emite un mensaje indicando si ha sido posible el borrado o no.

## Menú ( men )

Hay menús horizontales y verticales. Sólo hay que preocuparse de asignar la zona de la pantalla en la que se desea que aparezca el menú, el objeto se ocupa de la colocación. Las opciones pueden llevar una tecla caliente, un atajo o "hotkey", que permite elegir la opción simplemente pulsando esa tecla. Para indicar cuál es, se utiliza el carácter '>' antes de la letra en la llamada al objeto.

Sólo hay dos palabras para manejar menús: 'men_horizontal' y 'men_vertical'.

Si a un menú se le mandan más opciones de las que puede manejar, el exceso de opciones será simplemente ignorado.

## Programa ( prg )

Este objeto se ocupa simplemente de presentar información sobre el programa en la línea superior de la pantalla. Si no se utiliza, se dispone de una línea más para el desarrollo del programa.

La palabra que presenta esta información es 'prg_presenta'.

## Configuración ( cnf )

Este objeto permite leer información de un fichero filtrando todos los comentarios que se hayan escrito en él. El programador debe decidir después cómo lee la información entregada por el objeto, que lo hace en forma de cadena.

En los ficheros de configuración se considera línea de comentario toda aquella que tenga un '*' (asterisco) como primer carácter, aunque haya caracteres en blanco antes de él.

También se pueden poner comentarios dentro de una línea poniendo "//" (doble barra). Todo lo que haya a continuación, se considera comentario.

Las líneas en blanco se ignoran.

Un fichero se abre en modo configuración con la palabra 'cnf_abre', se va leyendo línea a línea con 'cnf_lee' y por fin se cierra con 'cnf_cierra'.

## Base de datos ( bdt )

Una base de datos se puede crear con la palabra 'bdt_crea' y abrir con 'bdt_crea'. Estas palabras devuelven un tipo "basedato", que es necesario usar en las futuras referencias a la base de datos.

Una vez creada o abierta es posible acceder a sus registros. Cada registro se mantiene en memoria, de modo que se pueden leer y cambiar sus campos.

También se puede examinar la estructura de la base de datos, viendo las características de cada campo.

Cuando se termine de usar una base de datos, se debe cerrar con 'bdt_cierra'. Esto es imprescindible, ya que si no se hace algunos datos no quedarán actualizados en el fichero correspondiente.

## Relación de palabras

```
abs           { n -- n }
          Calcula el valor absoluto de un número.
           -3 abs

azr_entero    { n1 n2 -- n3 }
          Obtiene un número aleatorio entre n1 y n2.
                    10 -10 azr_entero

azr_inicia    { -- }
          Inicia el generador con un número aleatorio.
            azr_inicia

bdt_activoregistro   { basedato -- Flag }
             Decir si el registro en memoria está activo (es decir:
             no está marcado como borrado)

             AGENDA @ bdt_activoregistro

bdt_actual   { basedato -- current_register }
             Decir el número de registro del registro actual de una base
             de datos.

             AGENDA @ bdt_actual

bdt_agregaregistro  { address.number basedato -- Flag }
             Agrega un registro desde memoria al fichero poniéndolo al
             final de éste.

             NUMERO AGENDA @ bdt_agregaregistro

bdt_anoultimo   { basedato -- year }
             Decir el año de la última modificación de una base de datos.

             AGENDA @ bdt_anoultimo


bdt_borraregistro  { number basedato -- flag }
             Marcar un registro como borrado.
```

```
                1 AGENDA @ bdt_borraregistro


bdt_camponombre  { address.string2 address.string1 basedato -- flag }
                Dar el contenido de un campo a partir de su nombre.

                AUX @ " EDAD" AGENDA @ bdt_camponombre

bdt_camponumero { address.string number basedato -- flag }
                Dar el contenido de un campo a partir de su número.

                AUX @  4 AGENDA @ bdt_camponumero

bdt_crea        { vector4 vector3 vector2 vector1 addres.string -- basedato }
                Crear un fichero de bases de datos y obtener un objeto en
                memoria para manejarlo.
                'address.string' contiene el nombre del fichero que hay que
                crear.
                'vector1' es un vector que contiene los nombres de los campos
                terminados en NIL.
                'vector2' es un vector de caracteres con los tipos de los campos.
                'vector3' es un vector de bytes con las longitudes de los campos
                'vector4' es un vector con la cantidad de decimales en cada
                campo.
                Tipos de campos:

                'C' -> Carácter.
                'D' -> Fecha (Longitud 8)
                'L' -> Lógico (Longitud 1)
                'M' -> memo (Longitud 10)
                'N' -> Númerico


bdt_decimaldecampo   { number basedato -- decimal }
                Decir el número de decimales de un campo a partir de su número.

                1 AGENDA @ bdt_decimaldecampo

bdt_diaultimo   { 'basedato' -- day }
                Decir el día de la última modificación de una base de datos.


bdt_leeregistro  { number basedato -- Flag }
                Leer un registro desde el fichero a memoria.

bdt_limpiaregistro  { 'basedato' -- }
                Limpiar registro en memoria.

bdt_longituddecampo   { number basedato  -- length  }
                Decir la longitud de un campo a partir de su número.

bdt_mesultimo   { basedato -- month }
                Decir el mes de la última modificación de una base de datos.

bdt_nombredecampo   { address.string number basedato -- Flag }
                Decir el nombre de un campo a partir de su número.

bdt_numerodecampo  { address.string address.number basedato -- flag }
                Hallar el número de un campo a partir de su nombre.

bdt_poncamponombre  { address.string2 address.string1 basedato -- flag }
                Poner el contenido de un campo dado su nombre.
                Pone el contenido de 'address.string2' en el campo
                'address.string'.

                 " 35" " EDAD" AGENDA @ bdt_poncamponombre

bdt_poncamponumero  { address.string number basedato -- flag }
                Poner el contenido de un campo dado su número.
```

```
                 " 35" 4 AGENDA @ bdt_poncamponumero

bdt_recuperaregistro  { number basedato -- flag }
             Recuperar un registro que estaba marcado como borrado.

bdt_tipodecampo   { number basedato -- type }
             Decir el tipo de un campo a partir de su número.

bdt_totalcampo    { basedato -- number }
             Decir el número de campos de una base de datos.

bdt_totalregistro  {  basedato -- number }
             Decir el número de registros de una base de datos.

bdt_abre        { mode address.string -- basedato }
             Abre una base de datos.
             'mode' es un número (1 = actualiza, 0 = lee).

bdt_cierra      { basedato -- }
              Cierra una base de datos.



cad_cambia    { char1 char2 addr -- addr }
          Cambia carácter 'char2' por carácter 'char1' en la cadena 'addr'

             TEC_A TEC_B CADENA @ cad_cambia

cad_carpertenece   { char addr -- n }
          Averiguar si un carácter pertenece a una cadena, y dejar la
          posición que ocupa el carácter en la cadena, o 0 (se empieza a
          contar desde 1).

            TEC_A NOMBRE @ cad_carpertenece

cad_copia           { addr1 addr2 -- addr }
          Copia la cadena 'addr1' en la cadena 'addr2'.

            " Eustaquio" NOMBRE @ cad_copia

cad_crea            { n -- addr }
          Reserva espacio para almacenar una cadena.

            80 cad_crea

cad_destruye         { addr -- }
          Libera la memoria reservada para una cadena.

            CADENA @ cad_destruye

cad_entero        { n -- addr }
          Convierte un número en una cadena (la nueva cadena debe ser
          destruida cuando no sea necesaria).

            130 cad_entero

cad_igual       { addr1 addr2 -- flag }
           Compara si dos cadenas son iguales.


cad_longitud    { addr -- n }
           Calcula la longitud de una cadena.

            NOMBRE @ cad_longitud

cad_mueve        { max position addr -- addr }
          Mueve hacia la derecha todos los caracteres de una cadena desde
          una posición determinada 'position', cuidando que la longitud
          no exceda de cierto máximo 'max' (la primera posición es 0).
```

```
            40 2 CADENA @ cad_mueve

cad_primerutil      { addr -- n }
          Dice el primer carácter de una cadena que no sea ni blanco, ni
          tabulador, ni retorno de carro ni nueva linea (se empieza a contar
          desde 0).

          "   Ahora" cad_primerutil

cad_quitacar         { n addr -- addr }
          Quita el carácter que ocupa una determinada posición 'n' en una
          cadena 'addr' (la primera posición es 0).

            2 CADENA @ cad_quitacaracter

cad_subcadena         { addr2 addr1 -- n }
          Averigua si la segunda cadena 'addr2' está contenida en la
          primera 'addr1'. Si es verdadero devuelve la posición, y si es
          falso 0 (se empieza a contar desde 1).

            " mar" " Calamares" cad_subcadena

cad_trozo           { n2 n1 addr -- addr }
          Extrae una cadena  desde la posición 'n1' a 'n2' (la primera
          posición es 1, y la cadena devuelta hay que destruirla cuando
          no sea necesaria).

            7 3 NOMBRE @ cad_trozo

cad_une             { NIL addr..... --- addr }
          Une varias cadenas terminadas con NIL (Hay que destruir la nueva
          cadena cuando no sea necesaria).

            NIL " de 1808" " Veinte de" cad_une

cdr_caja            { x2 y2 x y CDR -- }
          Dibuja una caja. Sus entradas son el tipo de caja 'CDR" y sus
          coordenadas 'x2 y2 x y'.

            70 10 1 1 CDR_DOBLE cdr_caja

cdr_color           { n -- }
          Establece el color con el que se dibujarán todas las figuras.

            ROJO cdr_color

cdr_dibuja  { x y number_columns  high_rows number_rows  CDR CDR -- }
          Dibuja un cuadrado. Las entradas son el tipo de líneas externas
          'CDR', el de las internas 'CDR', el número de filas
          'number_rows', el alto de cada una 'high_rows', el número de
          columnas "number_columns', la fila 'y' y columna 'x' de la
          esquina superior izquierda.

            3 3 1 4 3 2 CDR_SIMPLE CDR_DOBLE cdr_dibuja

cdr_linea           { x2 y2 x y CDR -- }
          Dibuja una linea. Las entradas son el tipo de línea 'CDR', y las
          coordenadas 'x2 y2 x y'.

            71 2 3 2 CDR_DOBLE cdr_linea

cnf_abre            { addr -- addr }
          Abre un fichero de configuración. La entrada es el nombre de
          fichero. Retorna fichero abierto o NULO si ha habido error.

            " Datos_dat" cnf_abre

cnf_cierra          { addr -- n }
          Cierra un fichero de configuración. Retorna 0 si todo va bien,
          EOF si hay error.
```

```
            FICHEROCONFIG @ cnf_cierra

cnf_lee             { addr -- addr }
          Obtiene la siguiente línea de un fichero de configuración. Deja
          la cadena si no hay error, o NIL (la cadena no contiene el fin
          de linea, y debe ser destruida cuando no sea necesaria).

           ENTRADA @ cnf_lee

fch_abre           { addr2 addr1 -- addr }
          Abre un fichero. Las entradas son el nombre del fichero 'addr1'
          y el tipo 'addr2'. La salida es un fichero abierto o NULO si ha
          habido algún error.

            " r" " Datos_dat" fch_abre

fch_abreactualizar  { n addr -- addr }
          Abre un fichero para actualizarlo. Las entradas son el nombre
          del fichero 'addr' y el modo 'n'. La salida es un fichero
          abierto o NULO si ha habido algún error.

           FCH_BINARIO " Datos.dat" fch_abreactualizar

fch_abregrabar      { n addr -- addr }
          Abre un fichero para grabar en él. Las entradas son el nombre
          del fichero 'addr' y el modo 'n'. La salida es un fichero
          abierto  o NULO si ha habido algún error.

            FCH_BINARIO " Datos_bin" fch_abregrabar

fch_abreleer        { n addr -- addr }
          Abre un fichero para leerlo. Las entradas son el nombre del
          fichero 'addr' y el modo 'n'. La salida es un fichero abierto
          o NULO si ha habido algún error.

            FCH_TEXTO " Datos_dat" fch_abreleer

fch_borra           { addr -- }
          Borra un fichero. La entrada es el nombre del fichero.

          " Viejo_txt" fch_borra

fch_cierra    { addr -- error }
          Cierra un fichero. La entrada es el fichero. La salida es
          cero si todo va bien y EOF si hay error.

                  SALIDA @ fch_cierra


fch_coloca         { n addr -- n}
          Coloca el puntero interno de un fichero en una determinada
          posición. Las entradas son el fichero 'addr' y la posición 'n'.
          La salida es 0 si todo va bien y no cero si hay error.

                 50 DATO @ fch_coloca

fch_colocafinal    { addr -- n }
          Coloca el puntero interno de un fichero al final. La entrada es
          el fichero. La salida es 0 si todo va bien y no cero si hay error.

                  DATO @ fch_colocafinal

fch_escribelinea    { addr1 addr2 -- addr }
          Escribe una linea 'addr1' en un fichero 'addr2'. Si todo va
          bien devuelve 0, y no cero si hay error.

          LINEA @ SALIDA @ fch_escribelinea

fch_escribeocteto   { n addr_string addr_file -- addr }
```

Escribe en un fichero 'addr_file' cierta cantidad de octetos 'n'
de una cadena 'addr_string'. Si todo va bien retorna el fichero,
o NIL si hay error.

1024 ITEM @ SALIDA @ fch_escribeocteto

fch_existe { addr -- flag }
Informa si un fichero existe.

" Datos_dat" fch_existe

fch_leelinea { addr -- addr }
Obtiene la siguiente linea del fichero. La entrada es el fichero.
Si no hay error devyelve una cadena, o NIL si hay error. La cadena
devuelta no contiene fin de linea, y debe ser destruida cuando no
sea necesaria.

ENTRADA @ fch_leelinea

fch_leeocteto { n addr file -- file/NIL }
Lee de un fichero cierta cantidad de octetos. Las entradas son
el fichero 'file', zona en la memoria donde dejar lo leido 'addr'
y la cantidad de octetos 'n'. Si no hay error devuelve el fichero,
y NIL si hay.

1024 INFO @ AGENDA @ fch_leeocteto

fch_nombre { addr2 addr1 -- addr }
Forma un nombre completo de un fichero uniendo el nombre el
separador y la extensión. Las entradas son el nombre 'addr1' y
la extensión 'addr2'. Devuelve el nombre completo.

" txt" " Datos" fch_nombre

max { n2 n1 -- n }
Calcula el máximo de dos números.

5 2 max

men_horizontal { option_initial addr_pointer_list x2 x y -- option}
Presenta al usuario un menú horizontal para que pueda elegir
entre varias opciones. mediante las teclas del cursor o
mediante un atajo (tecla caliente). Las entradas son la fila
donde se muestra el menú 'y', las columnas entre donde se
encuentra 'x2 x', un vector de cadenas 'addr_pointer_list'
terminando en NIL, en el que se señalan los atajos, y la opción
que hay que resaltar en primer lugar 'option_initial'.
Devuelve un número indicando la opción elegida, empezando a
contar en 1, o 0 si no se elige ninguna.

2 0 MENU 78 1 1 men_horizontal

men_vertical { option_initial addr_pointer_list x2 y2 x y -- option }
Presenta al usuario un menú vertical para que pueda elegir entre
varias opciones, mediante las teclas del cursor o mediante un
atajo (tecla caliente). Las entradas son las coordenadas de la
esquina izquierda 'x y' e inferior derecha 'x2 y2' de la zona
asignada al menú, un vector de cadenas 'addr_pointer_list'
terminado en NIL, en el que se señalan los atajos y la opción
que hay que resaltar en primer lugar 'option'.
Devuelve un número indicando la opción elegida, empezando a
contar en 1, o 0 si no se elige ninguna.

2 0 MENU 20 8 10 1 men_vertical

min { n2 n1 -- n }
Calcula el minimo de dos números.

5 2 min

```
pan_borra          { x2 y2 x y colour -- }
            Borra una zona de la pantalla. Las entradas son el color y
            las coordenadas.

               15 15 3 3 ROJO  pan_borra

pan_borralinea     { n -- }
            Limpia una linea determinada.

              0 pan_borralinea

pan_caracter        { n -- }
            Escribe un carácter en la pantalla.

              TEC_A pan_caracter

pan_cierra       { -- }
            Deja la pantalla preparada para volver al sistema operativo.


pan_color           { ink paper -- }
            Cambia el color del papel y la tinta.

               NEGRO VERDE pan_color

pan_cursor          { x y -- }
            Coloca al cursor en una posición determinada.

              0 0 pan_cursor

pan_cursorvisible   { flag -- }
            Poner y quitar el cursor.

              SI pan_cursorvisible

pan_define          { flag -- }
            Define la pantalla principal. La entrada es el modo que se
            va a usar.

              PAN_TEXTO pan_define

pan_entero          { length number -- }
            Escribe un número en la pantalla de un ancho determinado.

                6 12345 pan_entero

pan_limpia          { -- }
            Borra la pantalla completa.

               pan_limpia

pan_modo          { -- modo }
            Dice el modo en que está definida la pantalla.


pan_papel           { n -- }
            Cambia el color de la pantalla.

             VERDE pan_papel

pan_poncar          { char x y -- }
            Escribe un carácter en cierta posición.

              TEC_A 5 3 pan_poncar

pan_ponentero      { length number x y -- }
            Escribe un entero en cierta posición.

               6 12345 5 3 pan_ponentero
```

```
pan_pontexto       { addr x y -- }
            Escribe un texto en cierta posición.

             " Hola" 5 3 pan_pontexto

pan_resalta        { flag -- }
            Pone y quita la situación de resaltado de los caracteres.

                SI pan_resalta

pan_texto          { addr -- }
            Escribe un texto en la pantalla.

                " Hola, usuario" pan_texto

pan_tinta         { ink -- }
            Cambia el color con el que se escribe en la pantalla.

                ROJO pan_tinta

prg_presenta      { date author version name  -- }
            Presenta el programa.

             " 1994" " Salvador Merino" " 2.0" " MERINO TIL" prg_presenta

son_apaga         { -- }
            Apaga el sonido.

              son_apaga

son_bien          { -- n }
            Señala brevemente que algo es correcto.

             son_bien

son_cambia       { -- n }
            Cambia el estado del sonido (encendido/apagado).

                son_cambia

son_enciende      { --  }
            Encender el sonido.

               son_enciende

son_error         { --  }
            Señala un error.

               son_error

son_malatecla     { --  }
            Señala que la tecla pulsada no se admite.

               son_malatecla

tec_disponible     { -- flag }
            Decir si hay alguna tecla disponible.

              tec_disponible

tec_fijadomayus    { -- flag }
            Decir si está fijado el sujeta-mayúsculas.

              tec_fijadomayus

tec_mayus           { n -- n }
            Convertir una tecla en mayúscula.

              TEC_A tec_mayus
```

```
tec_minus          { n -- n }
            Convertir una tecla en minúscula.

             TEC_MAY_A tec_minus

tec_pertenece       { addr_pointer n -- n }
            Decidir si una tecla pertenece a cierto rango. Las  entradas
            son una tecla 'n' y un vector con las teclas admitidas
            terminado en NIL.
            Devuelve la tecla si pertenece, o NIL si no pertenece.

              0 TECLAS  TEC_F1 tec_pertenece

tec_pulsada         { -- n }
            Esperar a que el usuario pulse una tecla y devolver su código.

                   Tec_Pulsada

tec_ultima         { -- n }
            Devolver la última tecla pulsada por el usuario.

                   tec_ultima

tec_validada       { addr_pointer -- n }
            Devolver una tecla de un determinado rango. La entrada es un
            vector con las teclas admitidas terminado en NIL.

                 0 TECLAS tec_validada

Tec_validadarapido  { addr_pointer -- n }
            Devolver una tecla de un determinado rango o NIL si no hay
            disponible ninguna tecla correcta. La entrada es un vector con
            las teclas admitidas terminado en NIL.

                0 TECLAS tec_validadarapido

tim_ano        { -- año }
            Decir el año en curso.

tim_crono           { -- seconds}
          Devuelve el número de segundos del sistema.

tim_dia         { -- dia }
          Devuelve el día en curso.

tim_mes        { -- mes}
          Devuelve el mes.


tim_espera         { addr -- }
          Suspende la ejecución durante un cierto tiempo. La entrada es
          una cadena.

          " 0.1" tim_espera

usr_avisa           { addr -- }
          Avisar al usuario de algo.

          " Operación incorrecta" usr_avisa

usr_borrazona       { colour -- }
          Borra una zona de interacción con el usuario.

          NEGRO usr_borrazona


usr_consulta        { addr -- flag }
           Pregunta algo al usuario. Devuelve verdadero o falso, según
           contestación del usuario.
```

```
            " ¿Quieres seguir?" usr_consulta

usr_entero          {  ink paper x y max min length number -- number }
           Devuelve un número determinado por el usuario después de editar
           el que se le ofrece. Las entradas son el número que hay que
           editar 'number', el ancho asignado 'lenght', los valores mínimo
           'min' y máximo 'max' admitidos, las coordenadas donde se edita
           'x y' y los colores 'ink paper'.

               NEGRO BLANCO 1 1 2000 -900 4 100 usr_entero

usr_indica         { addr addr -- }
           Mandar un mensaje al usuario.

            " Pulsa ENTER" " Elige una opción" usr_indica

usr_informa         { addr -- }
           Mandar un mensaje al usuario.

             " Calculando" usr_informa

usr_pulsaunatecla   { addr -- }
           Mandar un mensaje al usuario y esperar a que pulse una tecla.

             " Operación terminada..." usr_pulsaunatecla

usr_texto          { ink paper x y length addr_string -- addr }
           Devolver una cadena determinada por el usuario después de editar
           la que se le ofrece. Las entradas son la cadena que hay que
           editar  'addr_string', el ancho asignado 'length', las coordenadas
           donde se edita 'x y' y los colores 'ink paper'.
           La cadena devuelta hay que destruirla cuando no sea necesaria.

             NEGRO BLANCO 1 1 20 " Artesonado" usr_texto


fdb_edita_tabla { -- }
             Invoca al editor de lista de definiciones (.def).

fdb_menu_ficheros { -- name_file_addr }
          Selecciona un fichero con las flechas arriba y abajo y enter.

fdb_inicio  {  name_file_addr -- struct }
          Abre un fichero. Deja fichero abierto  o cero si no pudo abrirlo.


fdb_visualizar { struct_file -- }
           Escribe en la pantalla todos los campos del registro actual.

fdb_insercion  { struct_file -- }
           Inserta un nuevo registro.

fdb_modificacion { struct_file -- }
           Modifica el registro actual.


fdb_borrar       { struct_file -- }
           Borra el registro actual.

fdb_leer         { struct_file -- }
           Lee el registro actual.

fdb_situacion_contenido  { struct_file -- }
           Lee el registro que contiene el contenido que apunta
           la variable clave.

fdb_registro_actual  { struct -- addr }
           Deja la dirección de la variable del registro actual.

fdb_total_registros  { struct -- total }
```

```
            Deja el total de registros.

fdb_clave   { struct -- addr }
            Deja la dirección de la variable clave.

fdb_mensaje_registro  { x y struct  -- }

            Guarda las coordenada de la cadena "REGISTER:    of   "

fdb_campo     { number struct --- addr }
            Deja la dirección de la variable número de campos.

fdb_impresora    {  struct pins --- }
            Volcado del registro actual a impresoras de 8 o 24 agujas
            compatibles EPSON.

fdb_dum_graf   { Field pins FILE struct -- }
            Volcado de un campo gráfico a un dispositivo o fichero.

pan_punto      { parm color x y --- }
             Dibuja un punto en pixel 'x'y'.
             parm = 1 XOR
             parm = 0 COPY


pan_getpixel   { y x --- color }
              Obtiene el color de un pixel.

pan_getimage    { address_$variable y1 x1 y x -- }
              Obtiene la imagen en las coordenadas 'y1 x1 y x'.

pan_putimage    { parm address_$variable y x -- }
              Pone una imagen en las coordenadas 'y x'
              parm = 1 :  XOR
              parm = 0 :  COPY

cad_makeshape { address_SHAPE address_PRESHAPE height width -- numbytes }
              Hace un modelado y retorna el número de bytes.


fch_poncar   { char FILE -- }
              Envia un carácter a un fichero.

fch_cogecar  { FILE -- char }
              Obtiene un carácter de un fichero.

fch_eof      { FILE -- flag }
              Obtiene EOF de un fichero.

fch_pon      { address FILE --- }
              Envia una cadena al fichero.

fch_coge     { n FILE -- address }
              Obtiene una cadena de un fichero de 'n' bytes o terminada con
              fin de línea.


lis_agrega      { address.string list -- list }
              Añadir un elemento a la lista.

lis_agregafin   { address.string list -- list }
              Añadir un elemento a una lista por el final

lis_contenido   { list -- address.1st }
              Dar el contenido de un elemento de una lista.

lis_crea        { -- list }
              Crea una lista.

lis_destruye    { list -- }
```

```
                Destruir una lista completa, incluyendo los contenidos.

lis_poncontenido   { address.string1 list -- address.string2 }
                Poner el contenido de un elemento de una lista. Las entradas
                son una lista 'list' y el contenido que se va a poner
                'address.string1', que debe ser una dirección de memoria
                reservada.
                Devuelve el contenido.

lis_ponsiguiente { list2 list1 -- list }
                Poner el siguiente elemento de una lista.  Las entradas son una
                lista 'list1' y la lista que se va a poner como siguiente
                'list2'.
                Devuelve la lista siguiente.

lis_siguiente    { list -- list2 }
                Dar el siguiente elemento de una lista.

lis_total       { list -- number }
                Calcular el número de elementos de una lista.

zon_altofisico   { zone -- heigth }
                Decir la altura en pixels de una zona.

zon_anchoenoxel  { zone -- length }
                Decir la anchura en oxels de una zona.

zon_anchofisico  { zone -- length }
                Decir la anchura en pixels de una zona.

zon_borra      { colour zone -- }
                Borrar una zona con un color.

zon_crea       { right  inf  lefth sup  -- zone }
                Crea una zona.

zon_destruye   { zone -- }
                Destruye una zona.

zon_oxel       { colour mask y x zone -- flag }
                Dibujar un oxel en una zona de un color.

zon_oxelseguro  { colour mask y x zone -- }
                Dibujar un oxel en una zona de un color.

zon_pixel       { colour mask y x zone -- flag }
                Dibujar un pixel en una zona de un color.

zon_pixelseguro  { colour mask y x zone -- }
                Dibujar un pixel en una zona de un color.


.           { n -- } Imprime el número en lo alto del stack.

s?          { n -- n } Imprime el número en lo alto del stack, pero no  lo
            destruye.

.s          { n1 n2 n3 -- n1 n2 n3 } Imprime en la pantalla todo el contenido
            del stack, pero no lo destruye.

bye         Destruye MERINO TIL y vuelve al sistema operativo.
```

QLIPER, El Club de Usuarios QL Español
---

La asociación C.E.I.U.Q.L. (Club Español Independiente Usuarios QL) fue fundada en noviembre de 1985 en Zaragoza por Serafín Olcoz y su grupo de amigos que le animaron a dar ese gran paso. Sin embargo, los estatutos del Club no fueron aprobados por el Ministerio del Interior y el Gobierno Civil hasta julio de 1986.

Los primeros QLs versión JM llegaron a España en diciembre de 1984. El QL MGE se vió por primera vez en las tiendas en septiembre/octubre de 1985. Cuando en Marzo de 1986 AMSTRAD compra SINCLAIR RESEARCH ltd, INVESTRONICA (el distribuidor SINCLAIR en España), que habia promocionado el QL junto colegios de arquitectos y desarrollado software para los mismos fines (Redacción de proyectos, Cálculo de estructuras e instalaciones para edificación), decide liquidar su stocks de QLs lo más rápidamente posible, y no importar nada en absoluto de hardware y software para el QL. Durante ese año y medio de vida comercial del QL se vendieron en España alrededor de 10.000 QLs.

La primera revista de C.E.I.U.Q.L. fue QLAVE. Estaba impresa en papel en un tamaño de 21x16 cm. Su contenido normal era: Editorial, Cartas abiertas, Actualidad del mercado, Preguntas y Respuestas, Ofertas (anuncios gratuitos a empresas comerciales), Comentarios de Software y Hardware, Trucos y Rutinas,.... Desde enero de 1986 a diciembre de 1988 se publicaron 26 revistas QLAVE haciendo un total de 972 páginas.

En noviembre de 1986 nacen los primeros Grupos Locales (Zaragoza, Valencia y Sevilla). A principios de 1987 nace el Grupo Local de Madrid, y a finales de ese año nacen los Grupos Locales de Málaga, Albacete, Alicante, Galicia, Mallorca, Navarra y Valladolid. La misión de los Grupos Locales era celebrar reuniones y descargar de trabajo al librero del Club en la realización de copias de programas escritos por los socios.

El número máximo de socios que tuvo C.E.I.U.Q.L. en sus mejores tiempos (finales de 1987) fue de 300 socios. Sin embargo, C.E.I.U.Q.L. tenia graves problemas económicos y de organización. Habia un grave agujero negro en la contabilidad del Club por donde desaparecian grandes cantidades de dinero, los discos/mdvs con colaboraciones se perdian en la redacción, la libreria no servia los pedidos de programas,.... Ante tan graves problemas y dimisión de sus colaboradores más intimos, Serafín Olcoz decide dimitir el 14 de abril de 1988. El Club cae en manos de Diego Alcalá, que prácticamente se nombra el mismo Presidente, Vicepresidente, Secretario, Tesorero y Librero. En otras palabras, este señor viola los estatutos del Club, pues se ha hecho con todo el poder directivo del Club sin que nadie lo hubiese votado (fue algo parecido a un Golpe de Club). Diego Alcalá tenia en sus manos la mayor recaudación de dinero de todos los tiempos del Club, y solamente fue capaz de lanzar una sola revista QLAVE, la cual habia sido compilada por Serafín Olcoz antes de irse, y una segunda revista seis meses después, que era la primera revista CUQ en disco que salió 4 meses antes, para pedir más dinero a los socios.

Durante el verano de 1988, y viendo que QLAVE va a desaparecer, Salvador Merino intenta convencer a Serafín Olcoz para crear un nuevo Club de Usuarios 68000 mixto (QL, ATARI ST, COMMODORE AMIGA y APPLE MAC), pero la idea no pudo hacerse realidad, porque a diferencia de QLAVE que era un Club Cultural, el nuevo Club seria lucrativo.

En septiembre de 1988 nace CUQ (Circulo usuarios QL) lanzando una revista en disco mensual que sustituia a la revista QLAVE, la cual no salia nunca a la calle y los socios de C.E.I.U.Q.L. estabamos desesperados por continuar en comunicación. La verdad era que aquellos dias eran muy malos (El FUTURA/QLT de Sandy/Tony Tebby se sabia que nunca iba a salir al mercado, la saga THOR estaba en visperas de defunción ante la desaparición del mercado de CST, y el emulador QL para ST se comentaba que dejaria de comercializarse por agotamiento de la ULA 8301).

Obtener la revista CUQ era bastante inédito. El usuario QL solamente tenia que enviar un disco con su colaboración (o sin colaboración) y un sobre franqueado para su retorno. Este sistema fue el que se utilizó

durante 2 años y 3 meses, e hizo uso de él alrededor de 60 usuarios QL. Las ventajas que tenia el sistema eran:

- El usuario arriesgaba muy poco.
- No habia que llevar ningún tipo de contabilidad.

Las desventajas:

- El intercambio se reducia a usuarios con discos. No es que los usuarios de microdrives no pudiesen participar (hubo algunos que participaron un poco), sino que el material era tan abundante y los cartuchos de microdrives tan escasos y tan caros que no podian permitirse el lujo de participar.
- Muy pocos usuarios participaban cumpliendo las fechas correctas. La mayoria lo hacian cuando le picaba el gusanillo, y hacian sus pedidos de atrasados a la carta.

En 1991 aprovechando que los discos ya tienen un precio bastante bajo, se decide poner una cuota anual de 1.500 ptas y poner la revista en disco 3.5" 720 Kbytes lleno con salida bi-mensual. También ese mismo año, Salvador Merino decide invertir todo el dinero sobrante de 1991 en participar y adquirir la libreria de software de la organización IFE.

En 1992, Salvador Merino pasa el relevo a Marcos Cruz en las tareas de Editor y Tesorero. También se decide cambiar el nombre de la revista por QLIPER por considerarlo mucho más adecuado.

Actualmente Salvador Merino vuelve a ser Editor, Tesorero y Librero de QLIPER, porque Marcos Cruz tiene problemas de tiempo debido a su trabajo. El Club solamente tiene 17 socios (1993). El tesoro del Club se encuentra muy saneado y podemos permitirnos algunos lujos en 1994. Nuestra libreria dispone del 85% del software de dominio público y ShareWare para QL disponible en todo el mundo. La producción de software del Club va a toda máquina siendo dos nuestras joyas más recientes: el Sistema OLIMPO (una libreria 'C' multiplataforma para QL y PC) y MERINO TIL (un lenguaje derivado del FORTH, basado en el sistema OLIMPO, y disponible para QL y PC).

Los equipos que utiliza QLíper son los mismos que dispone Salvador Merino: QL-TRUMP CARD (QDOS MGE y dos discos 3.5" DD), QL-GOLD CARD (MINERVA v1.97 y dos discos 3.5" ED) y un PC 486 con QXL.

18 de enero de 1994
Salvador Merino
Ctra Cádiz, Cerámicas Mary
29640 Torreblanca del Sol
Fuengirola (Málaga)
SPAIN.