TV3
TELEVISIÓ DE CATALUNYA

connecta el micro

1 FEM INFORMÀTICA

FUNDACIÓ CAIXA DE PENSIONS

connecta el micro

1 FEM INFORMÀTICA

FUNDACIÓ CAIXA DE PENSIONS

Edita:
FUNDACIÓ CAIXA DE PENSIONS

President Executiu de la Fundació
Caixa de Pensions:
JOSEP VILARASAU I SALAT

Vice-President Executiu:
RICARD FORNESA I RIBÓ

Director Executiu de la Fundació
Caixa de Pensions:
JOAN JOSEP CUESTA I TORRES

Cap de Control de l'Obra Social de la Caixa de Pensions:
JOSEP MARIA ARENAS I PASCUAL

Coordinadors de l'edició:
JORDI SALA
ALBERT SORIA

Autors:
MERCÈ GRIERA FISA*, LLORENÇ HUGUET ROTGER*,
DAMIÀ CASAS PESSAFERRER*, FRANCESC FRANCO JACOBO
*U.A.B. Departament d'Informàtica

Dibuix:
ROGER

Fotografia:
ESPARBÉ

Disseny gràfic i maquetació:
EQUIP 30/53

Gestió edició:
MUNDO CIENTÍFICO

Primera edició: maig 1985

Impressió:
TONSA, Herrera-Alza. Donostia

ISBN: 84-505-1475-4
Dipòsit legal: SS. 288-85

Presentació

Dia a dia la tecnologia informàtica, les aplicacions de l'ordinador, es van fent més presents en una gran diversitat dels nostres actes quotidians. Hi ha una bona colla de pautes de comportament personal i social que ja hem variat en funció d'aquesta presència, i fins i tot es diu que vivim immersos en «la cultura informàtica».

Socialment s'ha desvellat, de forma força majoritària, la consciència del fet que cal fer atenció a la informàtica, i es planteja la necessitat de comprendre-la per a poder-se'n servir.

Aquest llibre que ara enceteu, vol facilitar-vos, de manera planera i eficaç, l'accés a l'eina. I ho fa des d'una perspectiva que pretén ser àmplia, perquè entrar en el món de la informàtica no és únicament aprendre un llenguatge (el BASIC en aquest cas).

Voldríem poder facilitar coneixements i criteri per a saber servir-se d'aquesta nova tecnologia; una tecnologia que ens ha de permetre augmentar la nostra capacitat de creació, d'estudi, d'accedir a la documentació..., de donar resposta a les noves exigències.

L'Obra Social de la Caixa de Pensions entén que cal considerar el fenomen informàtic com un bé cultural i intenta, en la mesura de les seves possibilitats, contribuir a la seva difusió, igual com fa en tants d'altres àmbits culturals a través dels seus Centres i programes. L'edició d'aquests materials, complementaris del programa de televisió que s'ha realitzat conjuntament amb TV3, és una part del global d'accions que ja es realitzen –i que es pretén desenvolupar encara més–, principalment dirigides a la gent jove.

Confiem d'aquesta manera col·laborar a la millora dels recursos personals d'un bon nombre de ciutadans, que creiem que estaran en millors condicions d'avançar personalment i col·lectivament en el seu àmbit d'actuació.

Fundació Caixa de Pensions

Ni bo ni dolent

A cada època hi ha invents nous que revolucionen la manera de viure i fins i tot de pensar de la gent.
Us imagineu com era el món abans d'ésser inventada l'electricitat?
Hi ha una llista inacabable de coses que els nostres avantpassats no podien fer:

- Ni mirar la televisió
- Ni escoltar la ràdio
- Ni llegir a la nit, abans d'adormir-se
- Ni tenir les begudes fresques a l'estiu
- Ni fer una fotocòpia
- Ni jugar amb el tren elèctric.
- .....

I d'altres que, tot i que les podien fer, els eren una feinada, com ara:

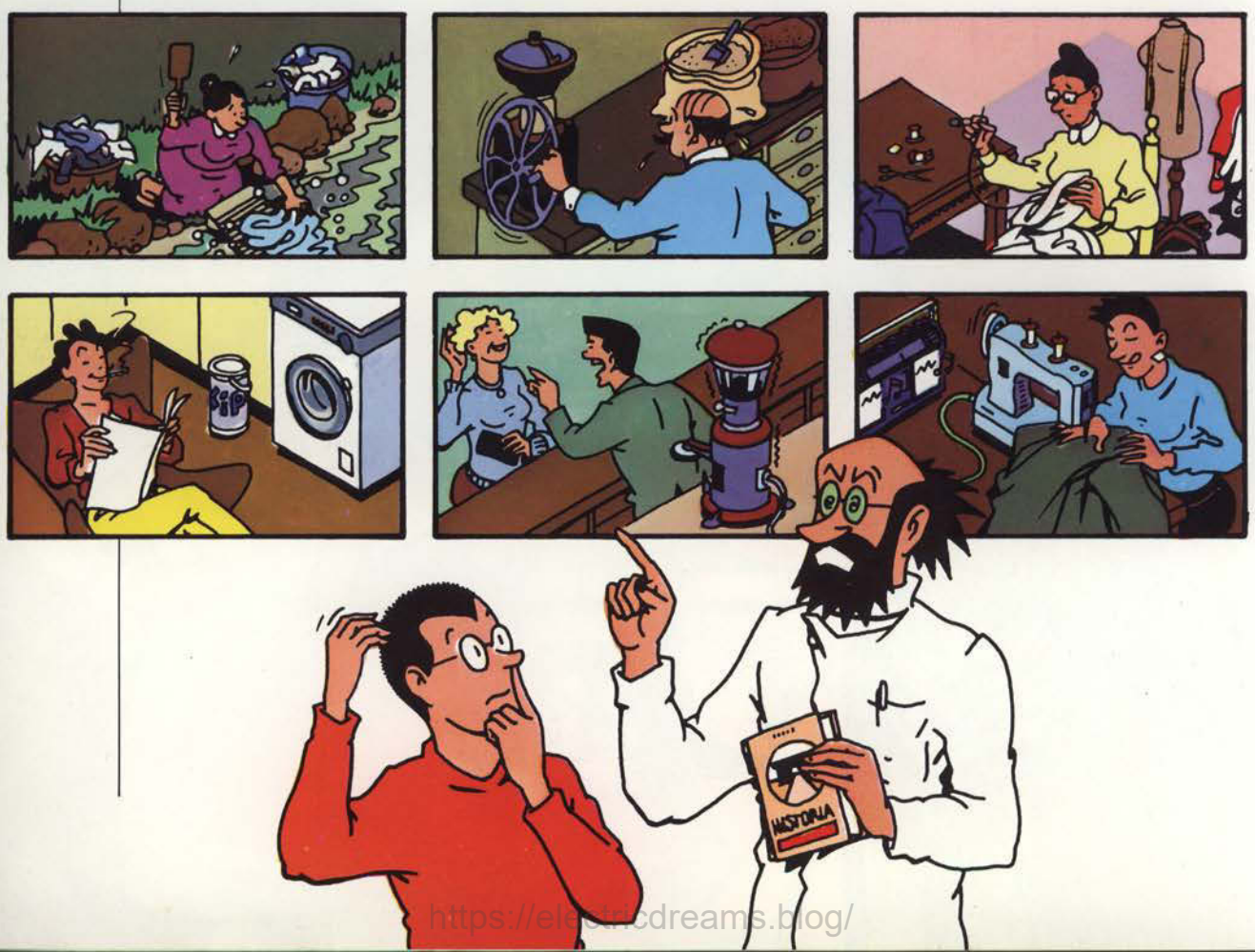
- Rentar la roba
- Moldre el cafè
- Cosir un vestit
- Fer un suc de taronja
- .....

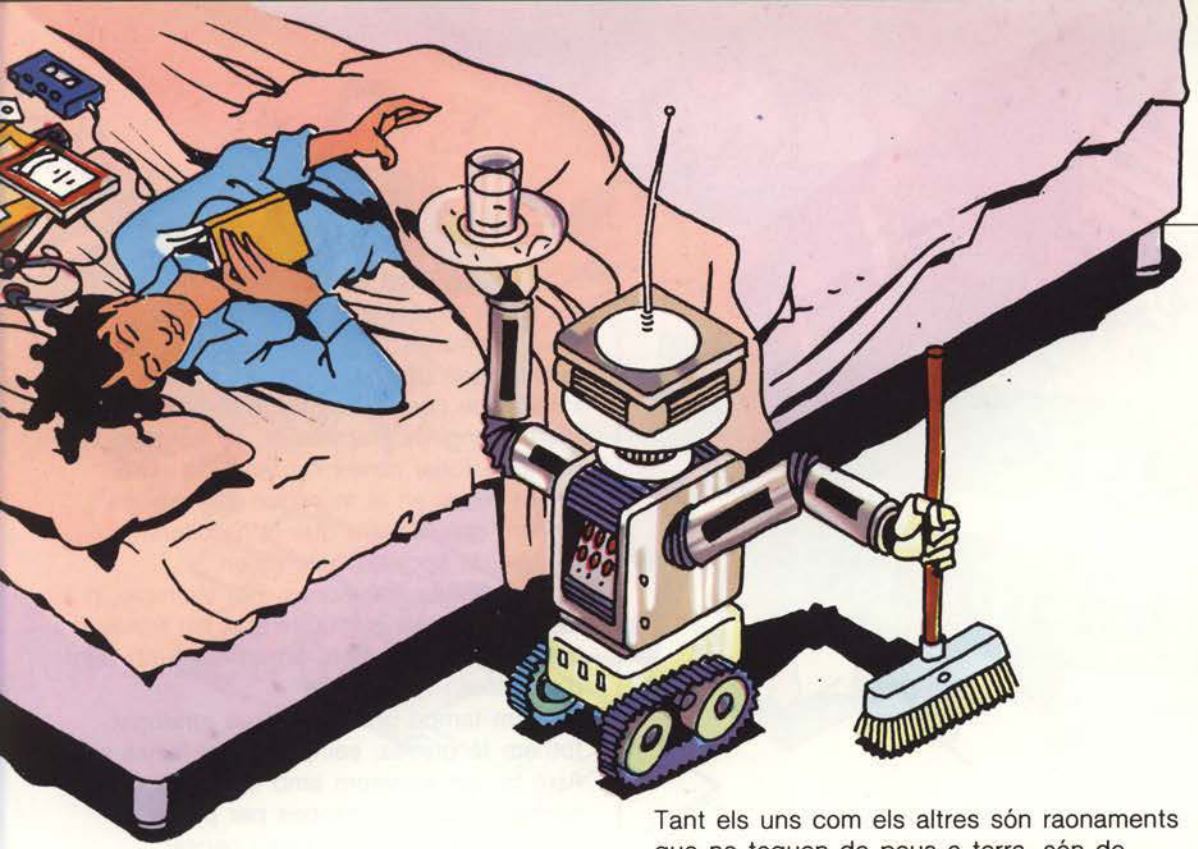
Tot això fa dir: que n'és de bona l'ELECTRICITAT!
això sí que és un INVENT!
Ah!, però i si un fica els dits dins un endoll?
i si hi ha un curtcircuit?
i si toquem un fil pelat?
Això fa mal, enrampa i fins i tot mata.
Aquest invent que tant bonic ens semblava, a l'hora de la veritat pot ésser perjudicial.
Bé, un raonament com aquest no és apropiat només al corrent elèctric; al llarg de la història ha passat igual per a cada cosa nova que s'ha creat. Penseu en la màquina de vapor, l'energia nuclear,..... De tot hi ha gent que n'és entusiasta i gent que no vol ni parlar-ne. S'ha d'escoltar tothom i saber trobar un punt mitjà.
La informàtica no s'escapa d'aquesta problemàtica; hi ha qui pensa que gràcies a ella:

- D'aquí a uns quants anys no caldrà treballar
- Es curaran les malalties
- Tothom viatjarà per l'espai
- .....

D'altres, al contrari, diuen que les màquines:

- Destruiran l'home
- Han creat l'atur i la reconversió
- Pensaran per nosaltres
- Ens tenen controlats
- .....



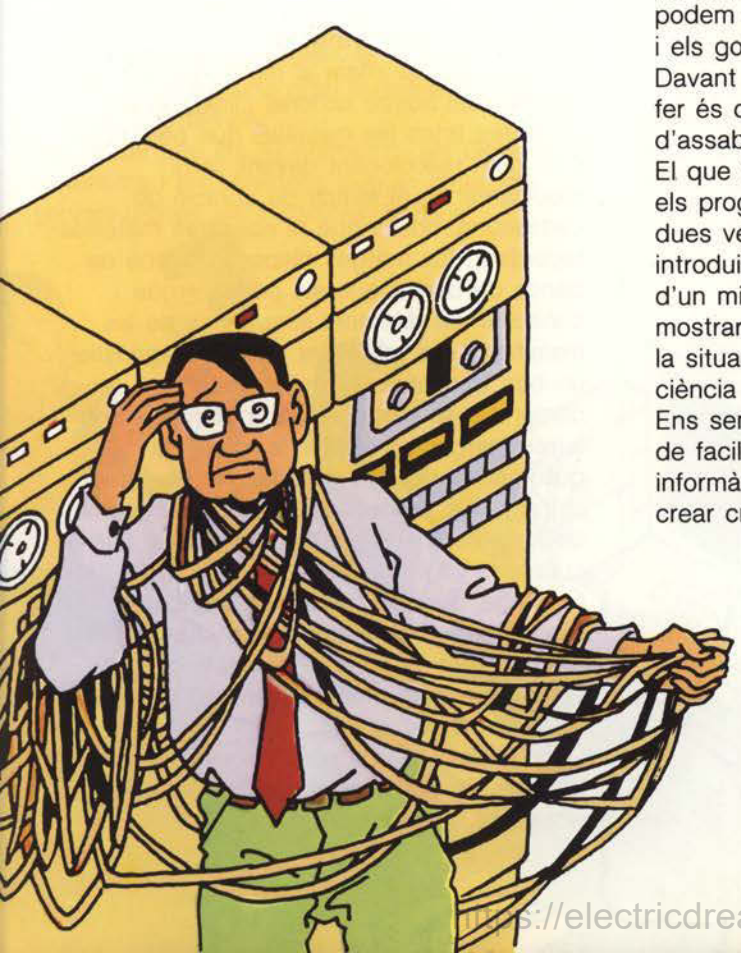


Tant els uns com els altres són raonaments que no toquen de peus a terra, són de «ciència ficció». Cal pensar que la informàtica no és ni bona ni dolenta; els qui, si un cas, podem ésser bons o dolents som els homes i els governs que en fem ús.

Davant aquesta problemàtica, el que s'ha de fer és cercar informació per tal d'assabentar-se ben bé del que passa.

El que intentarem en aquests petits llibrets i els programes de televisió és una feina en dues vessants; per una banda pretenem introduir el lector en el llenguatge i el maneig d'un microordinador i per l'altra volem mostrar els dominis, els camps d'aplicació i la situació actual de la informàtica, com a ciència i com a fet social.

Ens sentirem del tot satisfets si aconseguim de facilitar la comprensió de les situacions informàtiques que ens envolten i d'ajudar a crear criteri per a valorar-les.



Apropem-nos a la informàtica

Si hi ha quelcom que fa l'home diferent de les altres espècies del regne animal és la seva capacitat de CREAR. Aquesta és la qualitat que ens ha permès d'inventar infinitat de coses, des de la roda fins als darrers enginys electrònics; de confeccionar lleis i de forjar noves maneres de viure.

Si ens fixem en el món que ens envolta veurem que, encara que hi ha diferents models de societat, tots tenen característiques comunes. Per exemple, hi ha molts dels seus habitants que fan feines del tot rutinàries, sia manualment sia amb l'ajut d'aparells.

Veurem també que tothom va atrafegat, tothom té pressa; sempre tenim feines a fer. Això fa que esperem amb candeletes els nostres dies de vacances per poder asseure'ns, llegir, imaginar i pensar.

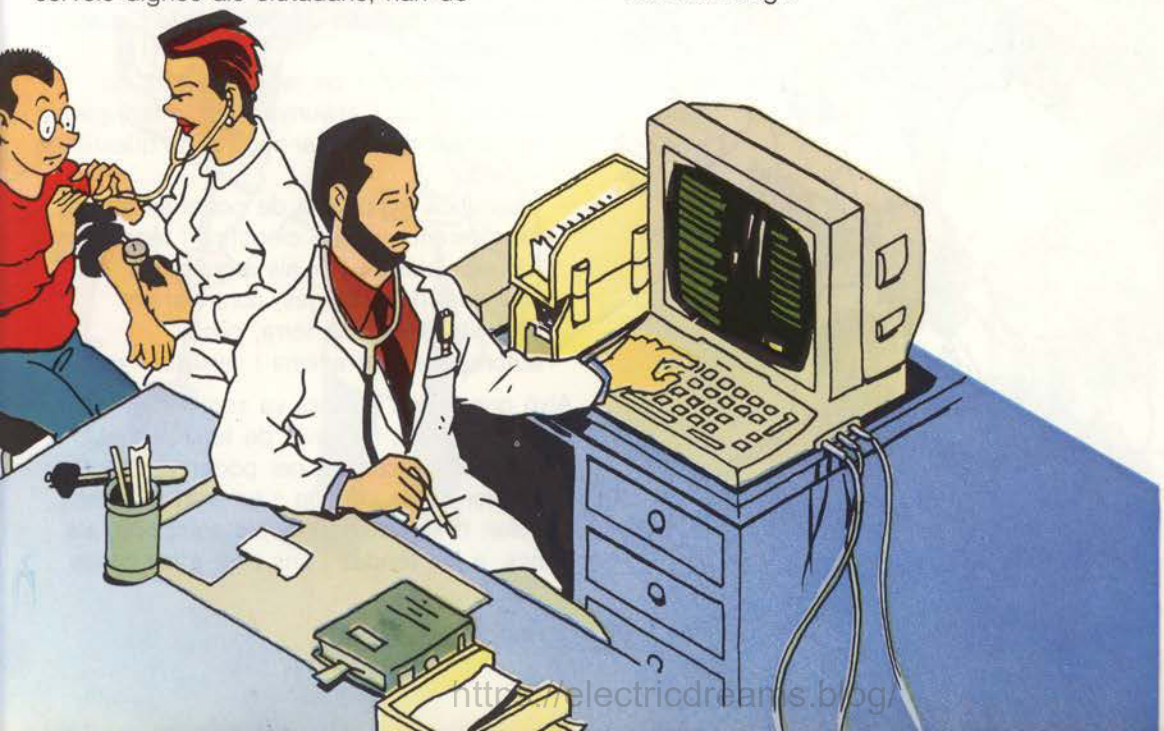
Podríem dir que les característiques del nostre temps són la **rutina** i l'**atabalament**. Però hi ha encara un tercer factor que ens defineix: és la **informació**.

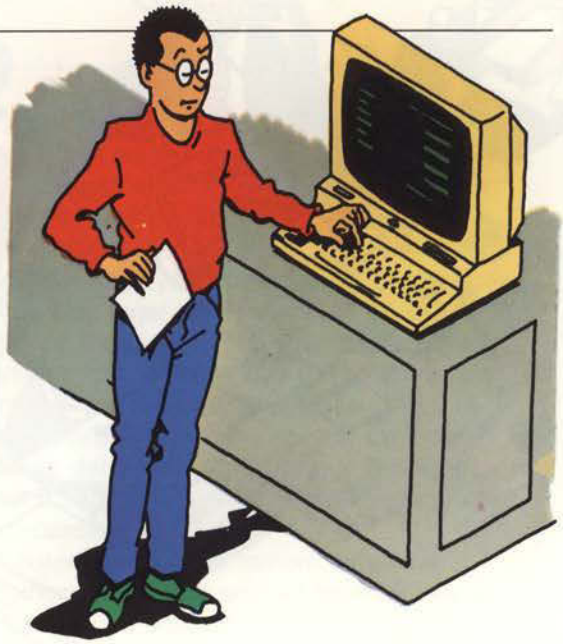
Per exemple, si anem al metge veurem que disposa del nostre historial clínic, on té anotades totes les malalties que hem tingut, com hem reaccionant davant certs medicaments, el temps de curació de cadascuna, coses que ni nosaltres mateixos recordem. Els metges disposen també de bancs de dades que els permeten de consultar els diferents símptomes de les malalties i així poder fer un bon diagnòstic i un bon tractament. A més, poden comunicar-se i intercanviar informació amb altres metges que estan a milers de quilòmetres del seu hospital. No és, doncs, estrany que en l'actualitat es curi un nombre de malalties deu vegades superior al que es curava el segle passat.

O també, avui dia, abans de fabricar un producte, una empresa ja sap si agradarà o no a la gent, el nombre d'unitats que en vendrà, i a quin preu cal vendre'l per obtenir un determinat benefici.

Els científics, si una cosa necessiten per avançar, és conéixer el que fan els altres en qualsevol lloc del món.

Tot això és possible gràcies als **ORDINADORS**, que tenen la qualitat de poder processar una gran quantitat de dades en molt poc temps.
Per tal de que els Ajuntaments, els Governs Autònoms i els Ministeris puguin oferir uns serveis dignes als ciutadans, han de maniobrar grans quantitats d'informació, com és ara el cens, i les estadístiques, inversions, pressupostos, impostos... etc.
Sense ordinadors, en un país democràtic, després de la jornada electoral, haurien de passar molts dies abans no se sabés qui ha resultat elegit.





Tal com presentem les coses, ja veieu que el que es necessita és **quelcom** que a l'hora que sigui capaç de moure grans quantitats d'informació, també ajudi l'home en les tasques repetitives i rutinàries per deixar-li més temps lliure per poder continuar inventant coses noves, apreciar l'art, viatjar... i parlar amb la gent que l'envolta.
Per aconseguir tot això cal resoldre els tres problemes que us hem esmentat:

- enllestir les feines rutinàries
- eliminar les presses
- accedir a molta informació en poc temps i que aquesta ocupi poc espai.

Es per això que fa cinquanta anys va néixer una nova ciència anomenada **INFORMÀTICA**, que té per objectius fonamentals:

- crear màquines que facin treballs rutinaris: robots, calculadores, ordinadors...
- fer les feines a gran velocitat:
  en un segon fer un milió de multiplicacions de números d'unes quantes xifres, o en un minut ésser capaç d'ordenar alfabèticament 500 fitxes.
- dissenyar aparells per maniobrar i guardar informació:
  les declaracions de renda de tots els ciutadans de Catalunya (que la fan) caben en un armari i cercar-ne una es qüestió de minuts.
- diversificar-se per tal de poder-se aplicar als diferents camps científics i tècnics:
  dibuixar plànols per als arquitectes, els scanners dels metges, controlar els satèl·lits des de la terra, pilotar automàticament avions i vaixells...

Això que en un principi va sorgir com un domini científic, i s'havia de tenir el títol d'enginyer o llicenciat per poder veure un ordinador, cada vegada s'ha anat fent més popular; hi ha ordinadors als aeroports, als bancs, a les tendes i fins i tot a les cases.

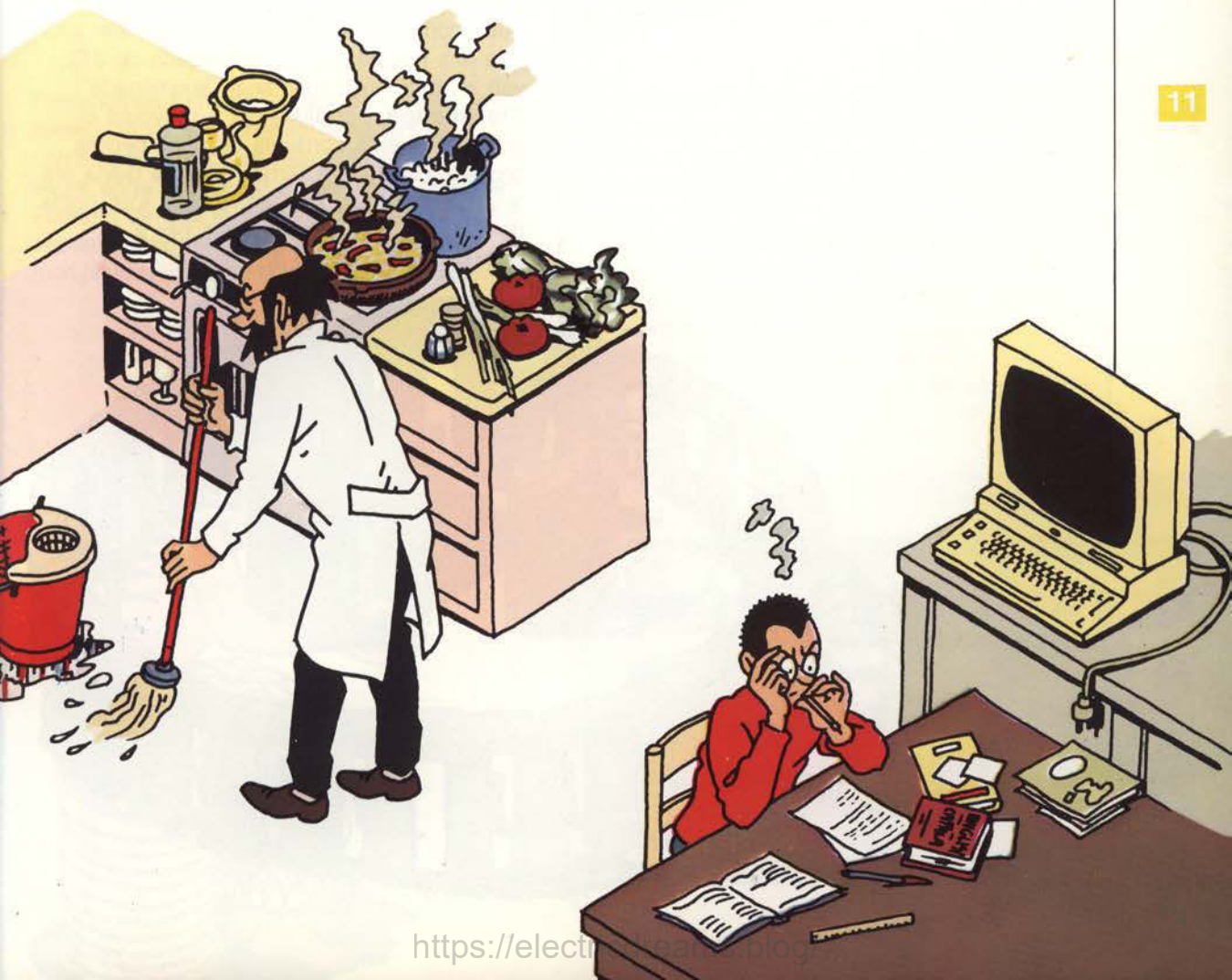
Ara tothom que hi estigui una mica il·lusionat pot tenir un microordinador.
Aquesta invasió és deguda no sols a una moda o al fet que els microordinadors estan a bon preu. Els microodinadors (familiarment en direm micros) són:

- divertits: gran quantitat de jocs electrònics per estimular els reflexos i la imaginació
  jocs educatius, com els escacs...
- avantatjosos per a les feines de casa:
  comptabilitat
  agenda
  correspondència
- educatius: permeten d'iniciar-se en la informàtica i aprendre llenguatges de programació i ajuden a les tasques de l'escola.

Ara, no cal esperar que:

- facin el dinar
- escombrin la casa
- facin els resums d'història o les traduccions de l'anglès.
- .....

Pensem, però, que la informàtica és una ciència jove i que si en cinquanta anys ha estat capaç de crear tot això, vés a saber amb què sorprendrà els ciutadans de l'any 2.200!!

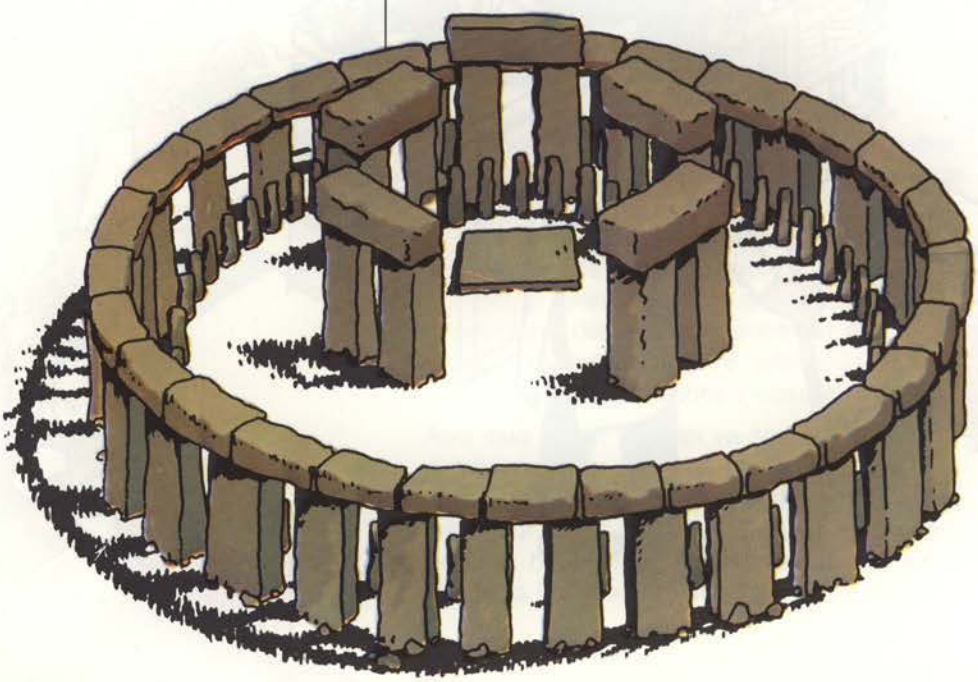


Una mica d'història

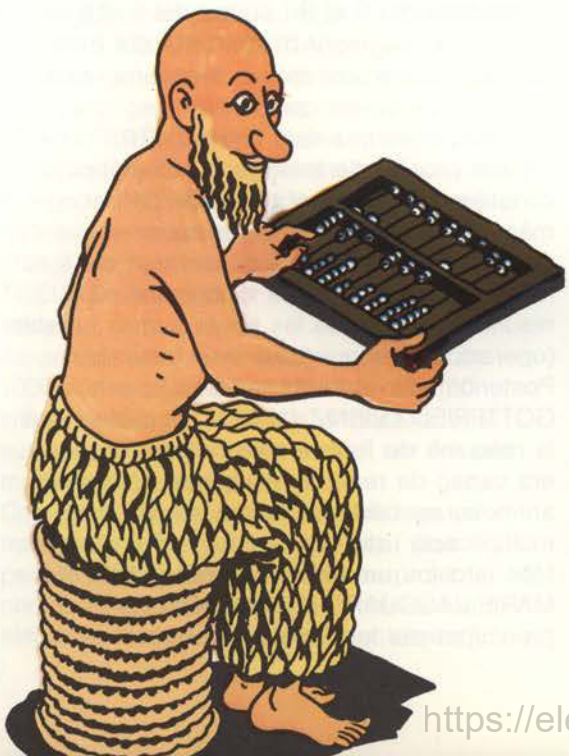
La informàtica, o més aviat l'ordinador, apareix gairebé pertot arreu en aquest món que ens envolta. Anem ara a donar-vos alguns noms, aspectes i dades de la seva recent història, que avui és a la cinquena generació.
Aquesta història comença amb el naixement del nostre segle XX, i rep l'embranzida més gran pels voltants de l'any 50 amb l'arribada dels elements electrònics.

Prehistòria de la informàtica

Quan l'home va deixar de viure a les coves per establir-se en poblats, el primer ofici que tingué fou el de pagès. Era, doncs, vital per a aquelles primeres societats poder saber quan començaven les estacions. Es per això que uns homes del segle V abans de J.C. van inventar «una màquina», anomenada **Stonehenge**, per ajudar-se a preveure el temps. Aquesta consistia en unes pedres grosses clavades a terra, i segons quina era la posició del sol ixent entre elles s'en podia deduir el temps que mancava per a l'arribada de l'estiu o de l'hivern.

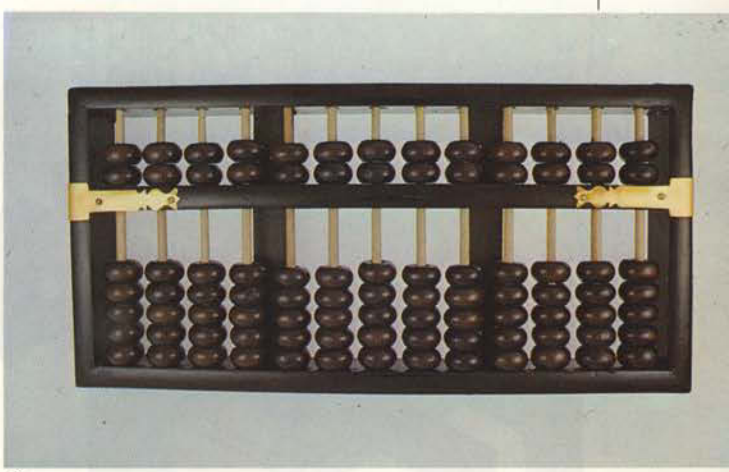


Posteriorment l'home, a part de conrear les terres, decidí de tornar-se ramader. Va fer corrals per tenir-hi animals i, és clar, es va veure en la necessitat de saber-los comptar. Podríem dir que la primera calculadora en aquest sentit van ésser els dits. Deu dits molt fàcils d'usar i portàtils, però de poca ajuda.

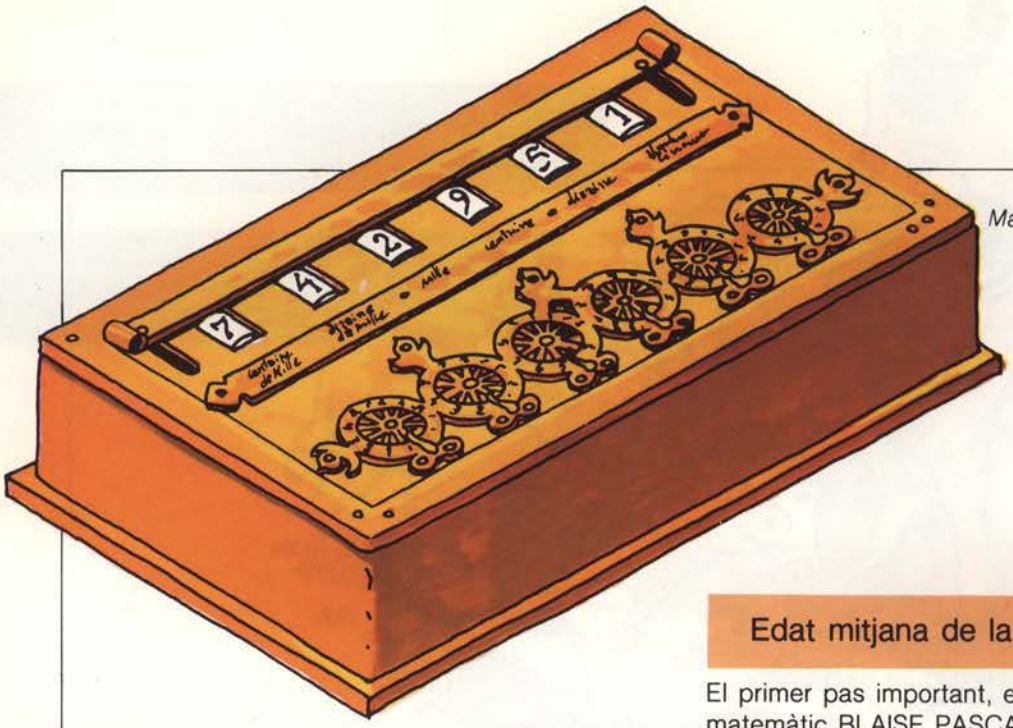


La civilització babilònica, el segles III o IV a. J.C., descobrí un mecanisme de càlcul anomenat **àbac**, que és un dispositiu que consisteix en un conjunt de boles sobre varetes.

L'ús de l'àbac és encara vigent en alguns pobles de l'Europa Oriental i d'Àsia. Malgrat que aquesta «màquina» facilita la manipulació de quantitats, no aporta res de nou al concepte de càlcul ni a la seva automatització (les operacions a fer són sempre manuals).

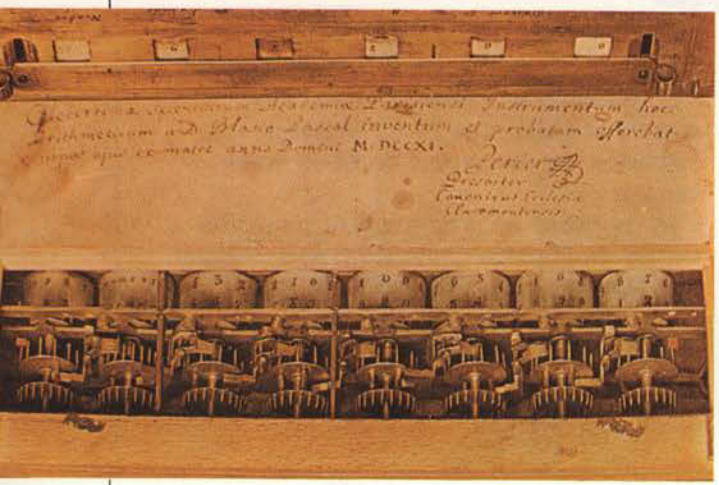


Àbac

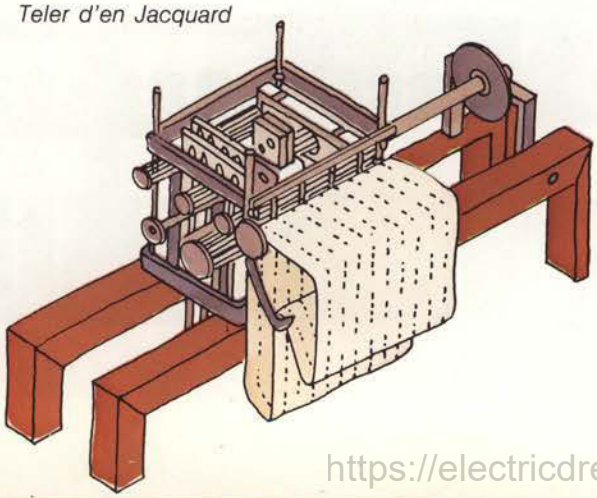


Màquina de Pascal (Pascalina

Pascalina: detall interior



Teler d'en Jacquard



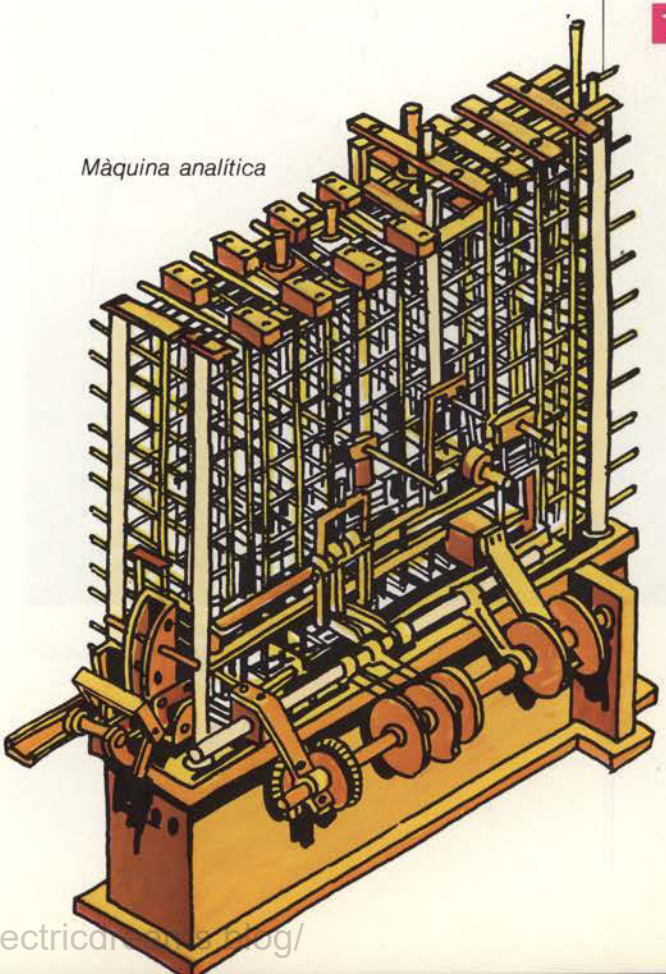
Edat mitjana de la informàtica

El primer pas important, el donà el matemàtic BLAISE PASCAL (1623-1662). Fill d'un recaptador d'impostos, es trovaba que cada vespre havia d'ajudar el seu pare a repassar les llargues sumes dels impostos cobrats. Com que aquesta feina tan rutinària l'avorria, decidí de crear una «màquina» que l'ajudés. Aquesta «màquina» podia fer les operacions de suma i resta (les que necessitava un cobrador d'impostos) mitjançant combinacions d'unes rodes dentades: cada roda tenia deu dents numerades del 0 al 9 i el pas del 9 al 0 provocava l'augment d'un en el valor de la roda inmediatament més a l'esquerra (és a dir, que funcionava com el comptaquilòmetres dels cotxes).
Aquest procés d'arrossegada automàtica constitueix la principal aportació de la màquina de Pascal. En sofisticacions posteriors va introduir-hi un element de memòria que li permetia d'acumular els resultats parcials de les seves sumes i restes (operacions sempre fetes amb base deu).
Posteriorment, el també matemàtic GOTTFRIED LEIBNIZ (1646-1716) perfeccionà la màquina de Pascal construïnt-ne una que era capaç de realitzar les quatre operacions aritmètiques bàsiques: suma, resta, multiplicació i divisió.
Més tard fou un mecànic francès, JOSEPH MARIE JACQUARD (1752-1834), més preocupat per la idea de construir

mecanismes d'enginyeria automàtics que no mecanismes de càlcul, qui inventà un teler automàtic que mitjançant un sistema de targetes perforades li permetia de realitzar còpies perfectes d'un prototip de roba.
No obstant això, no podem parlar de veritable avenç en el concepte d'automatització de càlcul fins l'any 1831, amb l'intent de creació de la «màquina analítica» per un professor de matemàtiques de la Universitat de Cambridge, CHARLES BABBAGE (1972-1871). Diem intent de creació, car Babbage mateix va fracassar en la consecució del seu propòsit, perquè les seves idees superaven la tecnologia existent en aquell temps.
Amb tot, el seu propòsit no assolit ha marcat el posterior desenvolupament de la concepció dels ordinadors.
L'objectiu de Babbage era construir una «màquina» capaç de realitzar qualsevol operació matemàtica sense la intervenció humana en el procés de càlcul. Aquest enginy és el que ell mateix denominava «màquina analítica», la qual era concebuda com la composició de quatre unitats bàsiques: una **MEMÒRIA** per poder emmagatzemar dades i resultats intermedis, una **UNITAT ARITMÈTICA** per poder efectuar els càlculs necessaris, un sistema d'engranatges i palanques per poder fer la **TRANSFERÈNCIA** de dades entre memòria i unitat aritmètica, i finalment uns dispositius por poder **INTRODUIR** dades (mitjançant un sistema de targetes perforades com les utilitzades per Jacquard en el seu teler) i **TREURE** els resultats obtinguts per la màquina.
Aquest disseny pretenia una memòria de 1000 números de 50 dígits decimals (50 xifres) i estimava el temps d'efectuar una suma en un segon, mentre que per fer una multiplicació caldria un minut.
Com hem apuntat abans, si Babbage va fracassar no fou pel seu mal cap, sinó perquè en aquell moment el seu projecte només es podia realitzar materialment amb elements mecànics (on era l'electrònica?).

Charles Babbage

Màquina analítica



Renaixement de la informàtica: els primers ordinadors

Estareu d'accord que Babbage, en el cas d'aconseguir el seu somni: la fabricació de la «màquina analítica», hauria aconseguit un gran avenç vers l'automatització del càlcul, no obstant que la informació que podia tractar la seva «màquina» era exclusivament numèrica (el que avui en diem digital).
Va ésser el nord-americà HERMAN HOLLERITH (1860-1929) qui, l'any 1890, aportà la idea genial de «codificar» qualsevol tipus d'informació mitjançant un sistema de targetes perforades, com ara el que usava Jacquard per al seu teler.
Com sempre, o quasi sempre, la funció crea l'òrgan. Hollerith es trobava amb la necessitat imperiosa de tractar un gran volum d'informació: el cens de la població dels Estats Units, que comptava llavors amb seixanta-cinc milions d'habitants.

Herman Hollerith

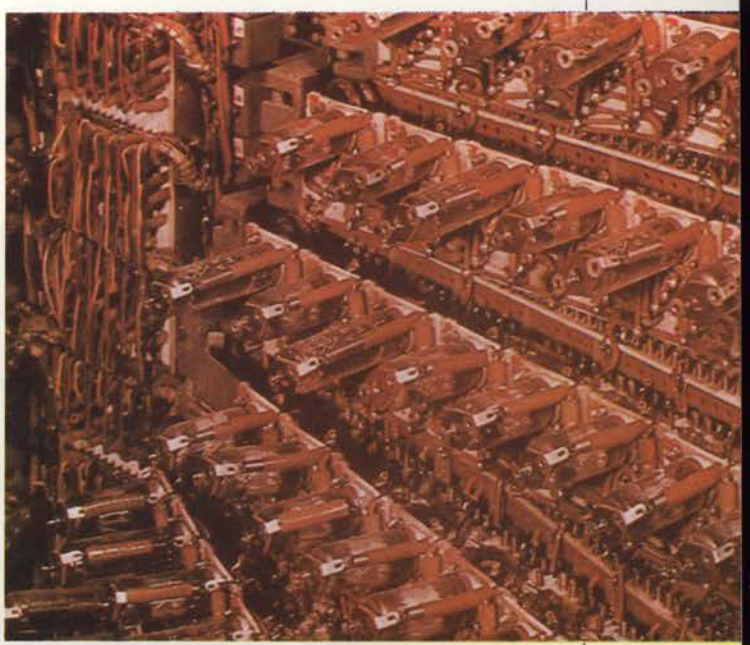
Va tenir la brillant idea de representar les diferents característiques de la població mitjançant uns forats en llocs específics d'una targeta, que després eren «llegits» per un dispositiu electrònic i comptats mecànicament. La màquina electromecànica construïda per Hollerith va trigar dos anys i mig a avaluar el cens dels Estats Units (avui consideraríem aquesta màquina molt lenta, però en aquell temps era un gran invent).
L'any 1896 el mateix Hollerith creà una empresa anomenada la Tabulating Machine Company per fabricar el seu enginy i el 1911, ajuntant-se amb altres companyies, fundà la Computing-Tabulating Machine Company, que donaria lloc al naixement de la coneguda companyia IBM (International Business Machines), l'any 1924.
A partir de la creació d'IBM, la informàtica ha experimentat un esclat en el seu desenvolupament, que coneixem avui com la revolució tecnològica.
Impulsat per la IBM, HOWARD AIKEN (1900-1973), professor de Física de la Universitat de Harvard, proposà el disseny d'una calculadora electromecànica.
Realitzar-la va comportar-li cinc anys d'esforços i suors, de 1939 a 1944. Rebé el nom de MARK I o A.S.C.C. (Automatic Sequence Controlled Calculator). Aquesta màquina incorporava les idees de Babage, ja que la «programació» era realitzada mitjançant una cinta perforada. L'element electromecànic de base era el relé (dispositiu que permetia d'obrir i tancar un circuit). La capacitat de la memòria del MARK I era de 72 números de 23 dígits decimals i necessitava 3 segons per fer una multiplicació de dos números de deu dígits (en realitat tenia una memòria molt més petita que la que pensava fer Babbage, però era molt més ràpida).
Aquesta màquina, que pesava 5 tones i tenia 5.000 relés, va estar en funcionament fins a l'any 1959.
Solapant-se en el temps, entre 1943 i 1946, a la Universitat de Pennsilvània, JOHN MAUCHLY i JOHN PRESPER ECKERT

L'ENIAC (vista parcial)

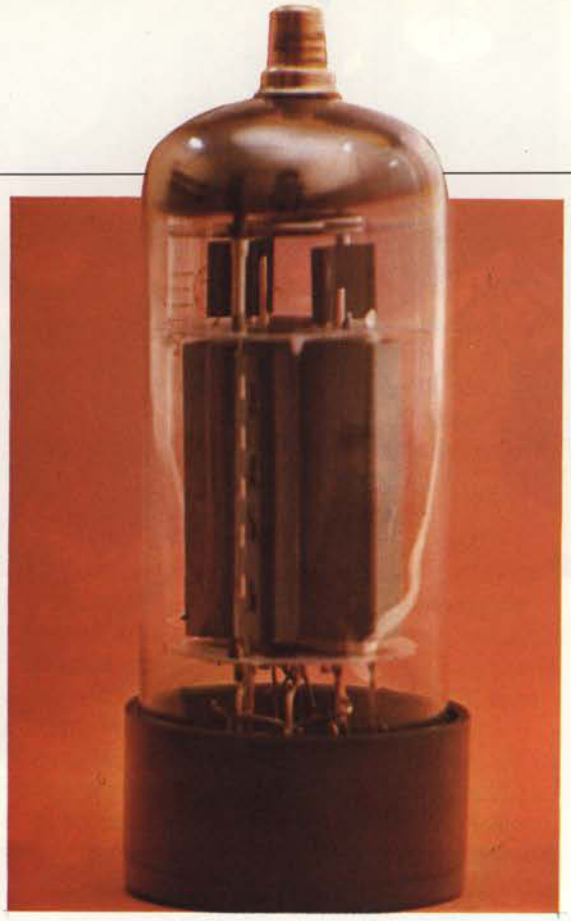
dirigien els treballs encaminats a concebre la primera calculadora electrònica, anomenada ENIAC (Electronic Numerical Integrator and Computer), la qual, com que no tenia components mecànics, reduïa el temps de càlcul (aquesta només necessitava 0,003 segons per fer una multiplicació de dos números de 10 dígits, és a dir, una reducció a la mil·lèsima part del temps necessari per al MARK I). «Programar» aquesta màquina consistia a fer connexions entre els cables elèctrics i accionar gran quantitat d'interruptors. Aquesta màquina pesava 30 tones, ocupava una habitació de 30 metres de llagarda per 3 d'alçada i 1 de fondària. Consumia tanta electricitat com mil rentadores funcionant alhora.

Aquests mateixos investigadors es van adonar que la cosa era força complicada, i basant-se en els estudis teòrics de VON NEWMANN, guiaren els seus esforços vers una màquina de calcular que contingués una sèrie de programes emmagatzemats per tal d'estalviar la feina d'anar connectant fils i més fils. Així crearen la màquina UNIVAC I per a l'oficina americana del cens.

Ja havía començat la competència comercial i, amb aquesta, l'era contemporània de la informàtica.



Plaques de vàlvules de l'ENIAC

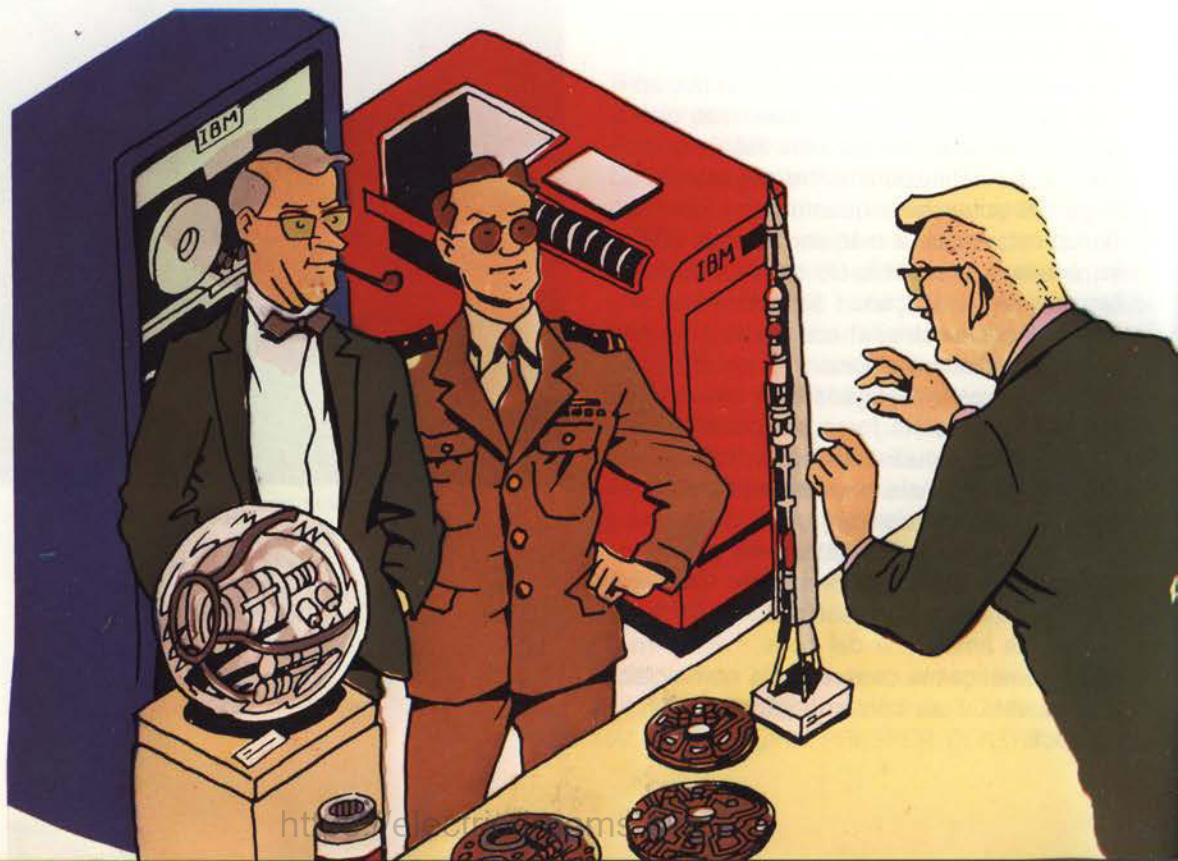


Vàlvula de buit

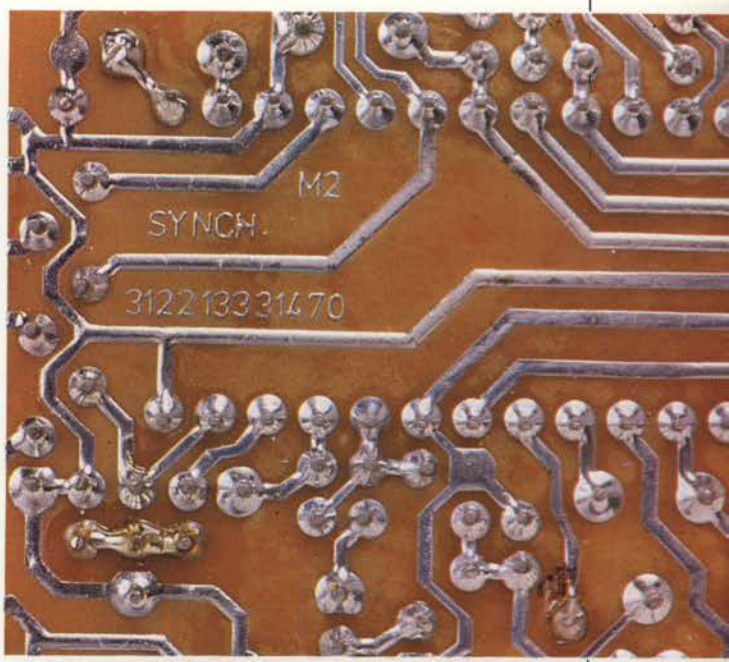
L'era contemporània de la informàtica: era comercial

Ens situem ja en els anys 1950, en què començà la comercialització de les màquines de calcular. El component electrònic essencial d'aquelles màquines era la vàlvula de buit (aquelles que també s'utilitzaven en els antics aparells de ràdio i televisió). Aquelles màquines avui ens semblen extremadament lentes i limitades, car només podien realitzar mil instruccions per segon i la seva capacitat de memòria oscil·lava entre 10.000 i 20.000 posicions. Foren **la primera generacio** d'ordinadors.

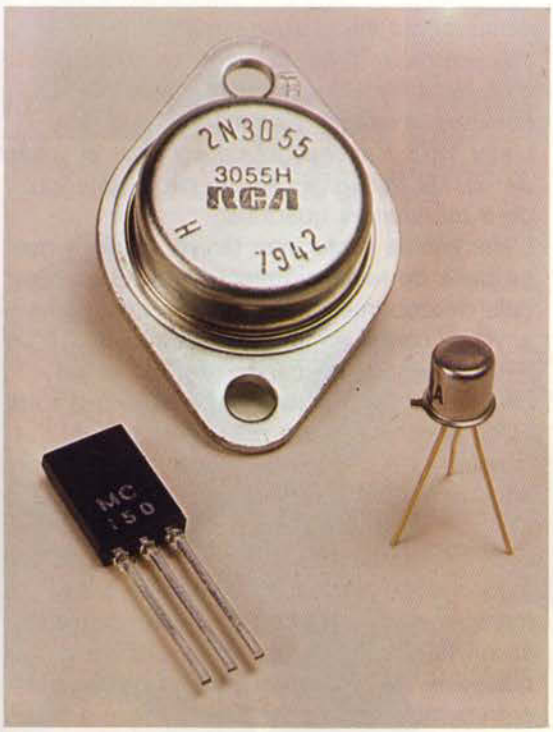
En aquella època les investigacions en electrònica es veieren potenciades i tigué lloc l'aparició del transistor (Premi Nobel de Física de l'any 1956). El transistor és un dispositiu electrònic que és capaç d'actuar com un interruptor. Consumeix molta menys energia i és més petit i menys fràgil que la vàlvula de buit. Al mateix temps, és més fàcil de construir en grans quantitats i, per tant més barat que les vàlvules.



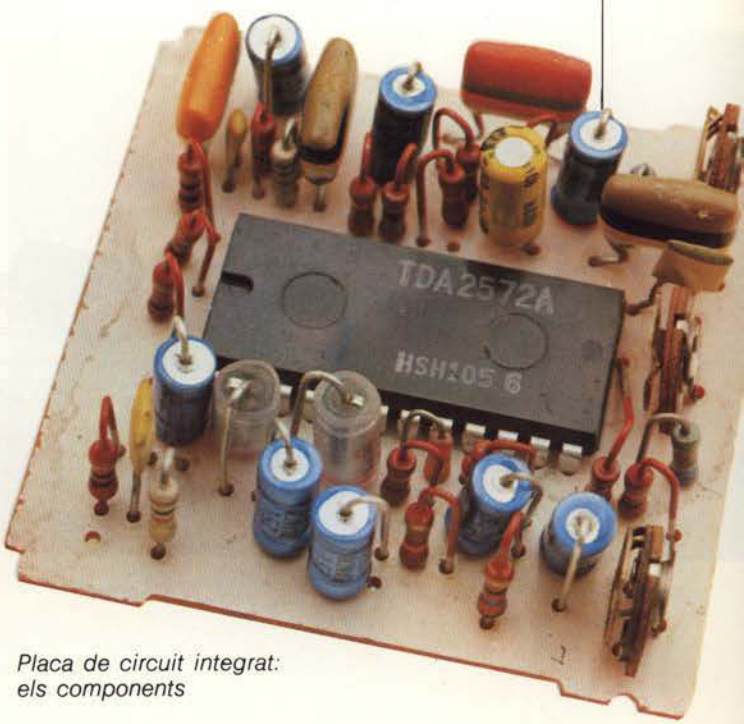
El transistor marca el pas a **la segona generació** dels ordinadors, en què les màquines són més petites, més ràpides (un milió d'instruccions per segon) i més fiables que les de la primera.
Quant al temps, es consideren ordinadors de la segona generació els nascuts entre 1960 i 1965. En aquesta època no només es progressà en el camp electrònic, sinó també en la tècnica d'utilització d'aquestes màquines: apareixen els conceptes de sistema operatiu, temps compartit, lleguatge d'alt nivell...
En aquells anys es continuà investigant en electrònica i aparegué el circuit integrat, el qual permet de col·locar sobre una pastilla de silici, utilitzant tècniques fotogràfiques, tots els components i les connexions necessaris per fer un circuit, en lloc de fabricar transistors aïllats i després conectar-los.



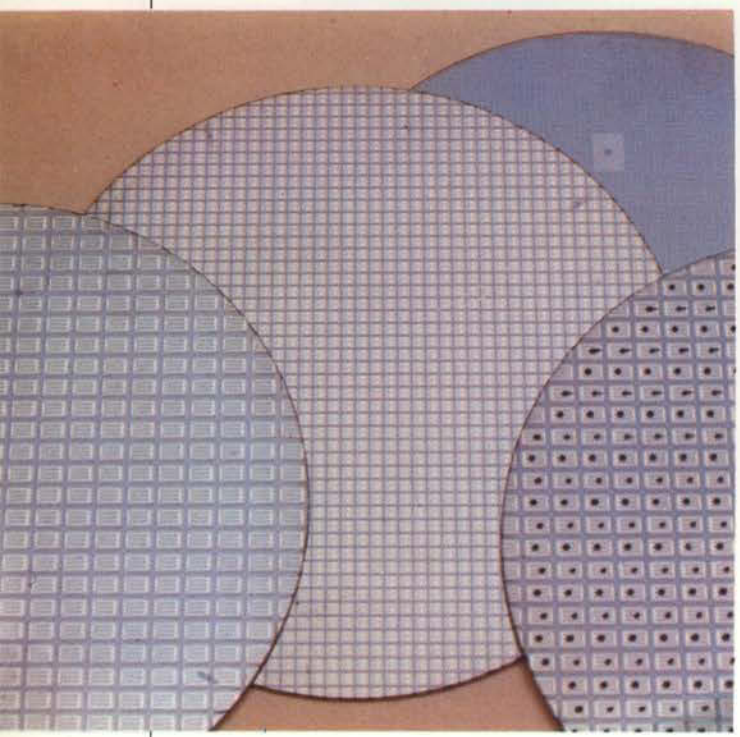
Placa de circuit integrat: les conexions



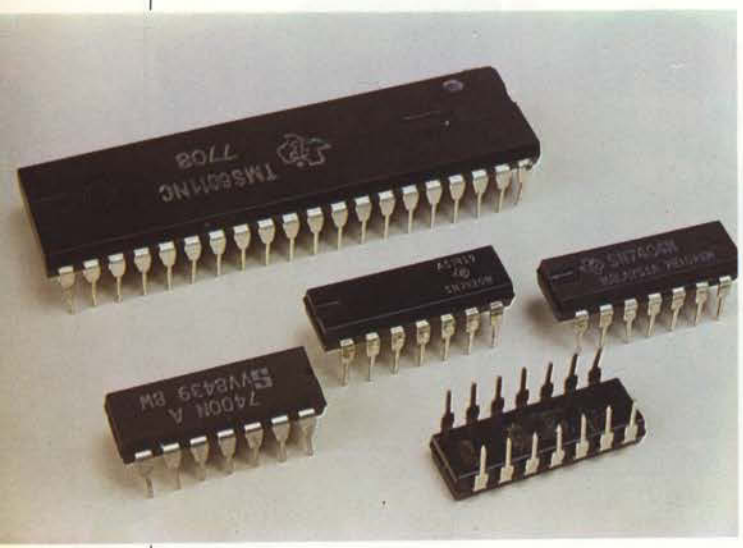
Tres models de transistors



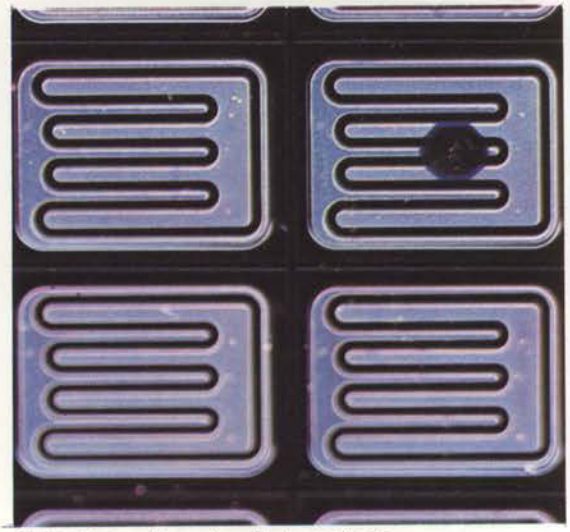
Placa de circuit integrat: els components



Oblees de silici



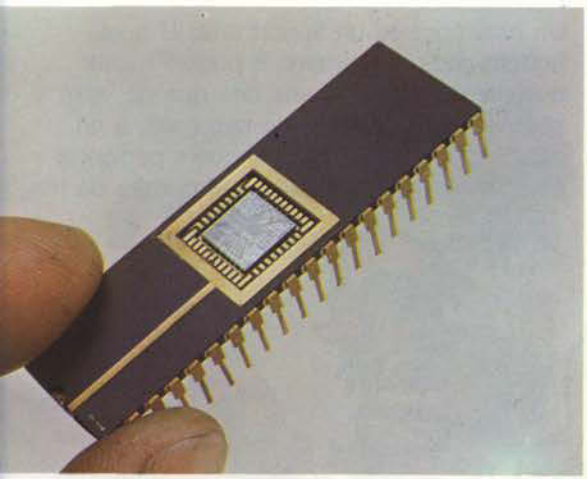
Chips



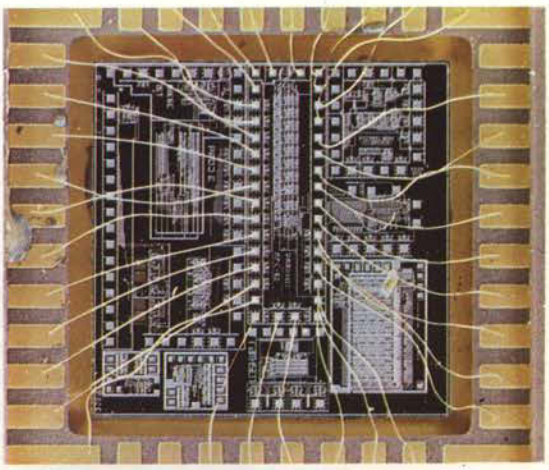
Part d'una oblea de silici ampliada

Amb els circuits integrats nasquè **la tercera generació** dels ordinadors, que es caracteritza per la reducció de l'espai d'emmagatzematge, i aconseguí cap al 1974, de col·locar circuits integrats d'uns 20.000 components en 5 mil·límetres quadrats (us imagineu com n'és, de petit, un circuit d'aquesta mida?), al mateix temps que es continuava reduint el consum d'energia.
L'any 1975 s'aconseguí de col·locar al voltant de 65.000 transistors en un circuit integrat de 5 mil·límetres quadrats.
L'electrònica evoluciona tan ràpidament que es parla de revolució electrònica. Gran part dels descobriments tecnològics dels darrers anys es basen en la microelectrònica.
Avui, la tècnica és capaç de ficar en un circuit integrat funcions complexes que totes soles ja constitueixen màquines. Neix el concepte de microprocessador, i amb ell una nova generació d'ordinadors: **la quarta**.
El microprocessador és la veritable clau de l'èxit de la informàtica. El que abans era una màquina mastodòntica que només era manipulada per especialistes s'ha convertit en un petit circuit integrat transferible a qualsevol tipus d'usuari. Només cal un petit entrenament per poder interaccionar amb un d'aquests enginys.

Una idea de la magnitud d'aquests desenvolupaments ens la pot donar aquesta analogia: Si el cotxe Volkswagen, «l'escarabat» que va sorgir a la mateixa època que el primer gran ordinador, hagués millorat en preu i rendiment com han fet aquests, avui correria a uns 10.000 km/h, gastaria 1 litre de gasolina cada 1.000 km. i ens costaria uns vint duros.
I l'electrònica no ha arribat al sostre de les seves possibilitats. Cada dia es redueix l'espai d'integració, s'augmenta la velocitat de càlcul, es redueix el cost..., van passant noves generacions, ja es parla de la cinquena. Cada vegada ens impliquen més i més, i ja se'ns fan gairebé imprescindibles uns coneixements mínims en la utilització dels ordinadors; cal endinsar-nos en els diferents dominis informàtics.



Microprocessador dissenyat al Dpt. d'Informàtica de la Universitat Autònoma de Barcelona



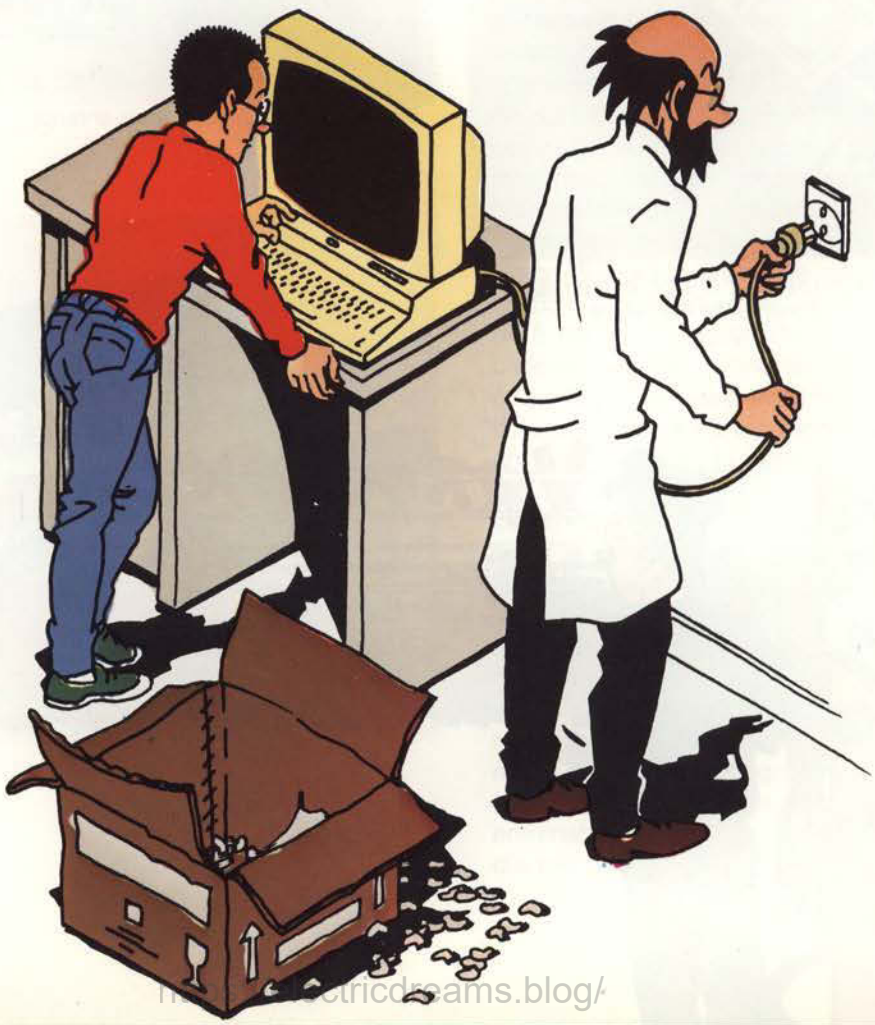
Detall del microprocessador anterior

Què es un ordinador?

Un ordinador és una màquina. Però, una màquina per a què?
Quan veiem un televisor sabem que si l'endollem veurem imatges. Quan posem en marxa un tocadiscos sabem que ens oferirà música. Quan tenim la roba bruta usem una rentadora. Al nostre voltant en tenim moltes, de màquines; i de cadascuna, en sabem ben bé el que fa. Però i els ordinadors, què fan? Hem de dir que quan endollem un ordinador aquest **NO FA RES**. Això decep? No! Al contrari, cal saber que un ordinador és, ni més ni menys, que **una màquina per acabar** i que som nosaltres els qui podem acabar-la per tal de que faci el que vulguem. Si en tenim ganes i en sabem una mica, un ordinador es pot convertir en una màquina per

- Portar la comptabilitat d'una empresa
- Dibuixar els plànols d'un arquitecte
- Controlar la venda i reserva de bitllets d'un aeroport
- Desafiar-vos als escacs
- Col·locar rebuts per ordre alfabètic
- Guardar articles en una editorial
- I un seguit de moltes coses més.

Un ordinador és un suport amb el qual podem comptar sempre, li podem manar qualsevol cosa i mai ens dirà que no. Això sí, ell no està assabentat de cap matèria en concret; cal, doncs, que tinguem paciència i li expliquem ben bé què és el que ha de fer.



Parts d'un ordinador

Encara que a les botigues hi hagi diverses marques, formes, colors i mides, tots els ordinadors són iguals. Ho explicarem amb un exemple:

Suposem un professional molt enfeinat que ens demana ajuda. Nosaltres ens comprometem a treballar coratjosament i a fer-ho tot molt ben fet, però, la veritat, de la feina no en sabem res.

Primer ell ens dóna una carpeta plena de fulls amb els quals treballar (n'hi ha que s'han ordenar; en altres, s'hi han de fer càlculs, escriure adreces, etc...).

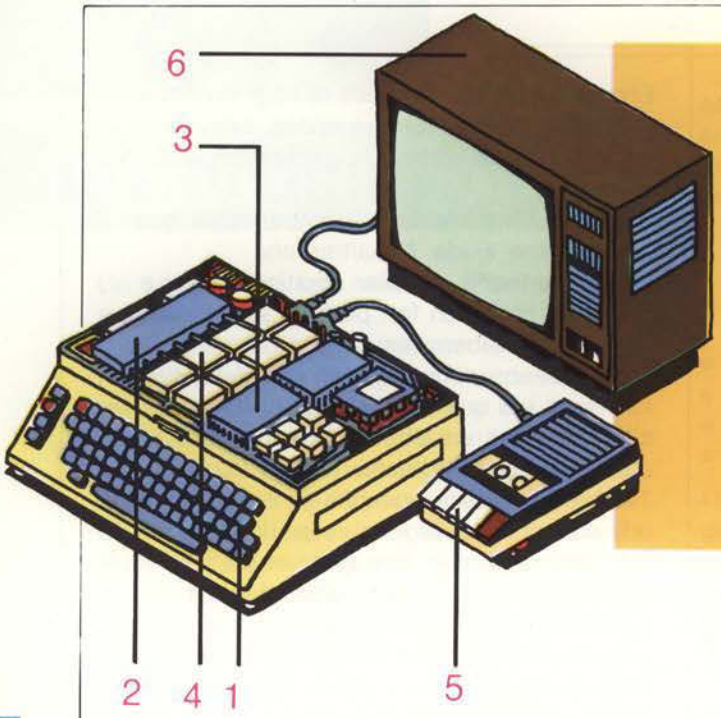
També ens facilita una pila de llibres per si ens cal consultar alguna cosa, una calculadora perquè ens ajudi en les operacions, un bloc per poder anotar quelcom que calgui tenir present i un arxivador.

Després ens explica a poc a poc, per ordre i ben detallat, què és el que hem de fer. Per tal que no se'ns oblidi, en prenem nota a la pissarra que tenim al darrera.

Ara ha arribat l'hora. Cal decidir per on posar-s'hi, com fer aquest reguitzell de coses que ens han manat i començar a treballar.

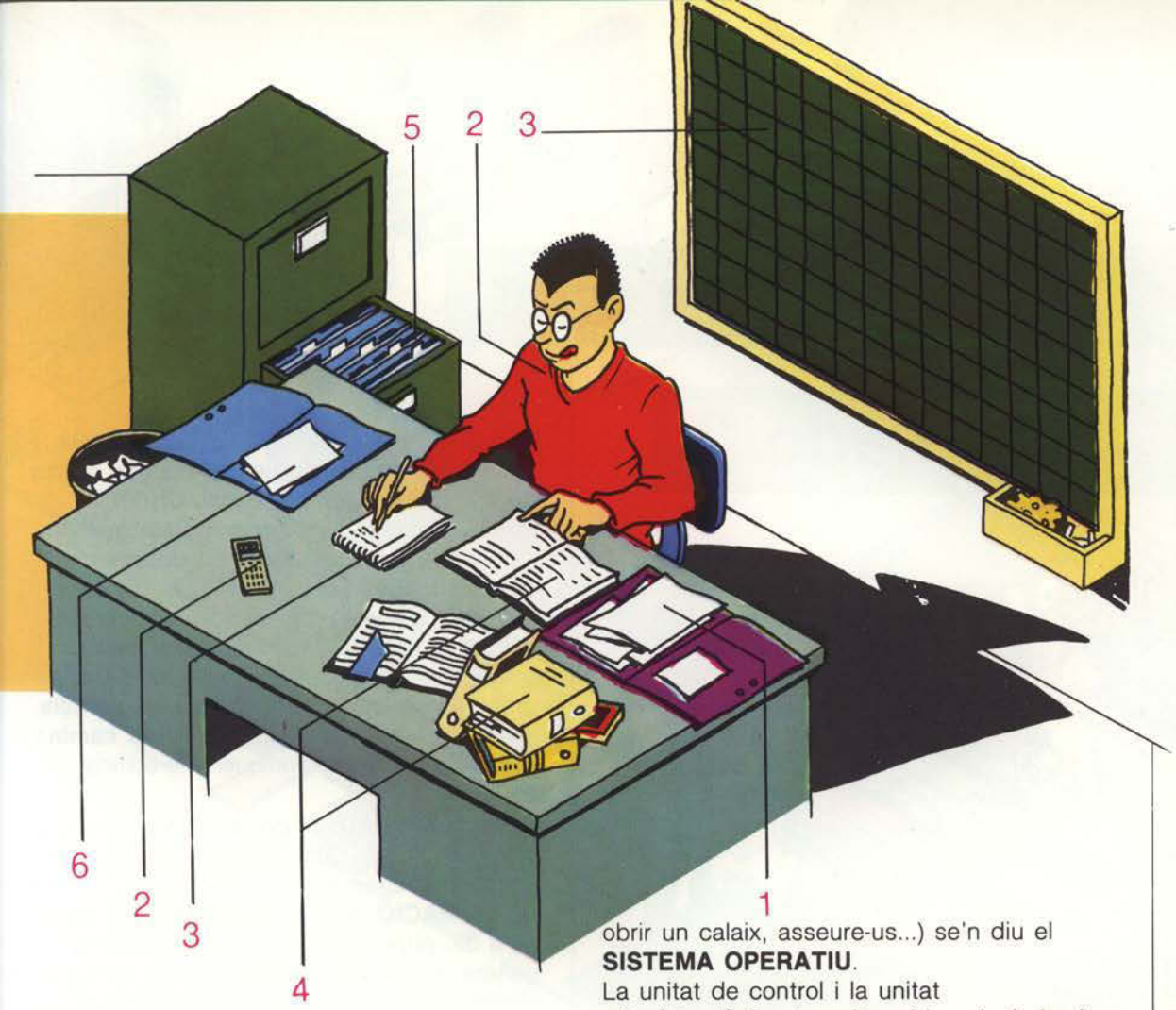
Tots els fulls acabats, cal col·locar-los en una altra carpeta.

Nosaltres, amb tots els materials que tenim al voltant, «fem» un ordinador.



1 Perifèric d'entrada
2 Processador central: Unitat aritmètico-lògica i unitat de control
3 Memòria RAM
4 Memòria ROM
5 Memòria secundària
6 Perifèric de sortida

Fixem-nos-hi:
La carpeta de les feines per fer és el que en direm **PERIFÈRIC D'ENTRADA**. En els ordinadors és un aparell de diskette, un enregistrador de cassettes o, el que és més corrent, un teclat. És el lloc per on «es posa» la feina a l'ordinador.
La carpeta de les feines fetes és el **PERIFÈRIC DE SORTIDA**, que pot ésser un aparell de diskette, un enregistrador de cassettes, ó el que és més comú, una pantalla o una impressora. És el lloc per on l'ordinador «ens entrega» els treballs que ens ha fet.
El que hi ha escrit en els fulls de dins les carpetes són les **DADES** i el paper on són escrites el **SUPORT**, que pot ésser tant un diskette, com una cinta, com fulls d'impressora.
La calculadora és el que anomenarem **LA UNITAT ARITMÈTICO-LÒGICA (UAL)**. Aquesta, a part de fer sumes, restes i les operacions que fan les calculadores de butxaca, és capaç de comparar dos números entre ells i saber si són iguals o quin és el més gran.
Totes les explicacions que ha fet el professional són el que s'anomena **PROGRAMA**.
Del bloc, la pissarra, els llibres i l'arxivador, en direm **MEMÒRIA**. Fixem-nos, però, que aquestes quatre coses són ben diferents:
El bloc, el fem servir per escriure-hi resultats intermedis dels nostres càlculs o de qualsevol feina massa llarga. És el que anomenem **REGISTRES**.
La pissarra és on guardem el programa. A més, podem escriure-hi, i esborrar-ne altres coses que necessitem consultar de manera ràpida. És el que en direm **RAM**. Per tal de tenir les coses més ordenades, suposarem que la pissarra està quadriculada i numerada. De cada quadret, en direm **POSICIÓ DE MEMÒRIA**.
Els llibres és on anem a consultar quan necessitem quelcom que no coneixem. Hi podem llegir el que vulguem, però no podem escriure-hi res. Són el que se'n diu



MEMÒRIA NOMÉS DE LECTURA O ROM. El bloc, la pissarra i els llibres, és a dir els registres la RAM i la ROM, són el que forma la **MEMÒRIA CENTRAL** de qualsevol ordinador.

L'arxivador és un lloc on podeu guardar i buscar qualsevol cosa que ens faci falta. En direm **MEMÒRIA SECUNDÀRIA**. Normalment són diskettes o cintes. La memòria central és de ràpid accés (només cal mirar), però hi cap poca cosa, mentre que la memòria secundària és lenta (cal aixecar-se, obrir el calaix i buscar), però, en canvi, és de gran cabuda.

Del qui organitza la feina (nosaltres en l'exemple), se'n diu **UNITAT DE CONTROL** i del conjunt de tasques que no cal manar perquè ja sabem fer, perquè les tenim apreses de fa temps (com ara gafar el llapis, obrir un calaix, asseure-us...) se'n diu el **SISTEMA OPERATIU**.

La unitat de control i la unitat aritmètico-lògica (nosaltres i la calculadora) formen en la majoria dels ordinadors un bloc comú anomenat **PROCESSADOR O MICROPROCESSADOR**.

De la pissarra, les carpetes, el llapis, la calculadora, l'arxivador, els papers, i de nosaltres, és a dir, les caselles de memòria, els aparells de memòria secundària, el microprocessador i els perifèrics, se'n diu el **HARDWARE**.

Del que hi ha escrit als llibres, als papers, a la pissarra, a l'arxivador i del que nosaltres sabem fer, sense que ens ho manin, en direm **SOFTWARE**.

Notem que és **HARDWARE** tot el que es pot veure i tocar com la pantalla, els chips, els circuits: tots els materials que componen un ordinador; en canvi, és **SOFTWARE** el conjunt de coses que sap i pot fer un ordinador, com el sistema operatiu, les dades, els programes, i els resultats.

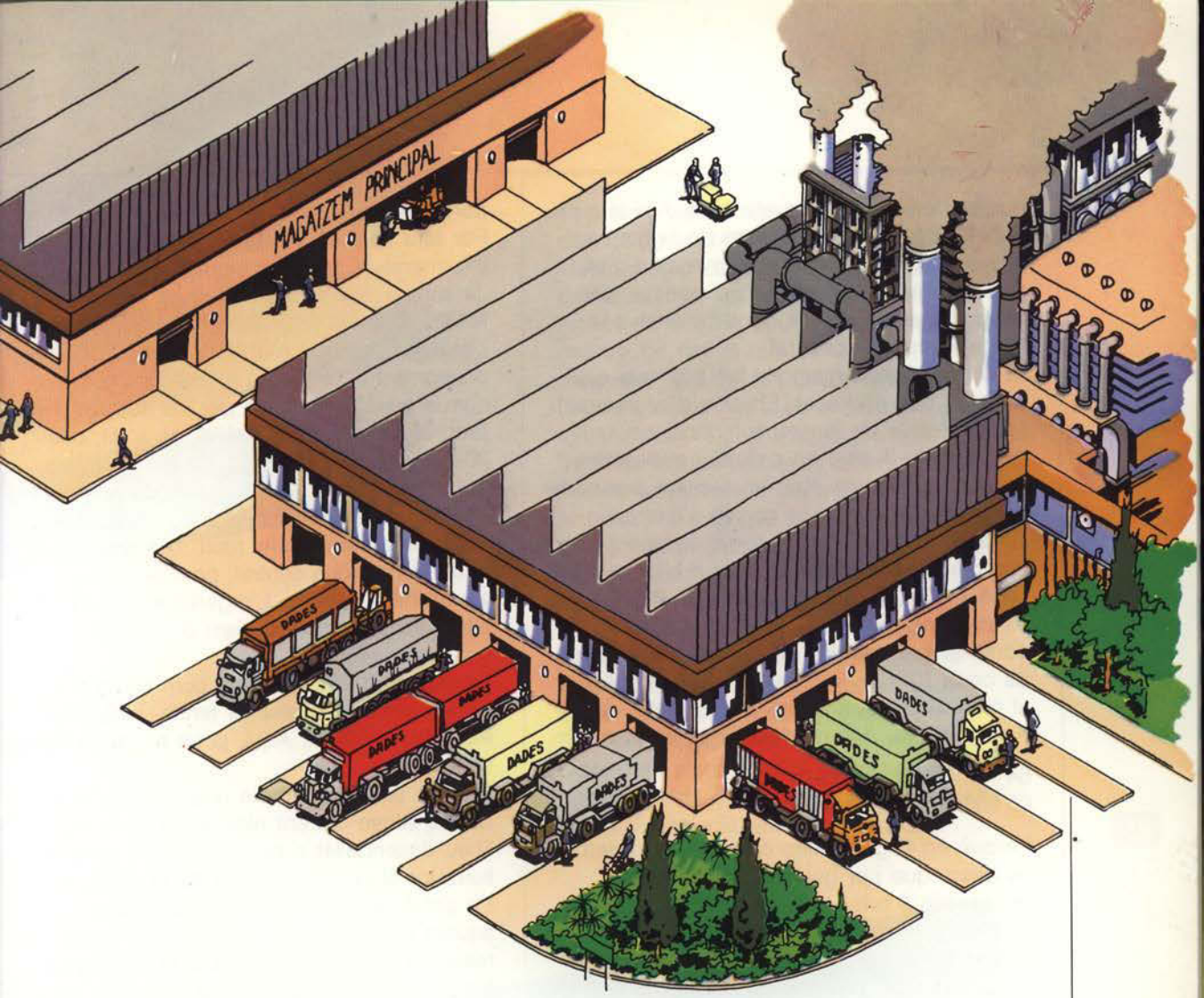
Què passa dins l'ordinador?

Doncs bé, si obríssim un ordinador, el que veuríem és una placa plena de peces negres de material de silici anomades **CHIPS**. Aquestes peces tenen com «potes» que s'enganxen en unes pistes conductores: en direm **BUSOS**.

Això que a primer cop de vista sembla tan complicat i estrany quedarà clar si pensem que aquesta placa és una vista aèria d'una gran fàbrica. Els «chips» són ara els **edificis**, les potes **les portes** i els busos **els camins** o carreteres que comuniquen els edificis entre ells.

Aquesta fàbrica, com gairebé totes, funciona amb electricitat. Quan es mana una feina a l'ordinador, direm que se li dona una **INFORMACIÓ**. Això es fa a través del teclat, és a dir, pitjant una tecla. Automàticament es genera un feix de senyals elèctrics carregats d'informació que corren pels busos, igual com un contenidor corre per una carretera. Els contenidors poden anar buits o carregats: igualment, els senyals elèctrics poden portar informació o no portar-ne.

Les primeres matèries de la nostra fàbrica són **LES DADES** i **COMANDES**. Els **productes acabats** són **els resultats**.



El lloc de treball, el **taller** de la fàbrica, és el **PROCESSADOR** del nostre ordinador; és on entren els contenidors carregats de primeres matèries (dades i comandes) i en surten contenidors carregats de productes acabats (resultats). Normalment, el taller no tracta les primeres matèries d'una a una, sinó que per començar a treballar necessita tenir-ne unes quantes alhora. És per això que el processador té 8 potes (portes), és a dir, li arriben contenidors a descarregar de vuit en vuit. Quan això passi, direm que tenim un **PROCESSADOR DE VUIT BITS** (Hi ha ordinadors que tenen processadors de 16 o 32 bits, però la majoria de micros el tenen de 8).

Fixem-nos que el nombre de bits del processador indica si el nostre ordinador treballa més o menys ràpidament. Si al taller li entren els contenidors de 32 en 32, en el mateix espai de temps en sortiran molts més productes acabats que si li entren els contenidors de 8 en 8 o de 16 en 16. En la realitat, aquesta diferència de temps és molt petita; hauríem de fer uns dos milions d'operacions per poder apreciar que l'un tarda més que l'altre.

L'activitat del processador consisteix, sobretot, a comunicar-se amb la memòria. A la memòria es guarda la informació, ja siguin primeres matèries (dades i comandes) o productes acabats (resultats). Igual com els constructors saben fer magatzems per guardar-hi productes, els electrònics saben fer **magatzems** per guardar-hi la informació

que porten els senyals elèctrics. Així la **MEMÒRIA** és el **magatzem** de la nostra fàbrica. Ja sabem que, a primer cop d'ull, això sembla estrany; però cal pensar, per exemple, que les piles o les bateries són magatzems d'electricitat.

Si entreu dins la memòria, veureu que està tot molt ben endreçat. Els bits d'informació (els materials de la nostra fàbrica) es guarden de 8 en 8 en caixes, i cadascuna porta una etiqueta amb un número per poder trobar-la fàcilment. Les caixes s'endrecen l'una al costat de l'altra correlativament. Els informàtics, de cada caixa de 8 bits en diem **UNA POSICIÓ DE MEMÒRIA O BYTE** (que es pronuncia «bait»). Per tal de poder conèixer la capacitat del nostre magatzem, de cada 1024 bytes, en diem una K. Així, si el manual diu que tenim un ordinador de 64 K vol dir que, a la memòria central, hi caben:

$64 \times 1024 = 65.536$ bytes
$65.536 \times 8 = 524.288$ bits

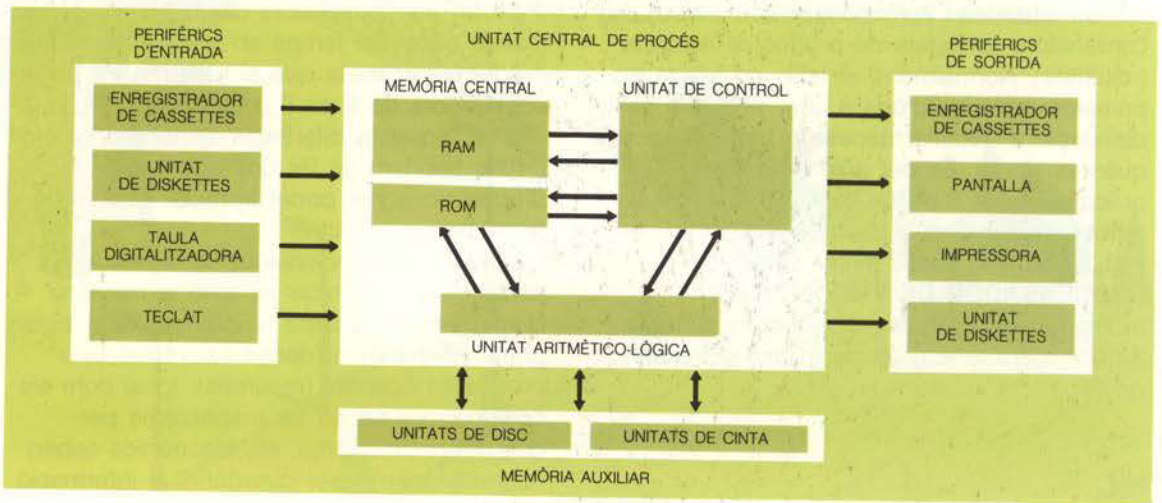
Perquè en tingueu més o menys una idea, us direm que per guardar una lletra es necessiten 8 bits, és a dir, dins la memòria central d'un ordinador de 64 K hi caben 65.536 lletres. Si considerem que el número mitjà de lletres per paraula del català és de 6, dins la memòria central d'un ordinador de 64 K hi caben, aproximadament, 10.936 paraules, és a dir, unes 32 pàgines de llibre. Per tant com més K té un ordinador, més informació hi cap a la memòria central.

Ja sabem que aquesta té dues parts: la **RAM** i la **ROM**. A la **ROM**, s'hi guarda el sistema operatiu i el **BASIC**, i això sol ocupar entre 15 K i 25 K. Per tant resten, com a espai disponible per als nostres programes, entre 39K i 48K, és a dir, unes 20 pàgines de llibre.

Tal com hem dit abans, a part de la memòria central, hi ha la memòria secundària, que serien uns magatzems molt més grans que el de la memòria central, però situats als afores de la fàbrica. L'organització d'aquests altres magatzems, podríem dir que és la mateixa.

La capacitat d'un diskette com els que es poden trobar a qualsevol tenda, està entre els 300 K i 500 K. Aquí, ja us hi caben llibres sencers.

Heu de tenir en compte que si el micro es queda sense corrent elèctric, ja sia perquè l'heu desendollat o perquè se n'ha anat la llum, tot el contingut de la **RAM** s'esborra. En canvi, la memòria secundària no té aquest problema. Així, tots el programes i resultats que vulgueu conservar al llarg del temps els haureu d'emmagatzemar en els diskettes o en els cassettes abans de tancar el micro.



Grans, mitjans i petits

Tots hem vist fotografies dels centres de control de la NASA o bé del Centre Meteorològic Europeu. Hi ha col·leccions de pantalles, l'una al costat de l'altra; però això no vol dir que hi ha un nombre molt gran de micros; no, tot el que es veu és un sol ordinador. **UN GRAN ORDINADOR**.

Un dels avantages dels ordinadors grans i mitjans davant dels micros, és que hi poden treballar moltes persones alhora; no tenen només una pantalla, un teclat i una impressora, sinó que tenen un gran nombre de cadascun d'aquests perifèrics. A més, no cal que estiguin l'un al costat de l'altre. Per exemple, hi ha Caixes d'Estalvis i Bancs que tenen d'aquests grans ordinadors, amb perifèrics repartits per tots els pobles on tenen sucursal; el mateix passa a les Universitats: és normal que existeixi un Centre de Càlcul on hi ha un ordinador, o més; però els perifèrics es troben repartits pels diferents departaments (Informàtica, Física, Psicologia, etc...) i negociats.

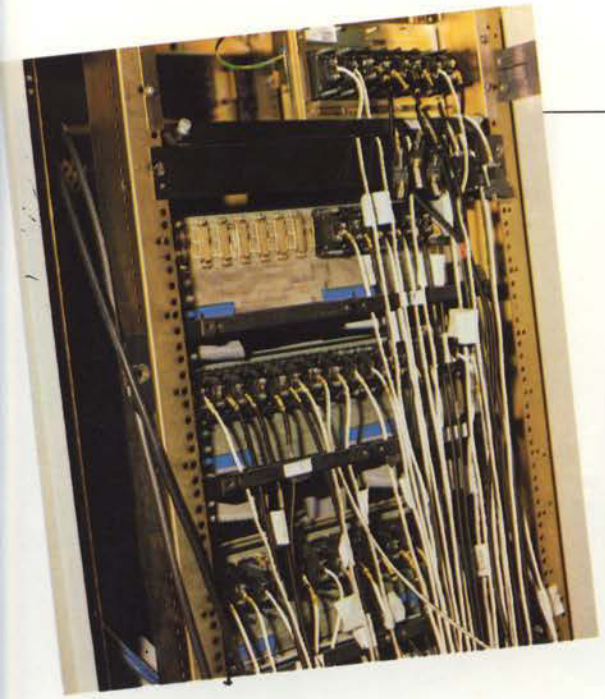
entre de control de la NASA

The Image Bank

Aquests grans ordinadors ofereixen, a més, la possibilitat de fer diverses feines alhora; poden estar llegint, escrivint llistes, enregistrant una cinta,..., i tot aparentment al mateix temps. Són capaços d'interpretar programes en llenguatges diversos. Per exemple, nosaltres podem estar fent un programa en **BASIC** i la persona de la cadira del costat pot fer-ho en **PASCAL**.
També és important el fet que se'ls poden deixar feines encarregades, en el sentit que, si tenim un programa que necessita molt de temps per executar-se, no cal que ens quedem «davant la pantalla», «mirant el cursor», sinó que podem deixar-lo amb una ordre que l'ordinador l'executi a la nit, i així queda el perifèric buit per a altres persones que puguin necessitar-lo.

El mateix comentari podríem fer per a la impressora. Suposem que volem fer 400 còpies d'una mateixa carta; aquesta operació trigarà bastant de temps; podem fer el mateix que abans, donar-li ordre de fer-ho a la nit. Mentre estem a casa ell treballa per nosaltres i l'endemà la feina estarà feta.
A part d'aquests avantatges, el que és realment important de ressaltar és el fet que aquests grans ordinadors tenen una capacitat de memòria molt superior a la dels micros que ens disposem a usar, i que l'execució de programes és molt més ràpida.

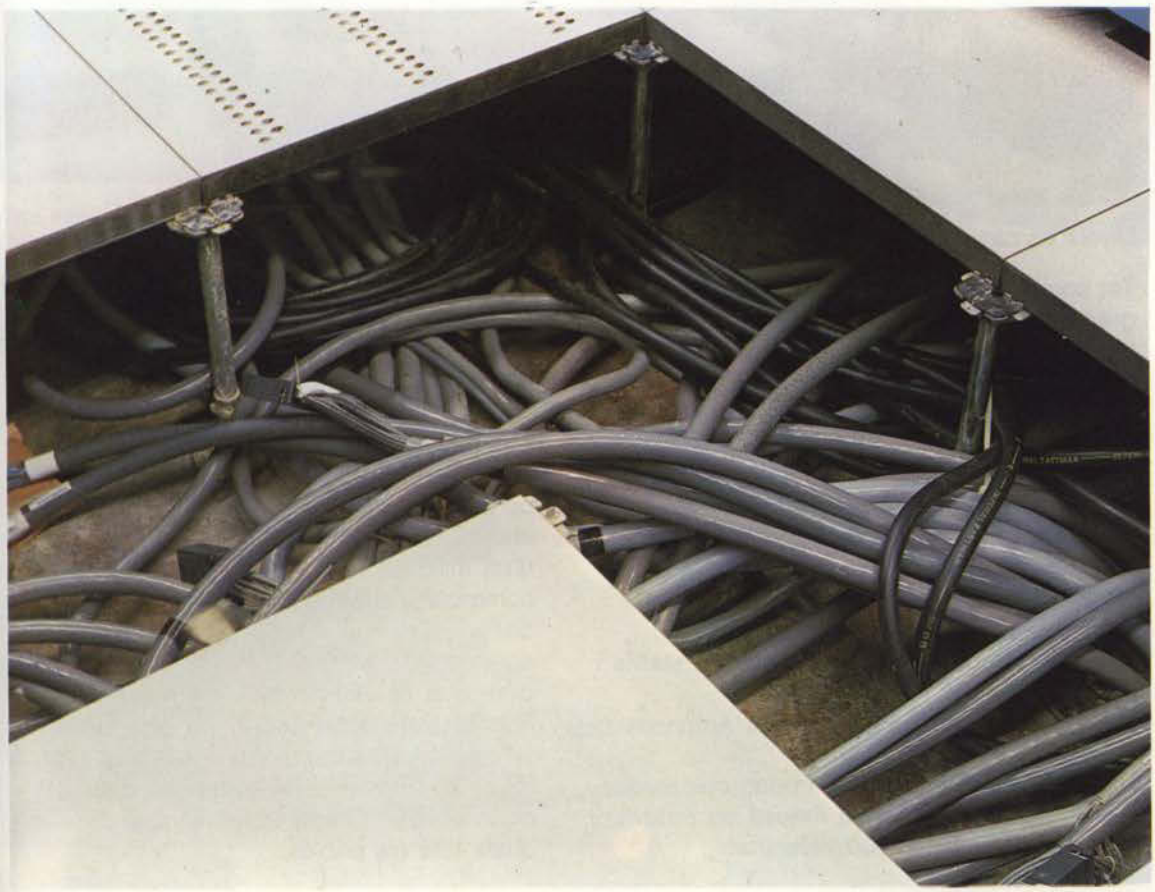
Elements de connexió (Modems) entre pantalles i ordinador (Centre de Càlcul de la U.A.B.)



Això no obstant, també necessiten unes atencions superiors. Per exemple, no poden pas estar a qualsevol lloc, han d'estar en habitacions que mantinguin la temperatura entre 15 i 29 graus centígrads i la humitat constant. A més, necessiten estar sempre «vigilats» per unes persones, anomenades **operadors**, car caldrà, en certs moments, canviar els diskettes, les cintes, connectar altres unitats de memòria secundària, establir prioritats de treballs quan l'ordinador vagi molt carregat de feina i, en general, estar a l'aguait per si hi hagués alguna complicació.

Controlador de comunicacions del ordinador (Centre de Càlcul de la U.A.B.)

Subsol d'una sala d'ordinadors



Centre de càlcul

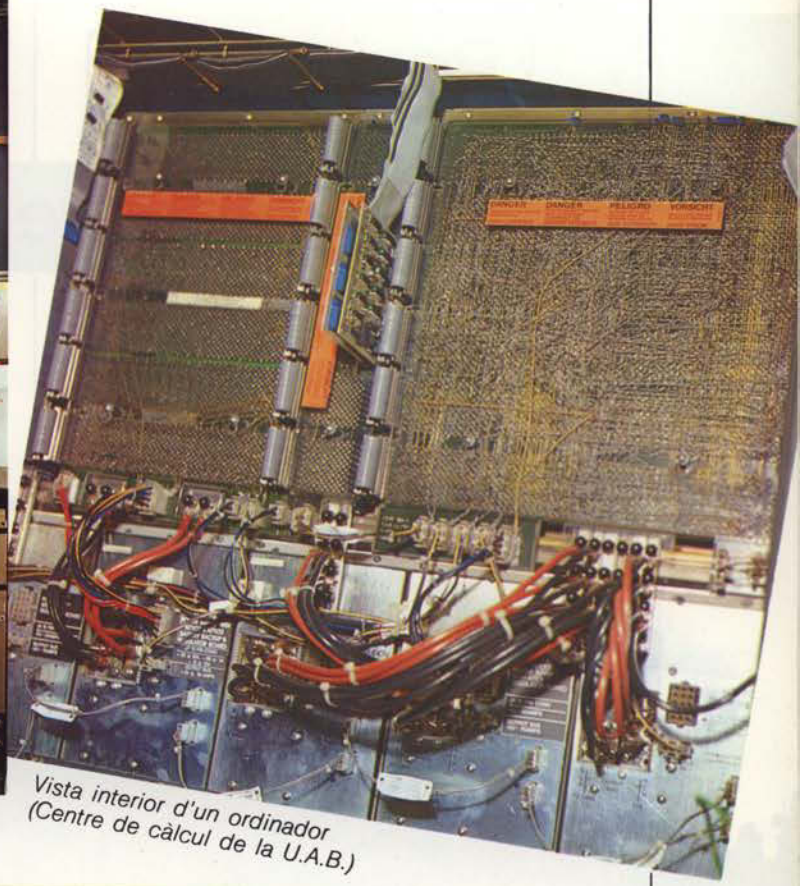
Tal com és de suposar, hi ha també una gran diferència en els preus. Per tal que ens en poguem fer més bé una idea, donarem un preu mitjà aproximat i uns quants llocs on els poden fer servir:

GRANS ORDINADORS: Costen, per terme mitjà, uns 800 milions de pessetes i els fan servir les grans institucions; per exemple, el Ministeri de l'Interior, per guardar-hi tots els afers policials; el Ministeri d'Hisenda, per regular el pagament d'impostos i les delcaracions de renda; les Caixes d'Estalvis i els grans Bancs; i fins i tot alguns Departaments de les Comunitats Autònomes.

ORDINADORS MITJANS: Poden costar, per terme mitjà, uns seixanta milions de pessetes i els podeu veure a les Universitats, Companyies de fonts energètiques (electricitat, gas,...), en sucursals de grans empreses multinacionals, en grans Ajuntaments i companyies d'assegurances.

MICROODINADORS: Són molt econòmics, es poden trobar per un preu entre vint mil i cent mil pessetes per a usos personals i entre les quatre-centes mil i una mica més d'un milió de pessetes per a feines comercials, administratives, gestió de botigues, petites empreses, magatzems, agències de viatges...
Ben aviat la situació de la informàtica estarà normalitzada. Igual com hi ha laboratoris i tallers, hi haurà microordinadors a gairebé totes les escoles, i tal com hi ha casals i biblioteques hi haurà aules d'informàtica a quasi tots els pobles.

Vista interior d'un ordinador
(Centre de càlcul de la U.A.B.)



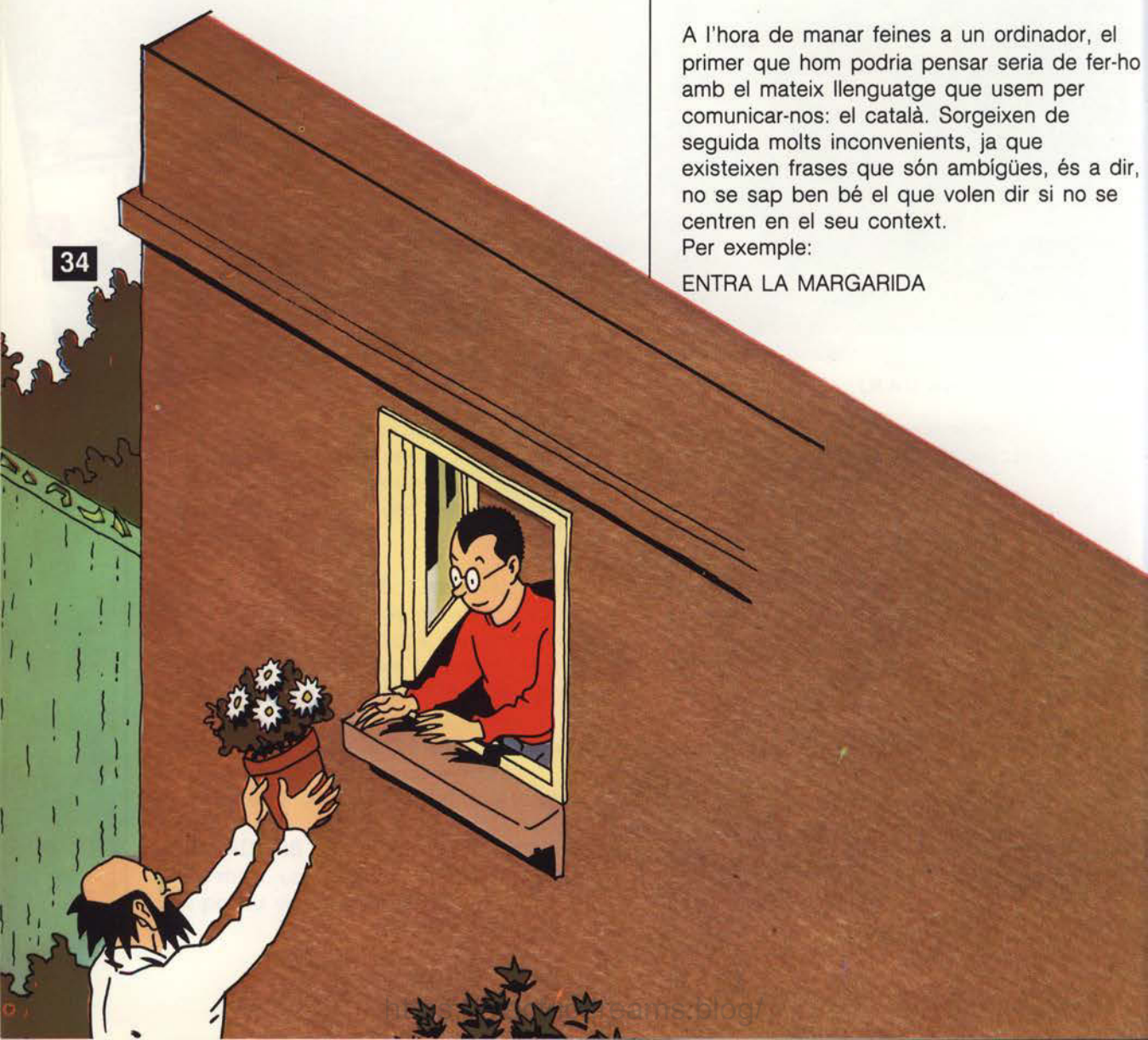
Vista interior d'un ordinador
(Centre de càlcul de la U.A.B.)

Microordinador

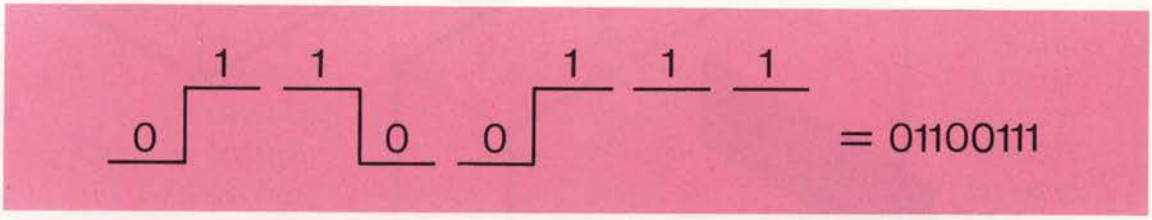
Per què s'ha d'aprendre BASIC?

A l'hora de manar feines a un ordinador, el primer que hom podria pensar seria de fer-ho amb el mateix llenguatge que usem per comunicar-nos: el català. Sorgeixen de seguida molts inconvenients, ja que existeixen frases que són ambigües, és a dir, no se sap ben bé el que volen dir si no se centren en el seu context.
Per exemple:

ENTRA LA MARGARIDA



ens estan dient que una noia anomenada Margarida ve a veure'ns? O ens estan manant que anem a fora i entrem una torreta amb una planta? Vés a saber!
Un ordinador, davant aquesta frase, quedaria ben desconcertat. Els ordinadors, doncs, no poden parlar ni català, ni anglès, ni francès ni cap dels llenguatges que a nosaltres ens van tan bé per entendren's.
Bé, si els ordinadors no poden parlar com nosaltres, per què no parlem nosaltres com ells?
Com ja s'ha dit, qualsevol ordre que donem a un ordinador és transformada en un seguit de senyals elèctrics que són una ona. Per escriure-ho millor en un paper podem adoptar l'escriptura 0 i 1, per indicar les crestes (1) i les valls (0) de l'ona.



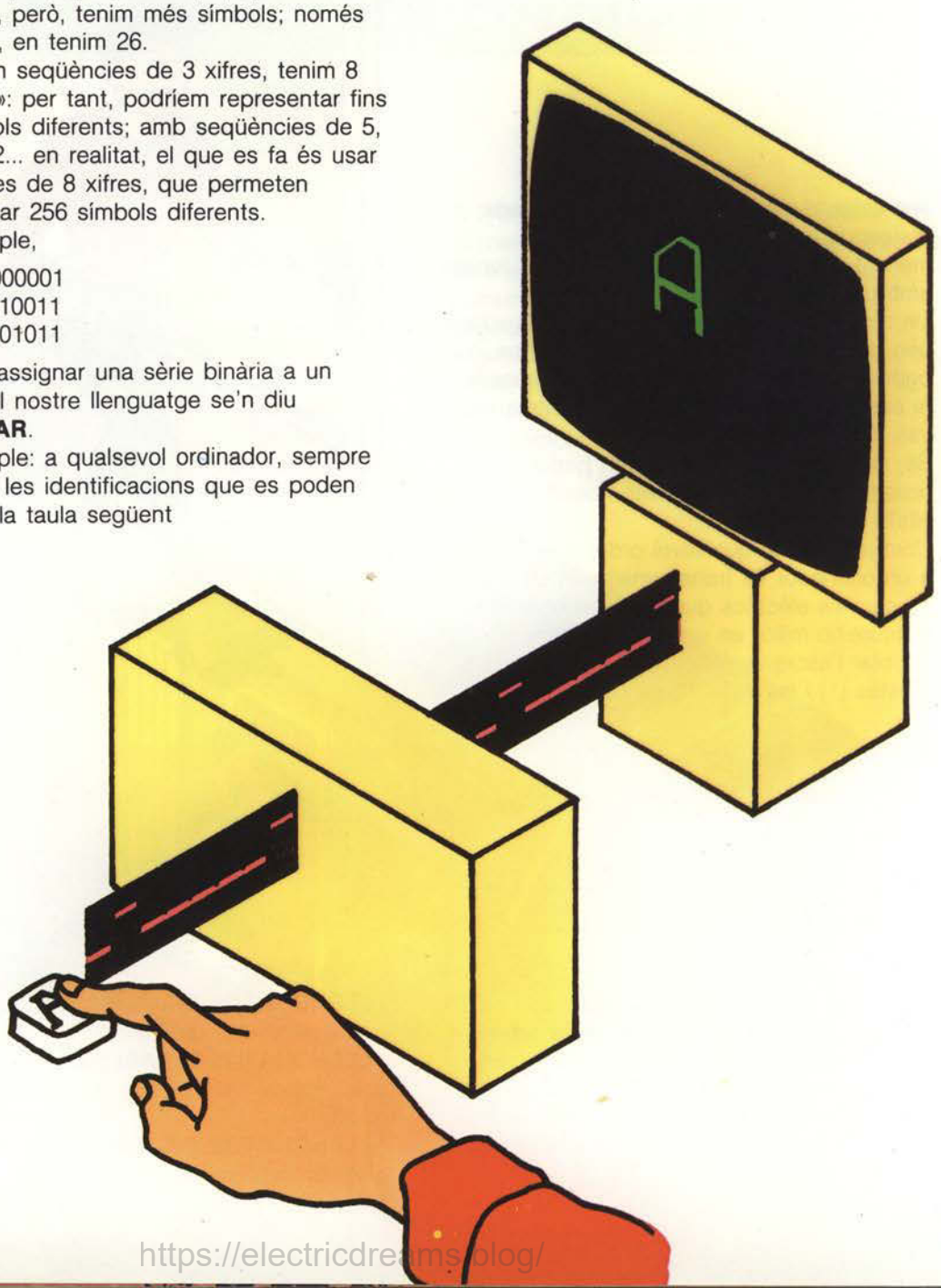
És a dir, per «parlar», un ordinador no usa 26 lletres com nosaltres. **Només en fa servir dues**. Combinant-les convenientment crea moltes paraules, combinant les paraules un plec de frases i així obté el seu idioma.
Aquest idioma propi dels ordinadors rep el nom de:

–**LLENGUATGE MÀQUINA** (perquè és el que «parlen» les màquines)
–**LLENGUATGE BINARI** (perquè només té dues lletres, i «bi» vol dir dos: pensem per exemple en «bifurcació, bilingüisme...»)
–**LLENGUATGE DIGITAL** (Del mot «dígit», que vol dir xifra).

Així per parlar amb un ordinador hem de convertir les nostres lletres, xifres i símbols habituals (com ara . , : + .. !) en seqüències de zeros i uns, és a dir en un senyal elèctric.
Si prenem seqüències binàries de dues xifres tenim quatre «números»: 00, 01, 10, 11, mitjançant els quals podem representar, per exemple A, B, C, D, o bé 1, 2, 3, 4.
Nosaltres, però, tenim més símbols; només de lletres, en tenim 26.
Si prenem seqüències de 3 xifres, tenim 8 «números»: per tant, podríem representar fins a 8 símbols diferents; amb seqüències de 5, n'hi ha 32... en realitat, el que es fa és usar seqüències de 8 xifres, que permeten d'identificar 256 símbols diferents.
Per exemple,

A = 01000001
3 = 00110011
+ = 00101011

Del fet d'assignar una sèrie binària a un símbol del nostre llenguatge se'n diu **CODIFICAR**.
Per exemple: a qualsevol ordinador, sempre es donen les identificacions que es poden veure en la taula següent



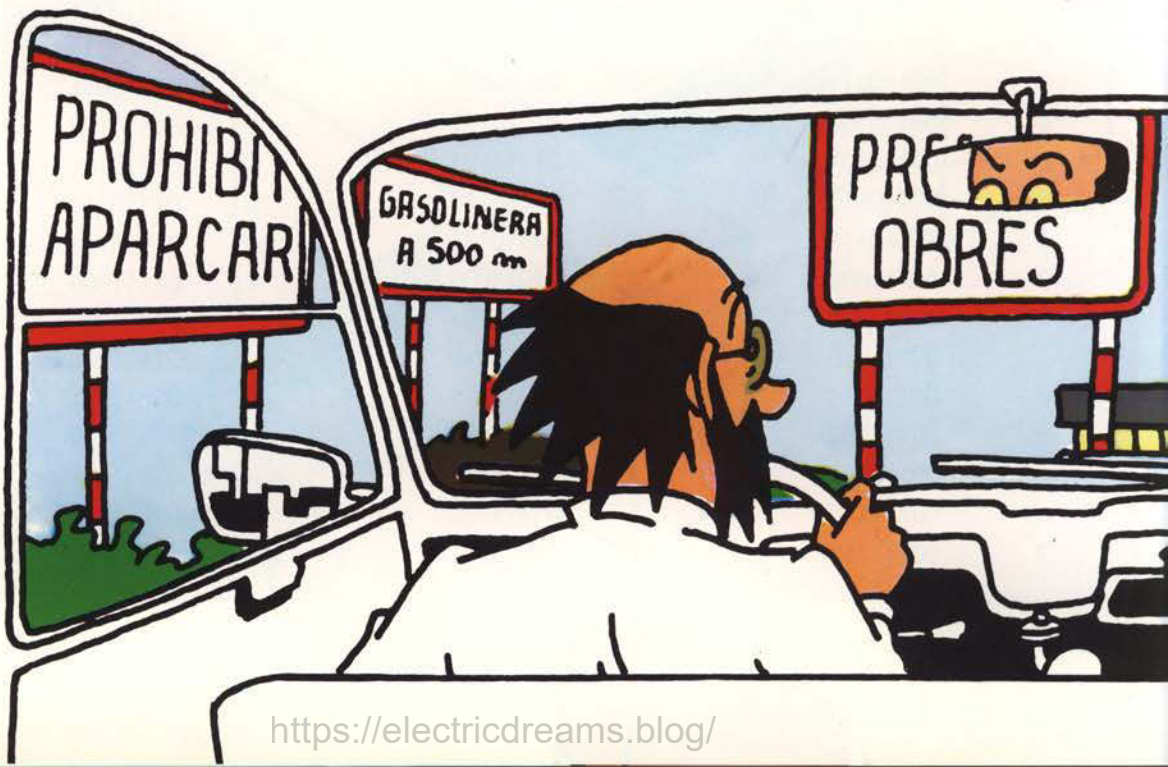
TAULA DE CORRESPONDÈNCIES

Caràcter	Octet	Caràcter	Octet	Caràcter	Octet
espai	00100000	å	01000000		01100000
!	00100001	A	01000001	a	01100001
"	00100010	B	01000010	b	01100010
#	00100011	C	01000011	c	01100011
$	00100100	D	01000100	d	01100100
%	00100101	E	01000101	e	01100101
&	00100110	F	01000110	f	01100110
'	00100111	G	01000111	g	01100111
(	00101000	H	01001000	h	01101000
)	00101001	I	01001001	i	01101001
*	00101010	J	01001010	j	01101010
+	00101011	K	01001011	k	01101011
,	00101100	L	01001100	l	01101100
-	00101101	M	01001101	m	01101101
.	00101110	N	01001110	n	01101110
/	00101111	O	01001111	o	01101111
0	00110000	P	01010000	p	01110000
1	00110001	Q	01010001	q	01110001
2	00110010	R	01010010	r	01110010
3	00110011	S	01010011	s	01110011
4	00110100	T	01010100	t	01110100
5	00110101	U	01010101	u	01110101
6	00110110	V	01010110	v	01110110
7	00110111	W	01010111	w	01110111
8	00111000	X	01011000	x	01111000
9	00111001	Y	01011001	y	01111001
:	00111010	Z	01011010	z	01111010
;	00111011	¡	01011011	{	01111011
<	00111100	–	01011100	|	01111100
=	00111101	¿	01011101	}	01111101
>	00111110	^	01011110	~	01111110
?	00111111	_	01011111	bloc	01111111

Així doncs, ja podem parlar amb els ordinadors.
És feixuc, oi?
Que lent! Quin embolic!
Doncs, a més, per acabar-ho d'adobar, no tots els ordinadors estan codificats igual. N'hi ha que tenen tecles especials, com ara la que permet d'esborrar la pantalla. En uns la codificació d'aquesta pot ésser, per exemple 1001001, en altres 11100011 o 11010111...
Abans hem dit que els ordinadors no poden parlar català, però ara sembla que també queda clar que nosaltres no podem parlar «llenguatge màquina».

Cal inventar un nou llenguatge per comprendre's!

El fet de crear un llenguatge diferent de l'habitual per resoldre certs problemes que se'ns presenten quotidianament és ben conegut per tothom. Per exemple:
– Per saber el temps que tardarem per anar de Barcelona a Figueres amb cotxe usem el llenguatge matemàtic:
$t = e/v = 120 \text{ km}/80 \text{ km/h} = 1$ hora i mitja.
En llenguatge escrit ens donaria una frase feixuga i poc entenedora; en canvi, així, amb una ratlla queda escrit i ben clar.
– Per dirigir el trànsit d'una ciutat s'usa el codi de la circulació; perquè si ho fessin amb llenguatge escrit, una determinada ordre necessitaria un cartell tan gran que no veuríem els edificis de darrera (penseu, per exemple en el símbol que representa la prohibició de girar a l'esquerra).
– Estant dalt d'un vaixell, per saber si hom s'apropa a la costa, mira de veure el far.
– Per enviar un telegrama, el «morse».
Així, doncs, no és estrany que per parlar amb els ordinadors ens inventem un lenguatge nou, específic per a aquest cas, un llenguatge més proper a l'home que el llenguatge màquina, però prou simple i precís perquè l'ordinador mateix se'l pugui traduir a binari. És a dir, nosaltres ens apropem a la màquina en el sentit que no parlarem en català, sinó en un llenguatge que li serà més simple i entenedor, al mateix temps que la màquina s'apropa a nosaltres i accepta de traduir ella mateixa aquest nou llenguatge a seqüències de zeros i uns.
En l'actualitat encara no s'ha trobat el llenguatge informàtic perfecte; hi ha al voltant de 400 llenguatges diferents, més o menys especialitzats en diferents temes. Els més importants, els teniu a la taula següent:



LLENGUATGE	DATA NAIXEMENT	APLICACIÓ
FORTRAN	1954	Càlculs científics
COBOL	1959	Gestió empresarial
ALGOL	1960	Càlculs científics
LISP	1960	Intel·ligència artificial
RPG	1962	Administració
BASIC	1964	De caire general. Especialment interessant per aprendre a programar
PASCAL	1970	Enginyeria
FORTH	1970	Robòtica
LOGO	1971	Ensenyament
C	1971	Sistemes gràfics

Connecta el micro

Si disposeu d'un micro, ben segur que ja teniu ganes de fer-lo funcionar. Ja n'hi ha prou d'explicacions retòriques; és l'hora de començar a usar-lo.

El primer que cal fer és endollar-lo, tenint en compte la tensió elèctrica. Normalment va amb 210-220 volts, però cal mirar-ho, en el manual (primers fulls) o en el micro (darrera la pantalla o sota el teclat). Si el vostre aparell és dels complets, dels que ja porten pantalla, només cal endollar-lo i accionar tots els botons a la posició ON. En l'altre cas, us cal usar el televisor de casa. Probablement,

Microordinador i alguns perifèrics

amb el micro us han venut un transformador que té dos fils. L'un ha d'anar connectat al micro i l'altre al corrent elèctric. També hi ha per dins la caixa, un fil que per una banda s'ha de connectar al micro i per l'altre al lloc on hi ha el fil que va a l'antena del televisor (així, doncs, cal treure el fil de l'antena i posar en el seu lloc el del micro). Engegueu el televisor, després el micro i en un canal qualsevol del televisor comenceu a sintonitzar-lo fins que desaparegui la imatge habitual de televisió i «la neu». Normalment es veurà la pantalla neta, amb el nom de la marca del vostre micro.

Cal tenir ben present que s'ha d'engegar sempre primer el televisor que el micro i en acabar de treballar és al revés, cal apagar primer el micro i després l'aparell de televisió. No s'han de desendollar mai els perifèrics mentre el micro estigui funcionant. És preferible no obrir el micro per mirar com

són els «chips» de memòria o per veure si hi ha algun fil de colors; els micros van precintats i si l'obriu i toqueu els «chips» amb els dits és molt probable que s'espatlli. A més es perdria la garantia, és a dir, el fabricant no es responsabilitzaria de l'avaria. La reparació pot costar molts diners, i fins i tot pot passar que no es pugui arreglar i s'hagi de llençar. Nosaltres sabem que fa il·lusió, això de veure un micro per dins; és per això que aquí mateix, en aquest llibre, n'hem posat fotografies, i prometem continuar posant-ne moltes més. Fins i tot mostrarem els «chips» per dins fent-ne ampliacions. Es veuran així coses que obrint el micro no podríeu veure.

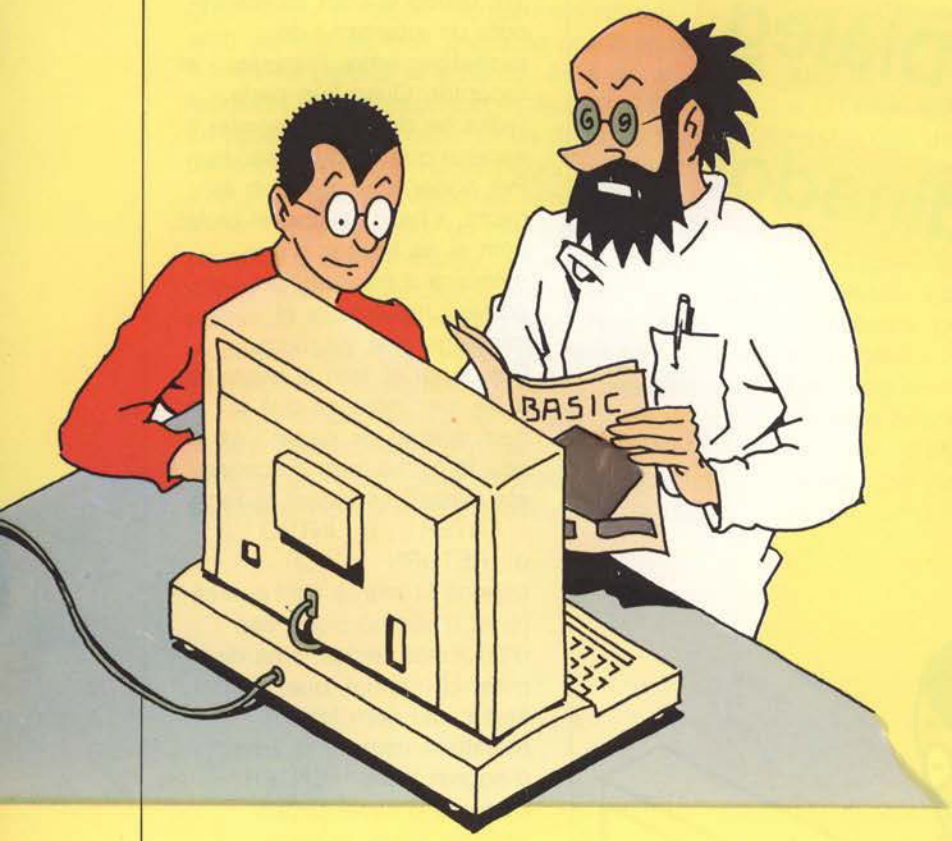
Sabem que, aquesta explicació, no l'hem fet gaire llarga ni completa. El motiu és que tot això depèn de la marca d'ordinador de què es disposi. El que no s'ha de fer és començar a endollar fils per aquí i per allà abans d'haver llegit atentament les instruccions que ens dóna el manual. Si tot i això no us en sortiu, es qüestió de preguntar a un amic que tingui un aparell com el vostre des de fa més temps o, si no, d'anar a la botiga on l'heu comprat.

Per poder començar a dialogar amb el micro necessiteu el teclat per entrar-hi els vostres missatges i la pantalla per obtenir les seves respostes. Va bé disposar, a més, d'una impressora, per poder conservar els resultats en paper.

Un micro sense perifèrics seria com una persona sense sentits, sense veure-hi ni escoltar, ni parlar, ni escriure.... Només un cervell que no rep informacions de l'exterior ni pot comunicar res als altres.

Diàleg amb l'ordinador

Tot diàleg educat es manté com un intercanvi de missatges entre l'emissor i el receptor. Quan l'un parla, l'altre ha de saber escoltar i esperar que li toqui el seu torn.
Per poder dialogar amb el micro, s'ha d'utilitzar el teclat, com si es tractés d'una màquina d'escriure, per poder-li transmetre el missatge que vulguem.
Per donar el torn al micro (és a dir, per indicar-li que ja hem acabat de parlar i és el seu torn), haurem d'acabar els missatges pitjant la tecla ENTER, o INTRO, o RETURN o CR, segons la marca de l'aparell (si el teclat no porta cap d'aquestes tecles, s'ha de mirar el manual, que, ben segur, diu com fer-ho).
Nosaltres usarem al llarg d'aquest llibre ENTER
Després d'haver pitjat aquesta tecla, el micro contestarà, o farà una pregunta, o esperarà que se li entrin més missatges. El que és cert és que ell haurà rebut aquell missatge que li hem volgut transmetre.
El teclat recorda el d'una màquina d'escriure, però segons quin sigui el micro hi ha tot un reguitzell d'altres tecles que aquella no tindria i que realitzen unes funcions especials que haurem de descobrir amb el manual, o tornant a demanar-ho al nostre amic experimentat.
Una d'aquestes tecles especials, i que no porten les màquines d'escriure, és la DELETE, que permet



d'esborrar els caràcters que ja hem escrit i que considerem equivocats. Aquesta tecla se sol indicar per DEL . Això sí, només esborra caràcters abans de pitjar ENTER . Si veiem que ens hem equivocat però ja hem pitjat ENTER , hem d'escriure el missatge de nou intentant de fer-ho bé aquest segon cop.
Per poder veure tots els missatges que es van introduint (transmetent) al micro, i per poder veure també els que ell enviarà, hem de disposar d'una pantalla.
Una dada important a conèixer de la pantalla que s'ha d'utilitzar és el nombre de línies que es poden escriure, així com quants caràcters es poden escriure en cada línia. També serà interessant de saber el nombre de colors de què es disposa.
Si volguéssim fer dibuixos molt complicats i molt ben perfilats, aleshores la pantalla hauria d'ésser un xic més sofisticada, parlaríem d'una pantalla d'alta resolució. En una pantalla normal (d'unes 600 línies) els dibuixos com ara una circumferència no queden ben arrodonits, sinó que es veuen fets de quadrets. Les pantalles d'alta resolució són, doncs, pantalles amb moltes més línies (de 3.000 a 4.000) i encara que els dibuixos també siguin fets de quadrets, com que són tan petits l'ull humà no els percep i fan la sensació d'un dibuix continuat.
Ja tenim teclat i pantalla connectats al micro.
– Però, i com ens entendrem?
Sembla ben difícil de poder iniciar un diàleg, oi?
Bé, ja sabem que els homes i les màquines han arribat a un acord creant llenguatges comuns. Un d'aquests llenguatges és el BASIC. Aprendrem a escriure els nostres missatges en BASIC i l'ordinador tindrà un intèrpret que convertirà el que li haurem dit a llenguatge màquina. Quan l'ordinador vulgui enviar-nos algun missatge, també l'haurà d'enviar traduït a BASIC perquè el poguem entendre.
La major part dels ordinadors, porten el lleguatge BASIC, i el seu intèrpret, incorporats a la memòria central, més concretament a la ROM. Si aquest no eś el nostre cas, ens hauran donat una **cassette**, o un **diskette**. Llavors, el primer que cal fer és «carregar» el BASIC. És a dir, traslladar-lo de la memòria secundària a la memòria central.

Què, sembla un embolic? No, home no! Provem un petit diàleg.
Com tot bon ciutadà abans de començar qualsevol treball, saludem. Escrivim el missatge HOLA i tot seguit pitgem la tecla ENTER , i esperem la resposta.
Què, l'ordinador no n'ha fet cabal? Diu que hem fet un error?
Això és perqué el micro no ha entès el que li volíem dir. Pensem que és molt intransigent, i ho vol tot clar i ben mastegat. Per fer això hem d'usar una paraula del BASIC. Es tracta de la paraula PRINT . Amb aquest mot es demana a l'ordinador que escrigui a la pantalla aquells missatges que nosaltres vulguem.
Provem-ho!
Escrivim ara PRINT HOLA i pitgem, la tecla ENTER .
Tampoc no ha funcionat?...
És clar! Encara no hem satisfet totes les exigències del nostre interlocutor. Els missatges que vulguem que se'ns escriguin hauran d'anar entre cometes (") .
Provem-ho ara:

Escrivim PRINT "HOLA".
Pitgem la tecla ENTER.

La tercera és la bona, oi? Ara sí, ha imprimit la paraula HOLA.

Utilitzant la paraula PRINT i recordant-nos de posar els missatges que volguem escriure entre cometes, podem repetir aquest exemple posant el nostre nom i cognoms, el nom de la nostra ciutat,... etc.
No hem de tenir por d'escriure; encara que ens equivoquem, l'ordinador no es trenca. Si fem una barrabassada, ell, com si estigué enfadat, ens enviarà un avís dient que hem fet qualsevol tipus d'error.
Si anem fent aquestes proves potser ens amoïnarem, perquè tot va quedant escrit a la pantalla.
Per poder netejar-la també tenim un paraula de BASIC que ens ho permet. Es tracta de la paraula CLS .
Bé, provem-ho!
Escrivim CLS i pitgem la tecla ENTER. Ha quedat la pantalla neta.
Podem encadenar dues o més ordres en una mateixa línia. Per tal que el micro ens entengui, les hem de posar separades per dos punts (:)
Per exemple, provem d'escriure CLS: PRINT "HOLA" i pitgem la tecla ENTER. Això és, s'ha esborrat la pantalla i després ens ha sortit només el missatge HOLA. Així queda més bonic.
Provem ara de fer el següent:
(recordem-nos de pitjar ENTER després de cada línia)

```
CLS
LET ANYS = 12
PRINT ANYS
>12
```

Ep! si nosaltres havíem posat PRINT ANYS, com és que ha escrit 12?
L'ordinador ha escrit 12 perquè ha identificat el mot ANYS amb el número 12 en escriure-li LET ANYS = 12 a la segona ratlla. Podem provar de fer-ho posant altres números (Fixem-nos que ara ANYS no va entre cometes).
Si en volem més, podem fer el següent:
(recordem-nos de pitjar ENTER després de cada línia)

```
CLS
LET CIUTAT$ =
"BARCELONA"
PRINT CIUTAT$
> BARCELONA
```

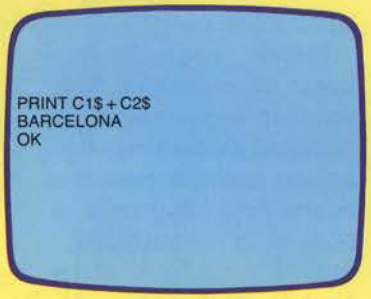
Aquí hem fet el mateix d'abans, però en lloc d'identificar un mot, en aquest cas CIUTAT, amb un número, ho hem fet amb un nom, BARCELONA; fixem-nos que en una identificació d'aquest tipus (mot = nom) darrera del mot posem el signe $ i hem d'escriure entre cometes ('').
Hi ha una altra cosa divertida: enganxar mots. Fem-ho!

```
CLS
LET C1$ = "BARCE"
LET C2$ = "LONA"
PRINT C1$
> BARCE
```



```
CLS
PRINT C2$
> LONA
```

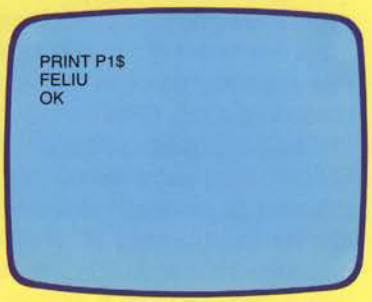

```
CLS
PRINT C1$ + C2$
> BARCELONA
```



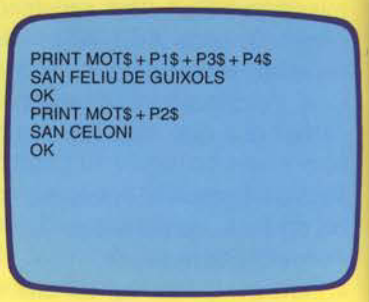
Un altre joc podria ésser:

```
CLS
LET MOT$ = "SANT"
LET P1$ = "  FELIU"
LET P2$ = "  CELONI"
LET P3$ = "  DE"
LET P4$ = "  GUIXOLS"
```


```
CLS
PRINT P1$
> FELIU
```




```
CLS
PRINT MOT$ + P1$ + P3$
+ P4$
> SANT FELIU DE
GUIXOLS
PRINT MOT$ + P2$
> SANT CELONI
```



El primer caràcter de la identificació P1$, P2$, P3$ i P4$ és un blanc, per tal d'aconseguir una escriptura clara.
Què passa si li posem PRINT MOT$ + P2$ + P4$? Podem continuar amb aquest o pensar-ne d'altres.

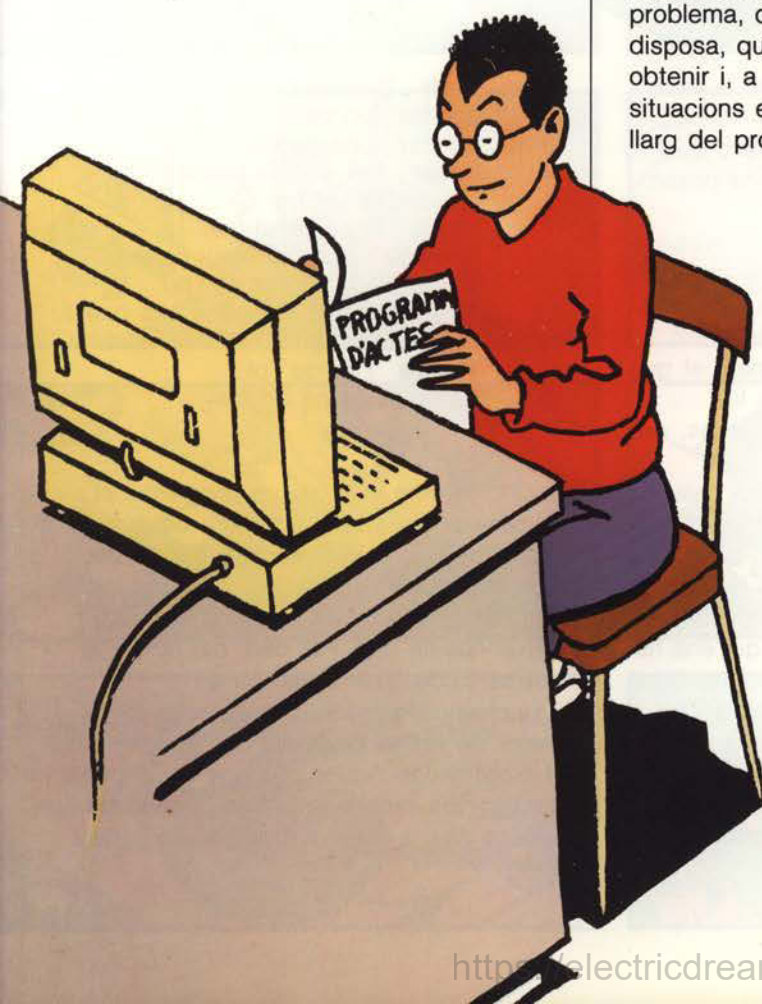
Una cosa darrera l'altra

Del problema a l'algorisme

Si tenim un problema i el volem resoldre amb l'ajut d'un ordinador, el que hem de fer és **un programa**. Hem d'ensenyar al micro a fer el que nosaltres necessitem, hem d'acabar la nostra màquina per tal que faci una feina determinada.

Ja coneixem el significat de la paraula programa. Ben segur que hem fullejat el programa d'actes de la Festa Major o hem vist, al final dels diaris, la programació de la televisió; a l'inici del curs, el primer que fa el professor és escriure el programa a la pissarra...; per als informàtics, el sentit de la paraula programa és el mateix. Però hem de tenir en compte que l'ordinador fa sempre una feina darrera l'altra i no en pot fer dues alhora; només entén ordres senzilles i no resol mai un imprevist.

Programar no és una feina fàcil, cal analitzar molt profundament quin és realment el problema, quines són les dades de què es disposa, quins són els resultats que es volen obtenir i, a més, cal preveure totes les situacions estranyes que es poden produir al llarg del procés.



Prenguem uns quants exemples de la vida quotidiana.

Suposem que volem fer un ou ferrat i que no en sabem; és el primer cop que ens posem davant els fogons.

Si ens han dit que «hem de tirar el rovell i la clara de l'ou a l'oli calent, esquitxar l'ou amb aquest oli calent durant mig minut i treure'l», segurament pensarem: això està fet! Però recordem que hem de fer exactament el que ens han dit i res més. Sabem que les accions queden determinades per verbs, així, la primera acció serà:

Tirar el rovell i la clara de l'ou a l'oli calent.

Fem-ho! ... Hi ha problemes, oi? On és l'ou? L'oli és dins l'ampolla i fred. I hem quedat que no podem fer res que no ens hagin dit.

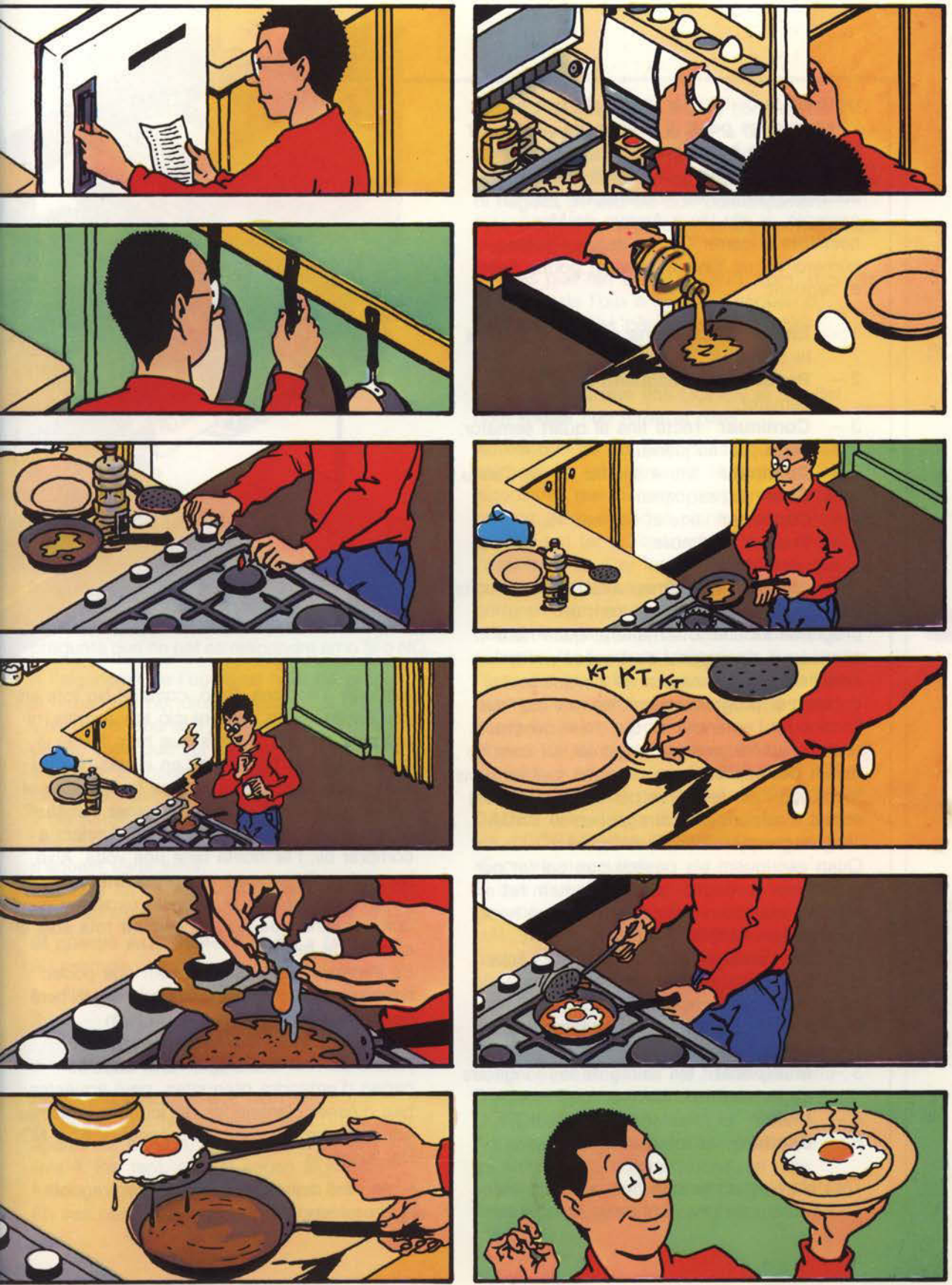
En aquest moment ens passa el mateix que a un ordinador mal programat. El qui ens ha passat la informació i ha fet de programador s'adona del que ha fet i es disposa a arreglar-ho ràpidament. El primer que recorda es que ens ha de dir les coses més detalladament i l'una darrera l'altra.

Ens ho torna a dir de la següent manera:

1 – **Obrir** la nevera.
2 – **Agafar** un ou.
3 – **Agafar** una paella.
4 – **Posar** l'oli dins la paella.
5 – **Encendre** el foc.
6 – **Posar** la paella al foc.
7 – **Esperar** que l'oli sigui calent.
8 – **Trencar** la closca de l'ou.
9 – **Tirar** el rovell i la clara a la paella.
10 – **Esquitxar** l'ou amb l'escumadora.
11 – **Treure** l'ou de l'oli.
12 – **Posar-lo** dins un plat.

Quan volguem escriure un programa, el primer que cal fer és entendre bé quin és el problema que volem resoldre per tal de descompondre'l en accions senzilles, com hem fet en el cas de l'ou ferrat, sense deixar-nos-en cap. Per cert: cal tancar la nevera i no deixar el foc encès.

En l'exemple següent, nosaltres farem el paper de programadors, i una amiga nostra el d'ordinador. Aquesta amiga viu al poble on passem les vacances, i l'hem convidada uns quants dies a casa nostra. Ella no coneix



gaire bé la ciutat. Per tant, com que no volem que es perdi, li haurem de programar l'itinerari que haurà de seguir: anirem amb compte i li donarem un seguit d'accions senzilles, completes i ordenades. Per exemple, si ella viu a Arenys de Mar i nosaltres al carrer Comte de les Pastes número 15, un itinerari detallat podria ésser el següent:

1 – **Entrar** a Barcelona per l'autopista de la costa.
2 – **Posar-se** a l'esquerra en el primer semàfor.
3 – **Continuar** recte fins al quart semàfor.
4 – **Girar** a l'esquerra.
5 – **Continuar** fins a l'edifici de «la Caixa».
6 – **Girar** a l'esquerra.
7 – **Continuar** fins al número 15.
8 – **Trucar** el timbre.

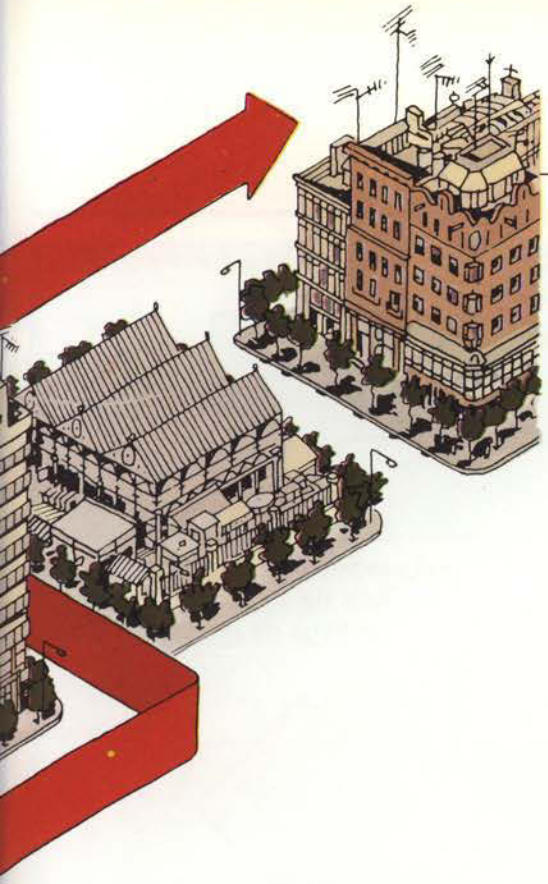
Ben segur que ens vénen a la memòria molts casos en què hem actuat com un programador. De tota manera, quan hem aconseguit de separar en trossos petits i senzills un problema, no vol dir que ja tinguem el programa escrit de manera que l'ordinador l'entengui. El que hem construït està en un llenguatge col·loquial, tal com ho diríem per explicar-ho a un amic; per tant, els mots que hem fet servir per escriure aquest seguit d'accions no corresponen al llenguatge que entén l'ordinador.
Quan escriguem els passos que cal fer per resoldre el problema tal com ho hem fet en els dos exemples d'abans, direm que hem construït un **ALGORISME**.
Ara farem un algorisme per calcular l'àrea d'un triangle:

1 – **Llegir** la longitud de la base.
2 – **Llegir** la longitud de l'altura.
3 – **Multiplicar** els valors de les longituds de la base i de l'altura.
4 – **Dividir** el resultat per 2.
5 – **Escriure** el resultat de la divisió.

Tots els algorismes que hem vist fins ara consisteixen en una seqüència única i ben

formada d'accions. Però, compte! No tots els problemes tenen una solució tan senzilla, ni tan sols els dos primers que hem fet. Què faríem si no hi hagués oli en el cas de l'ou ferrat, o si la nostra amiga es trobés amb el carrer tallat entre el segon i tercer semàfor? Nosaltres ho tenim molt clar, oi? Aniríem a comprar oli, i la xicota faria una volta. Això, ho podem fer perquè tenim **iniciativa** pròpia, però l'ordinador no té aquesta capacitat, és una màquina que **no pot decidir** tota sola el que ha de fer.
Es imprescindible que pensem que poden donar-se situacions d'aquesta mena. N'hem de preveure com més millor, i hem d'ensenyar a l'ordinador com sortir-se'n.
Abans de res, direm que l'ordinador és capaç d'entendre preguntes, però aquestes han d'ésser plantejades de forma que només admetin com a possibles respostes, SÍ o NO: Per exemple:

– és verd aquest cotxe? és una pregunta que es respon amb SÍ o NO.



– de quin color és aquest cotxe? és una pregunta que no pot es respondre amb SÍ o NO.

A l'algorisme de l'ou ferrat hem de preguntar si hi ha oli o no, dins l'ampolla. La pregunta s'ha de fer abans d'intentar posar l'oli a la paella, cosa que vol dir que ha d'anar entre les línies, o accions, 3 i 4 de l'algorisme. Per tal de no canviar la numeració, en direm línia 3 bis (entre 3 i 4).
Una manera natural de fer preguntes d'aquest tipus seria:
SI passa tal cosa, **ALESHORES** fer tal altra cosa; altrament, continuar com si res, o sia:

SI qüestió **ALESHORES** acció a fer.
A l'exemple:

3 – **Agafar** una paella
3 bis – **Si** no hi ha oli, **aleshores** anar a comprar-ne
4 – **Posar** l'oli la paella.

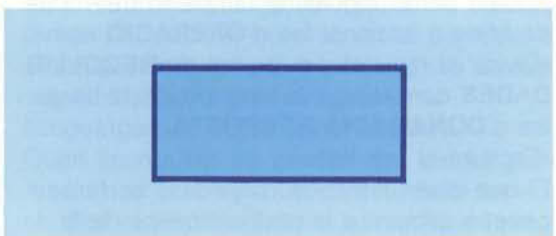
Anem a analitzar el que hem fet.
En el cas que hi hagi oli, la línia 3 bis no ens aporta res de nou, passem directament a la línia 4. Per tant, aquest segon algorisme funciona igual que el que havíem fet primer.
En cas que no hi hagi oli, hem de fer el que ens diu la linia 3 bis: anar a comprar-ne i seguidament fer el que diu la línia 4.
Segons sembla, hem fet un algorisme que ens permet de fer un ou ferrat fins i tot si no hi hagués oli des de bon començament.
Fins ara, els nostres algorismes només tenen tres tipus de línies:

– unes que fan OPERACIONS, com, «trencar la closca de l'ou» o «girar a l'esquerra»,
– unes que ens permeten d'OBTENIR l'objecte sobre el qual ha d'actuar l'algorisme, com «Agafar l'ou»,
– unes altres que ens DONEN el resultat, com «Posar l'ou ferrat al plat».

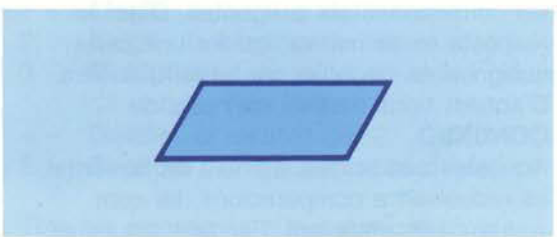
Notem que de qualsevol cosa que pensem, se'n pot fer un algorisme. Hi ha, però, algorismes que donaran programes i d'altres que no, perquè el que plantegen són feines que no pot fer un ordinador. Per exemple, l'algorisme de l'ou ferrat i l'algorisme que guia la nostra amiga per Barcelona no portaran a un programa, perquè, per més bé que li ho expliquem, un ordinador no pot cuinar ni fer turisme. L'algorisme de calcular l'àrea del triangle, en canvi, sí que es pot carregar a un ordinador, és a dir, es pot convertir en un programa.
Aquest darrer algorisme també té tres tipus de línies o accions: les d'**OPERACIO** com «Dividir el resultat per 2», les de **RECOLLIR DADES** com «Llegir la longitud de la base» i les d'**DONAR UNA RESPOSTA** com «Escriure el resultat».
El que diuen les línies d'operació es farà gairebé sempre a la part aritmètica de la UAL, mentre que les línies de recollida de dades i entrega de resultats faran treballar els perifèrics d'entrada i sortida.
Encara hi ha, però, un quart tipus de línies, les corresponents a preguntes. Quan la resposta és afirmativa, cal fer una certa acció i si és negativa cal fer-ne una altra. D'aquest tipus de línia, se'n diu de **CONDICIÓ**.
Aquestes condicions, a l'hora de la veritat, es redueixen a comparacions, tal com veurem més endavant. Per tant, aquestes línies estan lligades a la part lògica de la UAL.

De l'algorisme a l'ordinograma

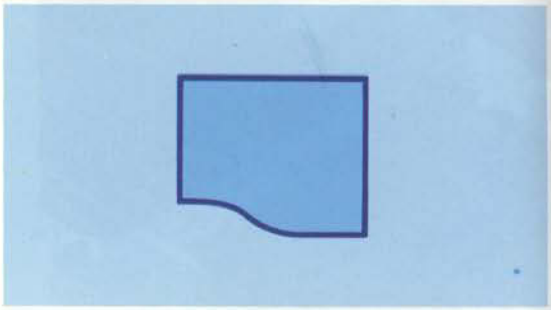
Sembla que això de fer algorismes es va complicant, sobretot des que hem trobat la línia de condició. Ja tenim per una banda quatre tipus de línies o accions, i per l'altra l'obligació de escriure-les de manera ordenada: una cosa darrera l'altra.
Per aclarir aquesta situació usarem els **DIAGRAMES DE FLUX,** anomenats també **ORDINOGRAMES**, que són una «representació gràfica d'un algorisme, mitjançant l'ús de signes convencionals subjectes a certes regles».
Quina tirallonga!
El que volem dir amb aquesta definició és que mitjançant algun tipus de figures o símbols hem d'ésser capaços de distingir els quatre tipus de línies. Amb algun altre dibuix hem de representar el fet de que hi ha un «flux» o «seqüència» de coses a fer.
Aquests dibuixos han d'ésser «convencionals», és a dir, que tothom ha de fer servir sempre els mateixos per tal d'evitar confusions.
Els símbols són els següents:



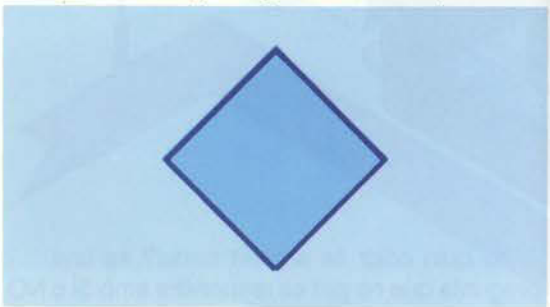
Rectangle: a dins posarem les operacions que fem en una línia d'operació.



Romboide: tindrà el contingut d'una línia d'entrada de dades.

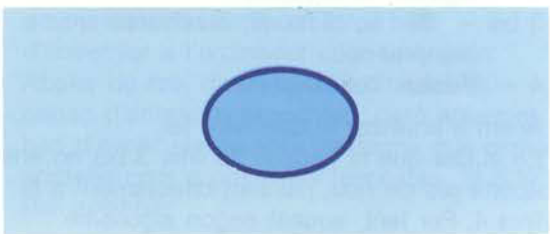


Full: indicarà el contingut d'una línia de sortida d'informació (entrega de resultats).



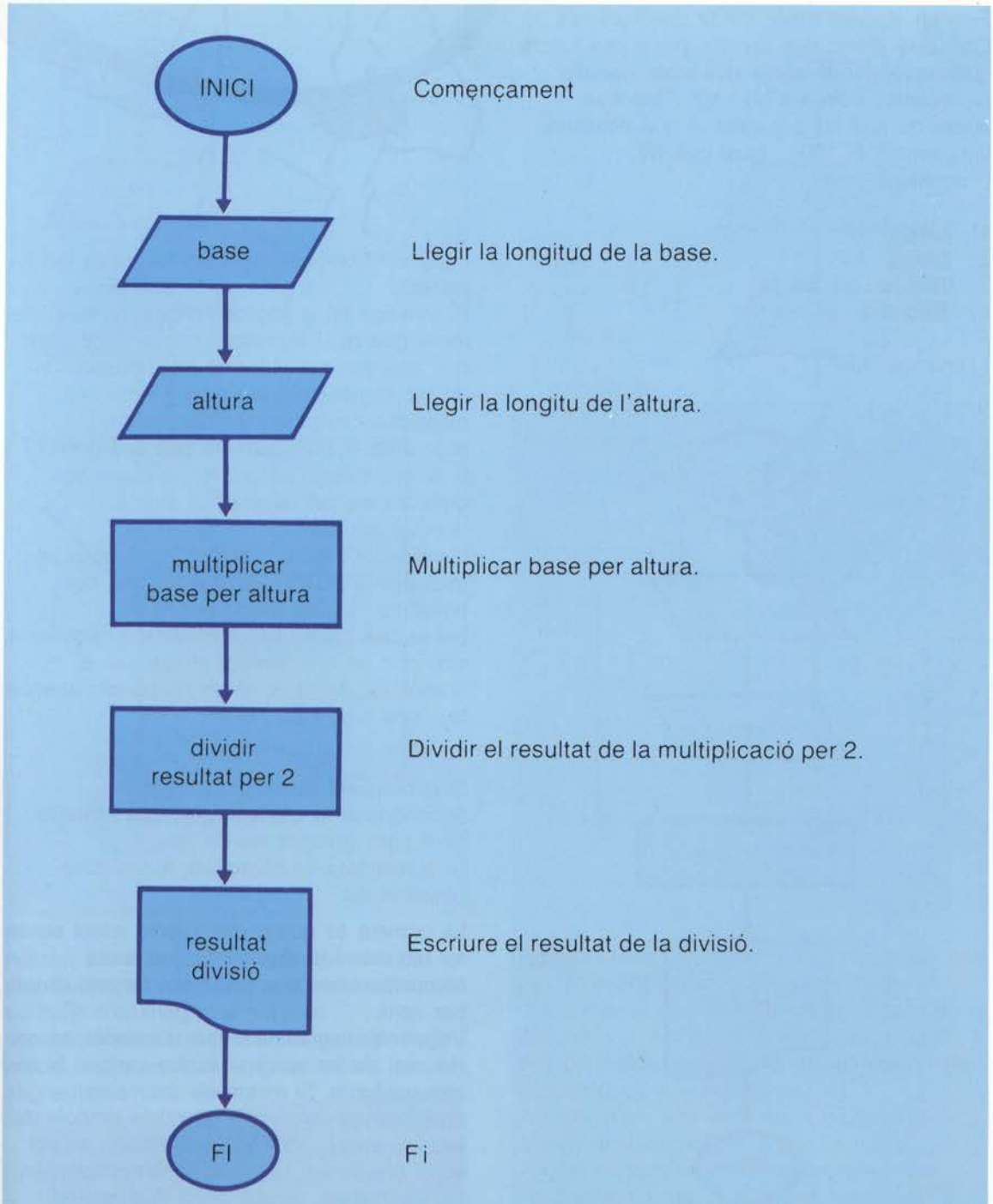
Rombe: per representar una línia de condició, el seu contingut serà la pregunta.

El dibuix que representarà el flux serà, com ja us podeu imaginar, una línia orientada o **fletxa**.
Fixem-nos que les fletxes estableixen molt clarament un «ordre»; d'aquí ve que també fem servir el nom d'ordinogrames per a aquestes representacions gràfiques.
Hi ha un altre símbol, que posarem sempre just on comença el diagrama i on acaba. Es l'el·lipsi.



El·lipsi: la del començament contindrà el mot INICI i la del final el mot FI.

Així, l'algorisme de l'Àrea del triangle quedaria descrit amb el diagrama de flux següent:

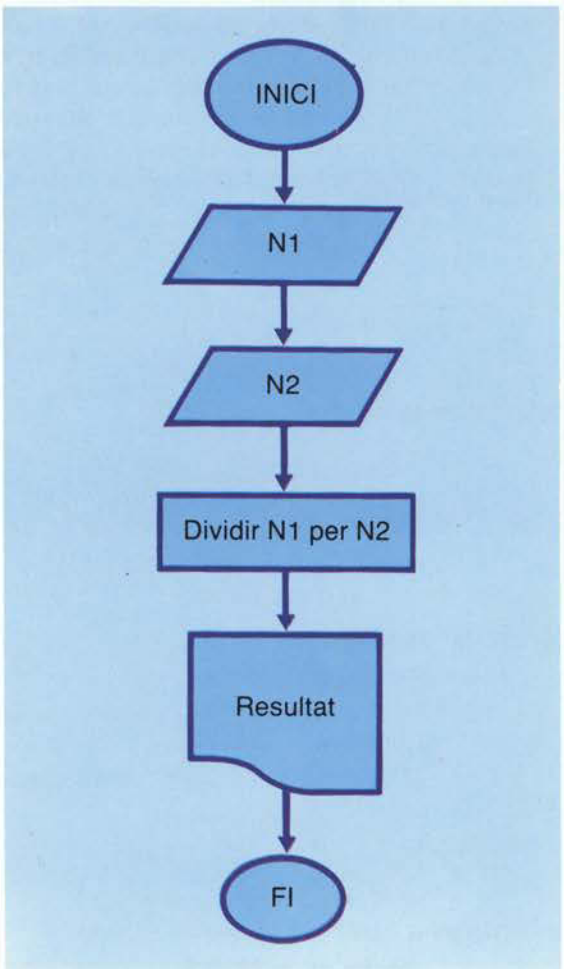


Sembla que aquest primer exemple no ha estat gaire difícil. Vegem-ne un altre:

Fem un algorisme per dividir dos números. Com que volem que serveixi per a dos valors qualssevol, farem servir dos mots que els representin, com ara N1 i N2. Amb això volem dir que N1 pot valer el que nosaltres vulguem: 2, 5, 100..., igual que N2.
L'algorisme serà:

1) **Llegir** N1
2) **Llegir** N2
3) **Dividir** N1 per N2
4) **Escriure** el resultat

I l'ordinograma:

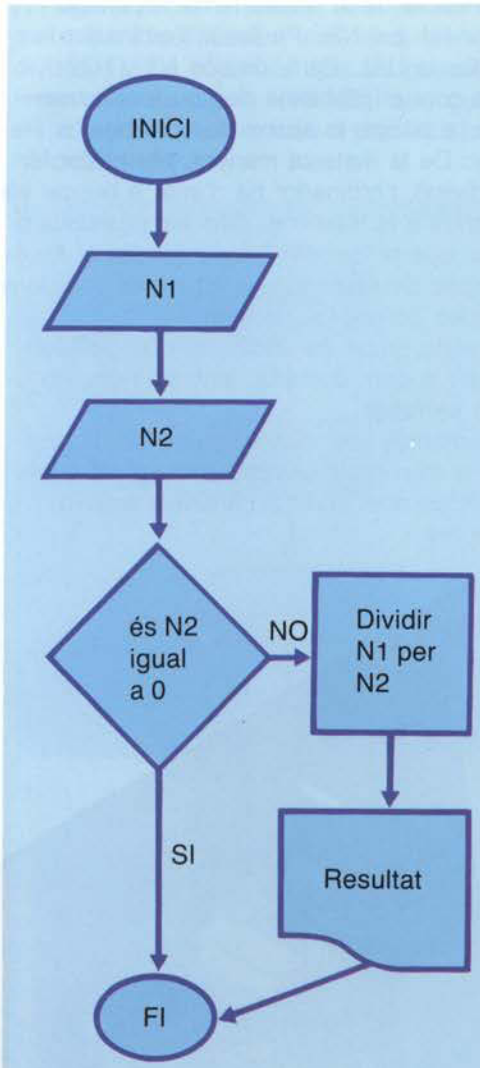


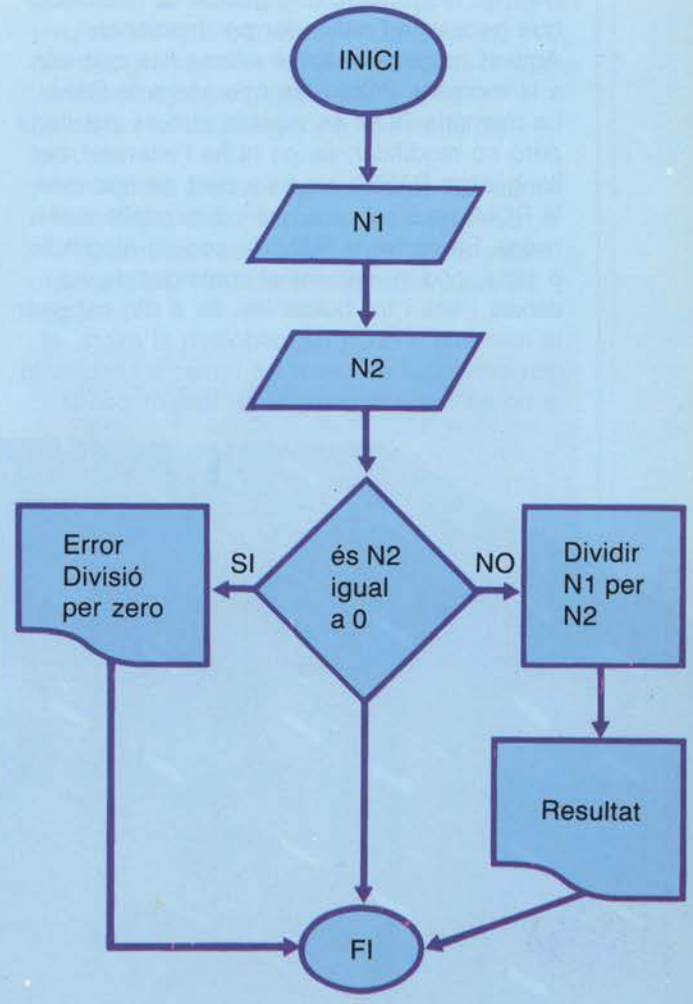
Fàcil, oi? Doncs el que hem fet no és del tot correcte.
Posem-nos en el lloc de l'ordinador. Hem de fer el que diu l'algorisme: primer esperarem que ens donin el valor de N1, i després el de N2. Suposem que siguin 8 i 2, respectivament. Seguint l'algorisme, escriurem 4. Suposem ara que ens donen 7 i 0. Direm que no es pot fer, ja que a les divisions no pot haver-hi un zero al denominador.
L'ordinador faria el mateix. Ens donaria un missatge d'ERROR, perquè la UAL, com nosaltres, no sap dividir per zero.
No és que hàgim escrit malament l'algorisme, sinó que no hem previst el cas que el divisor, N2, fos zero. Això, ho podem arreglar fent una pregunta:

És N2 = 0?

Si la resposta és negativa, el nou ordinograma funcionarà igual que l'anterior. No li hem introduït res de nou.
Si la resposta és afirmativa, tenim dues possibilitats:

La primera és que l'ordinograma acabi sense fer res més i la segona és que acabi comunicant-nos que pretenem fer una divisió per zero.
Vegem el diagrama de flux d'ambdós casos. Haurem de fer servir el rombe, perquè hi ha una pregunta. Si mirem els exemples anteriors, comprovarem que dels símbols de lectura, escriptura i operació només en pot sortir una fletxa; però del de condició n'han de sortir dues, perquè hi ha dos camins diferents a seguir.





Aquest exemple, encara que sigui senzill, ens diu força coses noves, sobretot d'aquella definició d'ordinograma: «signes convencionals subjectes a certes regles». En efecte, entre aquestes regles trobem les següents:

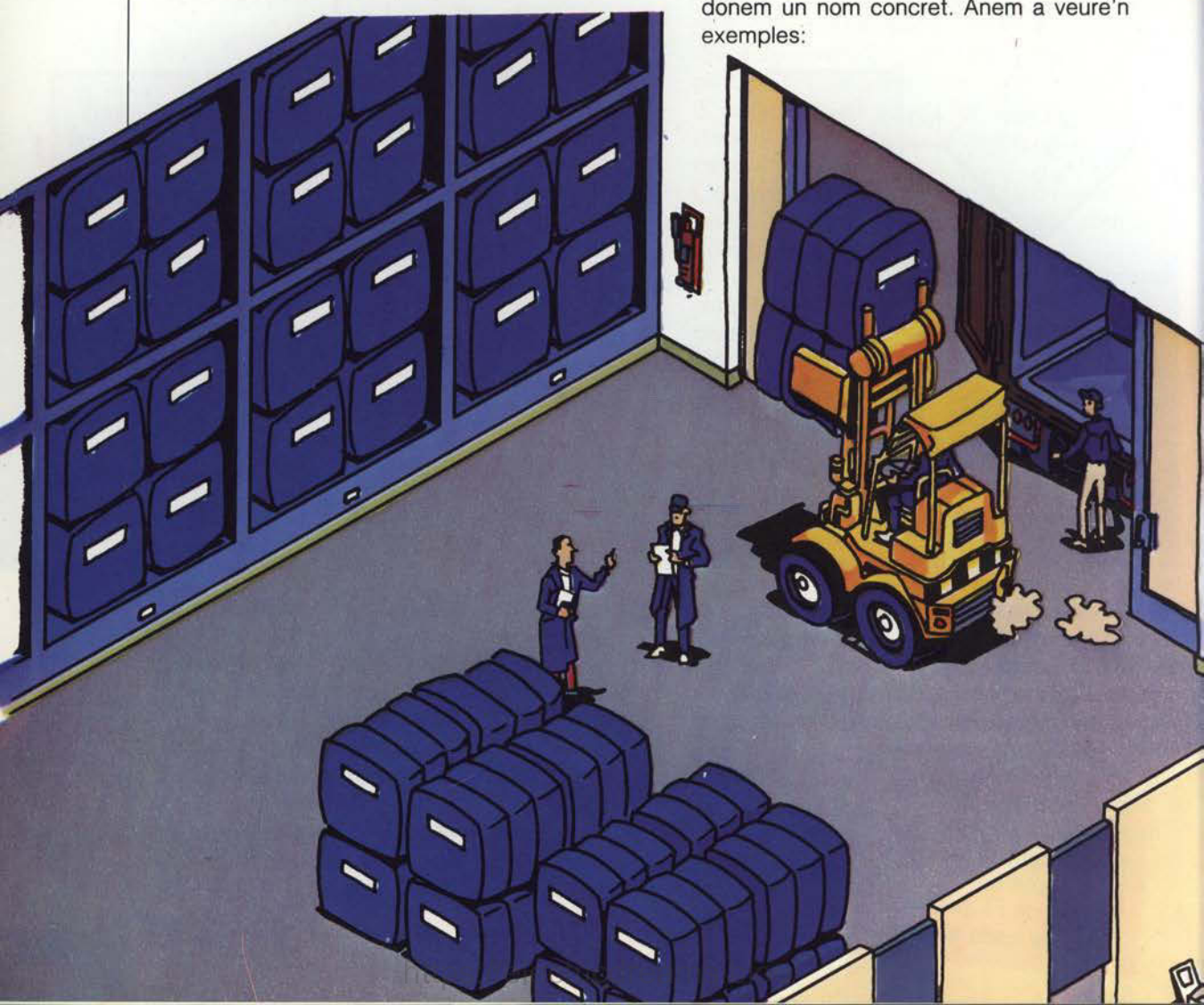
1) a qualsevol símbol (excepte l'el·lipsi d'INICI), podem fer-hi arribar tantes fletxes com sigui necesari.
2) excepte en el cas del rombe i de l'el·lipsi de FI, només pot sortir una fletxa de cada símbol.

Aquests exemples també ens fan pensar que cal fer les coses amb seriositat; hem d'analitzar el problema, escriure l'algorisme i després l'ordinograma. L'últim pas, el que ens permetrà de trobar la solució del nostre problema, serà el programa.
A partir d'ara ens dedicarem a conèixer quines accions pot fer el nostre micro. També explicarem com cal escriure-les a l'ordinograma per tal que ens sigui fàcil després convertir-lo en programa.

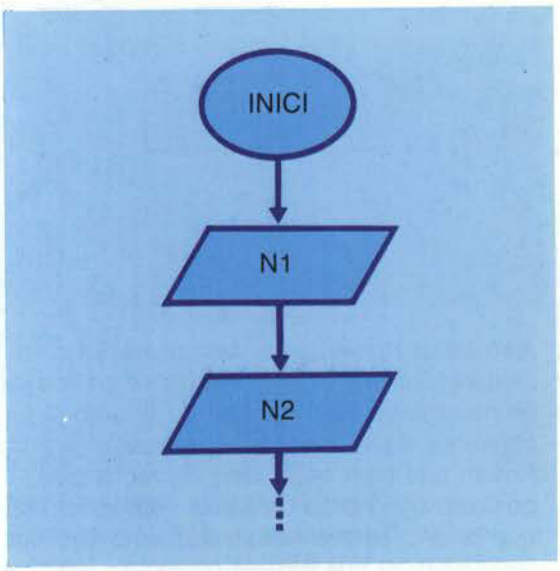
Memòria i variables: un magatzem de caixes amb nom

Fa una estona hem representat la memòria com un magatzem, que guarda la informació que necessita l'ordinador per funcionar. Aquest magatzem conté caixes, les que són a la memòria ROM i les que són a la RAM. La memòria ROM és aquella que es pot llegir però no modificar; és on hi ha l'intèrpret del llenguatge BASIC; aquesta part de memòria, la ROM, no s'esborra quan desendollem el micro. En canvi, la RAM, la podem modificar, o sigui, podem canviar el contingut de les caixes i fins i tot buidar-les, és a dir, esborrar la memòria; i quan desendollem el micro, el seu contingut es perd; en tornar a endollar-lo ja no es recorda del que hi havíem deixat.

A l'exemple de la divisió hi ha l'operació «dividir N1 per N2». Pe fer-la, l'ordinador fa treballar la UAL. Però, on són N1 y N2? Sí, és com el problema de l'ou ferrat: hem d'anar a buscar-lo abans de començar a cuinar. De la mateixa manera, per poder fer una divisió, l'ordinador ha d'anar a buscar els operands a la memòria, dins les posicions o caixes que la formen. A més a més, el micro és capaç de reservar unes quantes posicions seguides donant-los un nom.
D'aquests grups de posicions que guarden dades i podem demanar amb un nom, en direm **variable**.
Les variables són, doncs, grups de «bytes» seguits (numerats correlativament) als quals donem un nom concret. Anem a veure'n exemples:

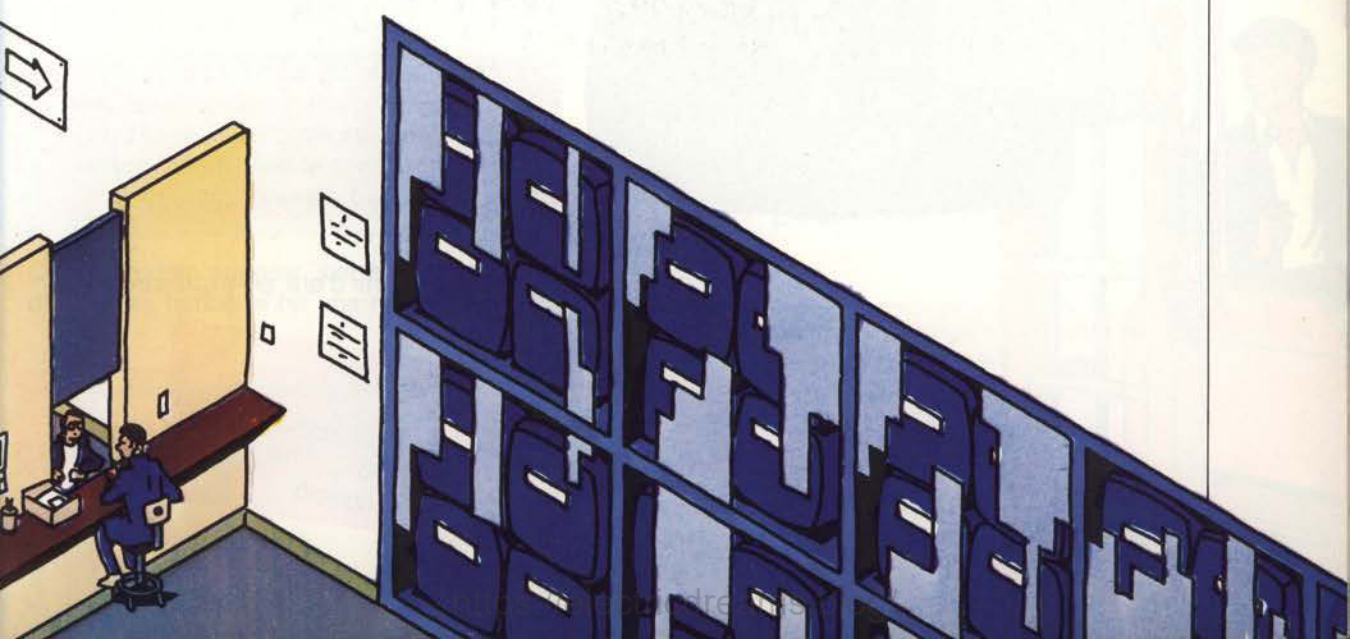


A l'algorisme de la divisió, hem fet servir dos mots que identificàvem amb el dividend i el divisor. El que volem és que les posicions reservades es diguin N1 i N2, i que guardin els números que li entrarem per teclat. N'hi ha prou de fer:

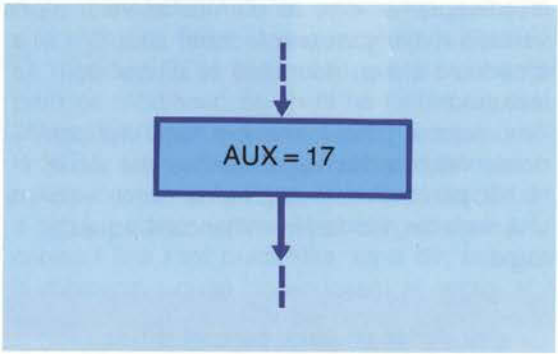


A l'ordinograma de la divisió ja ho havíem escrit així; sense saber-ho, havíem reservat posicions de memòria. Tenim una altra de les «regles» de què parlàvem en la definició d'ordinograma: «Per tal d'entrar el valor d'una variable mitjançant una lectura, s'ha d'escriure el seu nom dins el símbol de lectura».
Amb aquest primer exemple hem vist com donar valors a les variables des del teclat. Hi ha, però, altres maneres de donar valors a una variable. Ho farem mitjancant aquesta regla:

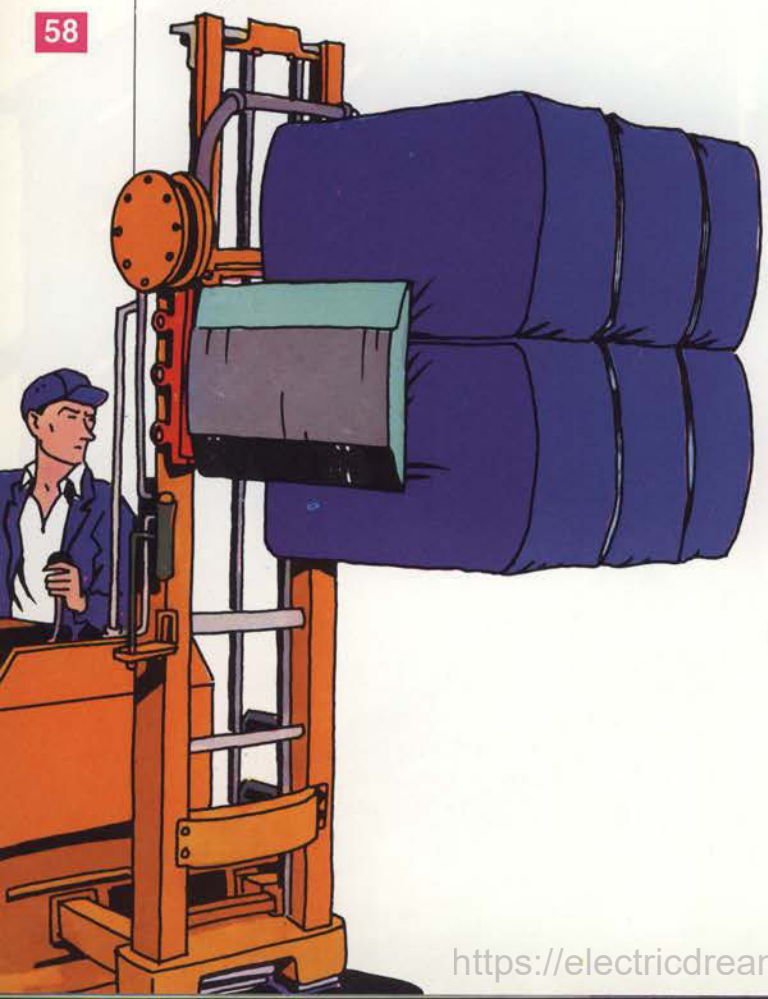
«Per donar un valor concret a una variable en un punt determinat d'un ordinograma, s'ha d'escriure dins un símbol d'operació el nom de la variable seguit del signe = (igual) i a la dreta d'aquest s'ha d'escriure el valor concret.»



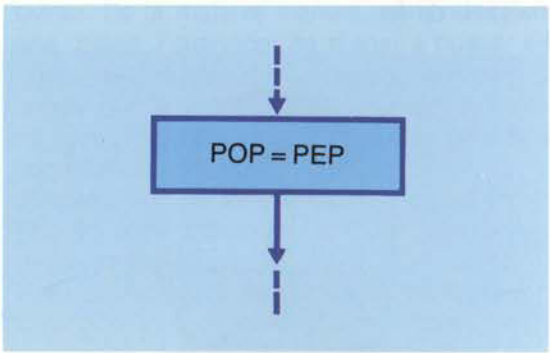
Suposem, per exemple, que, en un moment donat, ens interessa que una variable anomenada AUX prengui el valor 17. Dins l'ordinograma, ho representarem així:



Per tant, el que escriguem a la dreta del **signe igual** es carregarà dins les posicions que tenen per nom el que hem escrit a l'esquerra.



Si el que volem fer no és donar un valor concret a una variable, sinó un valor que está dins un altra variable, per exemple el contingut de la variable PEP a les posicions anomenades POP, escriurem a l'ordinagrama:



Això és una nova regla, molt semblant a l'anterior, ja que fa el mateix, però en lloc de fer-ho amb un valor concret ho fa amb el contingut d'una variable específica.
Notem que hem escrit dins els rectangles quelcom que no és un càlcul i us havíem dit que la UAL estava relacionada amb aquests. En realitat, el que havíem dit és una veritat a mitges, perquè els rectangles, a més d'estar relacionats amb la part de càlcul de la UAL, també els fem servir per indicar operacions de moviments dins la memòria, sense fer càlculs, com en els dos darrers exemples.
Si hi ha càlculs dins un rectangle hem d'obeir una nova regla:

> «El resultat de tot càlcul s'ha de guardar dins d'una variable. Quan dins un rectangle hàgim d'escriure càlculs, ho farem posant el nom de la variable seguit d'un = i darrera l'expressió».

Els càlculs, a partir d'ara, els expresarem per:

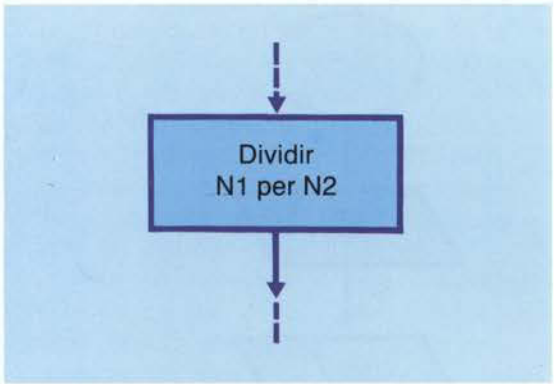
Suma	: +
Resta	: −
Multiplicació	: *
Divisió	: /
Potenciació	:^

Exemple: $(A + B)^2$ s'escriu (A + B)^2
Cal posar-hi els parèntesis, perquè, si no, només elevaríem al quadrat el contingut de la variable B.

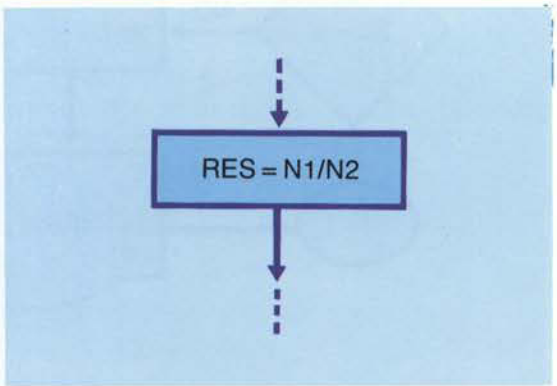
A l'algorisme de la divisió teníem dues variables, N1 i N2. La línia 3 deia:

3: **Dividir** N1 per N2

I per posar-ho a l'ordinograma havíem escrit



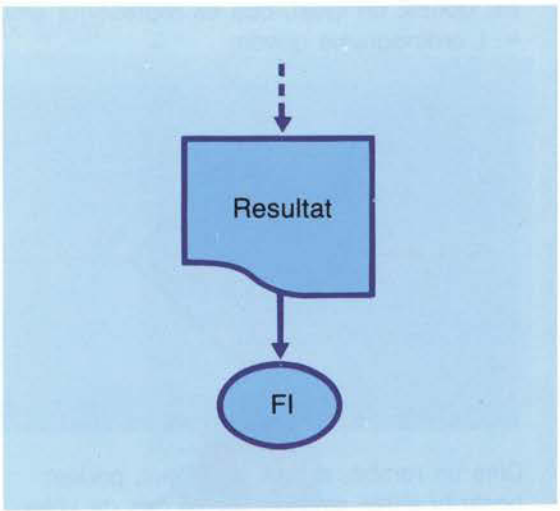
Ara, l'expressió «dividir N1 per N2», l'escriurem N1/N2 i guardarem el resultat, per exemple, a la variable RES. A l'ordinograma escriurem:



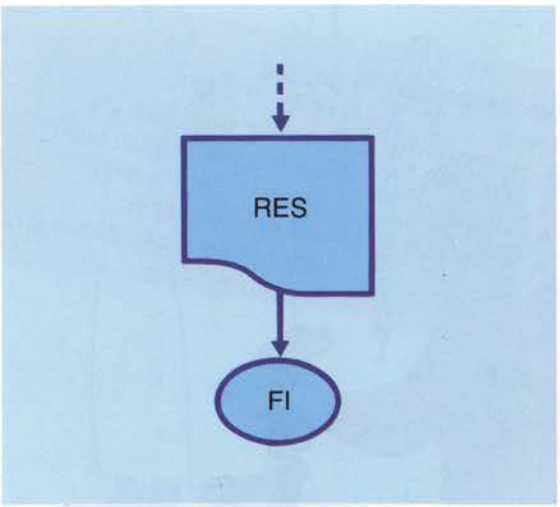
Com ja podem suposar, per al símbol d'esciptura també hi ha una nova regla:

«Per escriure el contingut d'una variable hem de posar el nom d'aquesta variable dins del símbol d'escriptura».

En el diagrama de la divisió, havíem posat:



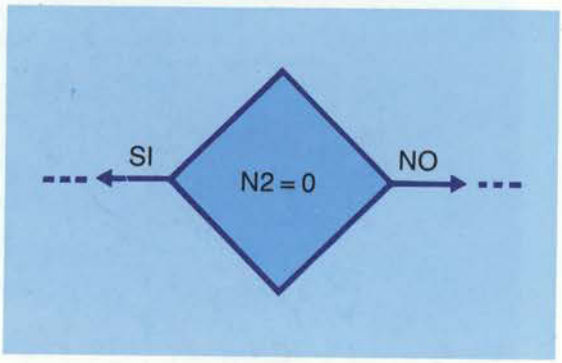
Ara ja sabem fer-ho més correctament. Escriurem:



I, finalment, per al símbol de condició la regla és:

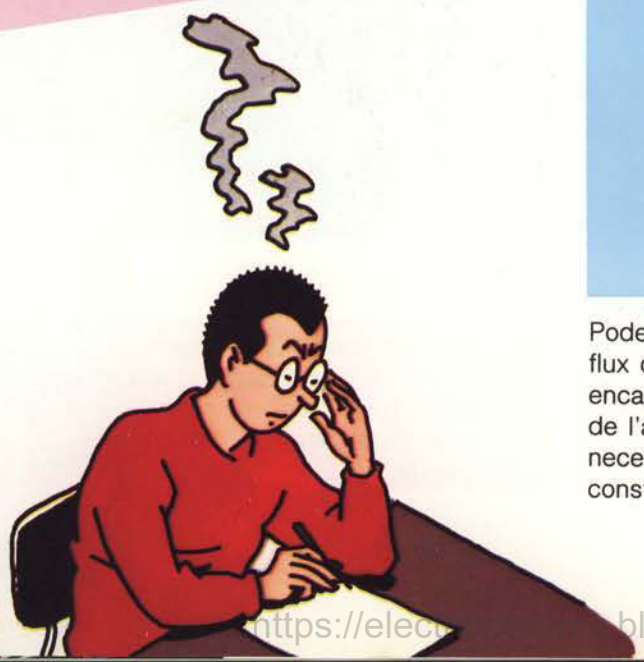
«Per comparar els contiguts de dues variables, o el contigut d'una variable amb un valor concret, s'han d'escriure ambdós separats pel símbol que calgui».

En el cas de la divisió el que volem saber és «si N2 es **igual** a 0»; en aquest cas el símbol és, doncs, un igual, que es representa amb =. L'ordinograma queda:

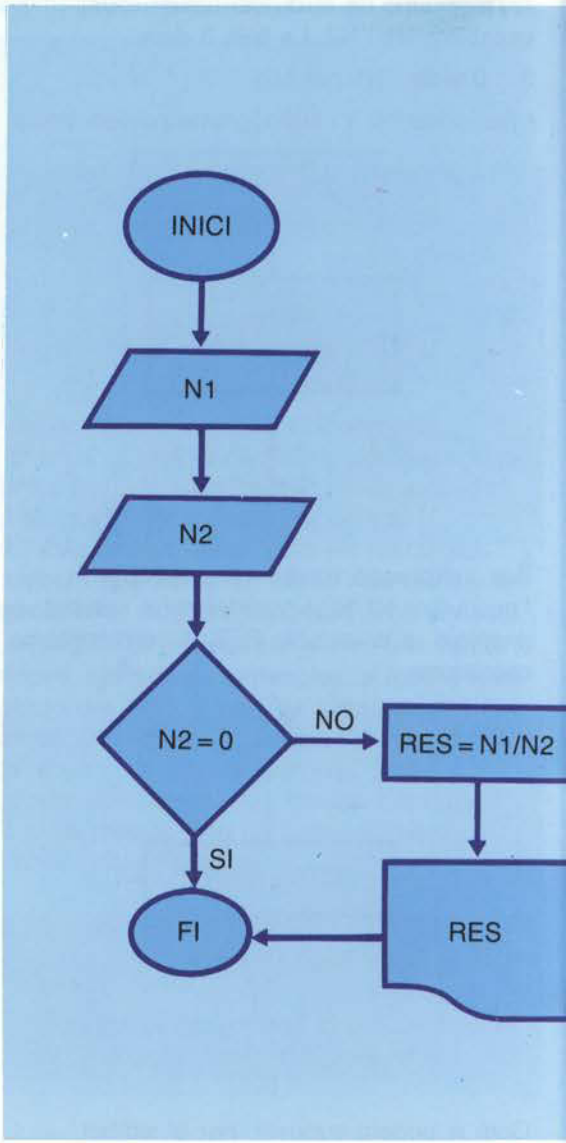


Dins un rombe, a part de l'igual, podem posar-hi altres símbols. En el cas de voler comparar la variable A amb la variable B tenim totes les següents possibilitats:

Representació	Significat
A = B	A és igual a B?
A > B	A és més gran que B?
A >= B	A és més gran o igual que B?
A < B	A és més petit que B?
A <> B	A és diferent de B?
A <= B	A és més petit o igual que B?



Ara ja som capaços d'escriure d'una forma totalment correcta l'ordinograma de la divisió. Per prevenir el cas que s'intenti fer una divisió per zero (primer diagrama), posaríem:



Podem provar també de fer el diagrama de flux de l'algorisme de l'àrea del triangle, però encara no podem fer el segon ordinograma de l'algorisme de la divisió, ja que necessitem saber abans què son les constants.

CONSTANTS: valors concrets

Hem vist que en un diagrama sortien, a part dels símbols i les fletxes:

- les paraules fixes: INICI i FI
- variables (per exemple N1)
- símbols de càlcul (/, +, ^, ...)
- signes de comparació (=, < >, <=, ...)

Notem que, a més, hi ha el que hem anomenat valors concrets. Per exemple el zero (0) del rombe en l'ordinograma de la divisió, o el 17 quan fèiem allò de AUX = 17. Aquests valors concrets s'anomenen **CONSTANTS**. En els nostres exemples són números. Però també hi ha **CONSTANTS** que són lletres, paraules, o una barreja de lletres, números i caràcters especials (com el &, o el %), anomenades **CADENES**.

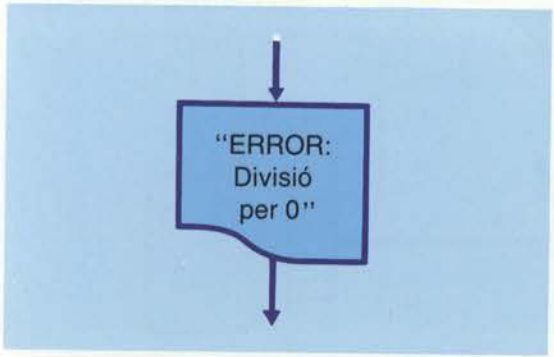
Les que són números, ja sabem la utilitat que tenen, perquè les hem fetes servir i les sabem escriure correctament. N'hi ha prou de posar el número.

Les CADENES s'han de posar entre cometes (").

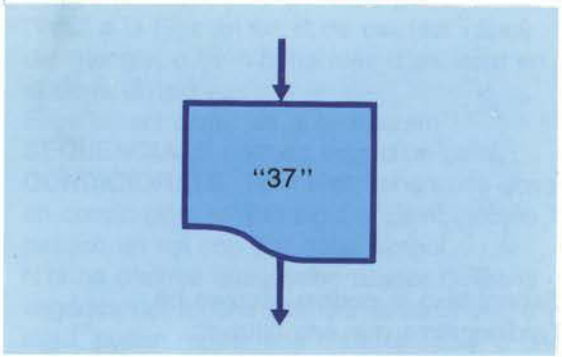
Si el que volem és escriure un missatge concret que no és dins de cap variable, com, per exemple, en el segon ordinograma de la divisió on hi ha el missatge:

ERROR: divisió per 0

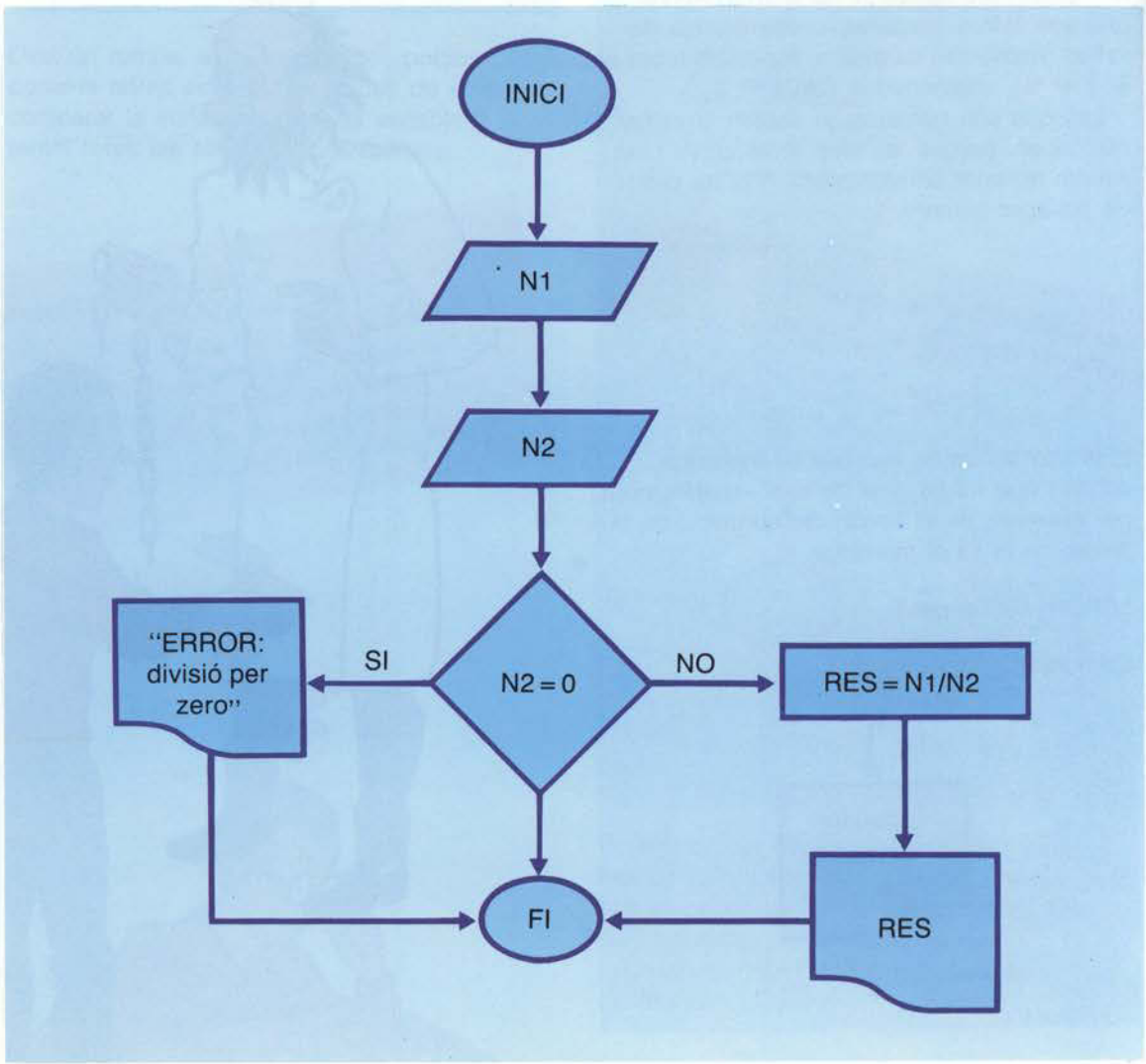
Escriurem:



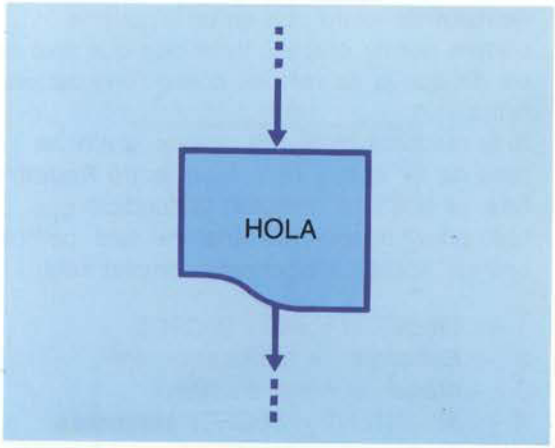
Hem de fer el mateix si el missatge és un número concret. Si volem escriure 37, posarem:



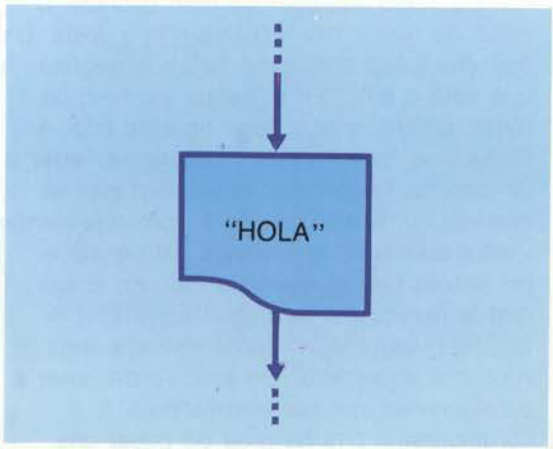
Sabent això ja podem escriure bé l'ordinograma que ens faltava:



Notem que si en un ordinograma posem:



el que volem és escriure el contingut de la variable HOLA, mentre que si posem:



el que volem és escriure la cadena HOLA.

Prosseguir, triar i repetir

En els diagrames que hem fet fins ara, o bé només hi ha un camí possible per anar de l'INICI a la FI, com en el de calcular l'àrea del triangle, o bé n'hi ha més d'un, com en el de la divisió.
Els d'un sol camí, els anomenarem **SEQÜENCIALS** i els de més d'un camí, **CONDICIONALS**. Tot i això, tenen una cosa en comú: sigui el que sigui el camí, només passen un sol cop per cada símbol.
N'hi ha d'altres que poden passar diverses vegades per alguns símbols determinats, o sigui, poden repetir una mateixa acció, o un grup d'accions, tants cops com creguem necessari. Anomenarem **ITERATIUS** els algorismes d'aquesta darrera mena.
Vegem-ne alguns exemples:

El primer és un joc per a dues persones. Consisteix en això: que l'una es pensa un número entre 0 i 9 i el guarda dins una variable, i l'altra ha d'intentar endevinar-lo com més aviat millor. Després és l'altra persona qui ha d'endevinar.
Fem l'algorisme.
El primer que hem de fer és entrar el número secret (que no ens vegin quan l'escrivim!), o sigui:

1 – **Llegir** el número SECRET

Després hem d'esborrar tota la pantalla perquè, si no, tots veurien el número que hem escrit pel teclat.

2 – **Esborrar** la pantalla.

Ara es comença a jugar. Es tracta d'escriure un número mitjançant el teclat i comparar-lo amb el que hi ha a SECRET per veure si ens l'han encertat. Per tant el que hem de fer serà una lectura guardant el que s'hagi llegit en una variable que en direm INTENT.

3 – **Llegir** el número INTENT.

I ja podem fer la pregunta «SECRET es igual a INTENT». Si la resposta és SI, vol dir que s'ha acabat el joc; si la resposta és NO, hem

de tornar a llegir un altre número, tornar a comparar i així anar fent, tants cops com calgui. L'algorisme, de moment, serà:

1 – **Llegir** el número SECRET
2 – **Esborrar** la pantalla
3 – **Llegir** el número INTENT
4 – **Si** INTENT = SECRET **aleshores** acabar
5 – **Llegir** el número INTENT
6 – **Si** INTENT = SECRET **aleshores** acabar
7 – **Llegir** el número INTENT
8 – **Si** INTENT = SECRET **aleshores** acabar

I un altre cop, si la resposta és NO tornaríem a escriure 2 línies més a l'algorisme idèntiques a les 3, 4 ó a les 5, 6... i així tantes vegades com s'equivoqui el qui juga. Nosaltres, però, no coneixem a priori aquest número; per tant no podem continuar per aquesta via. El problema és representar en l'algorisme una ordre complexa que de paraula s'escriuria així:

«Llegir INTENT. Si és diferent de SECRET, tornar a llegir INTENT».

Això és una acció que anirem repetint fins que es compleixi certa condició. Podem escriure-ho com:

Repetir
Llegir INTENT
preguntar si coincideix o no amb SECRET
fins que coincideixin

El que hi ha entre **Repetir** i **fins** és el que farem sempre que INTENT sigui diferent de SECRET. L'algorisme d'aquesta part és:

3 – **Llegir** el número INTENT
4 – **Si** INTENT = SECRET **aleshores** acabar

Per tant només cal fer:

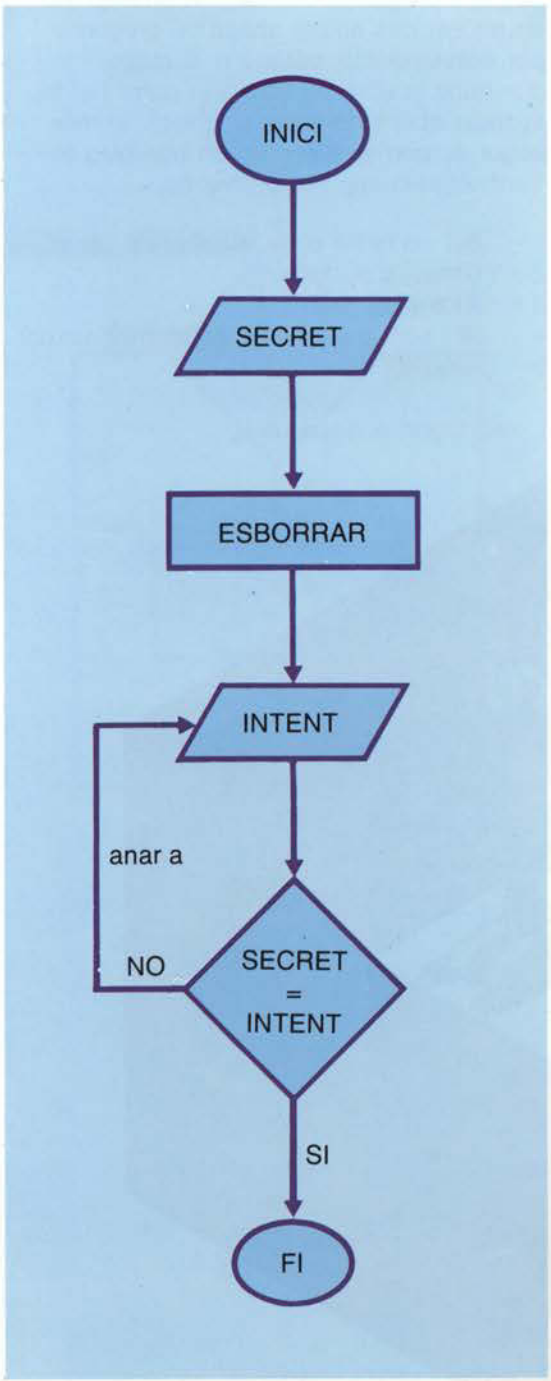
3 – **Llegir** el número INTENT
4 – **Si** INTENT = SECRET **aleshores** acabar
5 – **Anar a** 3

Acabem de veure que en un algorisme podem «tornar enrera», fixem-nos que això no vol dir que ja no fem les coses l'una darrera l'altra.
Si la resposta es SÍ, vol dir que ja n'hi ha prou de fer el que hem escrit entre **Repetir** i **fins**, ja que s'ha complert la condició que hem posat després de **fins**. Per tant, només ens cal acabar. L'algorisme complet serà:

1 – **Llegir** el número SECRET
2 – **Esborrar** la pantalla
3 – **Llegir** el número INTENT
4 – **Si** INTENT = SECRET **aleshores** acabar
5 – **Anar a** 3

Dins d'un algorisme, les coses es fan l'una darrera l'altra, tal com les hem escrites, o sigui, de dalt a baix i d'esquerra a dreta. Un cop efectuada una acció farem la següent, la que està a sota. En el nostre joc hem de tornar enrera, anar a llegir un altre cop. A l'algorisme hem fet servir l'expressió: **anar a**. Un cop fet l'algorisme, el següent pas és escriure l'ordinograma. Però, com representar l'ordre de **«anar a»**? Només cal pensar en les fletxes i en el que volen dir. En el cas que la resposta a la pregunta «INTENT = SECRET» sigui NO hem de tornar a llegir. Això que representàvem amb l'ordre **anar a** a l'algorisme, per representar-ho a l'ordinograma n'hi ha prou de posar una

fletxa que vagi des del rombe, per la sortida NO, fins al símbol de lectura d'INTENT. Així, l'ordinograma és:



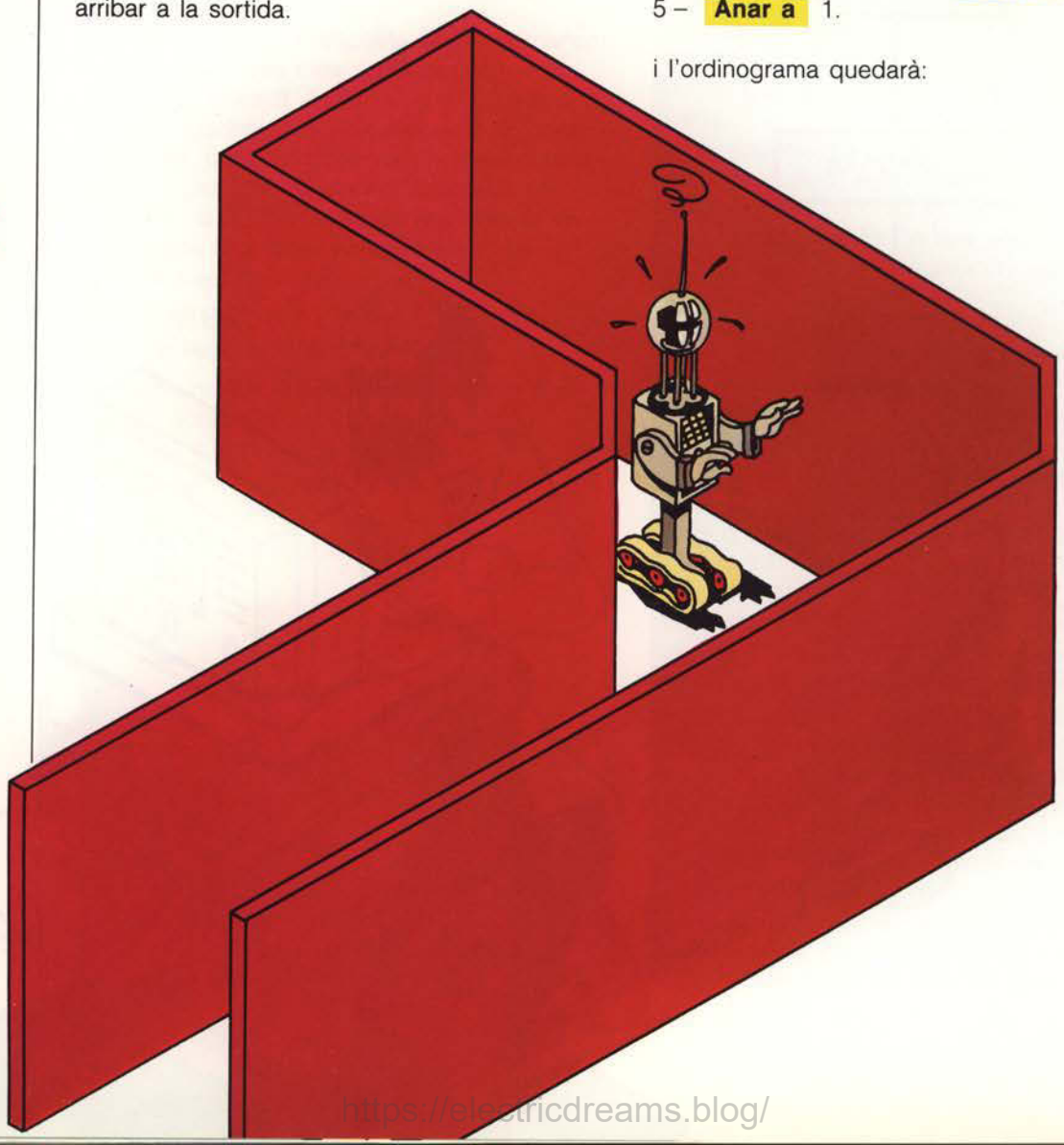
Hi ha molts problemes que es poden resoldre amb un ordinograma d'aquest estil. Per exemple, quan usem un caixer automàtic hi escrivim el codi personal, el que fa la màquina és comparar-lo amb el que llegeix de la banda magnètica de la targeta. Si coincideixen (SECRET=INTENT) ens deixa continuar; si no coincideixen, acaba. (Només tenim 3 oportunitats!!).

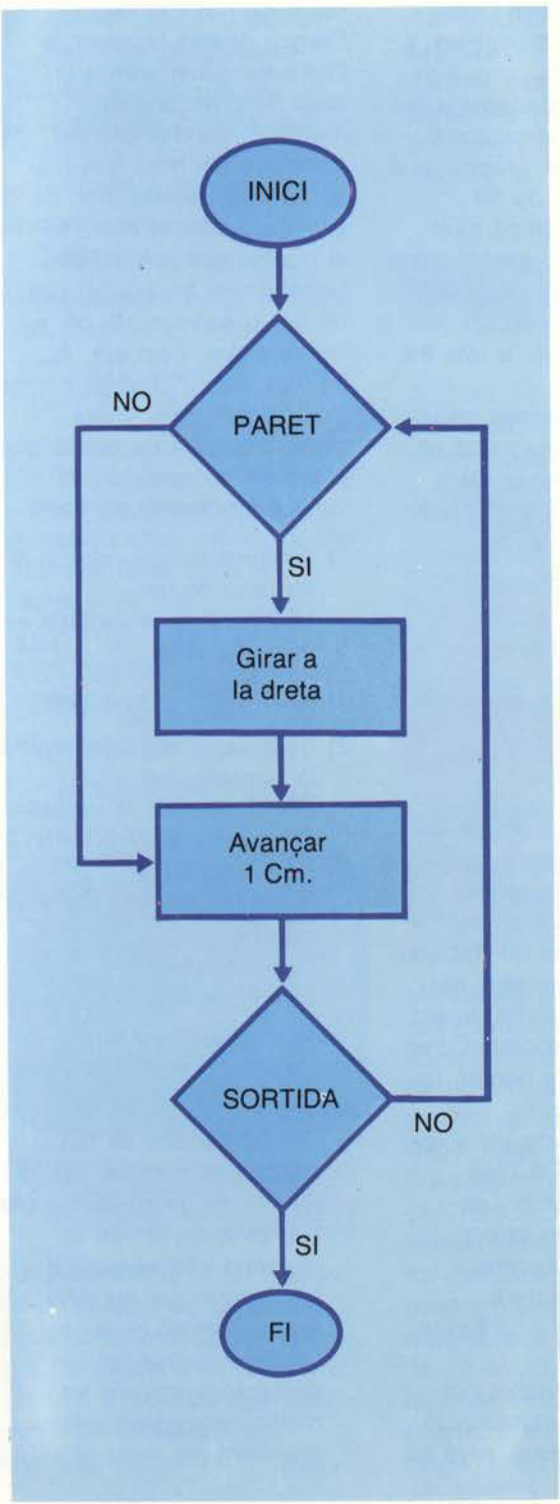
Anem a fer un altre exemple: tenim un robot que és cec; només sap fer 3 accions: avançar endavant 1 cm., girar un quart de volta a la dreta i, mitjançant uns sensors (una mena d'antenes), sap si toca o no una paret. Tenim el robot en un passadís en forma de L tancat per un costat, i ha d'anar a la sortida que hi ha a l'altra banda de la L. Com que el robot és cec, no sap on es troba; per tant, no sap en quina direcció ha d'anar. Haurà de fer-ho temptejant, és a dir, es posarà a caminar i quan es trobi amb una paret girarà a la dreta. Es fàcil comprovar que fent-ho d'aquesta manera el robot pot arribar a la sortida.

Hem de tenir en compte que si a l'exemple anterior primer actuàvem («llegir INTENT») i després preguntàvem («Si INTENT=SECRET»), en aquest cas actuar abans de preguntar pot ésser perillós, perquè si el robot d'entrada ja està de cara a la paret i el feu avançar abans de mirar si la toca, el més segur és que se n'endurà un bon blau al front. Aquest cop l'algorisme és:

1 – **Si** no hi ha paret **aleshores** **anar a** **3.**
2 – **Girar** a la dreta.
3 – **Avançar** 1 cm.
4 – **Si** som a la sortida **aleshores** acabar
5 – **Anar a** 1.

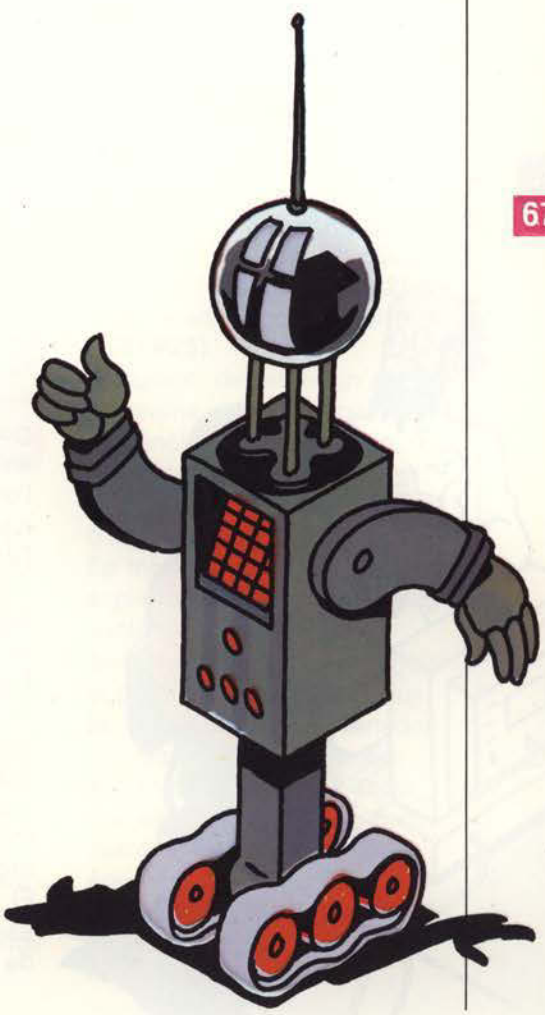
i l'ordinograma quedarà:





Si, aquest ordinograma, el volguéssim expressar en llenguatge col·loquial diríem:

Repetir
Mentre no hi hagi paret **fer** avançar de centímetre en centímetre
Girar a la dreta
Fins a trobar la sortida.



Programem en BASIC

L'últim pas a fer per tal que l'ordinador resolgui realment el nostre problema és confeccionar el programa. Tot el que hem escrit referent a algorismes i ordinogrames es vàlid sigui el que sigui el llenguatge de programació. No és així en el cas d'escriure els programes; aquests tenen escriptures diferents per a cada llenguatge de programació. Nosaltres hem triat el BASIC (Beginer's All-purpose Symbolic Instruction Code) perquè és el més usat en ordinadors petits, perquè és el que més bé s'adapta a les necessitats d'una persona que s'inicia en la programació i perquè permet de fer programes en camps molt diferents, des de gestió fins a jocs i dibuixos en color. A més a més no hi ha un BASIC standar per a tots els ordinadors.

Abans d'endinsar-nos en la tasca de la programació en BASIC, volem deixar clars dos conceptes fonamentals en programació: el de VARIABLE i el de CONSTANT.

Variables

Per variable entenem un cert conjunt de posicions de memòria que serveix per guardar dades i resultats amb els quals ha de traballar l'ordinador. La variable pot anar canviat de valor durant l'execució del programa. Per identificar-les, els donem un nom.

Hi ha dos tipus de variables: Les NUMÈRIQUES i les ALFANUMÈRIQUES. Les primeres guarden valors numèrics i les segones cadenes de caràcters (números, lletres i caràcters especials).

El nom d'una variable pot ésser qualsevol combinació de lletres i números, però ha de començar sempre per una lletra. Compte! el micro Dragon només fa servir, a l'hora de donar nom a un grup de posicions de memòria, els dos primers caràcters del nom que heu donat a la variable han de ser majúscules. Posar PEP és el mateix que posar PEIX. Recomanem en aquest cas fer servir noms curts per a les variables, com ara, AL, P1, KK, S3, ... D'altres micros distingeixen més lletres.

Hi ha una colla de coses que ja sabem fer amb les variables numèriques:

1) Assignar-los un valor. N'hi ha prou de fer:
   LET nom de la variable = valor numèric
   Per exemple:
   LET ANYS = 12
2) Escriure el seu contingut a la pantalla fent
   PRINT nom de la variable.
   Per exemple: PRINT ANYS
3) Fer operacions aritmètiques amb els seus continguts.
   Per exemple:
   LET RES = N1/N2

A l'hora d'escriure una expressió aritmètica, cal tenir present que:

- Les operacions és fan d'esquerra a dreta. Així N1/N2 vol dir N1 dividit per N2, és a dir, N1 és el dividend i N2 el divisor.
- La prioritat per resoldre una expressió és la següent: primer les potències, després les multiplicacions i divisions i finalment les sumes i

restes; per modificar aquesta prioritat es fan servir els parèntesis.

Sabem també que per assignar una cadena de caràcters a una variable alfanumèrica hem de posar:

Nom de la variable$ = "Cadena"

Per assignar BARCELONA a la variable CIUTAT fareu:

LET CIUTAT$ = "BARCELONA"

Recordem també que es pot fer una operació, **l'encadenament**, amb variables alfanumèriques. Per representar-la s'usa el signe +. Si fem:

LET P1$ = "HOLA!"
LET P2$ = " COM ESTEU?"
PRINT P1$ + P2$

ens sortirà per pantalla:

> HOLA! COM ESTEU?

```
LET P1$ ="HOLA!"
LET P2$ ="COM ESTEU?"
PRINT P1$+P2$
HOLA! COM ESTEU?
OK
```

Constants

Una **constant** és un número o un conjunt de caràcters que no es modifiquen al llarg de l'execució d'un programa. També n'hi ha de dos tipus: les **NUMÈRIQUES**, que representen valors numèrics, com 12, 1000 o 14.288; i les **ALFANUMÈRIQUES o LITERALS**, que són un conjunt qualsevol de caràcters, posat entre cometes (" "), com "HOLA", "12.3", "H1?/", ...

Programes

A l'hora de resoldre un problema, sabem que els primers passos a fer són:

1 – Analitzar-lo molt profundament.
2 – Tenir clar quines dades necessitem.
3 – Conèixer els resultats que volem obtenir.
4 – Plantejar tots els casos conflictius i trobar-ne la solució.
5 – Escriure un algorisme molt detallat de les accions necessàries.
6 – Construir l'ordinograma.

Manca només escriure el «programa». Un programa és un conjunt d'accions que pot fer un ordinador. Aquestes accions es distribueixen en línies (línies de programa). D'aquestes accions, en direm SENTÈNCIES. Per tal que aquest sàpiga l'ordre en què ha d'executar les línies, els posarem un número davant. Normalment les numerarem de 10 en 10, perquè, encara que hàgim estat molt previsors a l'hora de construir l'algorisme i l'ordinograma, ens podem haver deixat algunes accions sense advertir-ho. Si el que hem oblidat s'ha de fer després de la línia 30 i abans de la 40, n'hi ha prou de donar-li, per exemple, el número 35. Les sentències són molt semblants al que coneixem com a línies d'algorisme; per tant, tindran la mateixa estructura: sempre hi ha un verb, i darrera, depenent del verb, les dades sobre les quals aquest actua.

VERB	DADA
Obrir	La nevera
Llegir	Base

Aquests verbs, quan els escriguem en BASIC, els anomenarem INSTRUCIONS, i del conjunt de dades en direm EXPRESSIÓ. Aquesta pot ésser fins i tot un càlcul aritmètic (escriure **(4+3)/2)** o una encadenació (escriure **P1$+P2$)**.
Resumint, definirem un programa com un conjunt de sentències, agrupades en línies, on cada sentència té una instrucció seguida, o no, d'una expressió.
Es molt normal posar una explicació darrera de moltes sentències per tal que qualsevol persona que llegeixi el programa entengui el que fa. Farem saber a l'ordinador que aquell tros és perquè ens hi entenguem nosaltres, i que ell no cal que s'ho miri, usant la instrucció **REM.** Aquestes explicacions s'anomenen COMENTARIS i no afecten en absolut l'execució del programa.

Una línia de programa sol tenir la següent estructura:

número de línia instrucció expressió: ..... :instrucció expressió: REM comentari

sentència sentència

per exemple:

20 CLS: PRINT (4*3)/2:
REM àrea del triangle

fixem-nos que no hi ha expressió a la primera sentència.
Quan introduïm el programa a l'ordinador, usant el teclat, direm que CARREGUEM el programa, ja que el que fa l'ordinador és col·locar-lo a la memòria central. Un cop carregat, cal donar l'ordre per què es comenci a fer. Això s'aconsegueix amb la instrucció **RUN.** Quan l'ordinador està fent tot el que diu el programa direm que l'està EXECUTANT.

Ordres per a una màquina: instruccions, sentències i programes

El llenguatge BASIC té unes 200 instruccions, o millor uns 200 mots reservats per poder prendre'ls com a VERBS d'una sentència.
Aquest mots, per qüestions d'universitalitat, deriven de paraules del vocabulari anglès. No obstant això, hi ha països, com ara el Japó, amb tecnologia prou avançada per construir els seus propis ordinadors, i això els permet de crear llenguatges de programació derivats del seu propi idioma. Però per aprendre a programar no cal saber anglès, de la mateixa manera que a un nord-americà, tot i coneixent l'anglès des de petit, si vol saber programar també li caldrà aprendre'n. I tant difícil és per a ell com per a nosaltres.
Explicarem aquí uns quants d'aquests mots, els més utilitzats, per tal de poder començar ben aviat a fer programes.

Instrucció **NEW**

Esborra tota la memòria RAM, és a dir, posa a zero totes les seves posicions.
La posarem sempre abans de començar a escriure un nou programa. Si a la memòria hi hagués dos programes i poséssim RUN, el micro no sabria quin dels dos hauria d'executar, o el que és pitjor, barrejaria les instruccions de l'un amb les de l'altre.
Suposem que els dos programes tinguessin les línies numerades per
primer programa: 10, 20, 30, 45, 50
segon programa: 5, 15, 20, 25, 35, 40
després de fer RUN, el micro es pensaria que només en té un, numerat de la forma: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 i, per al cas de la línia 20, que es troba en tots dos programes, consideraria la del segon, ja que la del primer desapareix en escriure la del segon.

Instrucció **RUN**

Ordena al micro que comenci l'execució del programa que té carregat a la memòria.

Instrucció **END**

Aquesta instrucció diu al micro que ha d'acabar l'execució.

Instrucció **INPUT**

Diu a l'ordinador que ha d'esperar una entrada de dades pel teclat per guardar-les a la memòria. Després de la instrucció INPUT s'escriu la llista de noms de les variables on vulguem emmagatzemar la informació que donem a través del teclat:

INPUT AL

farà que el micro esperi que li entrem un número i pitgem la tecla ENTER. Tot seguit emmagatzemarà aquest valor a les posicions de memòria anomenades AL, a la variable AL. La instrucció INPUT es correspon, doncs, amb el símbol de lectura d'un ordinograma.
Si el que fem és:

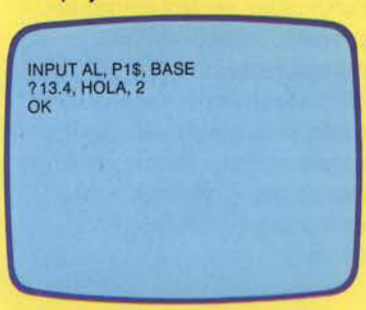
INPUT P1$

l'ordinador esperarà que li entrem una cadena de caràcters i pitgem la tecla ENTER. En aquest cas, la cadena de caràcters no cal que vagi entre cometes.
Si en lloc de posar un sol nom de variable n'hi posem una llista, els hem de separar per comes (,):

INPUT AL, P$, BASE

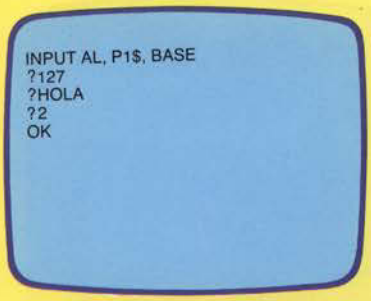
voldrà dir que hem de teclejar, separats per comes, un número, una cadena de caràcters i després un altre número.
Si volem que el contingut de la variable AL sigui 13.4, el de la variable P1$ sigui HOLA i el de la BASE sigui 2, podem fer-ho de les dues maneres que indiquem:

a) 13.4, HOLA, 2
i pitjar ENTER



o bé

b) 13.4 i pitjar ENTER
HOLA i pitjar ENTER
2 i pitjar ENTER



Quan un programa s'està executant i troba una instrucció INPUT no continuarà fins que li entrem quelcom. El més probable és, però, que no recordem l'ordre en què hem escrit les variables i no sapiguem quin valor entrar. Per resoldre aquest problema, la instrucció INPUT ens permet de posar un missatge que sortirà per pantalla. Aquest missatge s'escriu entre cometes i va abans de la llista de variables. Però entre el text del missatge i la llista de variables hi hem d'intercal·lar un punt i coma (;), com ara:

INPUT "INTRODUEIX EL VALOR DE LA BASE"; BASE

o bé

INPUT "INTRODUEIX DOS NUMEROS"; N1, N2

Instrucció **PRINT**

Tal com ja s'ha dit abans, aquesta instrucció serveix per escriure quelcom per pantalla:

– una constant:

PRINT 29.7485

escriurà

> 29.7485

i amb

PRINT "QUE?"

escriurà

> QUE?

– el contingut d'una variable:

PRINT BASE

escriurà

> 2

si el contingut que troba en la posició de memòria assignada a BASE és 2.

```
PRINT BASE
2
OK
```

PRINT P1$

escriurà

> HOLA

si el contingut que troba en la posició de memòria assignada a P1$ és HOLA

```
PRINT P1$
HOLA
OK
```

La instrucció PRINT es correspon amb el símbol d'escriptura dels ordinogrames.
Si volem posar una llista de constants i/o variables, ho podem fer de dues maneres: separant-les per comes (,) o bé per punts i comes (;).
En el primer cas, en què les separem per comes, la impressió es farà posant-ne dues tres o més a cada línia depenent del micro. Per exemple: si són dues per línia:

PRINT "HOLA", "COM ESTEU"?

escriurà

HOLA COM ESTEU?

```
PRINT "HOLA", "COM ESTEU?"
HOLA          COM ESTEU?
OK
```

Però si fem:

PRINT "HOLA", "COM", "ESTEU?", BASE

escriurà

HOLA COM
ESTEU? 2

```
PRINT "HOLA", "COM", "ESTEU?", BASE
HOLA          COM
ESTEU?        2
OK
```

si és que BASE encara té assignat el valor 2.
En el segon cas, en què les separem per punts i comes (;), ho enganxarà tot, però deixant un blanc darrera els números. Per exemple

PRINT "HOLA"; "COM"; "ESTEU?"

escriurà

HOLACOMESTEU?

```
PRINT "HOLA";"COM"; "ESTEU?"
HOLACOMESTEU?
OK
```

O si fem

PRINT "HOLA"; BASE; "COM ESTEU?";

escriurà

HOLA25 COM ESTEU?

si és que el contingut de BASE és ara 25.

```
PRINT "HOLA"; BASE; "COM ESTEU?";
HOLA25 COM ESTEU?
OK
```

Cal tenir en compte, en aquest darrer cas, que quan l'ordinador es trobi amb la propera instrucció PRINT del mateix programa, el que hagi d'escriure, ho posarà a la dreta del que havia escrit anteriorment, perquè hem deixat un (;) al final de la sentència.
Si feu

```
10 PRINT "HOLA";
   " COM ESTEU?";
20 PRINT " BE"
30 END
```

```
10 PRINT "HOLA"; "COM ESTEU?";
20 PRINT "BE"
30 END
RUN
HOLA COM ESTEU? BE
OK
```

Com que escrivim aquestes tres sentències com un programa, en fer RUN, sortirà a la pantalla:

HOLA COM ESTEU? BE

Instrucció **CLS**

Fa que l'ordinador, tal com ja s'havia dit, esborri la pantalla. Es correspon amb el símbol d'operació d'ESBORRAR, d'un ordinograma.
El micro Dragon permet afegir una expressió darrera la instrucció CLS que li digui de quin color la volem deixar després d'haver-la esborrada. Aquesta expressió serà, o bé una constant numèrica que corespongui amb un color, o bé una variable numèrica, escrita entre parèntesis, el contingut de la qual sigui el número del color desitjat.
La correspondència numèrica dels color és:

0 – negre,	1 – verd,	2 – groc,
3 – blau,	4 – vermell	5 – beige,
6 – turquesa,	7 – magenta,	8 – toronja

Fem-ne un exemple:

Posem NEW per esborrar tot el que hi hagi a la RAM.
Escrivim el programa:

```
10 CLS 1 : REM PANTALLA DE COLOR VERD
20 PRINT "DEMOSTRACIO DE COLOR"
30 PRINT "INTRODUEIX EL NUMERO DEL COLOR"
40 INPUT C : REM TRIAR COLOR
50 CLS (C) : REM PANTALLA DEL COLOR TRIAT
60 END
```

En posar RUN, el micro esborra la pantalla, la deixa de color verd i escriu dos missatges. Tot seguit espera que li indiquem de quin color volem que quedi el fons de la pantalla. Cal donar el número escollit i pitjar ENTER.
Ja està! La pantalla ha quedat tal com volíem, oi?
En el ZX Spectrum per a fer el mateix s'ha d'escriure: PAPER número de color: CLS i en el Commodore s'ha de posar POKE 53281, número de color.

Instrucció **LIST**

Si en un moment donat volem mirar el programa que tenim a la memòria, però que no veiem per la pantalla, hem de fer servir la instrucció LIST.
La sentència que la inclou com a instrucció pot tenir una de les expressions següents:

a) 40-100, o qualssevol altres números de línia del programa que volem mirar, sempre que el segon número que posem sigui més gran que el primer. En aquest cas:
LIST 40-100
permetria d'escriure, per pantalla, totes les línies del programa des de la 40 fins a la 100.
b) –80, o qualsevol altre número de línia del programa que volem mirar
LIST –80
permetria d'escriure, per pantalla, les línies del programa des de la primera fins a la 80.
c) 60–, o qualsevol altre número de línia del programa que volem mirar
LIST 60–
permetria d'escriure, per pantalla, des de la línia 60 fins a la darrera del programa.
d) sense cap expressió
LIST
permetria d'escriure, per pantalla, tot el programa.
Amb aquestes sentències ja podem comenccar a fer programes, com ara:

Programa Ninot

Aquest programa dibuixa un ninot a la pantalla mitjançant un seguit d'instruccions PRINT. Cadascuna dibuixarà una línia del ninot. Hem fet servir la lletra «x» per configurar el barret i la samarreta, la «w» per als pantalons, la «z» per a les sabates,...
Si ara volguéssim canviar les lletres que dibuixen la samarreta i els pantalons hauríem de canviar les sentències 120, 130 i 140, posar la nova lletra que vulguem fer correspondre a la samarreta i canviar les línies 160, 170 i 180, posant la nova lletra que vulguem fer correspondre als pantalons. Això tant molest, ho podem resoldre fent servir variables. Com que les tres línies de dibuix de la samarreta són iguals, n'hi haurà prou de fer un PRINT de la mateixa variable tres vegades. Aquesta variable serà alfanumèrica i estarà associada a una cadena de caràcters que tingui la mateixa forma que la línia de dibuix que vulguem.
En el programa, en diem CA$, i se li pot assignar, per exemple, la cadena: "0000000".

```
10 REM ******************************
20 REM *** PROGRAMA NINOT  1   *****
30 REM ******************************
40 CLS
50 PRINT "          xxxxx"
60 PRINT "        xxxxxxxxx"
70 PRINT "         ( o o )"
80 PRINT "         (  !  )"
90 PRINT "         (  -  )"
100 PRINT "           ,,,"
110 PRINT "        ,,,,,,,,,"
120 PRINT "        x xxxxx x"
130 PRINT "        x xxxxx x"
150 PRINT "          ,,,,,"
160 PRINT "          ww ww"
170 PRINT "          ww ww"
190 PRINT "         zzz zzz"
200 GOTO 200
```

Per als pantalons es pot fer el mateix amb la variable PA$.
Les sentències:

INPUT "DIBUIX SAMARRETA:"; CA$

INPUT "DIBUIX PANTALON"; PA$

permetran d'assignar els corresponents valors de les variables CA$ i PA$ per pantalla.

Instrucció **LET:** Sentència d'assignació

La sentència d'assignació és la que representa, en un programa, el fet de carregar en una variable o bé un valor concret, o el resultat d'una expressió aritmètica, o bé el contingut d'una altra variable. El seu format és:

LET variable = expressió

```
10 REM ********************************
20 REM *** PROGRAMA NINOT  2    *****
30 REM ********************************
40 CLS
50 INPUT "DIBUIX CAMISA : ";CA$
60 INPUT "DIBUIX PANTALO: ";PA$
70 CLS
80 PRINT "              xxxxx"
90 PRINT "            xxxxxxxxx"
100 PRINT "            ( o o )"
110 PRINT "            (  !  )"
120 PRINT "            (  -  )"
130 PRINT "              ,,,"
140 PRINT "           ,,,,,,,,,"
150 PRINT "           ";CA$
160 PRINT "           ";CA$
170 PRINT "           ";CA$
180 PRINT "            ,,,,,"
190 PRINT "            ";PA$
200 PRINT "            ";PA$
210 PRINT "            ";PA$
220 PRINT "           zzz zzz"
230 GOTO 230
```

Podem posar:

LET RES = N1 * N2

Aquesta sentència es caracteritza per la instrucció LET i el signe =.
Per a la majoria dels micros, posar la instrucció LET és optatiu, no és necessari; per tant, no apareixerà en algunes dels programes que escriurem, (si el manual del vostre micro diu que és obligatori usar LET, només caldrà afegir-lo davant les sentències d'assignació).
El que si que és del tot necessari és el signe «=». Aquest signe, però, no té el mateix significat que quan fèiem comparacions; aquí no estem preguntant res, sinó assignant; per tant, serà interpretat com a **assignar** a la variable el valor de l'expressió»
Fixem-nos en la sentència següent:

P = P + 1

No vol pas dir que el valor de P és igual al valor de P incrementant en 1.
Això és impossible matemàticament, seria com dir 3 = 3+1 (3=4). Aquesta és una sentència d'assignació i, per tant, s'ha d'interpretar com: «agafar el valor de la variable P, sumar-li 1 i deixar el resultat en la mateixa variable P».
Podem dir que una sentència d'assignació es correspon amb els símbols d'operació dels ordinogrames que impliquin càlculs (A=B+2), o moviments de dades dins la memòria (B=2, o B=C).

Instrucció **GOTO**

Quan en un algorisme és necessari «trencar» l'ordre normal de fer les coses fem servir l'expressió **anar a**. En l'ordinograma això ho representem amb una fletxa que, o bé «torna enrera», o bé «se salta» uns quants símbols.
Dins un programa **anar a** s'escriu mitjançant la instrucció GOTO.
Per tant, si en un moment donat d'un programa ens interessa anar a una sentència que no és la següent (la de sota), cal posar:

GOTO número de línia

Sense carregar-ho a l'ordinador, sabríem dir el que escriurà el següent programa?

```
10  CLS
20  GOTO 70
30  PRINT "A"
40  GOTO 110
50  PRINT "E";
60  GOTO 90
70  PRINT "P";
80  GOTO 50
90  PRINT "R";
100 GOTO 30
110 END
```

Sentencia **IF...THEN...**

És una sentència de condició. Si es compleix una certa condició, farem una cosa, i si no es compleix, en farem un altra.
És la representació, en un programa, del rombe dels ordinogrames o de l'expressió SI...ALESHORES... dels algorismes. El seu format és:

IF condició THEN acció

on IF es correspon amb el «SI» i el THEN amb «ALESHORES». Sabent això, el programa del joc d'endevinar un número entre 0 i 9 seria:

```
10 CLS
20 INPUT "ENTRA EL NUMERO SECRET"; SECRET
30 CLS
40 INPUT "ENTRA EL NUMERO INTENT"; INTENT
50 IF INTENT = SECRET THEN END
60 GOTO 40
```

o bé, canviant la condició:

```
10 CLS
20 INPUT "ENTRA EL NUMERO SECRET"; SECRET
30 CLS
40 INPUT "ENTRA EL NUMERO INTENT"; INTENT
50 IF INTENT <> SECRET THEN GOTO 40
60 END
```

Quan inmediatament després de THEN s'hi troba un GOTO número de línia, es pot simplificar escrivint només el número de línia.
En aquest cas es podria escriure:

```
50 IF INTENT <> SECRET
      THEN 40
```

En el cas que volguéssim escriure el missatge «T'HAS EQUIVOCAT» abans de demanar un nou número INTENT, aprofitaríem que en una línia de programa es poden escriure diverses sentència separades per dos punts (:). Així, la línia 50 quedaria:

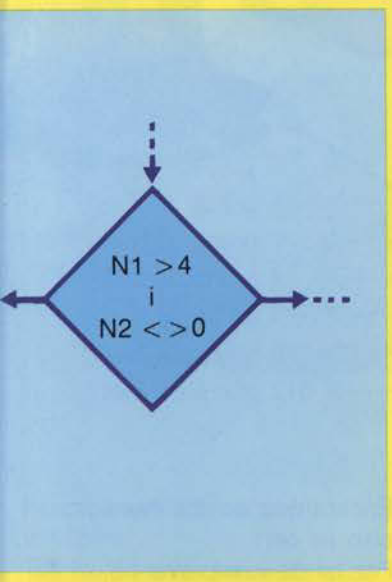
```
50 IF INTENT <> SECRET
THEN PRINT
"T'HAS EQUIVOCAT":
GOTO 40
```

Aquest exemple ens fa notar que un determinat problema té diversos algorismes que el solucionen i, per tant, molts programes possibles. Igual que en la vida quotidiana hi ha moltes maneres de fer una feina determinada, en infomàtica hi ha moltes maneres de programar un ordinador.

Operadors lògics: AND, OR i NOT

És normal que a vegades necessitem que es compleixin dues o més condicions alhora per tal de fer un seguit d'accions determinades. Per exemple, suposem que volem multiplicar el contingut de dues variables, N1 i N2, però només si la primera es més gran que 4 i la segona és diferent de 0. En aquest cas hem de posar:

- a l'algorisme:
  Si N1 és més gran que 4 **i** N2 és diferent de 0, **aleshores** multiplicar N1 per N2.
- a l'ordinograma:



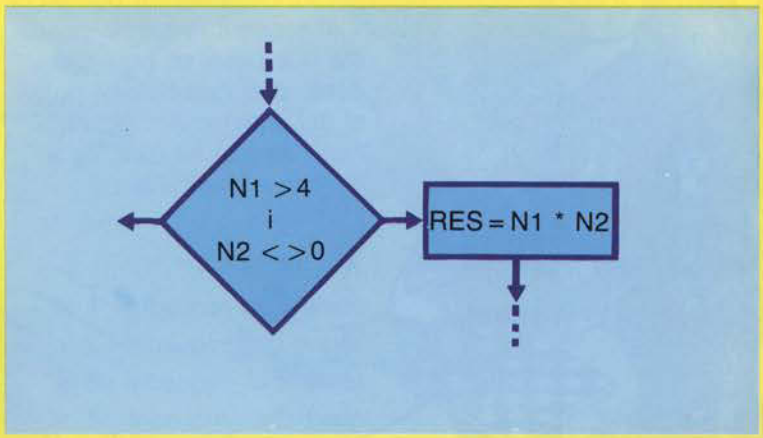
Per escriure aquesta situació en un programa, ens falta representar el mot **i** de la pregunta; la qual cosa farem mitjançant la paraula **AND**.

Així, escriurem, en el programa:

```
IF N1 > 4 AND N2 < > 0
   THEN RES = N1 * N2
```

Suposem ara que no necessitem que es compleixin alhora un seguit de condicions, sinó que en tenim prou que se'n compleixi una qualsevol d'elles.
En aquest cas cal posar:

- a l'algorisme:
  Si N1 és més gran que 4 **o** N2 és diferent de 0, **aleshores** multiplicar N1 per N2
- a l'ordinograma:



Com representar la **o** de la pregunta? ho farem mitjançant la paraula **OR**. Així escriurem en el programa:

```
IF N1 > 4 OR N2 < > 0
   THEN RES = N1 * N2
```

I finalment, si ens interessa fer quelcom només si no es compleix certa condició, en el programa farem servir el mot **NOT** davant la condició. I si aquesta és composta (com les dues condicions anteriors) l'escriurem entre parèntesis. Podríem escriure, per exemple:

```
IF NOT (N1 > 4 AND N2 < > 0)
   THEN RES = N1 * N2
IF NOT (N1 > 4ORN2 < > 0)
   THEN RES = N1 * N2
```

que seria fer exactament el contrari dels dos casos anteriors.
Mentre dels mots AND, OR i NOT se'n diuen OPERADORS LOGICS, els símbols de comparació (=. < > , < =...) s'anomenem OPERADORS RELACIONALS.

Ara que ja en sabem una mica més, anem a explicar com fer alguns programes:

Programa del blat i l'escaquer

Aquest programa resol un problema ben conegut. La història diu que: Hi havia una vegada un rei molt ric que devia un favor a un pagès del seu regne. El rei li va dir: «demana'm el que vulguis». El pagès va demanar-li quelcom d'enunciat molt senzill: "demà poseu un gra de blat al primer requadre d'un escaquer, l'endemà poseu-ne 2 al següent requadre i cada dia que passi en poseu el doble dels que haviem posat el dia anterior. Així, fins que hagin passat 64 dies, és a dir, fins a completar l'escaquer.

primer dia: requadre 1 <-------> 1 gra de blat
segon dia: requadre 2 <-------> 2 grans de blat
tercer dia: requadre 3 <-------> 4 grans de blat
quart dia: requadre 4 <-------> 8 grans de blat
cinquè dia: requadre 5 <-------> 16 grans de blat
desè dia: requadre 10 <-------> 512 grans de blat

Penseu que en una tona de blat (més o menys uns 20 sacs) hi ha 50 milions de grans!
Al cap d'un mes, el rei va fer cridar el pagès i li va dir: «En tota la terra no hi ha prou blat per fer el que m'has demanat».

Encara que sembli mentida, això es cert.
Per tal de comprovar-ho, farem el següent programa: entrant un nombre determinat de tones, s'haurà de calcular fins a quin requadre de l'escaquer es pot arribar fent el que deia el pagès.

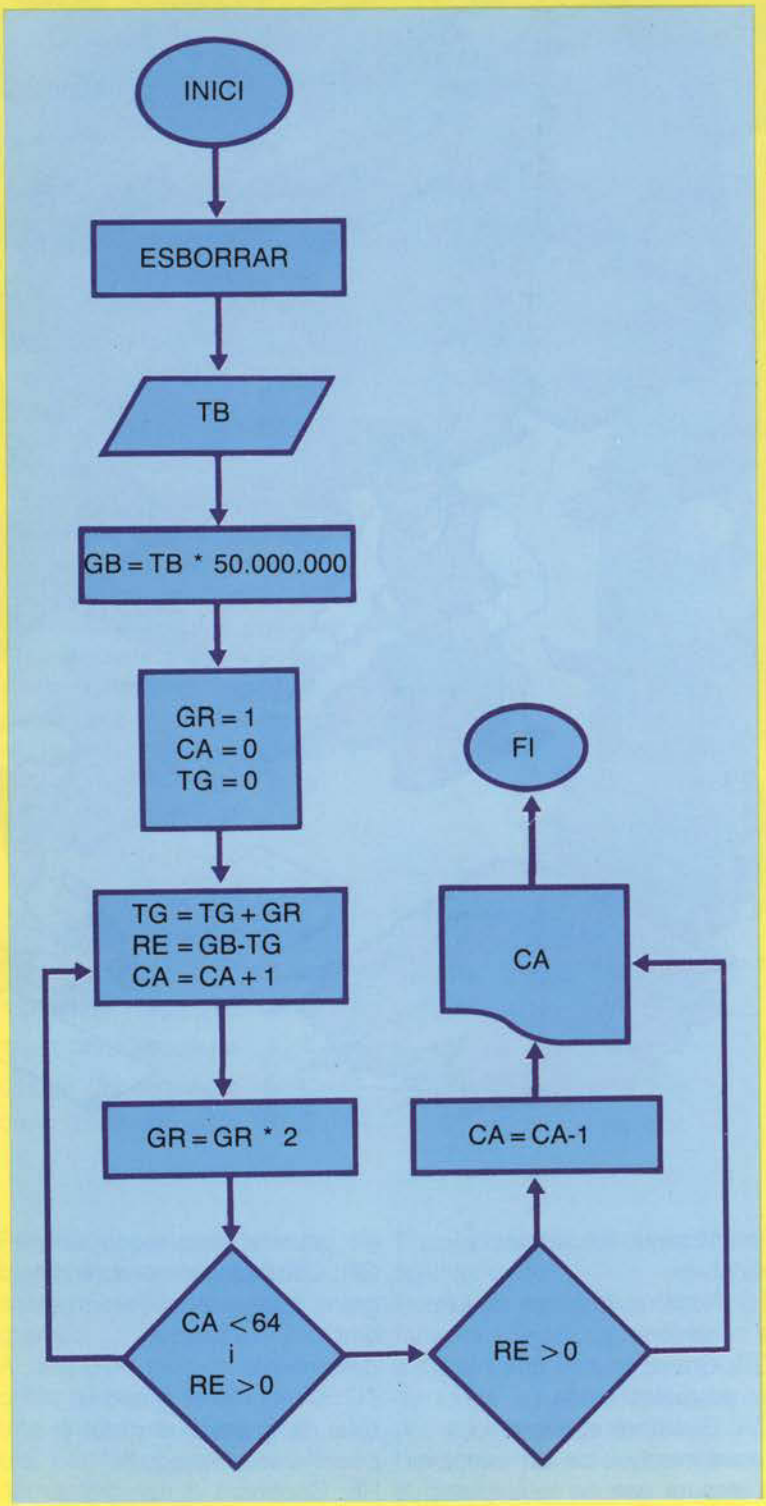
Farem servir les següents variables:
TB: Nombre de tones de blat a considerar.
GB: Grans de blat que hi ha en aquestes tones.
CA: Guardarà el nombre de requadres que es van omplint a mesura que es col·loquen els grans a cada requadre.
GR: Contindrà el nombre de grans que es necessiten per omplir un requadre determinat.
TG: Contindrà la quantitat total de grans que s'han posat a tot l'escaquer.
RE: Contindrà el nombre de grans que encara queden, a mesura que s'omplen els requadres. És el nombre de grans inicials menys els que ja s'han col·locat:

GB – TG.

L'algorisme serà:

1 – **Esborrar** la pantalla.
2 – **Llegir** TB
3 – **Calcular** l'equivalent en grans GB (50 milions per tona).
4 – **Inicialitzar** GR a 1, CA a 0 i TG també a 0. Ens disposem a omplir el primer requadre.
5 – **Calcular** el nombre TG de grans posats a l'escaquer.
6 – **Calcular** quants ens en queden.
7 – **Incrementar** el nombre de requadres, CA, per saber quin estem intentant omplir.
8 – **Calcular** el nombre de grans, GR, que hem de posar al requadre següent.
9 – **Si** no hem omplert l'escaquer i ens queden grans **aleshores anar a** 5.
10 – **Si** encara ens queden grans o ens han arribat just per omplir el requadre actual **aleshores anar** a 12.
11 – **Tornar** al requadre anterior, perquè no hem pogut omplir completament el requadre on som.
12 – **Escriure** el total de requadres plens.
13 – **Acabar.**

Fixem-nos en que la condició de la linia 10 seria:
Si RE > 0 o RE = 0 **aleshores...**
Que és equivalent a posar
Si RE> = 0 **aleshores...**
Així, l'ordinograma quedarà:



El programa BASIC seria:

```
10 REM ********************************
20 REM *** PROGRAMA DEL BLAT I L'ESCAQUER ***
30 REM ********************************
40 CLS
50 PRINT "          PROGRAMA ESCACS"
60 PRINT "          ================"
70 INPUT " TONES DE BLAT?: ";TB
80 LET GB = TB*50000000#
90 LET GR = 1 : LET CA = 0 : LET TG = 0
100 LET TG = TG+GR : LET RE = GB-TG : LET CA = CA+1
110 LET GR = GR*2
120 IF CA<64 AND RE>0 THEN GOTO 100
130 IF RE>=0 THEN GOTO 150
140 LET CA = CA-1
150 PRINT
160 PRINT "EL DARRER REQUADRE PLE ES: ";CA
170 END
```


Programa del caixer automàtic

Quan fèiem el joc d'endevinar un número, vam dir que un caixer automàtic funcionava de la mateixa manera. Per tal que un caixer ens doni els diners que li demanem, ha de passar:

– Que escrivim correctament el nostre codi personal.
– Que no demanem més diners dels que tenim al compte. Per tant, el programa, abans de fer res, ha de saber el nostre codi i el nostre saldo.

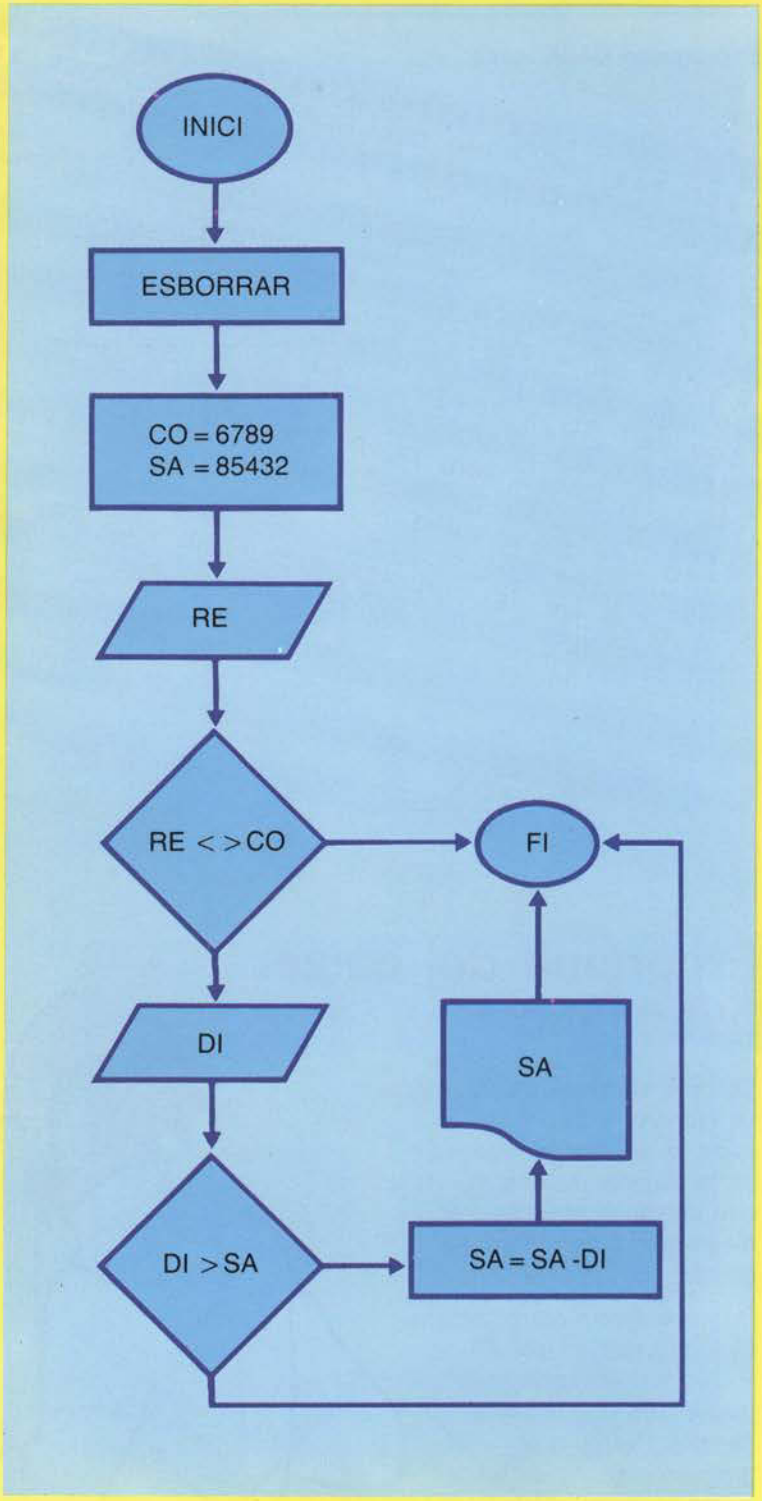
Usarem les següents variables:

CO: Codi personal secret.
SA: Saldo disponible.
RE: Codi personal que escrivim per treure diners.
DI: Diners que volem treure.

L'algorisme serà:

1: **Esborrar** la pantalla.
2: **Donar** el valor inicial al codi (està imprès a la banda magnètica de la targeta) i al saldo.
3: **Llegir** el codi que introduïm pel teclat.
4: **Si** no coincideixen amb el codi secret, **aleshores** anar a 8.
5: **Llegir** els diners que volem treure.
6: **Si** són més dels que tenim **aleshores anar a** 8.
7: **Escriure** el nou saldo.
8: **Acabar**.

En el cas que el nostre codi personal sigui 6789 i el saldo 85432, l'ordinograma seria:



I el programa:

Si teniu un micro SINCLAIR recordeu-vos d'afegir LET davant les sentències d'assignació.
Aquest programa actua de la mateixa manera que ho faria el caixer automàtic. Ja veieu, però, que hi ha diferències. El nostre ordinador no llegeix cap banda magnètica. El que hem fet és introduir aquestes dades dins el programa.
Si un programa actua tal com ho faria una màquina que ja existeix, direm que és un SIMULADOR. Aquest programa és un simulador d'un caixer automàtic.

```
10 REM ****************************
20 REM *** PROGRAMA CAIXER  ******
30 REM ****************************
40 CO=6789:SA=85432!:CLS
50 PRINT:PRINT" BENVINGUT AL CAIXER AUTOMATIC"
60 PRINT:PRINT "     TECLEJEU EL VOSTRE CODI"
70 PRINT "         SI US PLAU"
80 INPUT RE
90 IF RE <> CO THEN PRINT"* CODI INCORRECTE *"
   :END
100 PRINT:PRINT"  QUANTS DINERS VOL TREURE:"
110 INPUT DI
120 IF DI > SA THEN PRINT :
    PRINT  "SOLAMENT DISPOSEU DE  "; SA: END
130 SA = SA - DI
140 PRINT:PRINT"JA PODEU RECOLLIR LA QUANTITAT"
150 PRINT " EL NOU SALDO ES DE: ";SA
160 PRINT:PRINT  "MERCES PER LA SEVA VISITA"
```


Programa del test de multiplicació:

En aquest programa es tracta del següent: l'ordinador «genera» dos números aleatòriament i us pregunta quant val el seu producte. El micro té la capacitat de «treure's de la màniga» un número aleatori, del qual només sabem que no pot superar un valor concret. Nosaltres li direm, mitjançant l'expressió:

RND (dada)

en el cas del Dragon; o bé INT (RND*dada) per al ZX Spectrum i PC d'IBM; o be INT (RND[1]*dada) en el Commodore, que vol dir «generar un número entre 0 i el valor que representi **dada**. Aquest pot ésser una constant o una variable; això sí, sempre numèrica».
Seguirem la notació del Dragon. El resultat d'aquesta expressió, el podem escriure fent PRINT RND(10) o bé PRINT RND(AL), on AL és una variable. El que farem més sovint, però, serà guardar-lo en una altra variable per tal d'operar-hi. Aleshores podem fer, per exemple:

BA=RND(22) o X1=RND(AL)

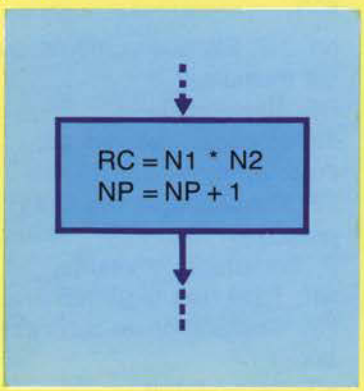
El test de multiplicació, el farem amb un número no superior a 10 i l'altre entre 0 i 100. Per generar-los posarem RND(10) per al primer i RND(100) per al segon.
El programa calcularà internament, sense dir-vos-ho, el producte dels dos números. Us donarà 3 oportunitats per encertar-ho. Si l'encerteu amb menys de 3 intents us comptarà la resposta com a bona. A la fi ens donarà el tant per cent de respostes bones sobre el total de preguntes. Les variables que farem servir són:

N1, N2: Els dos números que cal multiplicar.
RC: Resultat de la multiplicació (calculat per l'ordinador).
RE: Resultat que nosaltres proposem.
P: Comptador d'intents.
NP: Total de preguntes.
NA: Comptador de respostes bones.
NT: Percentatge.

L'algorisme serà:

1 – **Esborrar** la pantalla.
2 – **Inicialitzar** NP i NA a zero.
3 – **Generar** els dos números N1, N2.
4 – **Calcular** el seu producte RC.
5 – **Incrementar** el nombre de preguntes NP.
6 – **Escriure** N1 i N2.
7 – **Llegir** intent RE.
8 – **Si** RE és igual a RC, **aleshores anar a** 12.
9 – **Incrementar** comptador d'intents P.
10 – **Si** P és menor que 3, **aleshores anar a** 6.
11 – **Anar a** 13.
12 – **Incrementar** el comptador d'encerts NA.
13 – **Si** volem continuar, **aleshores anar a** 2.
14 – **Calcular** el percentatge NT.
15 – **Escriure** el percentatge.
16 – **Acabar**.

Per tal de simplificar una mica l'ordinograma, si hi ha dues o més operacions semblants seguides, les posarem dins el mateix símbol d'operació; així les linies 4 i 5 de l'algorisme queden:

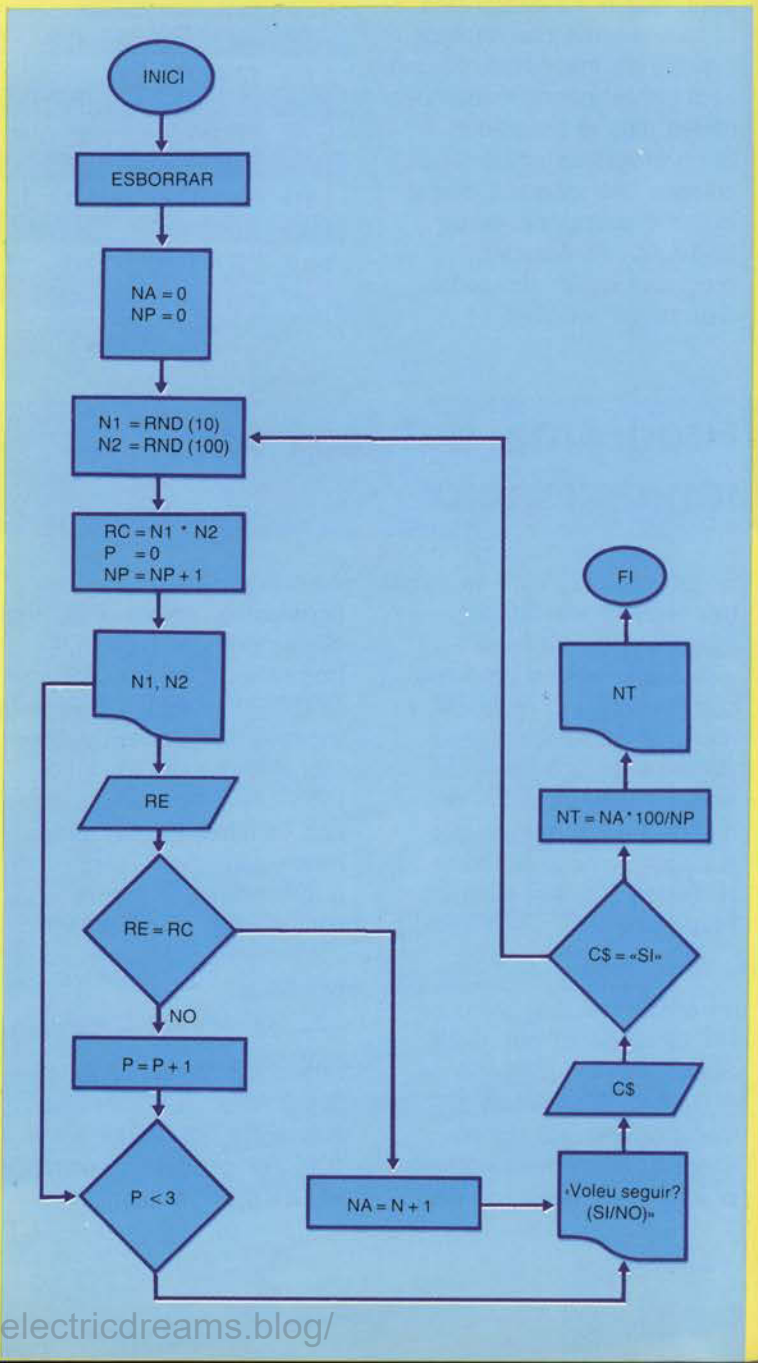


Compte!, això es pot fer sempre que no hi hagi a l'algorisme cap ordre d'anar a alguna de les línies que ajuntem (exceptuant la primera).
Per representar la pregunta: «SI voleu continuar, ALESHORES...» farem:

13.1: **Escriure** "Voleu continuar? (SI/NO):"
13.2: **Llegir** C$
13.3: **Si** C$ = "SI" **aleshores anar a** 2.

Així substituïrem la línia 13 de l'algorisme per aquestes tres.

L'ordinograma serà:



Hem afegit al tercer símbol d'operació l'assignació P = 0, ja que cada vegada que el micro genera dos números hem de comptar de nou amb 3 possibilitats per encertar el resultat.

El programa és, doncs:

```
10 REM **********************
20 REM *** PROGRAMA TEST ***
30 REM **********************
40 CLS
50 PRINT "      PROGRAMA TEST"
60 PRINT "      ============="
70 NA = 0 : NP = 0
80 N1 = RND(10) : N2 = RND(100)
90 RC = N1*N2 : P = 0 : NP = NP+1
100 PRINT " N1      : ";N1
110 PRINT " N2      : ";N2
120 INPUT " N1*N2 =? ";RE
130 IF RE = RC THEN 170
140 P = P+1
150 IF P <3 THEN PRINT "RESPOSTA INCORRECTA" : GOTO 100
160 PRINT "RESPOSTA CORRECTA = ";RC: GOTO 180
170 NA = NA+1
180 INPUT "VOLS SEGUIR?(SI/NO) ";C$
190 IF C$ = "SI" THEN 80
200 NT = NA*100/NP
210 PRINT "-----------------------"
220 PRINT "NUMERO PREGUNTES: ";NP
230 PRINT "NUMERO ENCERTS  : ";NA
240 PRINT "PERCENTATGE     : ";NT;"%"
250 END
```

Salvament i càrrega de programes

Si volem treballar l'aprenentatge de la multiplicació, el que podem fer és generar els dos números a multiplicar entre 0 i 10. Aleshores l'ordinador es convertirà en una màquina per preguntar les taules de multiplicar. A més, posa notes!

Ja hem fet alguns programes, els hem provat, i comencem a estar convençuts que, amb paciència i constància, això de programar en BASIC no és tan difícil com ens havíem arribat a imaginar.
Però on són aquests programes que acabem de fer? a la RAM, oi?

Apagar el micro ara voldria dir que perdríem tot el que acabem de fer, i si un altre dia volem veure el ninot, o el simulador de caixer automàtic, o qualsevol dels que s'han carregat, hauríem de tornar-los a escriure un altre cop. Cal, doncs, abans d'apagar el micro, saber com

s'ha de salvar el treball que acabem de fer.
Es necessita un enregistrador de cassettes.
L'enregistrador de cassettes servirà com a perifèric per poder guardar els programes y les dades que es tenen a la RAM, i viceversa, per poder passar a la RAM qualsevol dada o programa que tinguem guardat en una cinta de cassette.
Perque l'enregistrador de cassettes permeti aquesta operació cal que:

1) tingui una entrada amb la indicació AUX o LINE IN, que permetrà d'enregistrar d'una font externa a la RAM.
2) tingui una sortida per a altaveu exterior o auricular. Normalment posarà MONIT, EAR, LIS, SPKR.
3) permeti de parar i engegar per control remot. Això serà si té una entrada amb l'etiqueta REM.
4) es pugui connectar a la xarxa elèctrica.
5) disposi de tots els fils que permetin la connexió amb el micro.

Malgrat aquestes condicions, és aconsellable de consultar el manual del micro i seguir totes les indicacions que doni; en cas de dubtes, podem recórrer a l'amic més expert o a la casa on s'ha comprat l'aparell. Demanar a quina posició de volum ha d'estar l'enregistrador per permetre una bona connexió (normalment és a la mateixa posició en què s'enregistra qualsevol música, si fa no fa a la meitat de volum).

Instrucció CSAVE o (SAVE)

Aquesta és la instrucció que usarem per salvar o desar un programa.
Un cop feta la connexió de l'enregistrador de cassettes amb el micro, rebobinarem la cinta fins al començament, en el cas que encara no hi tingui cap programa o dada, i posarem el comptavoltes a zero.
Pitgem les tecles de l'enregistrador: PLAY i RECORD (com si es tractés d'enregistrar una cançó), escrivim per teclat:

CSAVE "nom del programa" i pitgem la tecla ENTER.

El nom del programa ha de tenir, com a màxim, vuit caràcters i sempre ha de començar per una lletra.
Suposem que volem salvar el programa NINOT1. Faríem:

CSAVE "NINOT1" i pitjar la tecla ENTER.

Al cap d'una estona, depenent de com sigui de llarg el programa, sortirà per pantalla el missatge OK i es pararà l'enregistrador. Ja s'ha salvat, enregistrat, el programa NINOT1.
Cal apuntar el número del comptavoltes, perquè si hi volem enregistrar altres programes necessitarem saber quant ocupa cada programa que ja hi tenim, per saber on s'ha de començar la propera vegada que vulguem salvar un nou programa. És prudent que entre programa i programa s'hi deixin algunes voltes buides.
Encara que s'ha salvat el programa, en aquest cas NINOT1, continua estant a la RAM, i per això es pot continuar usant.

Instrucció CLOAD o (LOAD)

Aquesta és la instrucció que serveix per carregar un programa que tenim guardat en una cinta de cassette, a la memòria (RAM) del micro.
Els passos a fer per carregar un programa són:

1) Posar NEW, per un si de cas teníem algun programa dins la memòria.
2) Rebobinar la cinta fins al començament. O bé, en el cas de saber el número de volta en què comença el programa, rebobinar-la fins aquesta volta.
3) Escriure, per teclat: CLOAD "nom del programa" i pitjar ENTER

En el cas de voler carregar el programa NINOT1, farem:
NEW
Rebobinar la cinta
CLOAD "NINOT1"
Ara, només cal esperar que surti, per pantalla, el missatge OK.
Sempre el nom del programa a carregar ha de coincidir amb el nom que li hem donat quan l'hem salvat.
Per assegurar-se que el programa s'ha carregat bé, es pot fer amb la instrucció LIST.

Altres programes

Al final de cada llibre hi haurà una colla de programes, sense l'ordinograma, ni comentar-los pas a pas.
N'hi ha uns quants en què sobren les explicacions, perquè són només petites variacions dels que ja s'han exposat; uns altres no es comenten perquè tenen instruccions que encara no coneixem; es posem, però, perque és important que, amb l'ajut del manual, ens hi barallem una mica.
El que es pot fer, també, és pensar el problema i fer el programa més curt (menys línies). Si aquest és el cas, s'haurà perfeccionat.
Millorar un programa no és treure'n els missatges ni els comentaris, sinó que vol dir aconseguir els mateixos resultats amb menys sentències:
S'ha de variar l'estructura lògica del programa.

```
10 REM ************************
20 REM *** PROGRAMA CASA ****
30 REM ************************
40 CLS
50 PRINT"                    OOOO"
60 PRINT"                    OO"
70 PRINT"                  HHH"
80 PRINT"                 HHH"
85 PRINT"                 HHH"
90 PRINT"        ################"
100 PRINT"       ################"
110 PRINT"      ##################"
120 PRINT"      H                 H"
130 PRINT"      H   FFFF          H"
140 PRINT"      H   FFFF          H"
150 PRINT"      H   FFFF          H"
160 PRINT"      H          DDD    H"
170 PRINT"      H          DDD    H"
190 PRINT"      H          DDD    H"
200 PRINT"    HHHHHHHHHHHH DDD HHHHH";
210 GOTO 210
```

```
10 REM ************************
20 REM *** PROGRAMA CARA ****
30 REM ************************
40 CLS
50 PRINT"              *****"
60 PRINT"           **********"
70 PRINT"       ******      ******"
80 PRINT"     ****              ****"
90 PRINT"     ***     O       O    ***"
100 PRINT"    ***     O+O     O+O    ***"
110 PRINT"    ***      O  !!  O     ***"
120 PRINT"    ***         !!        ***"
130 PRINT"    ***         !!        ***"
140 PRINT"     ****       ##      ****"
150 PRINT"     ****              ****"
160 PRINT"      ****     ====    ****"
170 PRINT"        *****      *****"
180 PRINT"           **********"
190 PRINT"              ****"
200 GOTO 200
```

```
10 REM **********************
20 REM *** PROGRAMA TEST ***
30 REM **********************
40 CLS
50 PRINT "      PROGRAMA TEST"
60 PRINT "      =============="
70 LET NA = 0 : LET NP = 0
80 LET N1 = 1+INT(RND*10) : LET N2 = 1+INT(RND*100)
90 LET RC = N1*N2 : LET P = 0 : LET NP = NP+1
100 PRINT " N1      : ";N1
110 PRINT " N2      : ";N2
120 INPUT " N1*N2 =? ";RE
130 IF RE = RC THEN GOTO 170
140 LET P = P+1
150 IF P <3 THEN PRINT "RESPOSTA INCORRECTA" : GOTO 100
160 PRINT "RESPOSTA CORRECTA = ";RC: GOTO 180
170 LET NA = NA+1
180 INPUT "VOLS SEGUIR?(SI/NO) ";C$
190 IF C$ = "SI" THEN GOTO 80
200 LET NT = NA*100/NP
210 PRINT "----------------------"
220 PRINT "NUMERO PREGUNTES: ";NP
230 PRINT "NUMERO ENCERTS  : ";NA
240 PRINT "PERCENTATGE     : ";NT;"%"
250 END
```


Per executar aquest programa en un Commodore, només s'ha de substituir RND*10 i RND*100 per RND (1)*10 i RND (1)*100.
La versió per Dragon ja l'hem vist abans.

```
10 REM ****************************
20 REM *** PROGRAMA CAIXER ******
30 REM ****************************
40 CO=6789:SA=85432!:CLS
50 PRINT : PRINT "  BENVINGUT AL CAIXER AUTOMATIC"
60 PRINT : PRINT "      TECLEJEU EL VOSTRE CODI"
70 PRINT "         SI US PLAU"
80 INPUT RE
90 IF RE <> CO THEN GOTO 200
100 PRINT : PRINT "    QUANTS DINERS VOL TREURE: "
110 INPUT DI
120 IF DI > SA THEN PRINT : PRINT  "SOLAMENT DISPOSEU DE  "; SA: GOTO 100
130 SA = SA - DI
140 PRINT:PRINT " JA PODEU RECOLLIR LA QUANTITAT"
150 PRINT " EL NOU SALDO ES DE: ";SA
160 PRINT : PRINT  "MERCES PER LA SEVA VISITA"
170 A$=INKEY$
180 IF A$ <>"" THEN GOTO 40
190 GOTO 170
200 PRINT:PRINT "*** ERROR - CODI INCORRECTE ***
210 PRINT"      TECLEJEU DE NOU EL CODI"
220 PRINT"      SI US PLAU":GOTO 80
```


Per ZX Spectrum i ZX 81 s'ha d'afegir LET davant les assignacions.

```
10 REM **********************
20 REM *** PROGRAMA RENDA****
30 REM **********************
40 CLS : PRINT"   DECLARACIO DE RENDA "
50 PRINT "   ==================":PRINT
60 PRINT:PRINT "=INGRESSOS:"
70 PRINT:INPUT"T. PERSONAL    ";TP
80 INPUT "INT.CTES.CORR ";CC
90 LET TI=TP+CC
100 PRINT "                ---------------"
110 PRINT"T.INGRESSOS:             ";TI
120 PRINT:PRINT"=DESPESES: "
130 INPUT"COTIZAC. S.S. ";SS
140 PRINT"DESP.DIF.JUST.    ";TP/100
150 LET TD=SS+TP/100
160 PRINT"                ----------------"
170 PRINT"T.DESPESES:               ";TD
180 LET DI=TI-TD
190 PRINT:PRINT"DIF.INGRESSOS/DESPESES";DI
195 PRINT
200 PRINT"SI US PLAU, MIREU A LA TAULA";
205 PRINT"  QUINA QUANTITAT US CORRESPON PAGAR"
210 PRINT:INPUT"QUOTA INTEGRA:PESSETES: ";PS
220 LET DG=16500
230 PRINT:PRINT"DEDUCCIO GENERAL:";DG
240 INPUT"PER MATRIMONI    ";MA
250 INPUT"PER FILLS        ";FI
260 LET ML=10000
270 PRINT"DESPESES MALATIA:";ML
280 LET RN=(TP-SS-TP/100)/100
290 PRINT"REND.NETS TREBALL ";RN
300 PRINT"                  -------------"
310 LET SD=DG+MA+FI+ML+RN
320 PRINT"SUMA DEDUCCIONS:          ";SD
330 LET DI=PS-SD
340 PRINT" DIFERENCIA:....          ";DI
350 PRINT"=A DEDUIR:"
360 PRINT:INPUT"RET.REN.TRE.PER.";RT
370 INPUT"RET.REN.CAPITAL ";RC
380 PRINT"                   --------------"
390 LET TR=RT+RC
400 PRINT"TOTAL RETENCIONS          ";TR
410 PRINT:PRINT"TOTAL A PAGAR .........: ";DI-TR
```

```
10 REM ******************************************
20 REM *** PROGRAMA DEL BLAT I L'ESCAQUER *******
30 REM ******************************************
40 GR=1 : CA=1 :TG =1
50 CLS:PRINT"          PROGRAMA ESCACS"
60 PRINT"      ---------------"
70 INPUT "TONES DE BLAT: ";TB
80 GOSUB 180
90 GB = TB * 50000000#
100 IF CA=64 THEN GOSUB 340:
PRINT@449,"    L'ESCAQUER ES PLE":GOTO170
110 GOSUB 340
120 CA=CA+1:GR=2*GR
130 TG=TG+GR
140 RE=GB-TG
150 IF CA <= 64 AND RE > 0 THEN 100
160 PRINT @449,"L'ULTIMA CASELLA PLENA ES:";CA-1
170 GOTO 170
180 REM ** DIBUIX DEL TAULER **
190 CLS
200 PRINT : PRINT
210 GOSUB 300:PRINT CHR$(142);:FOR I=1 TO 16:
    PRINT CHR$(131);:NEXT I: PRINT CHR$(141)
220 FOR I = 1 TO 4
230 GOSUB 310
240 FOR J = 1 TO 4: GOSUB 320:GOSUB 330:NEXT J:
     PRINT CHR$(133)
250 GOSUB 310
260 FOR J = 1 TO 4: GOSUB 320:GOSUB 330:NEXT J:
     PRINT CHR$(133)
270 NEXT I
280 GOSUB 300:PRINT CHR$(139);: FOR I = 1 TO 16:
    PRINT CHR$(131);:NEXT I:PRINT CHR$(135)
290 RETURN
300 PRINT "        ";:RETURN
310 PRINT "        ";:PRINT CHR$(138);:RETURN
320 FOR K=1 TO 2:PRINT CHR$(128);:NEXT K:RETURN
330 FOR K=1 TO 2:PRINT CHR$(143);:NEXT K:RETURN
340 IF (CA - INT(CA/2)*2)=0 THEN GOTO 350 ELSE
    GOTO 370
350 IF (INT(CA/8,1)+1 - INT(CA/8,1+1)/2)*2)=0 THEN
    GOSUB 440 ELSE 390
360 RETURN
370 IF (INT(CA/8,1)+1 - INT((INT(CA/8,1)+1)/2)*2)=1 THEN
          GOSUB 440 ELSE 390
380 RETURN
390 FOR I = 1 TO 10
400 PRINT @70+INT(CA/8,1)*32+(CA-((INT(CA/8,1)-1*8)+8)*2,
   CHR$(128);:PRINT CHR$(128);
410 PRINT @70+INT(CA/8,1)*32+(CA-((INT(CA/8,1)-1*8)+8)*2,
   CHR$(143);:PRINT CHR$(143);
420 NEXT I
430 RETURN
440 FOR I = 1 TO 10
450 PRINT @70+INT(CA/8,1)*32+(CA-((INT(CA/8,1)-1*8)+8)*2,
   CHR$(143);:PRINT CHR$(143);
460 PRINT @70+INT(CA/8,1)*32+(CA-((INT(CA/8,1)-1*8)+8)*2,
   CHR$(128);:PRINT CHR$(128);
470 NEXT I
480 RETURN
```

Versió no més per Dragon.

L'Obra Social de la Caixa de Pensions
posa a la vostra disposició,
dins la seva xarxa de Biblioteques
el nou servei de

MICROTEQUES

La biblioteca,
entesa com un espai de recursos per a la informació i la formació,
amb aquest servei incorpora noves tècniques i mitjans
que permetran l'usuari
interactuar amb la documentació i les dades.

A la MICROTECA trobareu:

- Uns microordinadors i uns perifèrics que han estat triats en funció dels objectius del servei.
- Uns programes i uns paquets informàtics seleccionats en funció de la seva qualitat, autonomia explicativa i llengua.
- Una informació escrita i documentació variada que permetrà usar eficaçment les eines informàtiques.
- Els serveis normals de la biblioteca i dels seus professionals que us posaran a disposició bibliografia especialitzada i l'eficàcia organitzativa del servei.
- Un programa de dinamització que promourà accions de reflexió, informació i estudi a l'entorn dels temes informàtics de major interès.

Les primeres Microteques
començaran a funcionar properament a les següents Biblioteques:

TARRAGONA
C. Colom, 2

LLEIDA
C. Bisbe Torres, 2

GIRONA
C. Migdia, 32

BARCELONA
C. Clot, 21-25

MANRESA
Guimerà, 1

Podeu demanar més informació al telèfon (93) 302 54 04 ext. 206

Col·lecció

connecta el micro

1 FEM INFORMÀTICA

2 IMATGES I SONS

3 TRACTAMENT DE LA INFORMACIÓ

4 ROBOTS I INTEL·LIGENCIA ARTIFICIAL

5 NOVES TECNOLOGIES

Servei de Publicacions de la

FUNDACIÓ CAIXA DE PENSIONS

El programa «connecta el micro» és una iniciativa conjunta de Caixa de Pensions «la Caixa» i TV3

275 Ptes.

9 788450 514759