TV3
TELEVISIÓ DE CATALUNYA

connecta el micro

3 TRACTAMENT DE LA INFORMACIÓ

CURS DE BASIC

FUNDACIÓ CAIXA DE PENSIONS

connecta el micro

3 TRACTAMENT DE LA INFORMACIÓ

CURS DE BASIC

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

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

Coordinadors de l'edició:
JORDI SALA
ALBERT SÒRIA

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

Disseny de portada i compaginació:
EQUIP 30/53

Dibuix:
ROGER

Fotografia:
ESPARBÉ

Fotocomposició:
CATALANA DE FOTOCOMPOSICION, S.A.

Gestió edició:
MUNDO CIENTÍFICO

Guió de televisió:
JAUME AGUSTÍ I CULELL

Primera edició: juny 1985

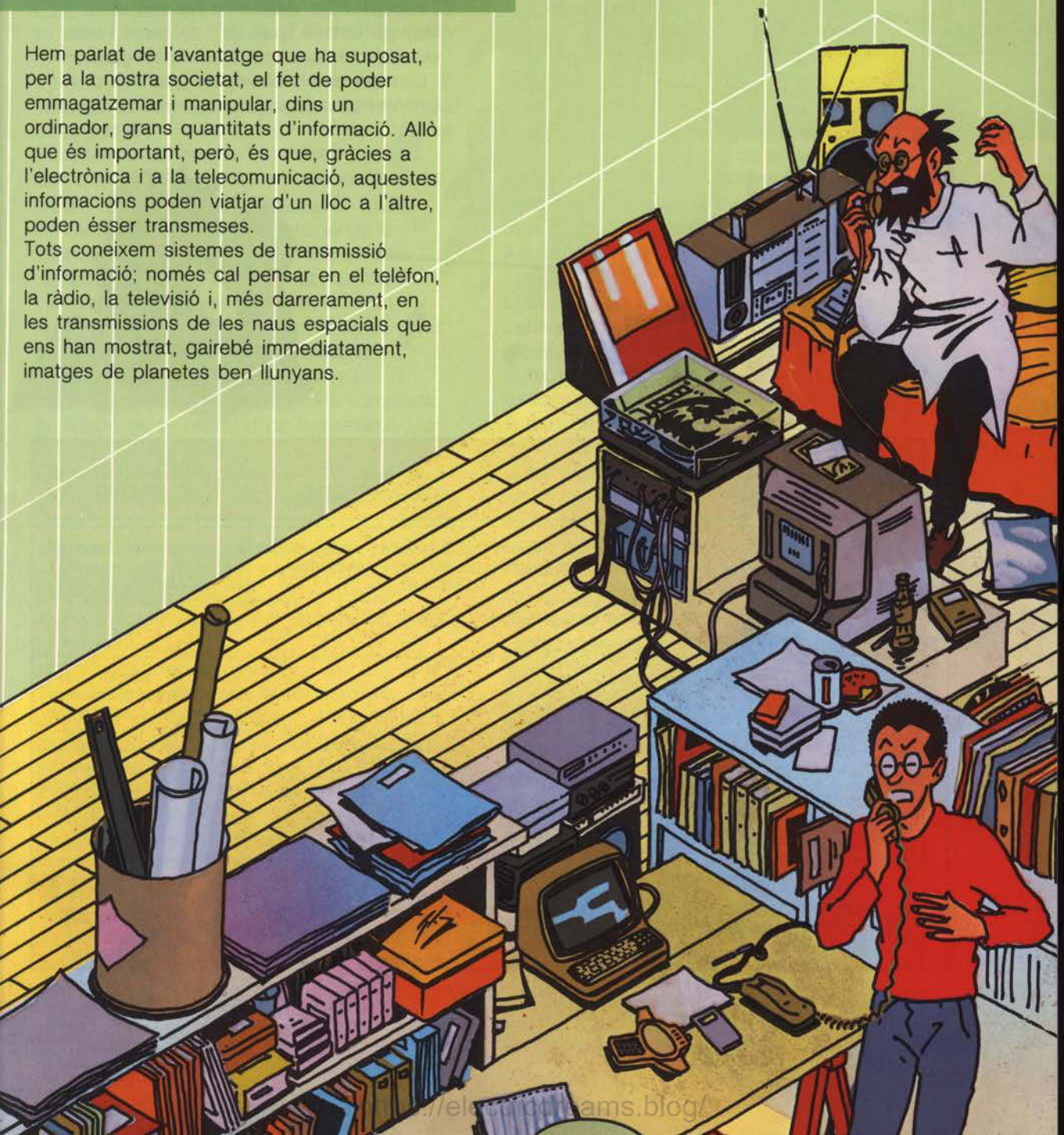
Impressió:
TONSA, Herrera-Alza. Donostia

ISBN: 84-505-1477-0
Dipòsit legal: S.S. 300-85

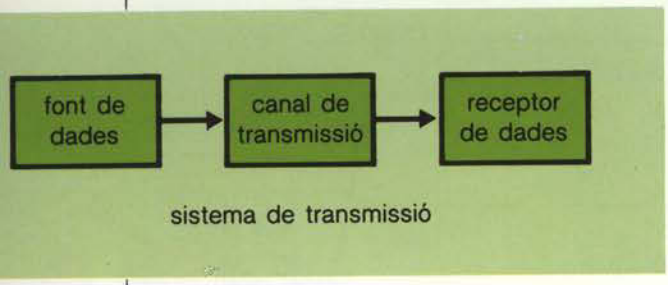
La transmissió d'informació

Hem parlat de l'avantatge que ha suposat, per a la nostra societat, el fet de poder emmagatzemar i manipular, dins un ordinador, grans quantitats d'informació. Allò que és important, però, és que, gràcies a l'electrònica i a la telecomunicació, aquestes informacions poden viatjar d'un lloc a l'altre, poden ésser transmeses.

Tots coneixem sistemes de transmissió d'informació; només cal pensar en el telèfon, la ràdio, la televisió i, més darrerament, en les transmissions de les naus espacials que ens han mostrat, gairebé immediatament, imatges de planetes ben llunyans.



Aquests sistemes de transmisió, per més que a cop de vista semblin ben diferents, tots ells poden representar-se pel següent diagrama:



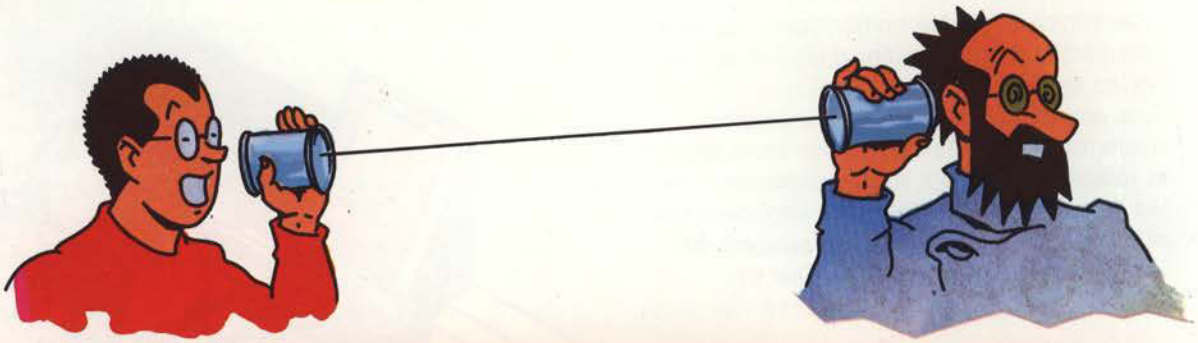
sistema de transmissió

La font de dades és la que emet els missatges i la que els assigna un valor numèric, per tal que puguin ésser tractats per un ordinador. És a dir, la font decideix el missatge a emetre, i li assigna un "paquet" de zeros i uns.

El canal de transmissió és el medi per on viatja la informació.

El receptor de dades és qui rep el missatge com a "paquet" de zeros i uns, i els transforma en quelcom intel·ligible.

Si pensem en els exemples que s'han esmentat abans, podem construir la següent taula:



SISTEMA DE TRANSMISSIÓ	FONT DE DADES	CANAL DE TRANSMISSIÓ	RECEPTOR DE DADES
EL TELÈFON	És l'aparell de qui marca el número i comença a parlar. Les paraules són transformades per la mateixa font en estímuls elèctrics.	El fil telefònic, per on viatja la informació: les nostres paraules.	És l'aparell de qui escolta. El receptor transforma els estímuls elèctrics en sons intel·ligibles per l'home.
LA RÀDIO	És l'emissora que realitza un determinat programa. Aquí es transforma el so en ones electromagnètiques	L'aire	L'aparell de ràdio de casa nostra, que transforma les ones electromagnètiques en sons audibles per l'home.
NAU CENTRE ESPACIAL	La nau que viatja per l'espai, transforma les imatges en "paquets" de zeros i uns que donen lloc a ones.	L'aire	L'ordinador del centre espacial rep els "paquets" de zeros i uns, i els tracta per tal de reproduir la imatge original.
CLIENT BANC DE DADES	L'ordinador de qui vol saber la informació. Transforma les seves comandes en zeros i uns, que donen lloc a estímuls electrics.	Fil telefònic	L'ordinador central on hi ha la informació d'un determinat tema a la qual volem accedir.

Aquest últim exemple i el del telèfon són diferents dels altres dos. Notem que, aquí, tenim allò que en podem dir un **sistema d'anada i tornada.** En el sistema client-banc de dades, quan comença la transmissió, la font de dades és l'ordinador del client, mentre que el receptor és l'ordinador de la central de dades. Un cop feta la comanda, els papers s'inverteixen: l'ordinador del client és ara el receptor i, el de la central, la font. Això no passa amb altres sistemes de transmissió, com ara la ràdio o bé la televisió, on la informació viatja sempre en un mateix sentit.

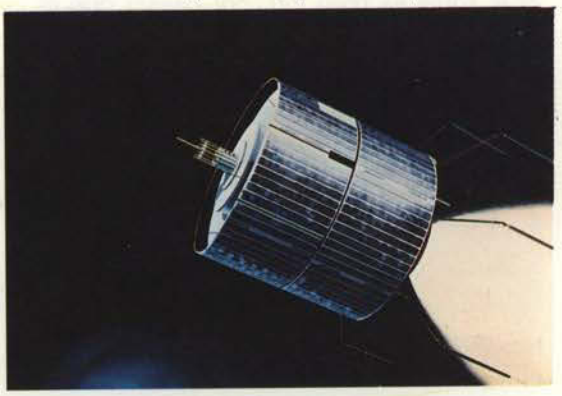
Suposem que hem de "transmetre" aigua des d'un riu fins als dipòsits generals d'una ciutat. Per fer això, cal estudiar el "canal" (la canonada) que ha de transportar la informació (l'aigua) des d'un lloc a l'altre. La física ens dóna fòrmules per a calcular l'amplada de la canonada, sabent les necessitats dels habitants de la ciutat. S'ha de fer un estudi previ per a calcular el nombre de litres per segon que s'hi necessiten: aleshores es decidirà l'amplada, en funció de la velocitat de transmissió i del cost de la canonada.

Cal fer-ho així perquè, si no es fes així, ens podríem trobar amb una canonada molt ample –i, per tant, cara– que mai no aniria a plena del tot; o bé al contrari, amb una canonada estreta –més barata– que provocaria que molts usuaris quedessin sense aigua a les hores punta. Cal analitzar prèviament la situació.

Uns estudis com aquests, són els que fa la **Teoria Matemàtica de la Informació.** És una nova branca de la Matemàtica, tal com indica el seu nom, que té per objectiu de **mesurar la informació.** Igual que la Física mesura l'aigua transmesa en litres/segon, la velocitat en metres/segon o km/hora, les distàncies en metres o també en anys llum... i ens dóna fórmules per a calcular l'amplada de la canonada que hem de fer servir, la teoria matemàtica de la informació mesura la transmissió d'informació en bits/segons, i també arriba a donar fórmules per a calcular l'"amplada" dels canals de transmissió: la **capacitat.** Per tant, donat tot un conjunt d'informacions a transmetre, i segons siguin les seves característiques, podrem determinar quin canal cal triar per transmetre-les més ràpidament, més fiablement i també, si s'escau, de la manera més barata possible.

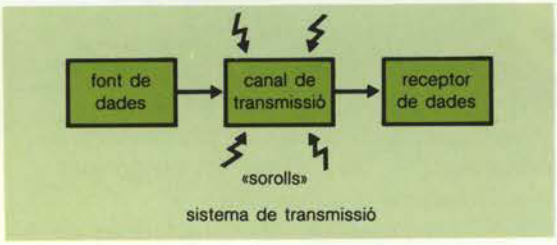
Aquesta teoria sorgí a finals dels anys 1940, i podem dir que "el pare de la criatura" és Claude E. SHANNON, qui treballava als Bell Laboratoires (AT&A) dels E.U.A. Les seves idees són vigents encara avui, i en cada país hi ha grups de gent que dirigeixen els seus treballs de recerca en aquesta direcció.

A part de saber trobar el millor canal per a transmetre unes determinades informacions, hi ha el problema de saber contrarrestar les **interferències** que es produeixen al pas de les informacions pel canal de transmissió. Quan la informació ha de recórrer grans distàncies, el més probable és que, pel camí, es trobi amb interferències **(sorolls)** i que en resulti alterada (penseu, per exemple, l'efecte

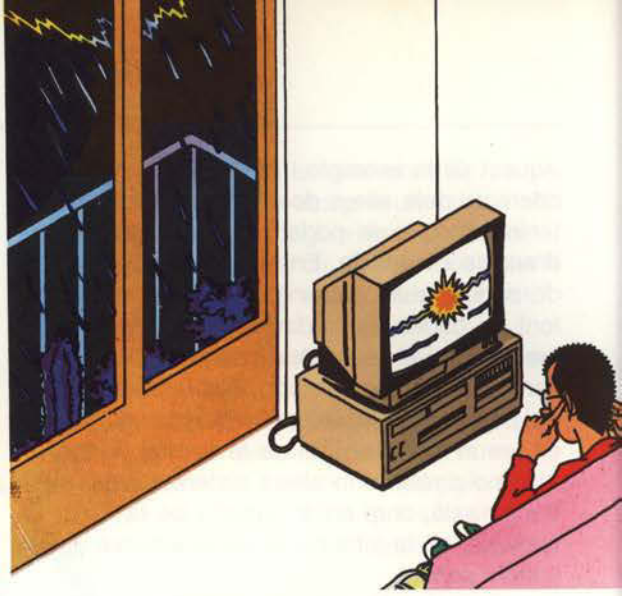


Satèl·lit de comunicacions ATS-B (Salmer)

que pot produir una tempesta en la recepció dels senyals de ràdio o televisió).
Això vol dir que, en realitat, cal representar el sistema de transmissió per:



sistema de transmissió

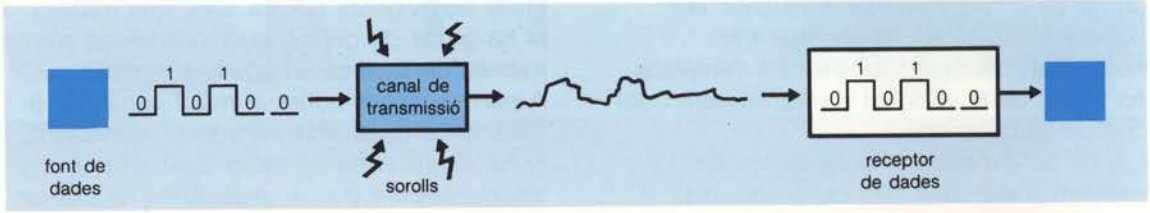


Ja hem explicat que la mateixa font de dades assigna un valor numèric a cada missatge que s'ha de transmetre, un "paquet" de zeros i uns. Podem pensar que la font de dades emet un senyal, una ona, que travessa el canal de transmissió i arriba, encara que degradada, al receptor.

Hi haurà tants errors com el nombre de bits (xifres) diferents hi hagi entre el missatge enviat i el rebut. Així:

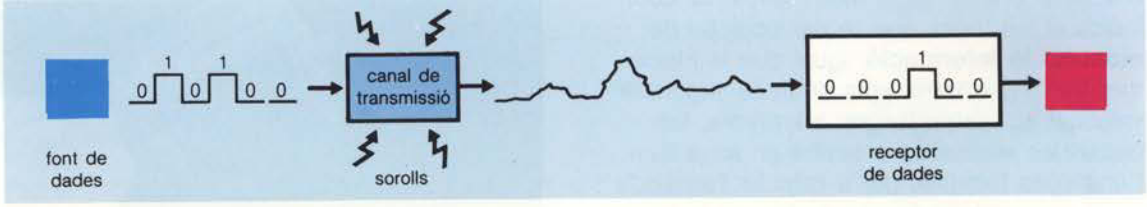
si enviem 0 0 1 0 1 0 0 1
i rebem 0 1 1 0 0 0 1 1

direm que se'ns han produït 3 errors.



És el receptor el que cerca l'ona que més s'assembla al senyal rebut. Un cop recuperat, el transforma en el missatge que representa (un color, un so, una lletra...).
Si el soroll és prou fort, pot passar que el receptor es confongui, a l'hora de recuperar l'ona rebuda:

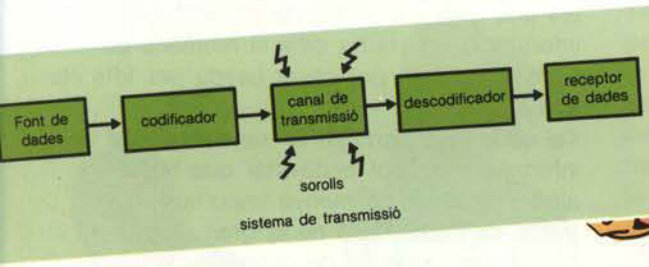
Per solucionar aquest problema, sorgí la **Teoria de la Codificació** i els anomenats **codis correctors.** Aquesta teoria no fa referència al canal, sinó que introdueix dos nous elements al sistema de transmissió: són el **codificador** i el **descodificador.** Aquesta teoria de la codificació tracta,



Nosaltres havíem enviat l'ona que representa el missatge 010100 i hem rebut la que representa el missatge 000100. Diguem que se'ns ha produït **un error,** ja que, en el missatge rebut, el segon bit està malament.

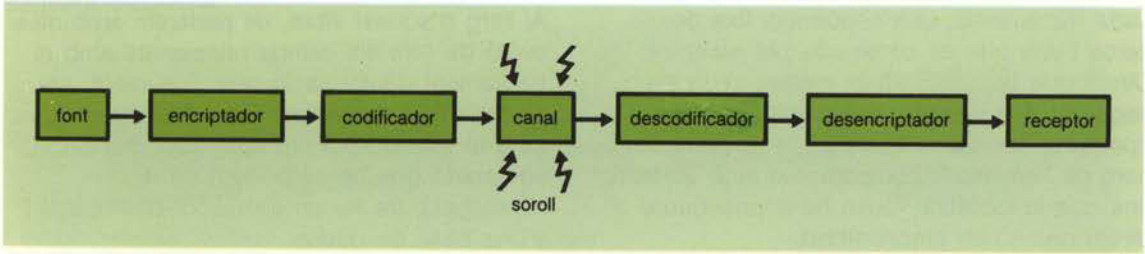
donada una determinada informació, de cercar com codificar-la de manera que el descodificador sigui capaç de "detectar" que s'hi han produït errors (aleshores donarà ordre a la font perquè torni a emetre el

missatge). A més, en alguns casos no només es detecten els errors, sinó que el propi descodificador és capaç de corregir-los.
El sistema de transmissió ens quedarà, ara:

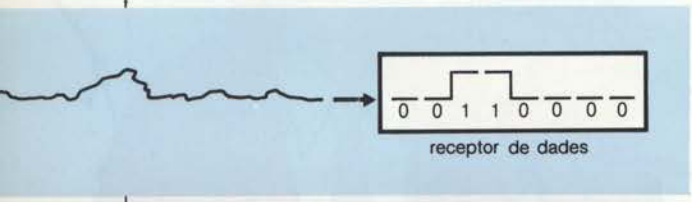


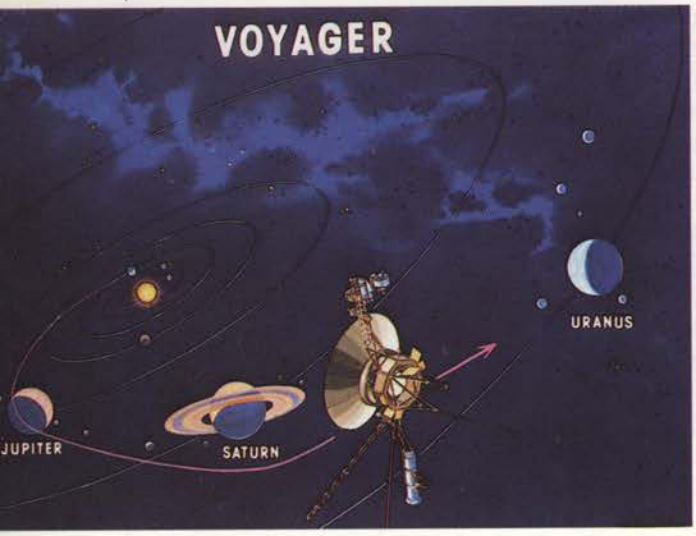
sistema de transmissió

El codificador transforma un "paquet" de zeros i uns (que li envia la font) en un altre "paquet" més llarg, també de zeros i uns (o potser en altres símbols). Aquesta transformació comporta augmentar el temps de transmissió, i eventualment també el seu cost, però s'hi guanya en fiabilitat.
Aquesta teoria nasqué, a finals dels anys quaranta, amb els treballs de Hamming i de Golay, i tingué la seva època de màxim esplendor a començaments dels anys setanta, amb els treballs de Reed i de Solomon per als JPL ("Jet Propulsion Laboratoires") de la NASA. Actualment, molts científics treballen en aquest domini, per a trobar bons codis i mètodes de descodificació cada cop més ràpids.
En transmetre informació d'un cantó a l'altre, apareix un nou problema: **el secret,** ja que, avui, les informacions confidencials ja no viatgen en maletes tancades amb "pany i clau" ni en "caixes fortes", sinó que ho fan per un medi tan vulnerable com és l'aire.
Una nació petita com la nostra no es pot permetre el luxe de construir naus que viatgin per l'espai per enviar-nos imatges de planetes llunyans. Però, si els americans ja ho fan, i és cert que la informació viatja per l'aire, per què gastar nosaltres diners i enforços? Només cal invertir en unes antenes potents, d'abast "ben llarg" i, a l'igual que la informació arriba a Califòrnia, ens arribarà a nosaltres.
El problema no té una solució tan senzilla. És cert que, tal i com reben la informació de l'espai els americans la podríem rebre nosaltres; però aquesta informació va **encriptada.** És a dir, quan la rebéssim, **mai** no seríem capaços de saber què ens volen dir en veritat aquelles col·leccions de zeros i uns: tindríem la informació, però no la podríem utilitzar mai.
La **Criptografia** és una "ciència" vella; els romans ja encriptaven els seus missatges abans de donar-los als missatgers perquè els portessin a altres persones. Ha estat, però, l'aparició dels ordinadors allò que ha anul·lat tots els mètodes antics i ha fet néixer noves tècniques.
Amb aquesta innovació haurem de representar el sistema de transmissió d'informació com:



Quan un sistema de transmissió no envia cap missatge, és a dir, quan la font no emet res, allò que arriba al receptor només són sorolls. Pensem, per exemple, en l'ordinador que rep missatges des d'una nau espacial.





Esquema de vol del Voyager *(Salmer)*

L'ordinador pot estar informat que el dia 4 d'abril, a les 10 hores, 57 minuts i 32 segons rebrà un missatge; però ell no pot saber on acaben els sorolls i on comença exactament allò que se li vol transmetre.
Per això, caldrà **sincronitzar** l'emissor (nau) i el receptor. Això s'aconsegueix enviant durant un cert període de temps, abans de cada transmissió, una seqüència fixa de zeros i uns que és coneguda pel receptor. Direm que l'emissor envia moltes vegades seguides el seu **nom.** El receptor va triant "paquets" de zeros i uns de la llargada del nom de l'emissor i comparant-lo amb aquests fins que el localitza. Quan ho aconsegueixi direm que s'han **sincronitzat.**

Els sistemes de transmissió d'informació més usuals són les **xarxes d'ordinadors,** també anomenades **xarxes de teleprocés.** Això és: una col·lecció d'ordinadors interconnectats els uns amb els altres, de manera que la informació que hi ha dins la memòria de cadascun d'ells pot ésser usada per tots els altres.
Tal com s'ha plantejat la transmissió de la informació, es pot endevinar que hi ha un altre problema, no menys important, que tracta de l'ordenació de la gran quantitat d'informació que ha de processar o transmetre un ordinador, de manera que el seu accés sigui ràpid i fiable. Per solucionar aquest problema s'han creat les anomenades **bases de dades.**
Per acabar, volem fer notar que un sistema de transmissió complet com el que descriu la figura (A) és laboriós i molt car de portar a la pràctica; aleshores, allò que es fa, davant de situacions d'aquest tipus, és una **simulació.** L'ordinador serveix perquè, en executar uns programes, en doni la sensació que estem enviant i rebent informació. Hi ha programes que simulen el soroll; d'altres, el codificador, l'encriptador, etc. Així, abans de construir realment el sistema ja s'ha treballat amb ell i se sap quins resultats donarà: se'ls sotmet a situacions límit, en unes condicions extremes, i així es va perfeccionant sense necessitat de construir-lo. Al Departament d'Informàtica de la Universitat Autònoma de Barcelona podem treballar amb la Universitat Paul Sabatier, de Toulouse, sobre un simulador de transmissions d'imatges via satèl·lit, anomenat LOUSTICC. Podem fer simulacions connectats a la xarxa espanyola de teleprocés **IBERPAC** i a la xarxa francesa de teleprocés **TRANSPAC.**
Al llarg d'aquest llibre, us parlarem amb més detall de tots els camps relacionats amb el tractament d'informació que, breument, ara s'acaben de presentar. També intentarem, amb el vostre MICRO i amb l'ajut del BASIC (ja sabem que no es podran enviar missatges), de fer un simulador d'un canal i d'una base de dades.

La detecció d'errors

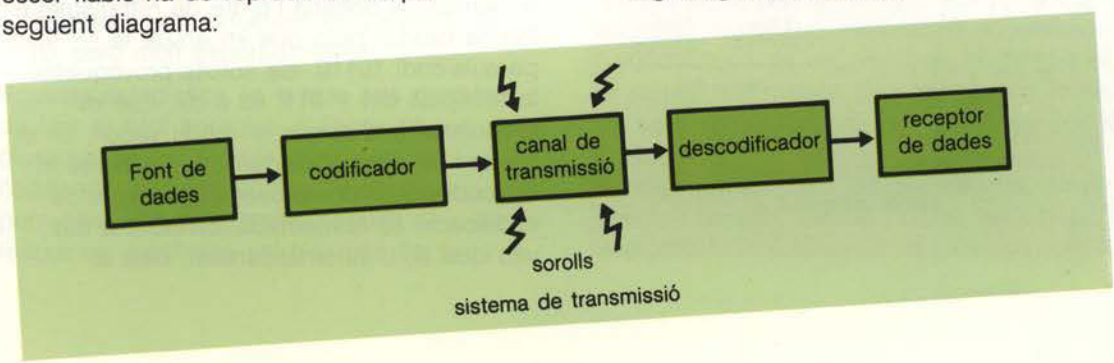
Ja sabem que qualsevol sistema de transmissió d'informació que pretengui ésser fiable ha de representar-se pel següent diagrama:

A causa de les interferències que hi ha sempre en qualsevol canal, el receptor rep amb alguns dels símbols alterats el missatge enviat per la font; és a dir, hi ha canvis de zeros per uns i d'uns per zeros. Aquest canvis són allò que hem anomenat **errors.** Si aconseguim **detectar** quan es produeixen els errors, ja haurem fet una passa important cap a l'objectiu d'aconseguir un sistema fiable, perquè només caldrà donar l'ordre, a la font, que ens torni a enviar, a transmetre, el missatge, i així fins que arribi bé. Aquest procediment és lent, però assegura que el text que arriba al receptor és precisament allò que se li ha volgut enviar.

Per poder detectar els errors, cal enviar, juntament amb la informació, unes quantes xifres (bits) complementàries, anomenades **bits de control.**

El codificador afegeix, segons una certa **regla,** una quantitat de bits als símbols d'informació. Aquesta regla també és coneguda pel descodificador. La primera cosa que fa el descodificador, quan li arriba un missatge, és comprovar si la regla emprada pel codificador és respectada o no. En el cas que no ho sigui, sap amb tota certesa que hi ha errors en el missatge rebut respecte de l'enviat. Notem que:

- si la regla és respectada, pot ésser una transmissió correcta o no, encara que el descodificador la donarà sempre per bona i esperarà un nou missatge.
- si la regla no és respectada, es **segur** que hi ha errors, i el descodificador dóna la transmissió sempre com errònia. En aquest cas, ordena a la font de repetir la transmissió del missatge i, així, fins que la regla sigui respectada.



Suposem, per exemple, que volem enviar aquesta imatge:

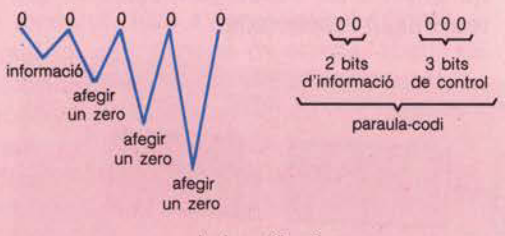
la font la quadricula en nou parts i assigna a cadascun dels seus pixels un valor numèric, segons la següent taula de colors.

missatge a transmetre	**símbol d'informació**
	00
	10
	01
	11

El codificador afegeix a cadascun dels símbols d'informació uns quants bits complementaris, que seran els que hem anomenat **bits de control.**
Una regla de codificació podria ésser la següent:

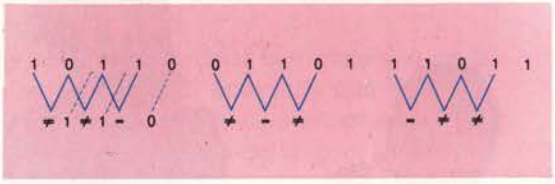
"si els dos símbols anteriors són iguals, afegir un zero; si són diferents, afegir-hi un u; repetir aquest procés tres vegades"

Així, per a la informació 00, el codificador obtindria:



regla de codificació

La seqüència de zeros i uns format pels bits d'informació i els bits de control s'anomena **paraula-codi.** El conjunt d'aquestes forma el **codi.**
Calculem les paraules-codi que correspondrien als altres colors:



així obtenim:

missatge real	informació	bits de control	paraula-codi
	00	000	00000
	10	110	10110
	01	101	01101
	11	011	11011

Si en transmetre el color blau (informació 10) es produís un error al segon bit, el receptor rebria el color negre (informació 11). Com que els missatges no estan codificats, ens equivocaríem, ja que ens pensaríem que ens han enviat el color negre.
En canvi, si es codifiquen, s'enviaria la paraula-codi 10110 que, amb l'error considerat, arribaria al descodificador 11110, el qual aleshores comprovaria si la regla de codificació és respectada. En aquest cas, s'adonaria que no ha estat respectada, ja que, en ésser iguals els dos primers bits, el tercer hauria d'ésser un 0. Per tant, detectaria l'existència d'errors en la transmissió i donaria l'ordre de retransmetre. Podria passar però que en enviar la paraula-codi 10110, els sorolls provoquessin la recepció del 11011: és a dir, que es produïssin tres errors: un en el segon, un el tercer i un altre al cinquè bit. Aleshores el descodificador comprovaria si la regla de codificació és respectada. En aquest cas, veu que sí, i, automàticament, diria al

receptor que ha rebut el missatge "negre", la qual cosa és totalment falsa. Aquí s'ha produit un error que no ha estat detectat. Noteu, però, que és realment estrany que en tant poc tros (5 bits) s'hagin produït tres errors. Això voldria dir que aquest canal equivoca més de la meitat dels bits transmesos: seria un canal realment dolent. Aquí no hi ha codi que valgui; potser seria millor llançar-lo.

Pensem que, a la pràctica, un canal normalment sol esguerrar 1 bit de cada 100.000 bits transmesos.

Si anéssim fent més exemples amb el codi anterior, veuríem que, sempre que es produïssin un o dos errors, el descodificador els detectaria. Ha d'haver-hi, com a mínim, tres errors perquè el descodificador s'equivoqui.

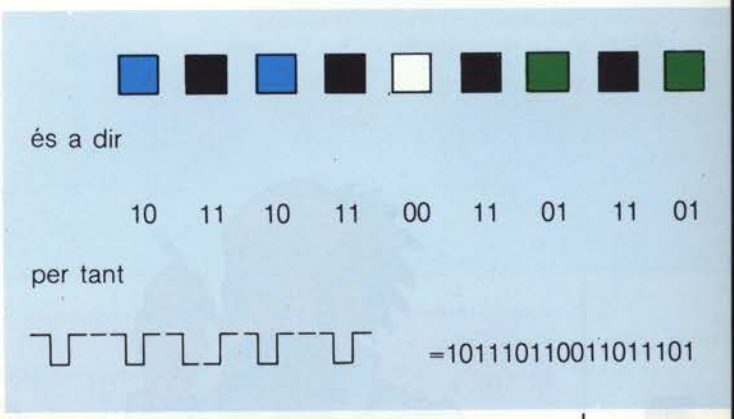
S'ha donat aquest exemple per remarcar que, sigui quina sigui la regla de codificació escollida per obtenir les pararules-codi, no es detectaran mai tots els errors possibles. L'objectiu de la teoria de la codificació és trobar el codi que més errors detecta en una transmissió concreta.

Per triar el codi o, millor dit, la regla de codificació, cal conèixer bé el canal. Això s'aconsegueix fent moltes proves: amb els resultats obtinguts es construeix una estadística que ens dirà aproximadament com es distribueixen els errors produïts pel canal. Per exemple:

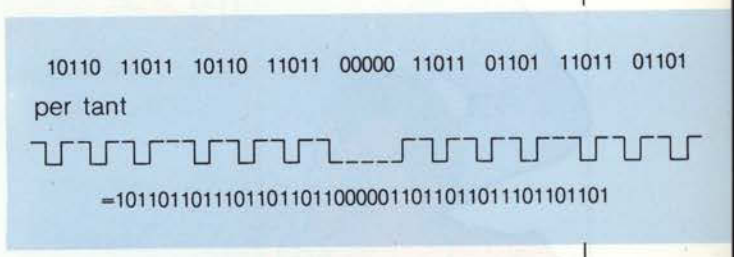
- els errors es produeixen d'un a un, en intervals gairebé constants;
- en començar la transmissió, no hi ha gairebé mai cap error; però, quan el canal s'"escalfa", se'n produeixen molts i molt seguits;
- els errors es produeixen en paquets de 10, però molt esporàdicament.

Aquesta estadística és la que ens permetrà de triar el codi apropiat. A l'hora de dissenyar-lo, cal pensar, però, que el codificador i el descodificador no són persones, sinó programes, circuits o màquines; aleshores cal triar una llei prou senzilla com per poder ésser programada o construïda mitjançant circuiteria. A més a més, el procés ha d'ésser el més ràpid i barat possible.

Observem que, en introduir la codificació, enlentim el procés de transmissió. Abans, sense codis, enviar l'anterior figura suposava transmetre:



mentre que, amb la codificació que hem fet, haurem de transmetre:



A més a més, cal comptar també el temps necessari perquè el codificador calculi els bits de control, el temps que trigarà a verificar la regla de codificació i el possible temps gastat a fer tornar a enviar els missatges "equivocats" (les unitats d'aquest temps, però, són de l'ordre dels nanosegons en molts casos [un nanosegon equival a 0,0000000001 de segon, o bé, cosa que és el mateix, mil milions de nanosegons formen un segon]). És el preu que cal pagar per la fiabilitat.

És allò que es diu: "De pressa i bé, no pot ser!"; o també: "De bo i barat, no n'hi ha al mercat!".

Autocorrecció d'errors

Si volem que un codi, a més de detectar errors, sigui capaç de corregir-ne uns quants, hem de fer servir un procediment lleugerament diferent del de la detecció. És clar que serà una mica més enrevessat, però ens anima a explicar-vos-el perquè creiem que us serà entenedor.

Donada una llei (que haurem triat tenint en compte totes les característiques del canal i de la informació que li volem fer passar), la primera cosa que farem serà construir **el codi,** tal i com ho fèiem abans; és a dir, ajuntarem els corresponents bits de control a la informació. Un cop fet això, ja ens podem oblidar de la llei: no la tornarem a fer servir més. Allò que ens importarà, a partir d'ara, és el codi i no com l'hem construït.

Si seguim amb l'exemple anterior, allò que cal fer és:

informació	control	paraula-codi
00	000	00000
10	110	10110
01	101	01101
11	011	11011

El nostre codi és un conjunt format per quatre elements que anomenarem C:

C = {00000, 10110, 01101, 11011}

El següent pas és calcular allò que en direm **distàncies** del codi. Podem definir la distància entre dues paraules-codi com el nombre de bits que tenen valors diferents. Així, la distància entre 00000 i 10110 és 3, car els bits primer, tercer i quart hi tenen valors diferents, mentre que el segon i el cinquè coincideixen.

Si col·loquem una paraula sota l'altre, n'hi ha prou amb un cop de vista per calcular la distància.

0 0 0 0 0	1 0 1 1 0	1 0 1 1 0
1 0 1 1 0	0 1 1 0 1	1 1 0 1 1
distància 3	distància 4	distància 3

Per abreujar, escriurem això com:

d(0000, 10110)=3, d(10110, 01101)=4 i d(10110, 11011)=3

Podem fer-nos una taula on figurin totes les possibles distàncies del nostre codi. Per a això, tindrem en compte que:

- No cal considerar la distància d'una paraula-codi a ella mateixa. Si de qualsevol paraula-codi en diem A, sempre tindrem que d(A,A)=0 ja que, evidentment, tenen els mateixos valors en els corresponents bits.
- la distància entre una paraula-codi A i una altra, anomenada B és la mateixa que la distància entre B i A. És a dir, d(A,B)=d(B,A). Per tant, a la taula només cal escriure'n una

Per al nostre exemple concret, la taula serà:

0 0 0 0 0 1 0 1 1 0	0 0 0 0 0 0 1 1 0 1	0 0 0 0 0 1 1 0 1 1	1 0 1 1 0 0 1 1 0 1	1 0 1 1 0 1 1 0 1 1	0 1 1 0 1 1 1 0 1 1	A B
3	3	4	4	3	3	d(A,B)

Això vol dir que, donades dues paraules qualsevols del nostre codi, sempre estan a distància o bé 3 o bé 4.

El següent pas és calcular la distància més curta entre dues paraules-codi qualsevols. D'aquest nombre en direm **distància mínima;** i, per abreujar, l'escriurem $\mathbf{d_{min}}$. En el nostre cas concret $d_{min}=3$.

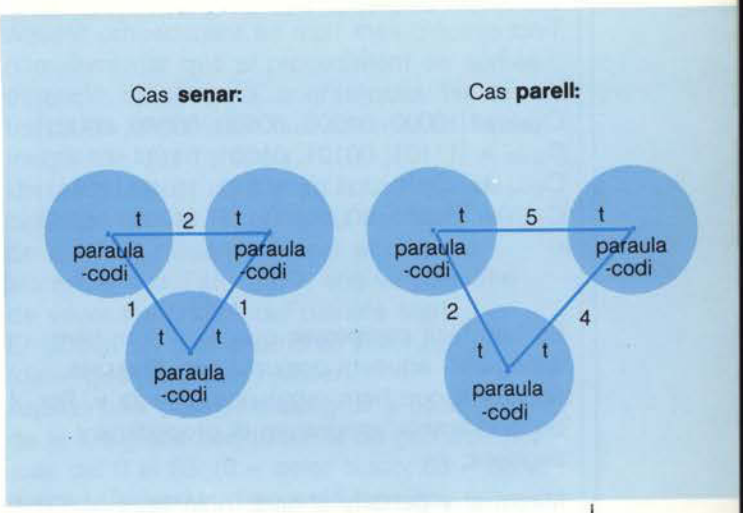
Ara anomenarem **t** al següent nombre:

$$t = \begin{cases} (d_{min}-1)/2, \text{ si } d_{min} \text{ és } \mathbf{senar} \\ (d_{min}-2)/2 \text{ si } d_{min} \text{ és } \mathbf{parell} \end{cases}$$

o, cosa que és el mateix:

$d_{min}= 2*t+1$ per al cas senar, o bé $d_{min}=2*t+2$ per al cas parell.

Això ens permet dir que les "esferes" de radi **t** i centre les paraules-codi són **disjuntes.** Per exemple, podríem tenir:



En el nostre cas concret, farem:

$d_{min}=3$, és senar, per tant:
$t= (d_{min}-1)/2 = (3-1)/2 = 1$

Això ens diu que les "esferes" de radi 1 i centre en les paraules-codi són disjuntes. Aquest nombre **t** ens permet de construir una col·lecció de conjunts disjunts, "esferes", que anomenarem **conjunts de descodificació,** un per a cada paraula-codi. El conjunt de descodificació corresponent a la paraula-codi A l'escriurem C_A, i consisteix en totes les paraules que és puguin obtenir a partir de la paraula-codi A, canviant el contingut de 1, 2, 3 ,....., t bits.

En el nostre exemple, com que la **t** val 1, els conjunts de descodificació consistiran en les paraules que és poden obtenir a partir de

les paraules-codi modificant el contingut de cadascun dels seus bits un a un.
Tindrem:

$C_{00000} = \{10000, 01000, 00100, 00010, 00001\}$
$C_{01101} = \{11101, 00101, 01001, 01111, 01100\}$
$C_{10110} = \{00110, 11110, 10010, 10100, 10111\}$
$C_{11011} = \{01011, 10011, 11111, 11001, 11010\}$

Es ben fàcil comprovar que, tal i com hem assegurat, aquests conjunts són **disjunts.** Suposem que hem rebut una paraula **v.** Per descodificar-la, emprarem el procediment següent:

Mirem si **v** pertany al codi. Això és: $v \in C$?, una pregunta que admet dues possibles respostes: sí o no.
1) Si la resposta és SÍ:

Direm que **no hi ha** hagut **errors** i que allò que ens volien enviar és precisament **v.** Podria ser, però, que hi haguessin hagut molts errors, d_{min} errors o més; nosaltres, però, **no els detectariem**

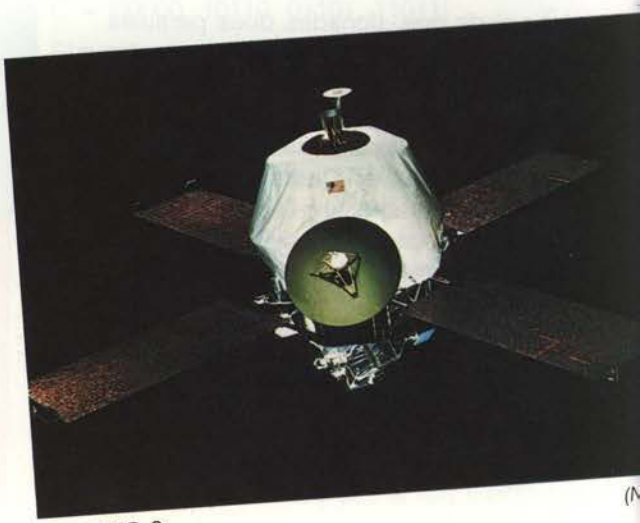
2) Si la resposta és NO:

Assegurem que **hi ha errors.**
Ara, mirem si **v** pertany a algun dels conjunts de descodificació.
Això és: hi ha alguna paraula-codi A tal que $v \in C_A$? Una pregunta que admet dues possibles respostes: sí o no.

a) Si la resposta es SÍ:
Assegurarem que hi ha t o menys errors a la paraula que hem rebut. Com que els conjunts de descodificació són disjunts, descodificarem **v** per **A** en el cas que $\mathbf{v} \in \mathbf{C_A}$. Hem rebut **v,** però ens havien enviat **A.** Hem **corregit** els errors.

b) Si la resposta és NO:
Sabem que **v** no pertany a cap dels conjunts de descodificació: per tant, hi ha més de **t** errors; els **detectem,** pero **no els podem corregir.** Donarem ordre a la font per tal que torni a transmetre el missatge.

Vista del planeta Mart des del satèl·lit Viking I



MARINER 9

(Salmer)

Aquest esquema ens mostra que, sempre que el canal ens produeixi **t** o menys errors, els corregirem. Això fa que el nombre **t** és conegui amb el nom de **capacitat correctora** del codi.

També ens mostra que, sempre que el canal ens produeixi d_{min}-1 o menys errors, els detectarem. Això fa que el nombre d_{min}-1 sigui conegut amb el nom de **capacitat detectora** del codi.

El nostre codi és, doncs, **1-corrector, 2-detector,** ja que la seva capacitat correctora val 1 i la seva capacitat detectora val 2.

Noteu que aquest és un procediment "fàcil" de programar, perquè es basa en preguntes contestables amb SÍ o NO. (Són IF... THEN). Això fa que l'ordinador tingui capacitat per donar ordres a l'home (a la font), com ara la de tornar a enviar el missatge. Ens fa pensar que l'ordinador (el programa) és un sistema **intel·ligent.**

Aquest procediment és molt més difícil d'implementar que el procediment de només detecció; noteu, però, la gran quantitat de temps de transmissió que estalvia. És fiable i menys car que l'anterior.

Un mètode com aquest és aquell que els científics del Jet Propulsion Laboratory (JPL) de la NASA dissenyaren per al projecte Mariner-9 que, l'any 1972, ens va permetre de veure fotografies del planeta Mart. Empraren un codi anomenat RM(1,5) (dissenyat per Reed i Muller).

Aquest codi permet d'assignar a cada pixel de la fotografia una tonalitat de gris que pot anar del 0 al 63. (0 = color blanc; 63 = color negre.) Cadascun d'aquests tons és transmés usant un nombre binari (compost per només zeros i uns) de 32 xifres. La capacitat detectora d'aquest codi és 15; és a dir, pot detectar errors a gairebé la meitat dels bits que transmet, i corregir-ne una quarta part.

A l'actualitat s'usen codis correctors d'errors en qualsevol sistema de transmissió d'informació: els projectes espacials, la televisió via satèl·lit, en els telèfons a llarga distància, per emmagatzemar informació a la memòria d'un ordinador, als bancs de dades,...

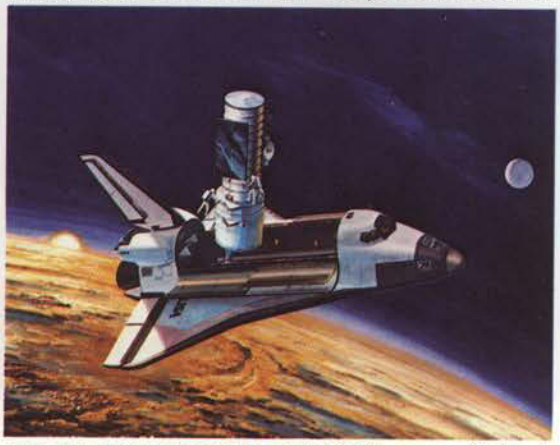
A cada país hi ha grups de científics que treballen en aquest tema. Encara que aquí l'hàgim presentat de manera senzilla, en realitat, els "bons codis" són basats en tècniques matemàtiques molt avançades.

Podríem dir que aquest és un tema que té necessitat: dels **enginyers,** per tal que dissenyin bones xarxes i circuits; dels **informàtics,** per a les parts del codificador i descodificador que seran programes i per a simular el sistema un cop pensat; i dels **matemàtics,** per a inventar codis on les seves paraules siguin nombres de poques xifres, per tal que la transmissió sigui **ràpida,** alhora que la d_{min} (i per tant la capacitat de detectar i corregir errors) sigui gran, per tal que la transmissió sigui **fiable.**

Una aplicació: imatges via satèl·lit

Tots sabem que la lluna és el satèl·lit de la Terra; és a dir, un cos que es mou voltant el nostre planeta. Dins el sistema solar, Mart, Júpiter, Urà i Neptú també tenen satèl·lits.
A finals dels anys cinquanta es pensà en la creació d'uns enginys per tal de col·locar-los en òrbita a voltant de la Terra. Aquests aparells es mouen tal i com ho fa des d'una época remota la Lluna; és per això que reben el nom de **SATÈL·LITS ARTIFICIALS.** Aquests són col·locats en òrbita per uns cohets, però no necessiten cap sistema de propulsió (motor) per mantenir-se tot fent voltes a la Terra.
Un d'aquests satèl·lits és el **LANSAT 5,** que té una òrbita periòdica tal que cada 16 dies passa sobre el mateix punt de la Terra, i té com a missió (entre d'altres) la de captar imatges digitals (fotografies) del total de la superfície terrestre. Aquestes, un cop captades, són transmeses a les estacions receptores, que en fan un primer processat per tal de corregir-ne els errors, tant de captació com de transmissió. Després, emmagatzemen aquestes imatges i les comercialitzen en cintes magnètiques.
La captació de la imatge pel satèl·lit és un xic diferent d'allò que havia estat explicat, en ocasió de la digitalització, en el segon llibre. La captació és feta per uns sensors que efectuen contínues escombrades de la zona d'influència en el moment del pas de satèl·lit.

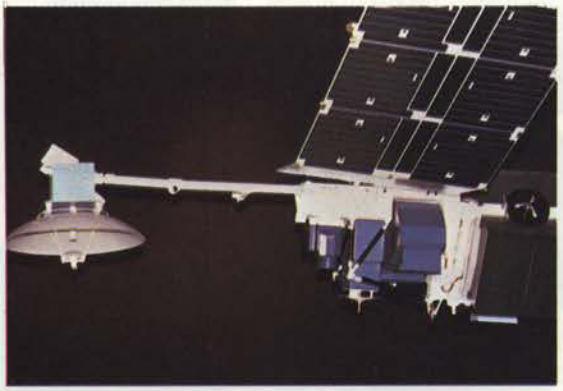
Llançament del Challenger STS-8
(Salmer)



Alliberament del satèl·lit PAYLOARD des de la nau STS-8. *(Salmer)*

La costa catalana captada pel satèl·lit LANSAT 5

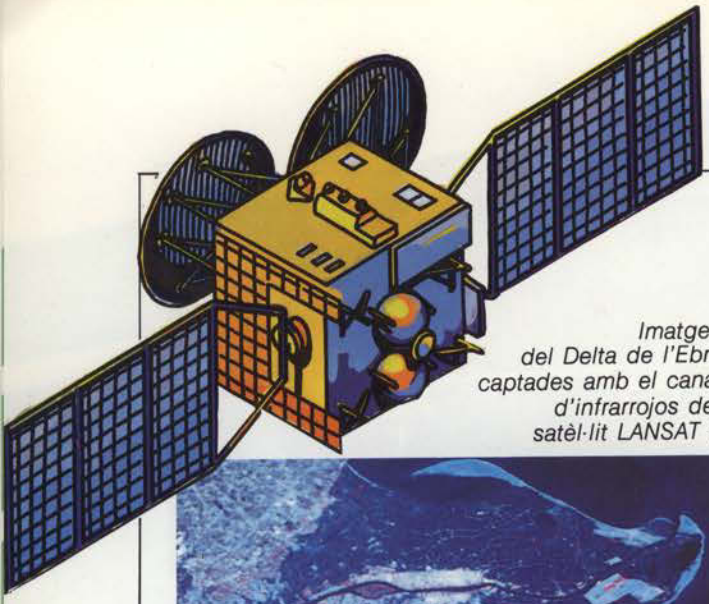
Aquest sensors són el TM *(temating mapper)* i el MSS *(multi-spectral scanner)* i obtenen les imatges digitals mitjançant la utilització de senyals a diferents longituds d'ona.
La captació de la imatge pot ésser feta mitjançant set canals diferents; entre ells, els canals corresponents al roig, al verd i al blau, i d'altres com els infrarrojos. Cada canal pot captar, i després enviar, fins a 256 nivells de gris per a cada pixel. En el cas del sensor TM, un pixel correspon a una zona de 30x30 metres, mentre que, per al MSS, representa una zona de 80x80 metres.
Els satèl·lits són capaços, així, d'enviar a la Terra fotografies de color del nostre planeta preses a gran distància, però això no seria pas important si no fos perquè, a més, els quatre canals restants, com ara els infrarrojos, ens permeten de fer exploracions especials.
En el Centre de Càlcul de la Universitat Politècnica de Barcelona hi ha un grup de recerca dedicat al tema del processament d'imatges digitals transmeses pel satèl·lit **LANSAT 5.** En particular, ens han explicat el seu treball de classificació dels usos del sòl en el Delta de l'Ebre.



Satèl·lit LANSAT 5 *(Nasa)*

COMTAL i Trackball (U.P.B.)

La imatge rebuda és passada a l'ordinador (en aquest cas un VAX/780) en forma d'una taula de tres dimensions. Les dues primeres corresponen a les coordenades del punt observat i el seu nivell de gris, mentre que la tercera diu amb quin canal ha estat obtinguda.
Per visualtzar una imatge es disposa d'un equip COMTAL Vision One/20 i d'un "Trackball" que no és altre cosa que un perifèric que permet mobilitzar un cursor per a tota la pantalla, així com desplaçar un *zoom* a través de tota la imatge. Es consideren 512x512 pixels, cosa que significa 262.144 bytes per banda de canal utilitzada; és a dir, si es vol visualitzar una imatge en la que s'han utilitzat els tres canals stàndard del color (blau, verd i vermell), és necessari tractar 3x262.144 = 786.432 bytes; és a dir, 768 KBytes.



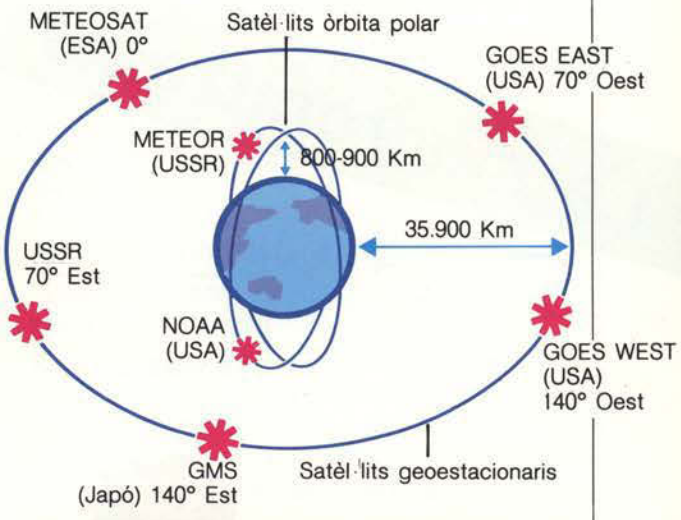
Imatges del Delta de l'Ebre captades amb el canal d'infrarrojos del satèl·lit LANSAT 5

Quan aquesta imatge ja és a l'ordinador, comença el procés de tractament.
En el cas del Delta de l'Ebre, es tracta de fer una classificació de l'ús del sòl;i es realitzen, entre d'altres, les següents manipulacions principals:

- **Radiometria de la imatge:** arreglar-la per la seva utilització, segons l'objectiu de l'estudi.
- **Correcció geomètrica:** corregir distorsions ocasionades per moviments del satèl·lit, per interferències atmosfèriques i, sobretot, per l'efecte del moviment de rotació de la Terra, que provoca distorsions dels punts visionats segons sigui l'angle d'incidència del raig utilitzat pel sensor.
- **Classificació de la imatge:** superposant la imatge obtinguda a la cartografia existent, o explorant una mostra del propi terreny.
- **Cercar signatures espectrals:** és a dir, cercar les característiques específiques de la imatge, tenint en compte amb quin canal ha estat captada, amb quina longitud d'ona.
- **Procés interactiu:** capacitat de tractar cada punt, associat a cadascuna de les signatures espectrals.

Es podria pensar que, per a fer una imatge com la del Delta de l'Ebre, no cal un satèl·lit: n'hi ha prou amb una avioneta. Si ho féssim així, però, hauríem de comprar tot l'equip de scanner, perquè per estudiar el sòl necessitem més canals que els de color; i, per altra banda, el moviment de l'avioneta provocaria errors damunt la imatge difícils de corregir. Es per això que hi ha empreses grans que es dediquen a l'explotació d'un satèl·lit; per a nosaltres, ens és del tot rendible, i ensems fiable, comprar-los la imatge.
L'estudi del Delta de l'Ebre que fan en aquest centre de càlcul permet de veure quina és la superfície real del Delta, com ha evolucionat en els darrers anys, quin és el tipus de sòl: aiguamolls, zones arrosseres,

Satèl·lit TELSTAR I *(Nasa)*



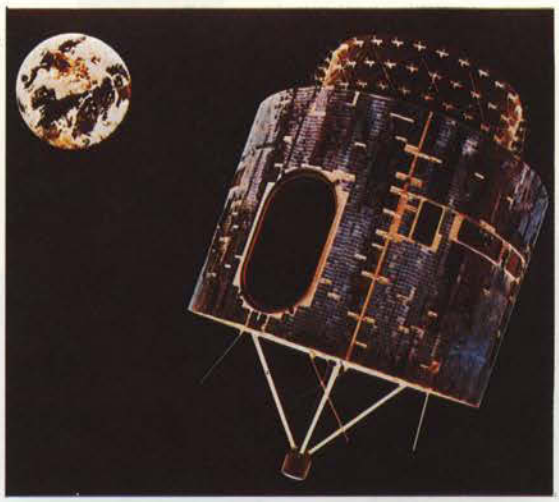
conreus,... Així, l'home pot explotar millor el sòl de què disposa, sense crear problemes ecològics irreversibles.

Aquest grup de recerca també col·labora en la simulació d'imatges transmeses pel satèl·lit SPOT, on cada pixel correspon a una zona de 20x20 metres. Aquest satèl·lit encara no és en òrbita, però abans de fabricar-lo ja se sap quin tipus de fotografies emetrà i si el projecte serà realment rendible.

A més de l'aplicació que us hem presentat, els satèl·lits actualment s'empren per:

- Prediccions meteorològiques (les imatges que podem veure a la televisió, són del satèl·lit METEOSAT).
- Comunicacions telefòniques a llarga distància, perquè és molt més senzill servir-se del satèl·lit que no pas instal·lar un cable per sota els oceans. Un d'aquest satèl·lits és el TELSTAR.
- Aplicacions militars (com per a saber l'emplaçament d'una determinada flota naval o d'uns míssils).
- Transmetre imatges de televisió, perqué ens estalviem de muntar un repetidor sobre cada muntanya.
- .....

Als Estats Units de Nord-amèrica poden saber com evoluciona la collita de blat de la Unió Soviètica (i viceversa).

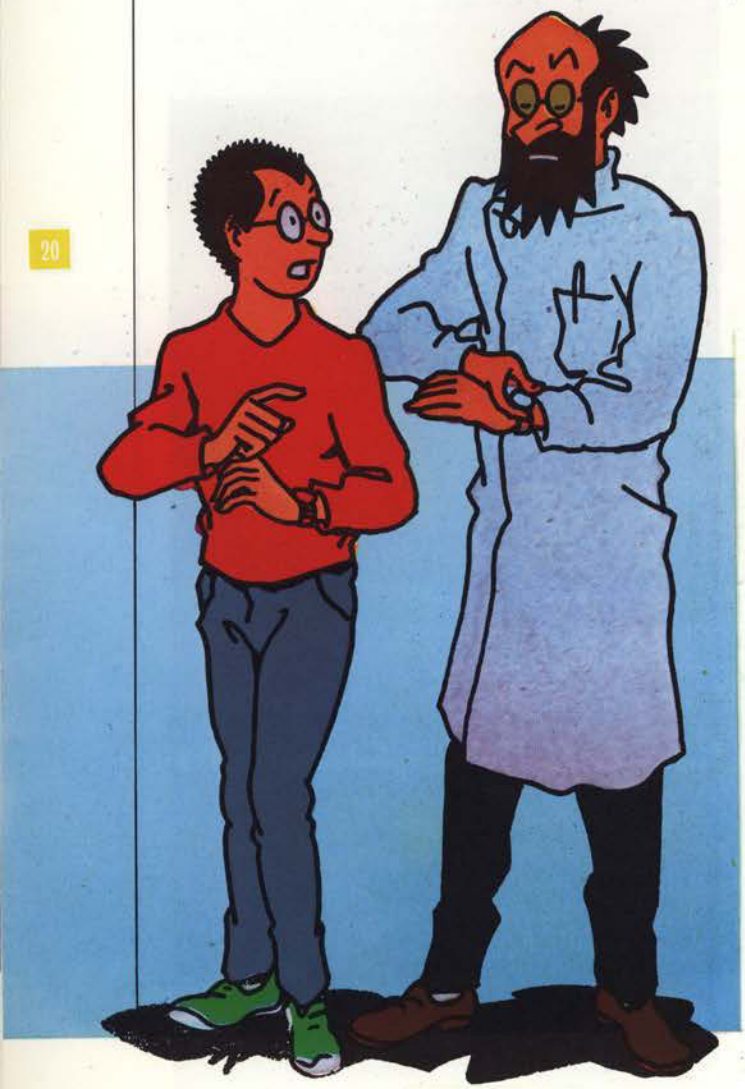


Satèl·lit METEOSAT voltant la Terra *(Nasa)*

Actualment, el països petits s'organitzen per tal de poder col·locar un satèl·lit propi en òrbita. Cal esmentar, per exemple, el projecte Europeu ARIADNA, en el qual també participa l'estat espanyol.

El món dels satèl·lits i les comunicacions espacials és realment sorprenent: cada dia sorgeixen noves aplicacions. Ara no ens acontentem amb tenir satèl·lits que voltin la Terra, sinó que enviem naus a fotografiar planetes ben llunyans. Aviat aconseguirem, fins i tot, de sortir del sistema solar per a conèixer com és realment la nostra galàxia.

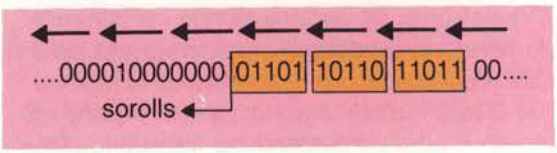
La sincronització



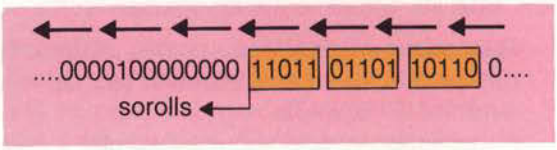
Sabem que, transmetre informació, consisteix a enviar sèries de zeros i uns a través d'un canal. Quan la font no emet cap símbol, a causa dels sorolls que sempre hi ha en tot canal, també arriben al receptor sèries de zeros i uns; aquesta vegada, menys uns que zeros, sèries gairebé nul·les però no totes zero. És important, doncs, saber distingir on acaba el soroll i on comença veritablement la informació.

Suposem que transmetem amb el codi $C = \{00000, 01101, 10110, 11011\}$, i que, al receptor, hi arriba la sèrie següent.

El receptor es pot pensar que el soroll acaba al bit 12, i que, per tant, allò que ens volien dir és:



Però també podria pensar que, on acaba el soroll, és al bit 13, i que la informació és, per tant:

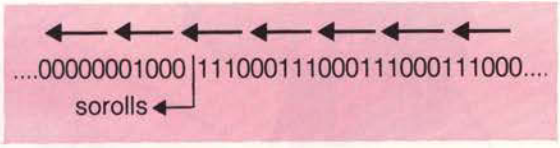


Notem que, només que ens equivoquem en un bit a l'hora de distingir l'inici de la informació, per més bé que hàgim triat el codi, rebrem tota la informació malament.

Ja veiem que no n'hi ha prou, per exemple, amb "una telefonada" (fer avisar al receptor que li volem enviar quelcom). S'ha d'emprar un procés una mica més sofisticat. Cal pensar-hi, per tal de trobar un mètode exacte.

És de vital importància, per a la transmissió d'informació, que la font i el receptor estiguin **SINCRONITZATS.** Per aconseguir-ho, es fa el següent procés:

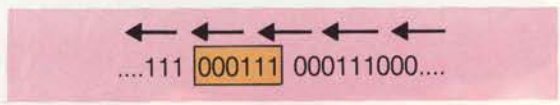
La font disposa d'un nombre binari (format per zeros i uns) d'una quantitat determinada de xifres, que és conegut també pel receptor. És anomenat **NOM DE LA FONT.**
La font, abans d'emetre les informacions, va enviant succesivament el seu nom. Així, abans de rebre la informació, el receptor rep una sèrie de zeros i uns que consisteix en el nom de la font moltes vegades seguides (és clar, amb algun error).
Per exemple, si el nom de la font és: 111000 el receptor rebrà:



Anomenarem **DECALAR** el procediment que consisteix a portar l'última xifra d'un nombre al davant, tants cops com vulguem. Si un nombre té n xifres, en decalar s'obtenen en nombres diferents.
Anem a calcular els decalats de 111000, el nom de la nostra font:

valor inicial	número de vegades que decalem	valor obtinguts
111000	0	111000
	1	011100
	2	001110
	3	000111
	4	100011
	5	110001
	6	valor inicial

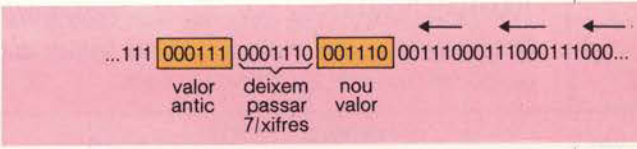
Com que el nostre nombre té 6 xifres, hem obtingut 6 valors decalats.
Si el nom de la font té n xifres (en el nostre cas n=6) el receptor "caça", a l'atzar, n xifres consecutives de la sèrie que li arriba, i les compara amb el nom de la font.
En el nostre exemple, podria ésser:



Observem que allò que troba el receptor és, precisament, un valor decalat del nom de la font.
Comparar aquest nombre amb el nom de la font serà cercar el nombre de bits que tenen valors diferents; és a dir, calcular-ne la distància.
A l'exemple:

d(000111, 111000) = 6
"valor caçat" — nom de la font

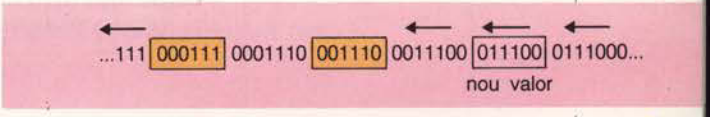
Si la **distancia és zero,** vol dir que hem trobat el nom de la font: direm que ens hi hem SINCRONITZAT. Ara, el receptor continuarà llegint els bits que li arribin de 6 en 6, i els anirà comparant amb el nom de la font.
Si la **distància és diferent de zero,** cal provar-ho amb un altre nombre. El receptor deixarà passar exactament n+1 instants de temps (n+1 bits) i "caçarà" les n xifres següents.
En el nostre cas



Ara, fem com abans:

d(001110, 111000) = 4

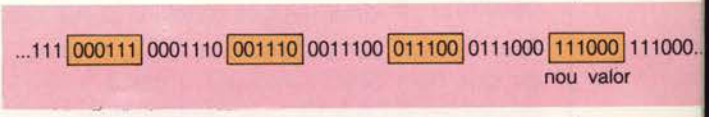
Com que és diferent de zero, deixem passar 7 xifres més.



i tornem a calcular la distància

d(011100, 111000) = 2

diferent de zero, deixem passar 7 xifres



d(111000, 111000) = 0. Ens hem
SINCRONITZAT.

Aquest procés ens pot suposar, com a màxim, n passos: tants com xifres tingui el nom de la font.
En aquest moment, el receptor ordena a la font que, quan vulgui, ja pot enviar la informació. Mentrestant, el receptor va llegint de n en n els bits que li arriben i va trobant les distàncies d'aquests amb el nom de la font. Quan aquesta distància sigui diferent de zero, el receptor sabrà que allí, en aquell bit precís, comença la informació.
A l'hora de la veritat, i a causa dels sorolls del canal, gairebé mai no s'aconsegueix que la distància sigui exactament zero. Allò que es fa és cercar per nom de la font uns nombres tals que la distància entre ells i tots els seus decalats sigui gran.
Quan s'obtingui un valor petit per a la distància, direm que ens hem SINCRONITZAT, ja que haurem sabut trobar el nom de la font, encara que aquest hagi arribat amb algun error.
Un valor bastant usat és:

100010011010111

El quadre que ens donarà tots els valors que es poden obtenir tot decalant serà:

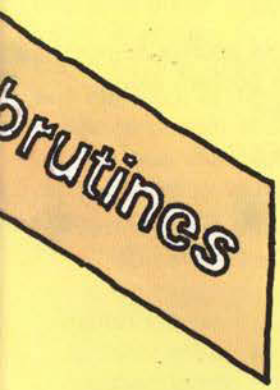
valor inicial	número de vegades que decalem	valor obtinguts
100010011010111	0	100010011010111
	1	110001001101011
	2	111000100110101
	3	111100010001100
	.	.
	.	.

Si el continuem fins a obtenir el nombre corresponent a decalar 14 cops, veurem que:

d(100010011010111, qualsevol valor decalat) = 8

Això fa que, per a distàncies 0,1,2, podem dir que hem SINCRONITZAT, i per a distàncies 6,7,8,9,10, que cal continuar.
El problema de la sincronització es redueix, doncs, a saber trobar nombres de poques xifres i tals que la distància entre ells i tots els seus decalats sigui la més gran possible. Com més curt sigui el nombre, abans aconseguirem la sincronització; i com més grans siguin les distàncies, més fiable serà el nostre procediment.

Subrutines



Una **subrutina** és un conjunt de sentències que s'ha de fer servir en diversos llocs d'un mateix programa. Per tal de no escriure-les cada cop que toqui, el BASIC ens permet de fer el següent procés:

- Escriure aquest grup de sentències (subrutina) a la fi del programa; per exemple, a la línia 1000.
- Un cop escrit, posar al final **RETURN.**
- Allà on toqui executar-les, posar-hi **GOSUB** 1000.

Quan el micro trobi la sentència:

GOSUB n.º línia,

anirà a executar les instruccions que hi hagi a partir de **n.º línia,** fins que es trobi amb un **RETURN;** aleshores tornarà a la sentència qui hi hagi sota el **GOSUB.** Això vol dir que el micro "recorda" el lloc des del qual hem anat a la subrutina. Per tant, quan s'executi el programa farà la mateixa cosa que si haguéssim escrit la subrutina a cada lloc on ens cal.

Una subrutina només es pot executar si hi hem arribat amb un **GOSUB;** això vol dir que abans on comenci la línia ha d'haver-hi un **GOTO** o bé un **STOP,** o un **END.**

Per entendre mes bé tot el que hem dit, vegem el següent exemple.

Farem un programa que ens calculi la mitjana aritmètica dels nombres sencers que hi ha entre 20 i 100, entre 10 i 200, i, després, entre dos nombres que entrarem per teclat. Si aquests nombres els guardem a les variables A i B, i el resultat a C, les sentències que calculen la mitjana seran:

```
LET C = 0
FOR Y = A TO B
LET C = C + Y
NEXT Y
LET M = B - A + 1
LET C = C / M
```

El programa començarà així:

```
LET A = 20
LET B = 100
```

i ara cal posar-hi el grup de sentències que calculen la mitjana; seguim amb

```
LET A = 10
LET B = 200
```

i, un altre cop, les sentències per a calcular la mitjana. Ara hi posarem:

```
INPUT A
INPUT B
```

i, un altre cop, el grup de sentencies que calculen la mitjana. Aquest és un cas clar per emprar una subrutina, perquè hem d'escriure tres vegades la mateixa cosa. La nostra subrutina, si suposem que comença a la línia 500, será:

```
500 REM CALCUL DE LA SUMA TOTAL
505 LET C = 0
510 FOR Y = A TO B
520 LET C = C + Y
530 NEXT Y
535 REM CALCUL DE LA QUANTITAT
536 REM TOTAL DE NOMBRES
540 LET M = B - A + 1
545 REM CALCUL DE LA MITJANA
550 LET C = C / M
560 RETURN
```

el programa quedarà així de senzill:

```
10 LET A = 20: LET B = 100
20 GOSUB 500
30 PRINT "LA MITJANA ES:";C
40 LET A = 10: LET B = 200
45 PAUSE 150
50 GOSUB 500
60 PRINT "LA MITJANA ES:";C
65 PAUSE 150
70 INPUT "PRIMER NOMBRE?";A
80 IMPUT "SEGON NOMBRE?";B
90 GOSUB 500
100 PRINT "LA MITJANA ES:";C
500 REM CALCUL DE LA SUMA TOTAL
...
560 RETURN
```

Fixem-nos, però, que, després d'executar-se la sentència de la línia 100, el micro buscarà la següent que ha d'executar; trobarà que és la 500 i anirà fent fins a la 560, on hi ha el **RETURN.** Com que aquest cop s'ha executat la subrutina sense haver-hi entrat per un **GOSUB,** el micro no sabrà on ha de tornar, i donarà error. Per tal d'evitar-ho, posarem:

490 STOP

o bé, en el Dragon: 490 END.

Una altra cosa que es pot fer és estalviar-nos d'escriure tres vegades:

PRINT "LA MITJANA ES:";C

Com que aquesta sentència apareix sempre darrera de GOSUB 500, la podrem afegir dins la subrutina, tot posant:

555 PRINT "LA MITJANA ES:";C"

Després, esborrarem les línies 30, 60 i 100. Per fer-ho, només cal escriure aquests tres nombres i pitjar <ENTER> després de cadascun d'ells.
El següent programa està pensat per donar una explicació global de l'ús de les subrutines. Si l'observem bé comprovarem que es pot anar a una subrutina des d'una altra.

```
5 CLS
10 PRINT "ESTIC DINS EL PROGRAMA
   PRINCIPAL. PITJEU <ENTER>"
15 IF INKEY$="" THEN GOTO 15
20 PRINT "ARA ANIRE A LA SUBRUTINA
   DE LA LINIA 200. PITJEU <ENTER>"
30 IF INKEY$="" THEN GOTO 30
40 GOSUB 200
50 PRINT "HE TORNAT AL PROGRAMA
   PRINCIPAL.PITJEU <ENTER>"
60 IF INKEY$="" THEN GOTO 60
70 PRINT "ARA ANIRE A LA SUBRUTINA
   DE LA LINIA 400. PITJEU <ENTER>"
80 IF INKEY$="" THEN GOTO 80
90 GOSUB 400
100 PRINT "HE TORNAT AL PROGRAMA
    PRINCIPAL I ACABO.PITJEU <ENTER>"
110 STOP
200 PRINT "ESTIC A LA SUBRUTINA DE
    LA LINIA 200. PITJEU <ENTER>"
210 IF INKEY$="" THEN GOTO 210
220 PRINT "ARA ANIRE A LA SUBRUTINA
    DE LA LINIA 400.PITJEU <ENTER>"
230 IF INKEY$="" THEN GOTO 230
240 GOSUB 400
250 PRINT "HE TORNAT A LA SUBRUTINA
    DE LA LINIA 200.PITJEU <ENTER>"
255 IF INKEY$="" THEN GOTO 255
260 RETURN
400 PRINT "ESTIC A LA SUBRUTINA DE
    LA LINIA 400. PITJEU <ENTER>"
410 IF INKEY$="" THEN GOTO 410
420 RETURN
```

La instrucció "ON"

Normalment, els programes estan dividits en diverses parts. Per exemple, podem dir d'un programa que, donats dos nombres, en calcula la suma, la resta, la multiplicació i la divisió, que està dividit en quatre parts.
Allò que passa, normalment, és que no ens interessa que, cada cop que fem **RUN,** s'executin totes aquestes parts, sinó que ens interessa poder triar quina és la part que volem executar.
De la mateixa manera que, en un restaurant, ens ofereixen el menú per tal que triem, nosaltres podem escriure un programa de manera que ens expliqui tot el que fa i, després, ens pregunti quina part en volem. D'això en direm "fer un menú".
Per exemple:

```
5 CLS
10 PRINT "OPCIO 1: SUMAR DOS
NOMBRES"
20 PRINT "OPCIO 2: RESTAR DOS
NOMBRES"
30 PRINT "OPCIO 3: MULTIPLICAR DOS
NOMBRES"
40 PRINT "OPCIO 4: DIVIDIR DOS
NOMBRES"
45 PRINT "OPCIO 5: ACABAR"
50 INPUT "TRIA OPCIO:";OP 60 CLS
70 INPUT "PRIMER NOMBRE?:";A
80 INPUT "SEGON NOMBRE?:"B
```

Després del menú, vénen normalment un seguit de preguntes amb les quals el micro sabrà quin conjunt d'instruccions ha d'executar, segons l'opció que hem triat:

```
 90 IF OP = 1 THEN GOTO 150
100 IF OP = 2 THEN GOTO 180
110 IF OP = 3 THEN GOTO 210
120 IF OP = 4 THEN GOTO 240
130 IF OP = 5 THEN GOTO 270
140 GOTO 5
150 LET C = A + B
160 PRINT "A+B=";C
170 GOTO 5
180 LET C = A - B
190 PRINT "A-B="; C
200 GOTO 5
210 LET C = A*B
220 PRINT "A*B =";C
230 GOTO 5
240 LET C = A / B
250 PRINT "A/B=";C
260 GOTO 5
270 STOP
```

Per tal de no escriure aquests cinc **IF** seguits, tots ells depenents del contingut de la mateixa variable, podem emprar la sentència:

ON expressió numèrica **GOTO** llista dels nombres de línia.

Aquesta sentència vol dir que, un cop avaluada l'expressió numèrica, si el valor que pren es 1, s'executarà un **GOTO** a la línia que correspon amb el primer nombre de la llista; si el valor és 2, s'executarà un **GOTO** al segon nombre, i així succesivament.
Per tant, en el nostre programa substituirem les sentències 90, 100, 110, 120 i 130 per:

90 **ON** OP **GOTO** 150, 180, 210, 240, 270

En aquest cas, OP ha de prendre valors entre 1 i 5: si el valor d'OP fos negatiu, segurament obtindríem un missatge d'error; i si fos més gran de 5 o bé si fos zero, el micro ignoraria la sentència i passaria a executar la següent.

La instrucció **ON** també es pot fer servir amb **GOSUB.** Si, per exemple, tenim un plec de programes on cadascun d'ells és una cançó, els podem ajuntar dins un programa més gran, tot posant al començament un "menú" de les cançons que s'hi poden escoltar. Aleshores, allò que podem fer és posar el primer programa-cançó a partir de la línia 1000, el segon a partir de la 2000, i, així, anar fent amb tots, i posar **RETURN** al final de cadascun d'ells. Si tinguéssim 10 cançons, el tros de programa que ens oferiria el menú i ens preguntaria quina cançó volem sentir, seria:

```
5 CLS
10 PRINT "nom de la 1.ª cançó: 1"
20 PRINT "nom de la 2.ª cançó: 2"


100 PRINT "nom de la 10.ª cançó: 10"
120 INPUT "NUMERO DE LA CANÇÓ?:";C
130 ON C GOSUB 1000, 2000, 3000, 4000
    5000, 6000, 7000, 8000, 9000, 10000
140 STOP
150 GOTO 5
```

Les funcions "INKEY$" i "SCREEN$"

Fins ara, ja hem vist alguns exemples de funció (CHR$,RND,...). La **INKEY$** és de les més útils que ens proporciona el BASIC. La seva missió és mirar si s'està pitjant alguna tecla i, en el cas que sigui així, guarda el caràcter que aquesta representa. El seu gran avantatge és que no interromp l'execució del programa, com feia la instrucció INPUT, ni és necessari pitjar ENTER després de les dades.
Si, en el moment de prémer la tecla, l'ordinador no es troba avaluant una funció INKEY$, será com si no haguéssim fet res. En el "Commodore", hi ha una instrucció molt similar, la **GET;** però necessita una variable alfanumérica on col·locar allò que s'ha pitjat en el teclat.

Per exemple:

GET A$

La instrucció GET no opera de la mateixa forma que la funció INKEY$. En un "Commodore", quan premem un seguit de tecles mentre un programa s'està executant, els caràcters que aquestes representin s'emmagatzemen dins una espècie de memòria temporal anomenada "*buffer* d'entrada". Si, en un moment donat, l'ordinador es troba amb una instrucció GET, allò que fa és portar el primer caràcter que hi ha dins aquest *buffer* a la variable alfanumèrica especificada en la instrucció. Si l'ordinador troba un altre GET, allò que farà llavors serà posar el segon caràcter dins la variable; i així, succesivament.
La funció INKEY$ (o la seva equivalent GET) és molt útil en els programes de jocs, on el jugador controla els moviments dels objectes de la pantalla mitjançant el teclat. Per exemple, en un programa per jugar al tennis no podem emprar INPUT perquè el jugador entri les ordres per a moure la "raqueta", ja que la pilota ha de continuar corrent per la pantalla, i INPUT, en aturar totalment l'execució en espera d'un ENTER, no permet que això passi.
Si ens interessa, per exemple, fer quelcom si s'ha pitjat la tecla P, per controlar-lo només hem de fer

IF INKEY$ = "P" THEN....

i, en el "Commodore",

GET A$
IF A$ = "P" THEN....

El següent programa és una ampliació d'aquell que va sortir al llibre anterior, on hi havia una pilota rebotant dins un rectangle. En aquest, hi ha una "raqueta" dibuixada amb quatre **X:** controlarem la posició d'aquesta raqueta mitjançant una taula, A(4), que contindrà els nombres de línia que ocupa la raqueta. Amb la tecla **o** la farem pujar, i amb la **p,** la farem baixar. S'ha de controlar el moment en què la raqueta arribi a dalt de tot o a baix de tot.

Per saber si la pilota ha tocat o no la raqueta, emprarem una nova funció

SCREEN$(X, Y)

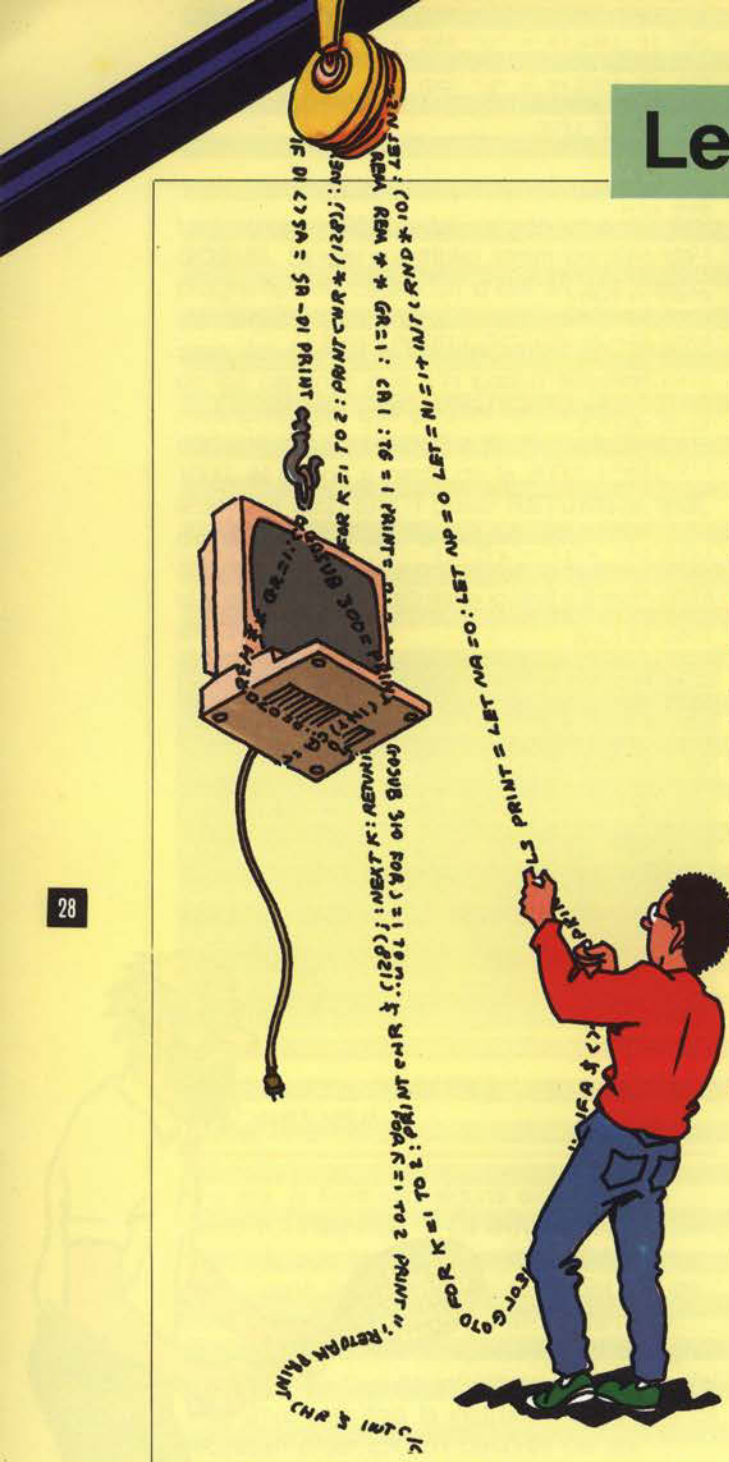
que obté el caràcter situat a la línia X, columna Y de pantalla. Així, per controlar-ho, preguntarem cada cop que s'hagi de moure la pilota:

IF SCREEN$(X, Y) = "X" THEN....

El programa serà doncs:

```
1 REM ************************
2 REM **** PROGRAMA FRONTO ****
3 REM *****Versio Spectrum*****
5 PAPER 6:BORDER 7:CLS
8 REM
10 REM CARACTER DE LA PILOTA
12 REM
20 POKE USR "a",24
30 POKE USR "a"+1,126
40 POKE USR "a"+2,126
50 POKE USR "a"+3,255
60 POKE USR "a"+4,255
70 POKE USR "a"+5,126
80 POKE USR "a"+6,126
90 POKE USR "a"+7,24
98 REM
100 REM DIBUIX RECTANGLE
102 REM
105 INK 1
110 FOR i = 0 TO 21
120 PRINT AT i,31;CHR$(143)
130 PRINT AT i,0;CHR$(143)
140 NEXT i
150 FOR j = 0 TO 31
160 PRINT AT 0,j;CHR$(143)
170 PRINT AT 21,j;CHR$(143)
180 NEXT j
185 DIM a(4)
188 REM
190 REM POSICIO INICIAL RAQUETA
192 REM
200 FOR i = 1 TO 4
210 LET a(i) = 8+i:NEXT i
218 REM
220 REM POSICIO INICIAL PILOTA
222 REM
230 LET dx = 1:LET dy = -1
240 LET x = 9:LET y = 9
248 REM
250 REM PINTAR PILOTA I RAQUETA
252 REM
260 INK 2
270 GOSUB 1000
280 GOSUB 1100
288 REM
290 REM CONTROL DE LES PARETS
292 REM
300 IF x+dx = 0 OR x+dx = 21 THEN
    LET dx = dx* -1
310 IF y+dy = 0 THEN LET dy = dy* -1
320 IF y+dy = 31 THEN PRINT AT x,y;
    INK 6;CHR$(128) : GOSUB 1400:
     GOTO 200
328 REM
330 REM CONTROL DE LA RAQUETA
332 REM
340 IF SCREEN$(x+dx,y+dy) = "X" THEN
    LET dy = dy* -1
348 REM
350 REM MOVIMENT DE LA RAQUETA
352 REM
360 IF INKEY$ = "o" AND a(1) > 1
    THEN GOSUB 1200
370 IF INKEY$ = "p" AND a(4) < 20
    THEN GOSUB 1300
380 GOSUB 1000
390 GOTO 300
985 REM
990 REM SUBRUTINA PINTAR PILOTA
995 REM
1000 PRINT AT x,y; INK 6; CHR$(128)
1010 LET x = x+dx : LET y = y+dy
1020 PRINT AT x,y; CHR$(144)
1030 RETURN
1085 REM
1090 REM SUBRUTINA PINTAR RAQUETA
1095 REM
1100 FOR i = 1 TO 4
1110 PRINT AT a(i),29; "X"
1120 NEXT i
1130 RETURN
1185 REM
1190 REM SUBRUTINA RAQUETA AMUNT
1195 REM
1200 FOR i = 1 TO 4
1210 LET a(i) = a(i)-1
1220 NEXT i
1230 PRINT AT a(1),29; "X"
1240 PRINT AT a(4)+1,29; INK 6; CHR$(128)
1250 RETURN
1285 REM
1290 REM SUBRUTINA RAQUETA AVALL
1295 REM
1300 FOR i = 1 TO 4
1310 LET a(i) = a(i)+1
1320 NEXT i
1330 PRINT AT a(4),29; "X"
1340 PRINT AT a(1)-1,29; INK 6; CHR$(128)
1350 RETURN
1385 REM
1390 REM SUBRUTINA ESBORRAR RAQUETA
1400 FOR i = 1 TO 4
1410 PRINT AT a(i),29; INK 6; CHR$(128)
1420 NEXT i
1430 RETURN
```

Les funcions de cadena



La major part de les funcions que hem vist fins ara només tractaven amb valors numèrics. En aquest apartat parlarem d'algunes operacions que podem fer amb el contingut de variables alfanumèriques: amb cadenes de caràcters.
Ja haviem vist, en el llibre anterior, que en el ZX-Spectrum, per tal d'obtenir els caràcters sisé, seté i vuitè d'una variable alfanumèrica, per exemple A$, haviem d'escriure:

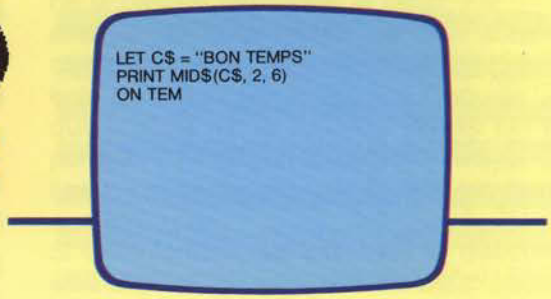
A$ (6TO8)

En els altres micros, o en quasi tots, hi ha una funció que fa el mateix la MID$. El seu format és:

```
MID$(variable$,X,Y)
```

on Y serà el nombre de caràcters que desitgem obtenir a partir de la posició X de la cadena. Així:

```
LET C$ = "BON TEMPS"
PRINT MID$(C$, 2,6)
```

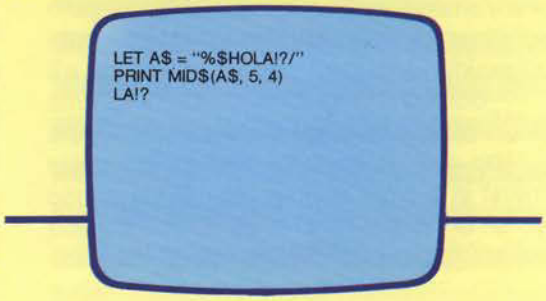


escriurà, a la pantalla

ON TEM

i si hi posem

```
LET A$ = "%$HOLA!?/"
PRINT MID$(A$,5,4),
```



a la pantalla sortirà

LA!?

Els valors X i Y poden ésser qualsevol tipus d'expressió numérica. Això ens pot ésser útil, per exemple, en el següent cas.
Suposem que tenim una taula d'una dimensió de 20 elements alfanumèrics:

DIM A$(20)

on cada element té 30 caràcters que conté el nom d'una persona, distribuïts de la següent forma:

- els 10 primers tenen el nom,
- els 10 del mig, el primer cognom, i
- els 10 darrers, el segon cognom.

Suposem també que la taula ja és emplenada correctament.
El següent programa escriu a la pantalla, en una línia, el primer cognom, i a la següent, el segon cognom seguit d'una coma i del nom; això ho fa per als 20 elements de la taula.

```
5 CLS
10 FOR I = 1 TO 20
20 PRINT MID$(A$(I),11,20)
30 PRINT MID$(A$(I),21,30);
40 PRINT ", ";MID$(A$(I), 1, 10)
50 NEXT I
```

Observem que, a la fi de la línia 30, hi ha un punt i coma; això vol dir que allò que escriurem a la línia 40 sortirà a la dreta d'allò que s'ha escrit a línia 30.
Per tal que no surti per pantalla tot el reguitzell de noms a la vegada, podem emprar la instrucció **INPUT,** tal i com ho fèiem en alguns programes del llibre anterior. Afegirem al programa la següent línia"

45 INPUT "PER CONTINUAR PITJEU <ENTER>";C$

Si el micro és un "Commodore", cal tenir en compte que els índexs de les taules, en lloc de començar per 1, comencen per 0; per tant, haurem de definir el vector com:

DIM A$(19)

Quan allò que ens interessi d'una cadena siguin els primers caràcters o bé els darrers, no és necessari usar la funció MID$; hi ha dues instruccions, la **LEFT$** i la **RIGHT$,** que fan això i que són més curtes.

LEFT$(B$,n)

ens dóna els n primers caràcters d'una cadena; per tant, és equivalent a posar

MID$(B$,1,n).

Si en el programa anterior substituïm MID$(A$,1,10) per **LEFT$(A$,10)** obtindrem els mateixos resultats. En el "ZX-Spectrum", aquesta funció no existeix; el seu equivalent es posar B$(TOn).
Si allò que ens interessa són els n darrers caràcters d'una cadena, posarem:

RIGHT$(B$,n).

Aquesta funció, tot i essent similar a la MID$, té un avantatge: no és necessari saber quina posició ocupa el darrer caràcter de la cadena. El "ZX-Spectrum" no té cap eina per a fer el mateix, si no és que sabem quina és la longitud de la cadena. Pero no ens hem de preocupar, perquè hi ha una altra funció:

LEN(X$)

que ens dóna la longitud de la cadena X$ i és general per a tots els micros.
Si fem:

LET L = LEN(B$)
PRINT B$(L−n+1 TO)

en un "ZX-Spectrum", obtindrem el mateix resultat que fent

LEFT$(B$,n)

en els altres micros.
El "Dragon" té una funció bastant útil, que permet posar dins una variable alfanumèrica un caràcter repetit diverses vegades; és la STRING$(n,c), on **c** es el caràcter que volem escriure n vegades. **c** pot ésser el codi ASCII del caràcter (en decimal) o bé el mateix

caràcter entre cometes. Així, si posem:

```
LET A$ = STRING$(20,"*")
PRINT A$
```

ens sortiran a la pantalla 20 asterics, l'un darrera l'altre; i si fem

```
PRINT STRING$(10,71).
```

a la pantalla hi sortiran deu lletres G seguides.
En els altres micros, per fer això cal posar:

```
10 FOR I = 1 TO 20
20 LET A$ = A$+"*"
30 NEXT I
40 PRINT A$
```

en el cas dels asterics; i

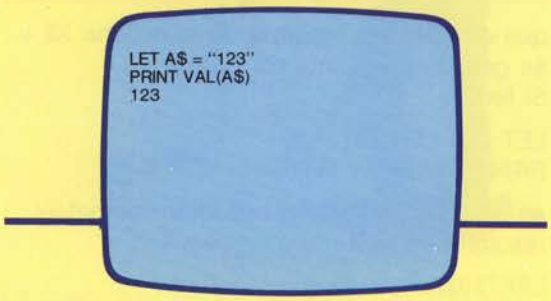
```
10 FOR I = 1 TO 10
20 PRINT CHR$(71)
30 NEXT I
```

en el cas de les lletres G.
Hi ha una altra funció que ens permet de convertir una cadena numèrica en la quantitat numèrica que representa. És la funció

VAL (caràcters numèrics).

Així

```
LET A$ = "123"
PRINT VAL (A$)
```

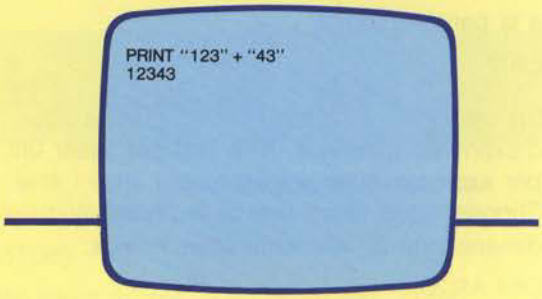


escriurà a la pantalla

```
> 123.
```

Perquè es vegi claramente la diferència, farem el següent:

```
PRINT "123" + "43"
```

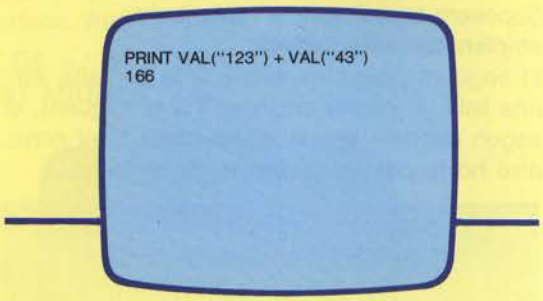


a la pantalla sortirà

```
> 12343
```

En canvi, si hi posem

```
PRINT VAL ("123") + VAL ("43")
```

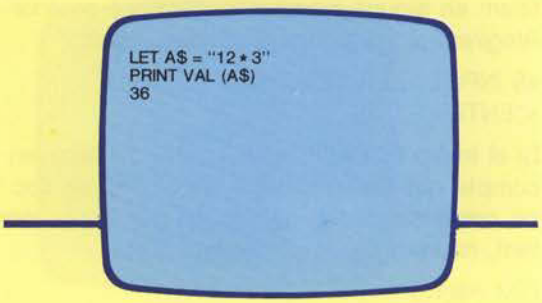


escriurà

```
> 166
```

perquè, com ja s'ha dit, la funció VAL ha convertit la cadena "123" en el nombre 123. Si dins la cadena hi ha simbols d'operació (+, −, *, /, ^), els deixa igual i efectua l'operació que indiquen.
Per tant, si escrivim

```
LET A$ = "12*3"
PRINT VAL (A$)
```



veurem, a la pantalla

> 36.

I passaria així mateix si hi posèssim

PRINT VAL ("123" + "3").

Si apareix una lletra al mig (no al principi ni al final) de la cadena, tot el que hi hagi a partir d'ella, i ella mateixa, es converteix en zeros.
Hi ha una funció similar a **VAL** que converteix un valor numèric en la cadena numèrica corresponent. Es la funció

STR$ (valor numèric).

Així

LET A = 123 : LET B = 43
PRINT SRT$ (A) + STR$ (B)

escriurà a la pantalla

> 12343.

Si ens hi fixem, veurem que **VAL** és la funció inversa de **STR$,** perquè, si fem

VAL (SRT$ (valor numèric))

recuperarem el mateix valor numèric que hi hem introduït. Però **STR$** no és la inversa de **VAL,** perquè

STR$ (VAL (cadena))

no ha de donar la mateixa cadena inicial, perquè si hi ha alguna lletra al mig de la cadena **VAL (cadena)** ens tornarà un seguit de zeros que **STR$** no convertirà en la lletra. Per comprovar-ho podem escriure

PRINT VAL (STR$ (12.46)).
PRINT STR$ (VAL ("42A3")).

Per acabar, parlem d'una funció que, en aplicar-la a una cadena, ens dóna el codi ASCII del primer caràcter de la cadena. Aquesta funció és

ASC (cadena).

En el "ZX-Spectrum" i "ZX-81", la mateixa funció s'escriu

CODE (cadena).

Així

LET A = ASC ("L")
PRINT A

escriurà

> 76.

De la mateixa forma

LET A = ASC ("HOLA")
PRINT A

escriurà

> 72.

Criptografia

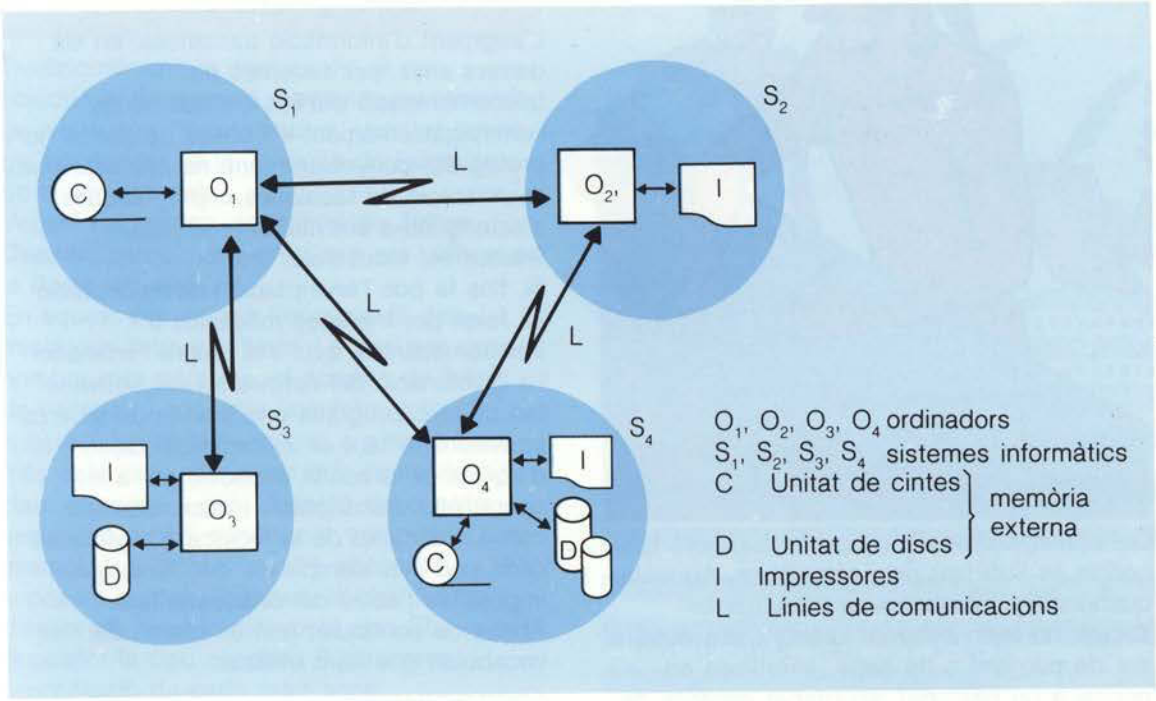
L'evolució dels sistemes informàtics apunta cap a la utilització de xarxes d'ordinadors que substitueixin el clàssic sistema informàtic enclaustrat i aïllat.

Avui es parla de sistemes distribuïts, amb tendència a compartir tota mena de recursos: *hardware,* discos, cintes, processadors, memòria, informació emmagatzemada en fitxers, etc. Ja es pot pensar en sistemes interconnectats, on cadascun d'ells realitza una tasca específica que podrà ésser aprofitada per qualsevol usuari de la xarxa.

Però aquesta possibilitat de compartir recursos planteja el problema de la **seguretat** i el de l'**autenticitat** de la informació transmesa o emmagatzemada. Seguretat, ja que hi ha informació que ha d'ésser confidencial; i autenticitat, o integritat, perquè hem d'ésser capaços d'assegurar d'on prové la informació que rebem. Cal protegir-la de malintencionades manipulacions.

A tall d'il·lustració, a la següent figura podem veure un cas típic de xarxa d'ordinadors i els problemes que se'ns presenten respecte de la seguretat i l'autenticitat.

Arxiu de cintes magnètiques



En qualsevol xarxa d'ordinadors, un element imprescindible és el medi per on ha de viatjar la informació: allò que abans n'hem dit el **canal de comunicació.** N'hi ha de diferents tipus: telefònics, de fibra òptica, vía satèl·lit, etc.

En general, quan un usuari vol treballar amb una xarxa, aquesta li exigeix la seva identificació: és allò que es coneix com a **LOGIN** (és el que ens demana la xarxa TRANSPAC, quan nosaltres volem comunicar-nos amb Toulouse per poder utilitzar el simulador de transmissions via satèl·lit de què ja hem parlat), que generalment es tracta d'un **mot de pas** *(password).* D'aquesta manera, es mantenen certs privilegis per a certs usuaris per tal de poder accedir a un determinat tipus d'informació, o poder llançar certs programes, connectar-se amb d'altres xarxes (per exemple Iberpac, amb Transpac), etc. El coneixement, per part d'un hipotètic "espía", del mot de pas d'algun usuari li possibilitaria de poder realitzar funcions associades a aquella persona en concret.

Deixant a part la possibilitat d'accedir al mot de pas d'un determinat usuari, un altre front d'atac, per part d'un "espía", serà el canal de comunicació, tot interceptant qualsevol missatge transmès entre els diferents components de la xarxa.

Per altra banda, dintre d'un sistema s'han de protegir tant els programes com la informació de les bases de dades, per tal d'evitar que un **espia** pugui violar o eliminar les proteccions del sistema. A més, cal custodiar:

- la informació, mentre viatja pel canal de transmissió, i
- la utilització dels recursos d'un determinat sistema.

La **CRIPTOGRAFIA,** ciència i tècnica que en els darrers anys ha tingut un reguany d'interès, tan teòric com pràctic, és la que s'ocupa de la protecció dels missatges transmesos sobre un canal, i de la informació emmagatzemada en sistemes digitals, tot contemplant els aspectes de seguretat i autenticitat esmentats abans.

Del primer sistema criptogràfic que hom té notícia és l'utilitzat pels **espartans.** Aquests, quan volien transmetre un missatge en secret, ho feien escrivint-lo sobre una estreta tira de pergamí o de papir, enrotllada en espiral a un bastonet, anomenat *escítola,* de tal forma que tota la superfície del bastonet restés recoberta de pergamí. Un cop desenrotllada la tira, allò que hi havia escrit en ella era inintel·ligible. No obstant, el destinatari tenia un bastonet igual que l'utilitzat per l'emissor; aleshores, només li calia enrotllar-hi la tira i llegir el missatge. Podríem dir que el bastonet era la **clau secreta.**

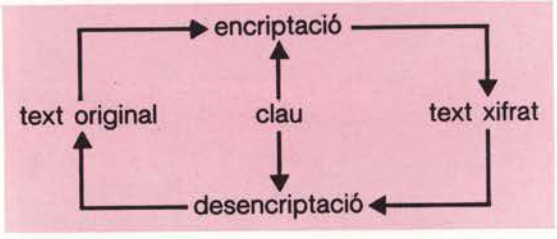
Fins ara, la criptografia era utilitzada quasi exclusivament en àmbits militars i governamentals. Un exemple recent de la seva utilització és en la Segona Guerra Mundial; els alemanys i japonesos utilitzaven una máquina, coneguda amb el nom d'**ENIGMA,** que **encriptava** els missatges que es transmetien entre ells. En principi, si els aliats podien interceptar-los, no s'assabentaven de res. Però, més tard, els aliats van ésser capaços de reconstruir una còpia d'**ENIGMA** i, a partir d'ella, van construir un **desencriptador** electrònic, anomenat **ULTRA,** que els permetia, llavors sí, de conèixer la informació que portaven els missatges encriptats. Aquest fou un exemple de **criptoanàlisi.**

L'augment d'informació transmesa, en els darrers anys, per sistemes de telecomunicació públics de fàcil accés, l'emmagatzemament en bases de dades no protegides convenientment, ha provocat que la utilització de tècniques criptogràfiques s'estengués a activitats comercials, financeres, industrials, etc.

Si, fins fa poc l'encriptació i desencriptació es feien per mètodes mecànics o electromecànics, avui s'hi empra l'ordinador. La combinació del *hardware* i del *software* fan de la criptografia una branca de recerca apassionant, tant en el camp del disseny d'equips de propòsit específic per a les operacions d'encriptació i desencriptació, com en el camp de la recerca d'algorismes cada cop més complexos que facin quasi impossible l'acció del **criptoanalista.**

Abans de continuar, fem un resum del vocabulari que hem utilitzat:

CRIPTOGRAFIA:
Ciència i art d'escriure, tot guardant inintel·ligible el contingut del text escrit.
ENCRIPTACIO:
Procés de transformació del text original en el text xifrat.
DESENCRIPTACIO:
Procés de transformació del text xifrat en el text original.
CLAU:
Paràmetres que controlen els processos d'encriptació i de desencriptació.



CRIPTOANÀLISI:
Ciència i estudi dels mètodes de descobrir els textos xifrats.
CRIPTOLOGIA:
Coneixement de la criptografia i de la criptoanàlisi.

Dos sistemes criptogràfics

Tradicionalment, la criptografia té per objectiu la transmissió i emmagatzemament de missatges "indesxifrables" per tot aquell qui no disposi de la "clau" o de l'algorisme de desencriptació.

Vegem l'algorisme, anomenat de **Julius Caesar,** perquè hom creu que era utilitzat en la Roma Imperial.

En aquest cas es feia l'encriptació dels missatges lletra per lletra. La clau era un nombre més petit que el nombre de lletres de l'alfabet, i encriptar una lletra volia dir substituir-la per la lletra que era tants llocs més enllà com el número que expressava la clau, de manera que, després de la **zeta** venia la **a,** i hom continuava comptant. Per exemple, utilitzant el nostre abecedari, suposant que té 27 lletres, la clau hauria d'ésser un nombre més petit que 27. Suposem la clau escollida 9 aleshores l'encriptació de cada lletra seria:

```
c ç d e f g h i j k l m n o p q r s t u v w x y z

riptació

l m n o p q r s t u v w x y z a b c ç d e f g h
```

i així, si volem encriptar el missatge **criptografia** ens donaria el text **text xifrat:** *karycxpaiori.*

Si una persona veiés el text xifrat que hem escrit no entendria pas què hem transmès; però, si s'hi trenca la closca i assaja de trobar-hi la clau, ben segur que podria **desxifrar, desencriptar,** el missatge original.

Si entreu els dos programes següents en el vostre microordinador, tindreu un simulador d'encriptador-desencriptador de Julius Caesar.

Ben segur que trobeu aquest procediment molt senzill. Aquest es pot complicar fent combinacions de claus i, fins i tot, no respectant els blancs del text original per tal de confondre més el possible "espía". A la secció d'altres programes n'hem posat un de més embolicat que els presents.

```
1 REM*****************************************
2 REM******  ENCRIPTADOR  D'EN  CESAR  ******
3 REM**********Versio Commodore***************
10 PRINT "<CLR>"
20 LET R$=""
30 INPUT "ENTRA LA CLAU (1-26)";A
40 IF A>26 OR A<1 THEN
   PRINT "ERROR, CLAU INCORRECTA ": GOTO 30
50 INPUT "ENTRA MISSATGE";B$
60 FOR I=1 TO LEN(B$)
70 LET N=ASC(MID$(B$,I,1))
80 IF N<65 OR N>90 THEN
   PRINT "ERROR, MISSATGE INCORRECTE": GOTO 50
90 LET M=N-A
100 IF M<65 THEN LET M=M+26
110 LET R$=R$+CHR$(M)
120 NEXT I
130 PRINT "<CLR>"
140 PRINT "MISSATGE:"
150 PRINT B$: PRINT
160 PRINT "MISSATGE ENCRIPTAT:"
170 PRINT R$
```

```
1 REM*****************************************
2 REM******  DESENCRIPTADOR  D'EN  CESAR  ******
3 REM**********Versio Commodore***************
10 PRINT "<CLR>"
20 LET R$=""
30 INPUT "ENTRA LA CLAU (1-26)";A
40 IF A>26 OR A<1 THEN
   PRINT "ERROR, CLAU INCORRECTA": GOTO 30
50 INPUT "ENTRA EL MISSATGE ENCRIPTAT";B$
60 FOR I=1 TO LEN(B$)
70 LET N=ASC(MID$(B$,I,1))
80 IF N<65 OR N>90 THEN
   PRINT "ERROR, MISSATGE INCORRECTE": GOTO 50
90 LET M=N+A
100 IF M>90 THEN LET M=M-26
110 LET R$=R$+CHR$(M)
120 NEXT I
130 PRINT "<CLR>"
140 PRINT "MISSATGE ENCRIPTAT:"
150 PRINT B$: PRINT
160 PRINT "MISSATGE DESENCRIPTAT:"
170 PRINT R$
```

Tots els sistemes criptogràfics utilitzats abans de la invasió dels ordinadors han quedat obsolets: formen part d'una història sense protagonisme, avui; perquè la velocitat

de tractament de la informació i la disponibilitat d'estudis estadístics sobre l'aparició de símbols i **n**-grames (**n** símbols consecutius) en qualsevol idioma, fan que el problema de trobar la **clau** (la criptoanàlisi) sigui gairebé un joc.

Vegem ara un altre exemple; però, aquest, actual; que permet utilitzar una part de la **clau** públicament. Aquest és el sistema d'encriptació conegut com R.S.A.

Es un sistema **criptogràfic de clau pública,** que és molt apropiat per utilitzar en una xarxa d'ordinadors. En aquest cas, la clau es desglossa en dues, una part és la **clau d'encriptació,** que es pot posar en un fitxer públic, al qual tothom que estigui connectat a la xarxa pot tenir accés, mentre que l'altra part és la **clau de desencriptació,** la qual serà secreta per a cada usuari.

El mètode que volem exposar és degut a tres investigadors (Rivest, Shamir i Adleman) d'un reconegut centre de recerca, (el M.I.T. de la Universitat de Standford, dels E.U.A.). Consisteix a donar un valor numèric a cada missatge a encriptar, i utilitzar propietats dels nombres primers. Recordem que un nombre es diu primer si no es pot dividir per cap altre nombre que no sigui ni l'u ni ell mateix, per exemple: 3, 5, 7,..., 31,...

Si poseu aquest programa que utilitza el mètode del **garbell** d'**Eratòstenes,** podreu trobar tots els nombres primers, entre 1 i 400.

```
1 REM*******************************
2 REM*** GARBELL D'ERATOSTENES ***
3 REM*****Versio Commodore********
10 DIM A(400)
20 FOR I=0 TO 400
30 LET A(I)=I
40 NEXT I
50 FOR I=2 TO 400/2
60 IF A(I)=0 THEN GOTO 100
70 FOR J=I TO 400-I STEP I
80 LET A(I+J)=0
90 NEXT J
100 NEXT I
110 PRINT "<CLR>"
120 FOR I=0 TO 400
130 IF A(I)<>0 THEN PRINT A(I),
140 NEXT I
```

Be, anem a fabricar les **claus** del R.S.A.:

1. Agafar dos nombres primers, que en direm **p** i **q** (no escollir-los massa grans, perquè el micro trigaria molt a fer els càlculs. Per exemple **p** = 3 i **q** = 11).
2. Calcular el producte de **p** per **q.** (En direm **n = p × q.** En el nostre cas, **n** = 33).
3. Calcular el producte de **p – 1** per **q – 1.** (En direm **m = (p – 1)·(q – 1).** En el nostre cas, **m** = 20).
4. Triar un nombre primer més petit que **m** i que no el divideixi (en direm **d**) i buscar un altre nombre (en direm **e**) tal que el producte de **e** per **d** sigui un múltiple de **m** més **u.** (Per exemple, **d** = 17, aleshores **e** = 13 [17*13 = 11*20 + 1]).

Tot això ho podeu fer fent servir aquest altre programa:

```
10 REM ****************************************
20 REM ********** CALCUL DE L'INVERS ********
30 REM ****************************************
40 INPUT " ENTRA EL VALOR DE P ";P
50 INPUT " ENTRA EL VALOR DE Q ";Q
60 INPUT " ENTRA EL VALOR DE D ";D
70 LET M = (P-1)*(Q-1)
80 FOR I = 1 TO M
90 LET A = I*M
100 FOR J = 1 TO M
110 LET B =  J * D
120 IF B > A+1 THEN GOTO 150
130 IF B = A+1 THEN GOTO 170
140 NEXT J
150 NEXT I
160 PRINT " EL VALOR DE D ES ERRONI ": GOTO 4
170 PRINT " EL VALOR DE E ES ";J
```

Un cop fets aquest càlculs, farem pública la **clau d'encriptació: (n,e)** i ens guardarem secreta la **clau de desencriptació: (n, d).**

La gràcia d'aquest algorisme rau, de primer, en el nombre **m** (que se sol anomenar **indicatriu d'Euler):** per això, també cal tenir-lo ben secret; i en segon lloc, en la dificultat de trobar la descomposició de **n** com a producte dels dos primers, quan aquests són escollits ben grans.

Si disposéssim d'un ordinador que fes un milió d'operacions elementals en un segon, necessitaríem els temps que indiquem a la taula següent per descomposar el valor de **n** amb producte de dos primers **p** i **q.**

nombre de dígits decimals de n	nombre d'operacions	temps de càlcul
50	$1.4 * 10^{10}$	4 hores
75	$9 * 10^{12}$	104 dies
100	$2.3 * 10^{15}$	74 anys
200	$1.2 * 10^{23}$	3.800.000.000 anys

El nostre exemple no és tan complicat, perquè hem pres un valor de **n** de només 2 dígits decimals. Però els autors del mètode aconsellen d'agafar dos nombres primers d'unes 100 xifres cadascun i fer públic el valor de **n** que tindrà al voltant de les 200 xifres. Evidentment, sempre s'han de guardar en secret els valors de **p** i de **q.**
Els mètodes d'encriptació i desencriptació són molt senzills:

Mètode d'encriptació:

– assignar un valor numèric entre 2 i n–1 a cada missatge que volem transmetre; en el nostre cas, a cada lletra de l'abecedari:

a	b	c	d	e	f	g	h	i	j	k
02	03	04	05	06	07	08	09	10	11	12

l	m	n	o	p	q	r	s	t	u	v
13	14	15	16	17	18	19	20	21	22	23

w	x	y	z	ç	.	,	;	:	'
24	25	26	27	28	29	30	31	32	

– Sigui **M** el valor numèric del missatge que volem encriptar; aleshores el **missatge xifrat,** que en direm **C,** serà obtingut fent:

1) calcular **M** elevat a **e,**
2) dividir aquest resultat per **n,**
3) **C** és la resta de la divisió anterior (dues xifres decimals) (essent **(n,e)** la clau d'encriptació que hem fet pública)

Una manera més ràpida de fer aquestes tres operacions està reflectida en el següent programa:

```
1 REM**************************************
2 REM******  EXPONENCIACIO  RAPIDA  ******
3 REM*********Versio Commodore*************
10 PRINT "<CLR>"
20 INPUT "BASE: ";BAS
30 INPUT "EXPONENT: ";XP
40 INPUT "DIVISOR: ";DIV
50 LET A1=BAS:LET Z1=XP:LET X=1
60 IF Z1=0 THEN LET RES=X: GOTO 120
70 LET I=Z1-INT(Z1/2)*2
80 IF I<>0 THEN LET Z1=Z1-1:LET X=X*A1-
   INT(X*A1/DIV(*DIV: GOTO 60
90 LET Z1=INT(Z1/2)
100 LET A1=A1*A1-INT(A1*A1/DIV)*DIV
110 GOTO 70
120 PRINT "RESULTAT= ";RES
```

Utilitzant aquest programa, si volem encriptar el missatge **Barcelona** resultarà, com a **text xifrat:** 092913163007252729

missatge original	ENCRIPTACIÓ assignació numèrica	text xifrat
B	03	09
A	02	29
R	19	13
C	04	16
E	06	30
L	13	07
O	16	25
N	15	27
A	02	29

Mètode de desencriptació

- Agafar les xifres decimals del text xifrat de dues en dues. Diguem-ne **C** d'aquest valor; aleshores obtindrem el text original fent:
  1) calcular **C** elevat a **d,**
  2) dividir aquest resultat per **n,** i
  3) **M** és la resta de la divisió anterior (dues xifres decimals), (éssent **(n,d)** la clau de desencriptació que ens hem guardat secreta).
- Trobar quin és el missatge (en el nostre cas quina és la lletra) que correspon al valor numèric **M.**

Notem que el programa anterior també ens serveix per a desencriptar, ja que l'algorisme és el mateix en ambdues operacions; l'única cosa que canvia és el valor de la clau.

DESENCRIPTACIÓ text xifrat	missatge original	assignació alfabètica
09	03	B
29	02	A
13	19	R
16	04	C
30	06	E
07	13	L
25	16	O
27	15	N
29	02	^

A la secció d'altres programes hi hem posat un anomenat "Criptosistema RSA" que conté aquest algorisme complet.

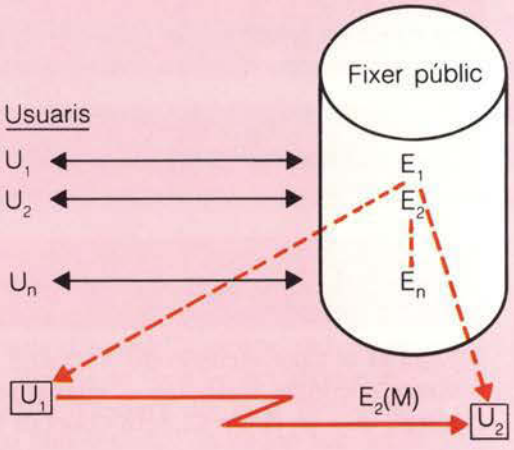
Criptosistemes amb clau pública

Aquest segon exemple que acabem de veure és el que s'anomena **criptosistema amb clau pública,** que permet d'ésser utilitzat en una xarxa d'ordinadors, mantenint el secret de la informació respecte a tots els altres usuaris.

En un criptosistema com aquest, podem considerar que cada usuari s'ha fabricat les pròpies claus d'encriptació i desencriptació, mitjançant un cert mètode, i ha dipositat la primera en un fitxer públic, mentre que, la segona, la guarda secreta.

Suposem que dos usuaris (U1 i U2) d'una xarxa d'ordinadors que disposa d'un criptosistema amb clau pública volen comunicar-se. Si U1 vol enviar un missatge a U2, aleshores mirarà quina és la clau d'encriptació d'U2 i enviarà el missatge xifrat segons aquesta clau. Encara que un altre usuari intercepti el missatge, no podrà desxifrar-lo, perquè no disposa de la clau de desencriptació d'U2.

CRIPTOSISTEMA AMB CLAU PÚBLICA



U_1 coneix: E_2 E_1 D_1 secret

U_2 coneix: E_1 E_2 D_2 secret

Missatge a transmetre M

Encriptació $E_2(M)$

No obstant, pot passar que un usuari U3 hagi interceptat la comunicació i vulgui fer arribar un missatge a U2, tot dient-li que és U1 qui l'envia. Per poder evitar una situació com aquesta, sorgeix la idea de **signatura digital.** Això és, quan U2 rep un missatge, es vol assegurar de saber qui l'hi ha enviat. Una manera de **signar** els missatges és la proposada per Rivest, Shamir i Adleman, que consisteix en:

a) Quan U1 vol enviar un missatge **M** a l'usuari U2, la primera cosa que fa és aplicar el seu mètode de desencriptació sobre el missatge **M.** Suposem que el resultat és **S.** Tot seguit, apliquem l'algorisme d'encriptació d'U2 a **S** i suposem el resultat **C.** Aquest text xifrat **C** és el que s'envia sobre el canal de comunicació.

b) Quan U2 rep **C,** li aplica el seu mètode de desencriptació i recupera **S.** Aquest és un missatge inintel·ligible, però sap que és U1 qui volia comunicar-se amb ell. En aquest cas, mirarà quina és la clau d'encriptació d'U1, aplicarà l'algorisme d'encriptació sobre **S** i recuperarà el missatge original **M.**
En aquest cas, **S** és la **signatura digital de M**

Missatge a transmetre	M
Signatura del missatge	$S = D_1 (M)$
Encriptació	$X = E_2 (S)$
Desencriptació	$D_2 (X) = S$

però S és indesxifrable per U_2,
però com que sap que és U_1
el qui li envia només li cal fer:

$E_1 (S)$

i retrobarà M.

(mirar figura anterior)

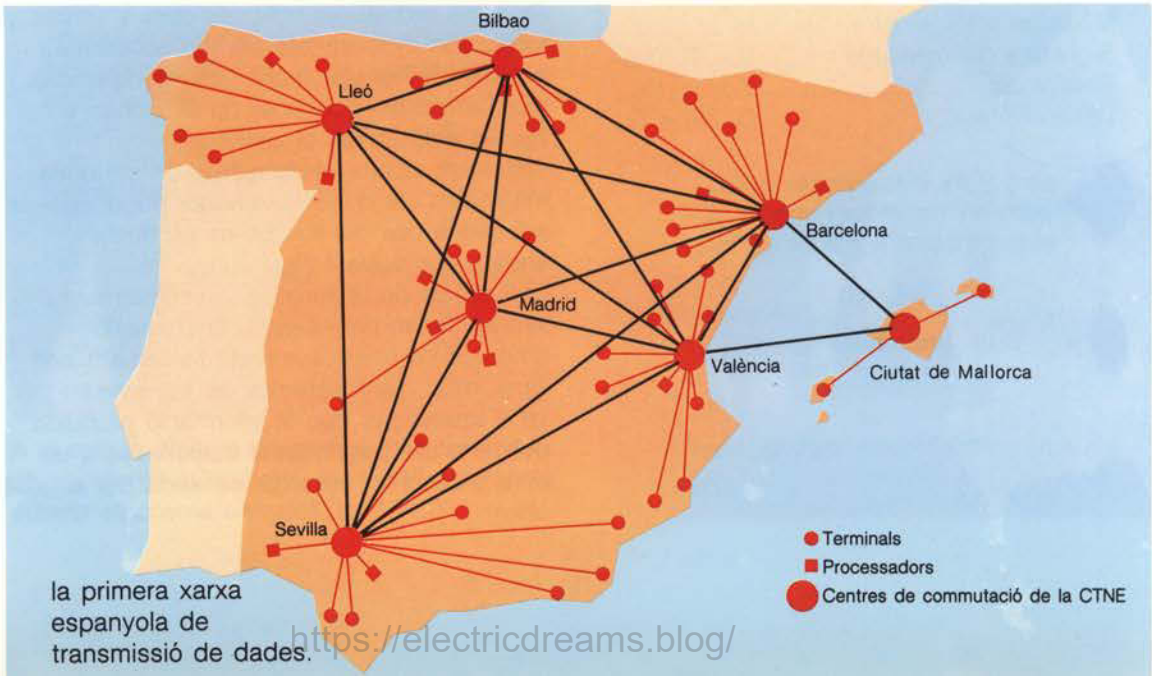
Xarxes d'ordinadors

El desenvolupament espectacular dels ordinadors i l'expansió de les telecomunicacions convergeixen en les denominades **xarxes d'ordinadors,** o **xarxes de sistemes informàtics,** o **xarxes de teleprocés...** conceptes dels quals ben segur hem sentit a parlar, i sobre els quals voldríem saber més coses.
Avui, els ordinadors utilitzats com a serveis locals són molt limitats per les necessitats actuals de manejar grans volums de dades que, en moltes ocasions, no es troben en el lloc on està ubicat el sistema. Es fa necessari, doncs, de disposar de mètodes mitjançant els quals l'ordinador pugui obtenir les dades des del lloc on es produeixin, encara que aquest sigui llunyà.
Les xarxes de teleprocés, o senzillament xarxes, tenen per objectiu la connexió d'equips informàtics, situats físicament ben lluny, mitjançant sistemes de transmissió per tal d'aconseguir que la informació produïda per un d'ells pugui ésser transmessa a un altre i, després, ésser processada per aquest últim.

Aquesta comunicació es fa, generalment, tot aprofitant la xarxa de comunicacions telefòniques; i per això es necessita un dispositiu que adapti els senyals, corresponents a informacions digitals (zeros i uns), al sistema de comunicació que en un principi havia estat pensat per a transmetre sons. Aquest dispositiu s'anomena **MODEM,** i allò que fa és modular el senyal emès per l'ordinador perquè pugui ésser transmès per fil telefònic i, a l'inrevés, recuperar el senyal rebut pel fil, tot transformant-lo en una informació digital.

MODEMS de comunicació

Un cop aconseguida l'adaptació ordinador-modem, cal definir el canal telefònic que uneix els dos modems. En realitat, les companyies telefòniques ofereixen o bé la possibilitat de connectar-se a la xarxa telefònica de la veu, que a Espanya se'n diu RAC *(Red Automática Conmutada),* o bé la de llogar unes línies privades entre dos punts determinats de la xarxa, cosa que s'anomena la **comunicació punt a punt.** Les raons d'utilitzar una o altra alternativa responen a motius de preus i a la qualitat de la transmissió. De manera general, la utilització d'una xarxa compartida, com és la RAC, resulta més barata que llogar un **circuit punt a punt;** no obstant, quan es prima la qualitat de la transmissió, resulta millor el lloguer del circuit punt a punt.



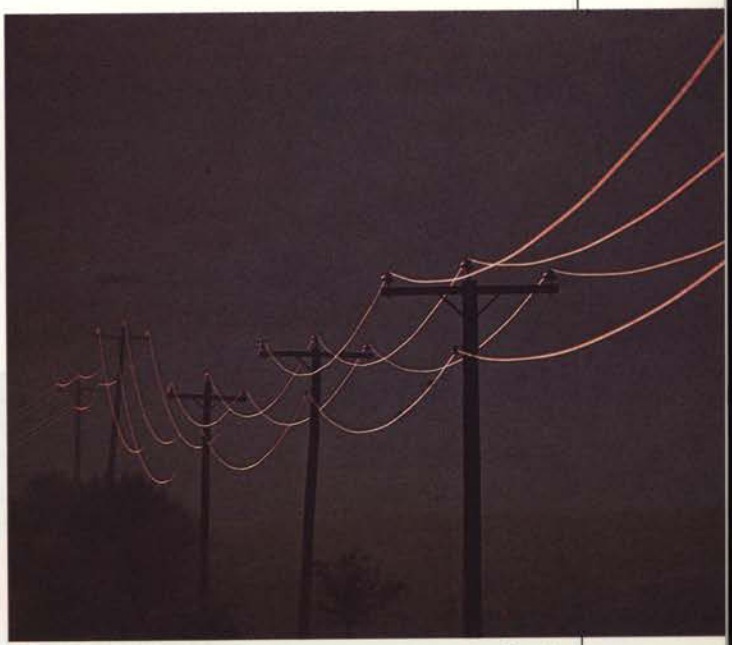
la primera xarxa espanyola de transmissió de dades.

Un usuari que desitgi usar la RAC per connectar els seus equips informàtics haurà de marcar, cada cop que vulgui establir una connexió, el número de telèfon assignat a l'ordinador amb el qual vol comunicar-se. Un cop establerta la comunicació telefònica, com en una conversació normal, es passa la comunicació a la modalitat de transmissió de dades i, un cop sincronitzats emissor i receptor, comença la transmissió pròpiament dita.

Tant si s'usa la RAC com els circuits punt a punt, les transmissions de dades es fan sobre la infraestructura de la xarxa telefònica; és a dir, les dades van entremesclades amb les converses d'altres usuaris.

Una diferència entre la utilització de la RAC i el lloguer de les línies punt a punt és que, la primera, transmet les dades utilitzant camins diferents, en cada cas, segons la saturació de la xarxa telefònica; mentre que, la segona, sempre passa pel mateix cami. Així, per transmetre una informació a través de la RAC, de Barcelona a València, segons estigui la xarxa en aquell moment, la informació pot seguit un camí passant per Barcelona-Lleó-Sevilla-València. Seria el camí de cost mínim sobre el gràfic determinat per la xarxa segons les ocupacions de cada nucli commutador.

En molts casos, malgrat les distàncies, es vol treballar en les informacions produïdes per un ordinador des d'un altre. Un exemple d'aquesta pretensió, que és de fet una realitat, és la comanda de bitllets per a vols d'una companyia aèria, on un gran ordinador central disposa d'una base **base de dades** que conté les informacions relatives a les places disponibles i ocupades de tots els seus vols i, fins i tot, pot consultar amb l'ordinador central d'altres companyies. En aquest ordinador central es connecten les terminals de cadascuna de les oficines, siguin al lloc on siguin, i això permet que, davant el mateix client, es pugui confirmar la reserva de plaça que sol·licita, al mateix temps que aquesta reserva queda enregistrada a la base de dades de



Línies de la xarxa telefònica *(The Image Bank)*

l'ordinador, de manera que un altre client ja no podrà obtenir aquesta plaça des de cap lloc. Aquesta sensació de resposta immediata s'anomena treballar en **temps real.**

El terme **sistema de temps real,** utilitzat en l'àmbit informàtic, significa que la transmissió de la informació es fa tan ràpidament que és rebuda pràcticament el mateix moment en què s'emet. Fa la sensació d'estar treballant amb l'ordinador central al costat.

La contraposició al terme de temps real és el de **sistema de procés per paquets** o **sistema en *"batch",*** on la informació és enviada de forma massiva perquè l'ordinador la processi al seu aire, sense presses, i, un cop processada, torni els resultats a l'usuari.

Un tercer tipus de sistema a tenir en compte en una xarxa de teleprocés és el **sistema de temps compartit** o ***"time-sharing",*** en el qual diversos usuaris, amb problemes i aplicacions diferents, poden utilitzar un mateix ordinador. Cadascun dels treballs es processen en l'ordinador central, de manera que sembla existir simultaneïtat en els processos, des del punt de vista de l'usuari,

perquè l'ordinador, alhora, va compaginant totes les tasques. Es evident que aquest procés requerirà ordinadors amb sistemes operatius ben complexos.

ORDINADOR CENTRAL

USUARI 1

USUARI 2...

USUARI n

Quant als tipus de sistemes de transmissió, podem considerar:

- **Transmissió en línia** *("on line"):* significa que els elements terminals es connecten directament amb l'ordinador.

- **Transmissió fora de línia** *("off line"):* significa que els elements terminals estàn connectats a uns dispositius auxiliars que, després, es connectaran a l'ordinador.

- **Sistema interactiu:** significa que l'usuari pot "dialogar" amb l'ordinador.

Mentre que pel que fa als tipus de canals –o línies– de transmissió, podem considerar:

- **línia símplex:** la que tan sols pot transmetre informació en un sentit.
- **línia semidúplex:** la que pot transmetre en les dues direccions, però no pas simultàniament.
- **línia dúplex:** la que pot transmetre en les dues direccions, fins i tot simultàniament.

Evidentment, una línia dúplex equival a dues línies simples usades en les dues direccions oposades.

El desavantatge d'una línia semidúplex és que, si es desitja que les dades viatgin en ambdues direccions (per exemple entre un ordinador i un terminal), quan l'ordinador rep la comanda que li envia l'usuari, el modem haurà d'invertir el sentit de les comandes perquè l'ordinador hi pugui respondre. Això fa que les comunicacions siguin més lentes que en les línies dúplex. No obstant, són les més utilitzades, perquè encara que el preu de la línia no varia gaire, els accessoris necessaris per a les dúplex són més complexos i molt més cars.

La possibilitat de comunicar-se amb altres usuaris a través de la línia telefònica fa que es pugui accedir, en temps real, a dispositius molt llunyans, que tenen una capacitat de tractament de dades molt més gran que el sistema informàtic propi, implantant així el concepte de descentralització de la informàtica i disminució dels costos dels recursos propis, al mateix temps que provoca una major utilització de les comunicacions. És el començament de l'era de la **telemàtica,** mot utilitzat per a designar la convergència entre telecomunicacions i informàtica.

Fitxers i bases de dades

El concepte de memòria central, o principal, com a dispositiu per a emmagatzemar temporalment dades, no es exclusiu dels micros; és el mateix, fins i tot, per als grans ordinadors. És un dispositiu que ha de suportar contínuament diferents programes i diferents col·leccions de dades; i la seva capacitat és insuficient per a moltes aplicacions i problemes; és necessari, per tant, disposar de mitjans auxiliars d'emmagatzemament. De la mateixa manera que els cassettes i diskettes són mitjans d'emmagatzemament "massiu" per als micros, les unitats de cinta i de disc ho són per als ordinadors grans i mitjans.

Quan un programa necessiti gran quantitat de dades d'entrada, o bé generi massa informació perquè pugui emprar la memòria central com a dispositiu d'emmagatzemament, serà necessària la utilització d'un fitxer.

Un fitxer és un conjunt de dades gravades en dispositius de memòria.

Aleshores, un programa que necessiti moltes dades, les llegirà d'un fitxer. Així farà una lectura, tractarà les dades llegides amb les seves instruccions i generarà uns resultats. El tractament d'aquestes dades d'entrada

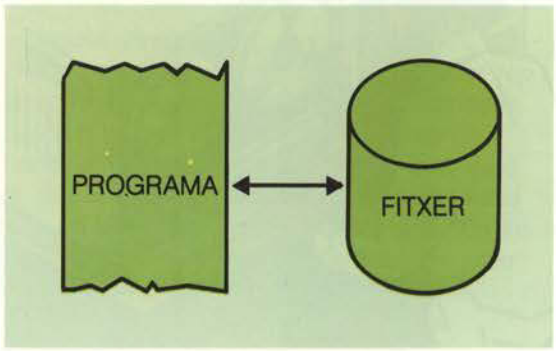
Unitat de discos

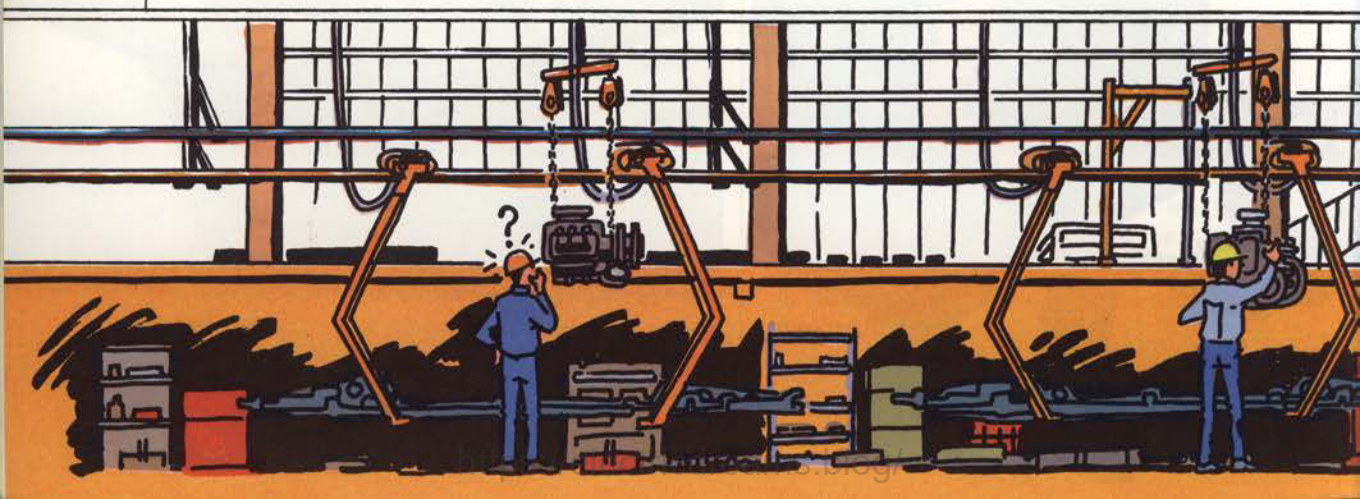
sempre serà el mateix, perquè el programa que les usa no canvia; això obliga al fet que tots els grups de dades que llegeixen un programa coincideixin en: TIPUS, ESTRUCTURA I DIMENSIÓ.
Per entendre bé aquestes restriccions compararem un programa amb una cadena de muntatge de cotxes. A l'entrada de la cadena hi ha una pila de xassis (dades d'entrades). Aquests van cap a un passadís on hi ha una colla d'obrers que els col·loquen els suports per al motor (actuen com les instruccions). Tot seguit, hi ha una pila de motors (unes dades d'entrada) que s'encaixen en els xassis... Aturem-nos aquí!
Què passaria si, en el primer pas, entréssim un motor en comptes d'un xassis? Res no aniria com ha d'anar. Per tant, les dades han d'ésser del mateix tipus.
Què passaria si entréssim un xassis (o un motor) d'un altre cotxe diferent al que volem fabricar en sèrie? Res no encaixaria al seu lloc. Per tant, les dades han de tenir la mateixa estructura.
Si a la pila de xassis n'hi hagués un que coincidís amb els altres, excepte que fos més ample que els normals, no cabria a la cadena i aquesta s'aturaria. Per tant, les dades han de tenir el mateix volum.
Aquest es un exemple clar d'un "programa" que necessita dos fitxers d'entrada: un de motors i un de xassis; dins de cadascun d'ells, tots els elements han d'ésser idèntics en tipus, volum i estructura.

De cadascun dels elements d'un fitxer se'n diu **REGISTRE.** Així com un motor té diverses peces, un registre conté diverses dades: són els **CAMPS.**
Així, un fitxer d'agenda estarà format per un seguit de registres amb els següents camps:

NOM	ADREÇA	TELÈFON

El camps NOM i ADREÇA seran alfanumèrics, mentre que el camp TELÈFON serà numèric. La longitud d'un mateix camp serà la mateixa per a tots els registres.
Per identificar un registre dins un fitxer, s'hi empra un camp de CLAU; en el cas d'una agenda, és el camp NOM, perquè és el que usem per cercar-hi un telèfon o una adreça.
Fins ara hem vist una estructura consistent en un programa que necessita unes dades (o que ha d'escriure-les) d'un fitxer i va a buscar-les.

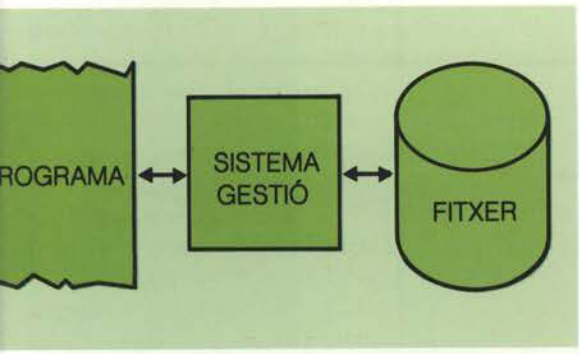




Aquesta estructura ens obliga a programar de manera que el programa entengui quina és la disposició física de les dades:

- quina estructura tenen, i
- quin lloc ocupen dins del dispositiu d'emmagatzemament.

Aleshores, si modifiquem el programa, hem de modificar el fitxer, i viceversa. Aquest inconvenient va portar els tècnics informàtics de finals dels anys 50 a investigar una nova forma d'enfocar el tema. Van pensar a estalviar feina al programador, tot intentant de crear dins l'ordinador un sistema de gestió de dades tal que, un cop definida l'estructura, ell mateix s'encarregués de distribuir les dades físicament dins el dispositiu d'emmagatzemament.

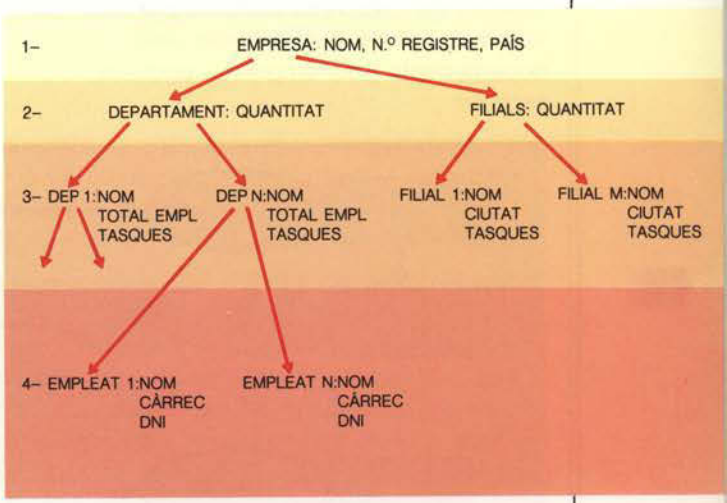


Després van pensar d'afegir dins aquests sistemes noves facilitats, per tal de connectar unes dades amb les altres a l'hora de fer un programa.

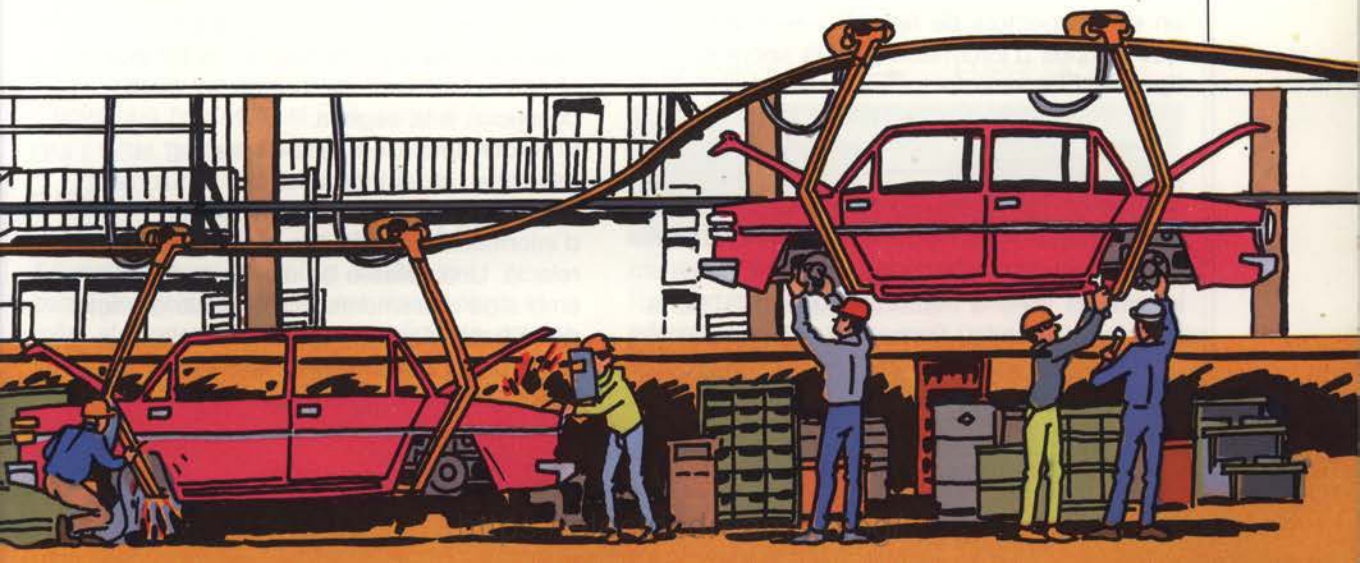
Havien creat les **bases de dades.**
Les bases de dades es classifiquen segons el tipus de connexió que s'estableixi. Els tres tipus principals són:

- JERÀRQUICA
- XARXA
- RELACIONAL

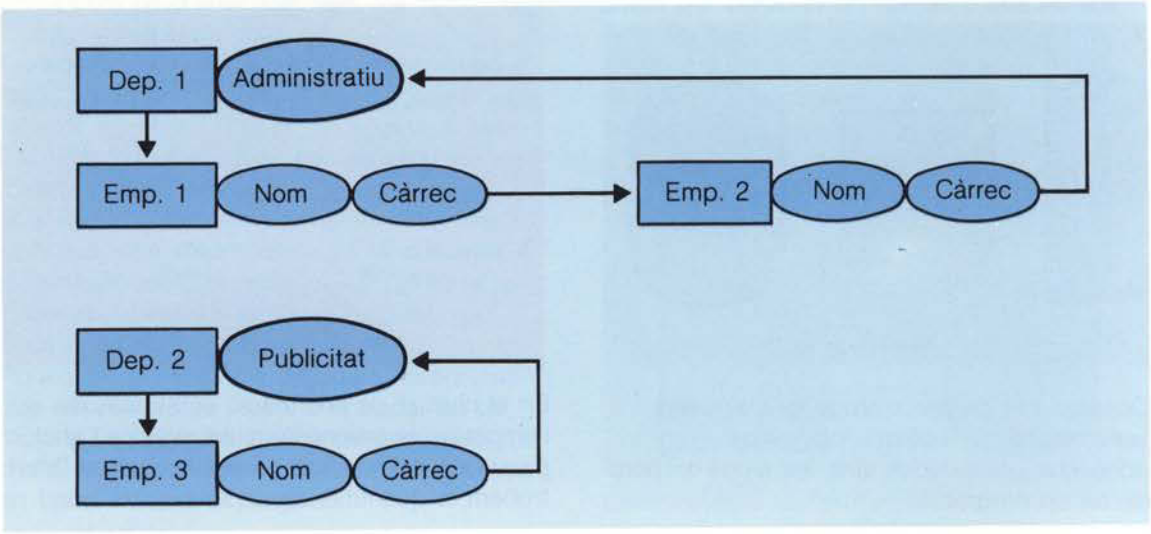
En el cas de les bases jeràrquiques, allò que es fa és definir un registre en el qual la seva informació està distribuïda per nivells enumerats segons una jerarquia (com un arbre genealògic). Per exemple:



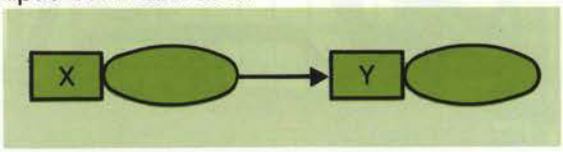
En el nivell 1, la informació es el nom de l'empresa, el seu número de registre i el país. Des del nivell 1, es pot accedir al 2, on trobem la quantitat de departaments i de



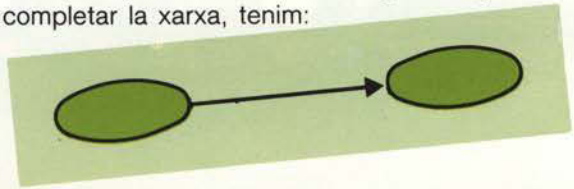
filials de què disposa l'empresa. Penjant del nivell 2 hi ha, en el 3, informació sobre cadascun dels departaments i filials,...
A les bases de dades de xarxa s'estableixen uns conjunts d'informació independents, i s'interconnecten de manera que es reflecteixi en tot moment si hi ha una jerarquia d'un tipus d'informació sobre un altre, o si hi ha una dependència, o si diverses informacions són dins la mateixa categoria; de tal manera que, al contrari del cas jeràrquic, des d'un bloc d'informació qualsevol es pugui arribar a qualsevol altre. Per exemple, seguint amb la situació de l'empresa anterior, per tenir en forma de xarxes la informació dels departaments i els empleats, podríem escriure:



on els connectors de tipus X ⟶ Y vol dir que el tipus d'informació X està sobre el tipus Y. El connector



representa que X i Y estan, com a informació, dins la mateixa categoria. I, per a completar la xarxa, tenim:



La idea de conjunts d'informació independents es reflecteix en el fet que no hi ha cap element de la primera xarxa connectat a la segona.
El tercer tipus de bases de dades, el relacional, és el més acceptat actualment com a eina important per la gestió d'informació. Es basa en el concepte de relació. Una **relació** és una taula formada amb alguns elements d'informació del total de la base. Amb el mateix exemple de l'empresa, podríem definir les següents relacions:

Relació 1

NOM DEP.	TOTAL EMPL.
Informat.	8
Publicit.	3
Adminis.	20

Relació 2

NOM DEP.	NOM EMPL.	DNI
Informat.	Joan	1112933
Publicit.	Pere	2120314
Adminis.	Mercè	9108240
Gestió	Jordi	1943128

Tota la manipulació i consulta d'informació distribuïda d'aquesta manera es basa en el fet que diversos tipus de dades estan dins una relació R, si hi ha un grup d'aquestes dades que "depèn lògicament" de la resta. Vegem-ho en el cas dels dos tipus de dades de la relació 1: És clar que, donat un nom de departament, a la relació hi trobarem un únic valor per al total d'empleats: direm que TOTAL EMPLEATS depèn de NOM DEP i escriurà:

NOM DEP ⟶ TOTAL EMPLEATS

Així, aquests dos blocs d'informació poden formar una relació (no podem posar-ho al revés, perquè no és estrany que dos departaments tinguin el mateix nombre d'empleats).
En canvi NOM EMPLEAT i NOM DEPARTAMENT no poden formar una relació (hi ha massa persones amb el mateix nom) a menys que hi afegim el DNI, perquè

DNI i NOM DEP ⟶ NOM EMPLEAT

o una altra

DNI i NOM EMPLEAT ⟶ NOM DEP

Aquestes bases de dades contenen bastantes relacions d'aquest tipus; però, a més, el programador en pot definir de noves: fins i tot, nous blocs i nous tipus d'informació. Quan algú vulgui consultar, per exemple, si una certa persona treballa en una tal empresa o en una tal filial, a la base de dades es buscaran relacions del tipus:

NOM EMP $\xrightarrow{?}$ EMPRESA

Els programes del sistema de gestió (escrit moltes vegades en llenguatges específics per bases de dades com, l'ALFA o el DDL/DML) ha de saber trobar un camí, si existeix, que porti des de el nom de la persona, via les relacions, fins a un nom de l'empresa, o sigui si

NOM EMPLEAT ⟶ ... ⟶ ... ⟶ EMPRESA

és possible.
També és factible obtenir més d'un d'aquests camins en una consulta a la base de dades

NOM EMPLEAT ⟶ EMPRESA 1-
.
.
NOM EMPLEAT ⟶ EMPRESA 2

si no hi creem relacions molt restrictives.
Per acabar aquest apartat, hem de recordar quelcom important.
Una base de dades no és un fitxer amb la informació col·locada d'aquesta o aquella manera. Consisteix en un **sistema de gestió** (conjunt de programes) que tracten la informació de manera molt determinada i d'un **fitxer** o d'un **arxiu** (conjunt de fitxers) on hi ha aquesta informació.

Sentències de gestió de dades

En programes anteriors hem observat que, per introduir dades i informacions a l'ordinador, hem utilitzat la instrucció INPUT. Imaginem-nos, però, que s'ha de fer una mateixa operació en diferents llocs del programa: resultaria molt avorrit! I, pitjor encara si les dades que s'han d'introduir sempre són les mateixes. Bé; el BASIC té un conjunt d'instruccions que ens faciliten aquesta feina, són: **DATA, READ i RESTORE.**

La DATA diu a l'ordinador que tot el que hi ha darrera seu, fins a la següent instrucció, són dades a emprar dins el programa. Aquestes dades s'han de col·locar dins de variables quan arribi el moment d'usar-les. La instrucció que s'encarregarà de fer-ho és la READ. Les dades es col·loquen dins la DATA, l'una darrera l'altra; i cada cop que s'executa un READ, l'ordinador recorda quina posició ocupa la dada llegida. Així, en el

proper READ, la lectura s'efectuarà sobre la dada que hi hagi a la dreta d'aquesta: és com si, dins l'ordinador, hi hagués una fletxa apuntant la dada que toca llegir.
Els formats de les sentències que inclouen el READ i el DATA són:

DATA dada 1, dada 2,...., dada N
READ nom de variable

Un exemple senzill de l'ús d'aquestes sentències és el següent programa:

```
10 FOR 1 = 1 TO 3
20 READ X
30 PRINT X
40 NEXT I
50 END
60 DATA 50, 3.5, 300
```

En aquest exemple, la primera vegada que passa el programa per la línia 20, el valor de X serà 50: el segon, 3.5; i el tercer i últim, 300. En el cas que el bucle fos de 4 passades, quan arribaria per quarta vegada a la línia 20 i llegís una nova variable, com que a la DATA no n'hi ha més, ens donaria un missatge d'error (OUT OF DATA ERROR).
Les dades de dins un DATA tant poden ésser numèriques com alfanumèriques, i sempre han d'anar separades per una coma.
Normalment, els DATA es col·loquen a la fi dels programes. Hi ha ordinadors que, si un **DATA** no és al final del programa, donen error.
Una altra forma de llegir les dades: en lloc de fer un bucle com el d'abans, es pot posar:

```
10 READ X,Y,Z
20 PRINT X,Y,Z
30 END
40 DATA 50, 3.5, 300
```

el resultat és el mateix.
Cal tenir en compte que les variables usades en el READ s'han de correspondre amb el tipus de dada que els toca llegir.

Per exemple:

```
10 FOR I = 1 TO 3
20 READ X$,Y$,Z
30 PRINT X$;" ";Y$;" TE ";Z;" ANYS"
40 NEXT I
50 END
60 DATA JOSEP, ROIG, 25, JOAN,
CANALS, 15, JAUME, FOLCH, 18
```

La primera pregunta que hom es fa, una vegada explicat tot això, és si, un cop efectuada la lectura de l'última dada d'un DATA, haurem de tornar a definir-nos una altra línia amb el DATA corresponent, per tal de poder llegir un altre cop la mateixa cosa.
En realitat, aquest problema es limita a trobar una ordre que digui a l'ordinador que posi la "fletxa" apuntant a la primera dada; i això, en BASIC, s'aconsegueix via la instrucció **RESTORE.** Es a dir, una vegada executada la instrucció RESTORE, quan fem un READ tornarem a llegir la primera variable que contingui la DATA.

```
10 FOR I = TO 3
20 READ X$,Y$,Z
30 PRINT X$;" ";Y$; TE ";Z;" ANYS"
40 NEXT I
 "
 "
 "
100 RESTORE
110 GOTO 10
160 DATA JOSEP, ROIG, 25, JOAN,
CANALS, 15, JAUME, FOLCH, 18
```

És del tot normal que totes les dades que siguin necessàries per a l'execució d'un programa no hi càpiguen en una línia, i s'hagin de distribuir dins diversos DATA. Per tal de situar la lectura del READ en una determinada línia de dades del programa, allò que podem fer és escriure la instrucció RESTORE, seguida del nombre de línia on hi ha les dades desitjades

RESTORE nombre línia

Altres programes

```
1 REM*******************************************************
2 REM********* SIMULADOR D'UN CANAL AMB SOROLLS **********
3 REM****************Versio  Commodore**********************
4 REM    AUTOR : MARTI GRIERA i FISA
5 PRINT "<CLR>"
20 PRINT "ENTRA EL MISSATGE A TRANSMETRE:";
25 INPUT A$
30 DIM A(6)
35 LET C=0
40 PRINT
50 INPUT "ENTRA EL % DE PROBABILITAT D'ERROR PER BIT"; ER
52 PRINT
55 IF ER>100 OR ER<0 THEN
   PRINT: PRINT "ERROR": GOTO 50
60 LET AL=1-ER/100
62 PRINT "MISSATGE REBUT:"
65 FOR I=1 TO LEN (A$)
70 LET CO=ASC(MID$(A$,I,1))
80 LET D=CO
90 GOSUB 200
100 PRINT CHR$(A);
110 IF CO<>A THEN LET C=C+1
115 NEXT I
120 GOTO 320
200 FOR J=0 TO 6
210 LET B(J)=D-(2*INT(D/2))
220 LET D=INT(D/2)
230 IF RND(1)>AL THEN LET B(J)=1-B(J)
240 NEXT J
245 LET A=0
250 IF B(6)=0 AND B(5)=0 THEN LET B(5)=1
260 FOR L=0 TO 6
270 LET A=A+2^L*B(L)
280 NEXT L
290 RETURN
320 LET TPC=C/LEN(A$)*100
325 PRINT: PRINT
330 PRINT "EL";TPC;"% DELS CARACTERS REBUTS SON ERRONIS"
```

```
1 REM **** ANTIAERI....Autor:Ferran Andreu Bustamante ******
4 REM **** Presentacio ******
5 PRINT "<clr>"
6 PRINT "<8 crsrd><ctrl 9> PITJA LES TECLES '<' I '>' PER A MOURE'T
7 PRINT "<ctrl 9> LA TECLA 'Z' ES PER A DISPARAR                "
8 PRINT "<ctrl 9> TENS TRENTA PROJECTILS.INTENTA FER LA MAXI-"
9 PRINT "<ctrl 9> MA PUNTUACIO !!!"
10 PRINT "<10 crsrd> PITJA UNA TECLA PER A JUGAR"
11 GET T$:IF T$="" THEN 11
29 REM *****INICIALITZA VARIABLES****
30 Q$=" "
40 A$="<30 commo +>"
50 X=10:I=0:B=0:SC=0
99 REM ****IMPRIMEIX LA PANTALLA DE JOC ****
100 PRINT "<clr>"
101 PRINT "<16 crsrd>"A$:PRINT <home>
102 PRINT "<22 crsrd><ctrl 9> PUNTUACIO:<ctrl 0>";SC:PRINT <home>
105 IF I>38 THEN PRINT "<home>":PRINTTAB(I) " ":I=0
110 PRINT "<home>":PRINTTAB (I) " >"
130 PRINT "<home>":PRINTTAB (X) "<13crsrd> <commo E> "
139 REM *****MIRA QUINA TACLA HA ESTAT PITJADA******
140 GET T$:IF T$="" THEN I=I+1:GOTO 105
150 IF T$="," THEN X=X-1:IFX<0 THEN X=0
160 IF T$="." THEN X=X+1:IFX>36 THEN X=36
165 IF T$="Z" THEN B=B+1:GOSUB 500
170 GOTO 110
499 REM *****SUBRUTINA DE DISPARAR *****
500 FOR K=1 TO 15:PRINTTAB(X+2) "<2 crsru><crsrl><shift
->":PRINTTAB(X+2)"<crsru><crsrl> "
503 NEXT K:PRINTTAB (X+2) "<2 crsru><crsrl> "
505 IF I=X THEN PRINT "<home>":PRINTTAB(X=2) "X":SC=SC+1:I=I+1:FOR T=1
TO100:NEXT
506 W$=W$+Q$: PRINT <home>:PRINT "<16 crsrd><ctrl 9>"W$:IF B=30 THEN GOSUB 700
507 PRINT <home><23 crsrd><ctrl 9> PUNTUACIO:<ctrl 0>";SC:PRINT "<home>"
510 RETURN
699 REM ******SUBRUTINA DEL FINAL DE JOC *****
700 FOR G=1 TO 1000:NEXT:PRINT <clr>:PRINT "<10 crsrd><6 crsrr>JOC ACABAT"
710 PRINT "<5 crsrr> <ctrl 9> PUNTUACIO =<ctrl 0>";SC
720 PRINT "<5 crsrd> VOLS TORNAR A JUGAR? (S/N)"
730 GET T$:IF T$<>"S" AND T$<>"N" THEN 730
740 IF T$="S" THEN 30
750 PRINT "<clr> <10 crsrd> ADEU!!!!!"
```

Versió per Commodore

```
1 REM************************************************
2 REM****** *    ENCRIPCIO  D'EN  CESAR       ******
3 REM******  AMB  VECTOR  DE  PERMUTACIONS  ******
4 REM**************Versio Commodore*****************
5 PRINT "<CLR>"
10 INPUT "ENTRA UNA CLAU";A$
20 INPUT "ENTRA EL MISSATGE";B$
25 FOR I=1 TO LEN(B$)
30 LET J=I-INT((I-1)/LEN(A$))*LEN(A$)
40 LET N=ASC(MID$(B$,I,1))
50 IF N=32 THEN LET N=91: LET PAL=2: GOTO 100
60 IF N>47 AND N<48 THEN LET PAL=0: GOTO 100
70 IF N>64 AND N<91 THEN LET PAL=1: GOTO 100
80 PRINT: PRINT "ERROR"
90 GOTO 20
100 LET M=N-VAL(MID$(A$,I,1))
110 IF M<48 AND PAL=0 THEN LET M=M+10: GOTO 150
120 IF M<65 AND PAL=1 THEN LET M=M+27
130 IF M=91 THEN LET M=32
150 PRINT CHR$(M);
160 NEXT I
165 PRINT
168 LET B$="": LET Z$="": LET Y=""
170 INPUT "VOLS ENTRAR UN ALTRE MISSATGE (S/N)?";Z$
180 INPUT "VOL CANVIAR LA CLAU (S/N)?";Y$
190 IF Z$="S" AND Y$="S" THEN GOTO 5
200 IF Z$="S" THEN GOTO 20
```

```
5 REM ****************************
6 REM **** CRIPTOSISTEMA RSA ****
7 REM ****************************
8 REM
9 REM **** MENU ****
10 PRINT "<CLR>"
20 PRINT "SIMULADOR RSA AMB P=3 I Q=11"
30 LET LET P=3 : LET Q=11
40 LET N=P*Q : LET M=(P-1)*(Q-1)
50 PRINT : INPUT "ENTRA D: ";D
60 GOSUB 1000
70 PRINT : PRINT "CLAU PUBLICA = (E,N): ";E,N
80 PRINT : PRINT "CLAU PRIVADA = (D,N): ";D,N
90 INPUT "APRETA <ENTER>";Z$
100 PRINT "<CLR>"
101 INPUT "VOLS ENCRIPTAR (1) O DESENCRIPTAR (2)";T$
102 IF T$="1" THEN GOTO 110
103 LET ED=E
104 LET E=D
105 LET D=ED
110 PRINT "ENTRA EL MISSATGE (ENTRE COMETES):"
120 INPUT R$
130 GOSUB 2000
140 PRINT : PRINT "ASSIGNACIO NUMERICA: "
150 PRINT A$
160 LET M$=A$
170 GOSUB 5000
175 IF T$="2" THEN PRINT : PRINT "ASSIGNACIO
    DESENCRIPTADA:" : GOTO 190
180 PRINT : PRINT "ASSIGNACIO ENCRIPTADA:"
190 PRINT A$
200 GOSUB 6000
205 IF T$="2" THEN PRINT : PRINT "MISSATGE
    DESENCRIPTAT:" : GOTO 220
210 PRINT : PRINT "MISSATGE ENCRIPTAT:"
220 PRINT S$
230 INPUT "VOLS FER UNA ALTRA OPERACIO (S/N) ";W$
240 IF W$="S" THEN GOTO 10
250 STOP
1000 REM ****** CALCUL DE L'INVERS ******
1010 FOR V2=1 TO M
1020 LET A2=V2*M
1030 FOR Z2=1 TO M
1040 LET B2=Z2*D
1050 IF B2 > A2+1 THEN GOTO 1080
1060 IF B2 = A2+1 THEN GOTO 1090
1070 NEXT Z2
1080 NEXT V2
1085 PRINT "VALOR DE D ERRONI" : GOTO 50
1090 LET E=Z2
1100 RETURN
2000 REM *** ASSIGNACIO NUMERICA ***
2010 LET A$=""
2020 FOR V4=1 TO LEN(R$)
2030 LET CO=ASC (MID$ (R$,V4,1))
2040 IF CO = 32 THEN LET A$=A$+"28" : GOTO 2100
2050 IF CO = 46 THEN LET A$=A$+"29" : GOTO 2100
2060 IF CO = 44 THEN LET A$=A$+"30" : GOTO 2100
2070 IF CO = 59 THEN LET A$=A$+"31" : GOTO 2100
2080 IF CO = 39 THEN LET A$=A$+"32" : GOTO 2100
2090 IF CO > 64 AND CO <91 THEN LET CO=CO-63 : GOSUB 3000
2100 NEXT V4
2110 RETURN
3000 REM *** CARACTERS ALFABETICS ***
3010 LET H$=STR$(CO-INT(CO/10)*10)
3020 LET I$=STR$(INT(CO/10))
3030 LET A$=A$+MID$(I$,2,1)+MID$(H$,2,1)
```

```
3040 RETURN
4000 REM *** EXPONENCIACIO RAPIDA ***
4010 LET A1=BAS
4020 LET Z1=E
4030 LET CH=1
4040 IF Z1=0 THEN GOTO 4100
4050 LET V1=Z1-INT(Z1/2)*2
4060 IF V1 <> 0 THEN LET Z1=Z1-1 :
     LET CH=CH*A1-INT(CH*A1/N)*N : GOTO 4040
4070 LET Z1=INT(Z1/2)
4080 LET A1=A1*A1-INT(A1*A1/N)*N
4090 GOTO 4050
4100 RETURN
5000 REM *** ENCRIPCIO / DESENCRIPCIO ***
5002 PRINT
5005 LET A$=""
5010 FOR V5=1 TO LEN (M$) STEP 2
5020 LET H$=MID$(M$,V5,2)
5030 LET BAS=VAL(H$)
5040 GOSUB 4000
5050 LET CO=CH : GOSUB 3000
5060 NEXT V5
5070 RETURN
6000 REM *** MISSATGE ENCRIPTAT / DESENCRIPTAT ***
6010 LET S$=""
6020 FOR V7=1 TO LEN(A$) STEP 2
6030 LET C7=VAL(MID$(A$,V7,2))
6040 IF C7=28 THEN LET S$=S$+"" : GOTO 6100
6050 IF C7=29 THEN LET S$=S$+"" : GOTO 6100
6060 IF C7=30 THEN LET S$=S$+"" : GOTO 6100
6070 IF C7=31 THEN LET S$=S$+"" : GOTO 6100
6080 IF C7=32 THEN LET S$=S$+"" : GOTO 6100
6090 IF C7 > 1 AND C7 < 28 THEN LET S$=S$+CHR$(C7+63)
6100 NEXT V7
6110 RETURN
```

```
1 REM ********************************
2 REM *** PROGRAMA DE L'ANTIAERI ***
3 REM ********Versio Spectrum*********
5 CLS
10 PAPER 7 : BORDER 2
20 DIM w$(4,8)
30 LET w$(1) = "*       "
40 LET w$(2) = "**      "
50 LET w$(3) = "***     "
60 LET w$(4) = "****    "
65 LET z=0 : LET f=0 : LET t=0 : LET m=0
70 FOR i = 0 TO 7
80 READ x : POKE USR "a"+i,x
100 READ x : POKE USR "b"+i,x
120 NEXT i
130 DATA 24,126,24,126,60,126,60,126,60,255,
        126,255,126,255,126,0
140 DIM d(32,2)
145 GOSUB 4000
150 FOR k = 1 TO 32
160 LET d(k,1)=21 : LET d(k,2)=k-1
180 PRINT AT 21,k-1; PAPER 4; CHR$(144)
190 NEXT k
200 LET q=0 : LET u=0 : LET s=16 : LET y=0: LET r=0
210 GOSUB 1000
220 LET a=INT(4.9*RND)
230 IF a = 0 THEN GOTO 220
240 LET b=INT(10*RND)+3
250 LET p=0
260 LET p=p+1
270 IF p+a = 33 THEN LET a=a-1 : IF a = 0 THEN
    PRINT AT b,p-1; CHR$(128) : GOTO 220
280 PRINT AT b,p; w$(a,TO a)
290 PRINT AT b,p-1; CHR$(128)
300 IF INKEY$ = "q" AND s > 1 THEN LET f=1 :
    GOSUB 1000
310 IF INKEY$ = "w" AND s < 30 THEN LET f=-1 :
    GOSUB 1000
320 IF INKEY$ = " " AND u < 32 THEN LET u=u+1:
    LET d(u,1)=19 : LET d(u,2)=s :
    PRINT AT 21,u-1; CHR$(128)
330 GOSUB 2000
335 IF m = 1 THEN LET m=0 : GOTO 240
340 GOTO 260
995 REM
1000 REM subrutina moviment del disparador
1005 REM
1010 PRINT AT 20,s; CHR$(128)
1020 LET s=s+2*f
1030 PRINT AT 20,s; CHR$(145)
1040 RETURN
2000 REM
2005 REM subrutina dels trets
2010 REM
2020 FOR l = r+q+1 TO u
2030 IF d(l,1)=99 THEN GOTO 2200
2040 IF SCREEN$(d(l,1),d(l,2)) = "*" THEN GOSUB 3000 :
     LET m=1 : GOTO 2200
2050 PRINT AT d(l,1),d(l,2); CHR$(128)
2060 LET d(l,1)=d(l,1)-1
2070 IF d(l,1) = -1 THEN LET d(l,1)=99 : LET r=r+1 :
     IF r+y <> 32 THEN GOTO 2200
2075 IF r+y = 32 THEN GOTO 5000
2080 PRINT AT d(l,1),d(l,2); CHR$(144)
2200 NEXT l
2210 RETURN
3000 REM
```

```
3010 REM subrutina d'impacte
3020 REM
3030 BEEP 2,-10
3040 PRINT AT d(l,1),d(l,2); CHR$(128)
3050 PRINT AT b,p; w$(a,9-a TO 8)
3060 LET d(l,1)=99
3070 LET y=y+1
3080 IF r+y = 32 THEN GOTO 5000
3084 IF l = 1 THEN GOTO 3090
3085 IF d(l-1,1) <> 99 THEN GOTO 3090
3087 LET y=y-1
3090 LET t=t+5-a
3100 RETURN
4000 REM
4010 REM subrutina instruccions del joc
4020 REM
4030 CLS
4040 PRINT AT 9,13; FLASH 1; INK 0; "JOC DE"
4045 PRINT AT 13,11; FLASH 1; INK 0; "L'ANTIAERI"
4050 PRINT AT 21,9; INK 1; "PITGEU <ENTER>"
4060 INPUT s$
4070 CLS
4080 PRINT AT 6,6; "DISPOSEU DE 32 BALES"
4090 PRINT AT 9,2; "PER DISPARAR PITGEU <SPACE>"
4100 PRINT AT 12,2;"PER MOURE L'ANTIAERI PITGEU"
4110 PRINT AT 15,10; "Q: A L'ESQUERRA"
4120 PRINT AT 17,10; "W: A LA DRETA"
4130 PRINT AT 21,9; INK 1; "PITGEU <ENTER>"
4140 INPUT s$
4150 CLS
4160 PRINT AT 6,2; "ELS OBJECTIUS I ELS PUNTS SON"
4170 PRINT AT 9,10; "*      : 4"
4180 PRINT AT 11,10; "**     : 3"
4190 PRINT AT 13,10; "***    : 2"
4195 PRINT AT 15,10; "****   : 1"
4196 PRINT AT 21,9; INK 1; "PITGEU <ENTER>" :
     INPUT s$
4197 CLS
4200 PRINT AT 5,4; INK 0; "SI DESPRES DE DISPARAR"
4210 PRINT AT 7,7; INK 0; "LES 32 BALES"
4220 PRINT AT 9,4; INK 0; "HEU ACONSEGUIT MES DE"
4225 PRINT AT 11,9; INK 0; FLASH 1; "25 PUNTS"
4230 PRINT AT 13,4; INK 0; "PODREU CONTINUAR JUGANT"
4240 PRINT AT 21,9; INK 1; "PITGEU <ENTER>" :
     INPUT s$
4250 CLS
4300 RETURN
5000 REM
5010 REM subrutina de puntuacio
5020 REM
5025 CLS
5030 LET z=z+t
5040 PRINT AT 8,10; INK 0; FLASH 1; "TOTAL DE PUNTS"
5050 PRINT AT 12,13; INK 0; z
5055 IF t < 26 THEN PRINT AT 6,10; "FI DE LA PARTIDA"
5060 LET t=0
5065 PRINT AT 15,8; INK 0; FLASH 1;
     "PODEU CONTINUAR JUGANT"
5070 PRINT AT 21,13; INK 4; "PITGEU <ENTER>" :
     INPUT s$
5080 CLS
5090 GOTO 150
```

```
**************************************************
********* PROGRAMA AGENDA TELEFONICA ************
***************Versió Commodore*******************

10 REM ****AUTOR: FERRAN ANDREU BUSTAMENTE ******
50 DIM AG$(50,4)
60 FOR S = 1 TO 50
61 FOR V = 1 TO 4
62 AG$(S,V) = "#"
64 NEXT V
65 NEXT S
100 PRINT "<CLR>"
110 POKE 53280,5
112 POKE 53281,1
114 PRINT "<CTRL-GRN>"
120 PRINT "<8CRSRD><12CRSRR>AGENDA TELEFONICA"
130 FOR T = 1 TO 2000 : NEXT T
150 ************** MENU D'OPCIONS *****************
160 PRINT "<CLR>"
170 PRINT "<1CRSRD><16CRSRR>OPCIONS"
171 PRINT "<16CRSRR>--------"
180 PRINT "<3CRSRD><CRTL-RVSON>1<CRTL-RVSOFF> VEURE
    UN REGISTRE"
190 PRINT "<1CRSRD><CTRL-RVSON>2<CTRL-RVSOFF> AFEGIR
    UN REGISTRE"
200 PRINT "<1CRSRD><CTRL-RVSON>3<CTRL-RVSOFF>
    ESBORRAR UN REGISTRE"
210 PRINT "<1CRSRD><CTRL-RVSON>4<CTRL-RVSOFF>
    CARREGAR UN FITXER"
215 PRINT "<1CRSRD><CTRL-RVSON>5<CTRL-RVSOFF> SALVAR
    UN FITXER"
217 PRINT "<1CRSRD><CTRL-RVSON>6<CTRL-RVSOFF> CANVIAR
    UN REGISTRE"
220 PRINT "<1CRSRD><CTRL-RVSON>7<CTRL-RVSOFF>
    DESPEDIR EL PROGRAMA"
230 PRINT "<3CRSRD><CTRL-RVSON>ENTRA LA TEVA
    OPCIO<CTRL-RVSOFF>"
250 REM ********* AGAFA EL NUMERO DE LA OPCIO *******
260 GET OP$ : IF OP$="" THEN GOTO 260
265 OP = VAL(OP$)
270 ON OP GOSUB 1000,2000,3000,4000,5000,6000,7000
280 PRINT "<CLR>"
282 PRINT "<2SPC><15CRSRD><1CTRL-RVSON>RESPOSTA
    INCORRECTA, TORNA-HO A PROVAR"
290 FOR T = 1 TO 1500 : NEXT T
295 GOTO 160
1000 REM ********** VEURE UN REGISTRE *****************
1010 PRINT "<CLR>"
1020 PRINT "<1CRSRD><3SPC><CTRL-RVSON>OPCIO 1: VEURE
     UN REGISTRE<CTRL-RVSOFF>"
1030 PRINT "<5CRSRD>"
1035 PRINT "<4CRSRD>NUMERO DEL REGISTRE QUE VOLS VEURE ";
1036 INPUT N
1040 PRINT "<CLR>"
1050 PRINT "<3CRSRD>NOM: ";AG$(N,1)
1052 PRINT "<3CRSRD>DIRECCIO: ";AG$(N,2)
1054 PRINT "<3CRSRD>POBLACIO: ";AG$(N,3)
1056 PRINT "<3CRSRD>TELEFON: ";AG$(N,4)
1058 PRINT "<5CRSRD><2CRSRR>PITJA LA BARRA DE ESPAI PER
     CONTINUAR"
1060 GET T$: IF T$ <> " " THEN GOTO 1060
1070 GOTO 160
2000 REM *********  AFEGIR UN REGISTRE *****************
2010 PRINT "<CLR>"
2012 PRINT "<1CRSRD><3CRSRR><CTRL-RVSON> OPCIO 2:
     AFEGIR UN REGISTRE<CTRL-RVSOFF>"
2020 FOR K = 1 TO 50
```

```
2024 IF AG$(K,1)= "#" THEN 2040
2030 NEXT K
2040 IF K = 50 THEN PRINT "<3CRSRD><CTRL-RVSON>ATENCIO!
     AQUEST FITXER ES PLE"
2041 FOR G = 1 TO 1000 : NEXT G : GOTO 160
2042 PRINT "<2CRSRD>TENS OCUPAT FINS EL REGISTRE N.º";K-1
2050 PRINT "<2CRSRD>ENTRADA NUMERO ";K
2060 INPUT "NOM: ";AG$(K,1)
2070 INPUT "DIRECCIO: ";AG$(K,2)
2080 INPUT "POBLACIO: ";AG$(K,3)
2090 INPUT "TELEFON: ";AG$(K,4)
2100 PRINT "<2CSRD>DADES CORRECTES?(S/N)"
2110 GET T$: IF T$ <> "S" AND T$ <> "N" THEN GOTO 2110
2120 IF T$ = "N" THEN PRINT "<CLR>": GOTO 2040
2130 PRINT "UNA ALTRE ENTRADA (S/N)"
2140 GET T$: IF T$ <> "S" AND T$ <> "N" THEN GOTO 2140
2150 IF T$ = "N" THEN GOTO 160
2160 PRINT "<CLR>" : K = K + 1 : GOTO 2040
2200 GOTO 2200
3000 REM ********* ESBORRAR UN REGISTRE ******************
3010 PRINT "<CLR>"
3012 PRINT "<2CSRD><CTRL-RVSON>OPCIO 3: ESBORRAR UN
     REGISTRE<CTRL-RVSOFF>"
3020 PRINT "<5CSRD>"
3025 INPUT "NUMERO DEL REGISTRE QUE VOLS ESBORRAR";E
3030 FOR H = E TO 49
3032 FOR J = 1 TO 4
3034 AG$(H,J) = AG$(H+1,J)
3036 NEXT J
3038 NEXT H
3040 PRINT "<4CRSRD>REGISTRE ";E;" ESBORRAT"
3050 PRINT "<3CRSRD>PITJA LA BARRA D'ESPAI PER CONTINUAR"
3060 GET T$ : IF T$ <> " " THEN GOTO 3060
3070 GOTO 160
4000 REM ******** CARREGAR UN FITXER *********************
4010 PRINT "<CLR>"
4015 PRINT "<1CRSRD><CTRL-RVSON>OPCIO 4: CARREGAR UN
     FITXER<CTRL-RVSOFF>"
4020 PRINT "<3CRSRD>NOM DEL FITXER QUE VOLS CARREGAR";
4025 INPUT FT$
4030 PRINT "<1CRSRD> COLOCA LA CINTA EN POSICIO, PITJA LA
     TECLA 'PLAY' I DESPRES LA BARRA D'ESPAI"
4040 GET T$ : IF T$ <> " " THEN 4040
4050 OPEN 1,1,0,FT$
4060 FOR V = 1 TO 50
4065 FOR D = 1 TO 4
4070 INPUT #1,AG$(V,D)
4080 NEXT D
4085 NEXT V
4090 CLOSE 1
4200 GOTO 160
5000 REM ******** SALVAR UN FITXER **********************
5005 PRINT "<CLR>"
5010 PRINT "<1CRSRD><CTRL-RVSON>OPCIO 5: SALVAR UN
     FITXER<CTRL-RVSOFF>
5020 PRINT "<3CRSRD>NOM DEL FITXER QUE VOLS SALVAR"
5025 INPUT NF$
5030 PRINT "<2CRSRD>COLOCA LA CINTA EN POSICIO, PITJA LES
     TECLES 'RECORD' I 'PLAY' A L'HORA I DESPRES PITJA LA
     BARRA D'ESPAI"
5060 GET T$ : IF T$ <> " " THEN GOTO 5060
5070 OPEN 1,1,1,NF$
5080 FOR P = 1 TO 50
5085 FOR L = 1 TO 4
5090 PRINT#1,AG$(P,L)
5100 NEXT L
5110 NEXT P
```

```
5120 CLOSE 1
5200 GOTO 160
6000 REM ********* CANVIAR UN REGISTRE ***************
6005 PRINT "<CLR>"
6010 PRINT "<1CRSRD><CTRL-RVSON>OPCIO 6: CANVIAR UN
     REGISTRE<CTRL-RVSOFF>"
6030 PRINT "<7CRSRD>NUMERO DEL REGISTRE QUE VOLS
     CANVIAR"; : INPUT R
6040 PRINT "<CLR>"
6045 PRINT "<3CRSRD><CTRL-RVSON>NOVA ENTRADA
     <CTRL-RVSOFF>"
6050 PRINT "<5CRSRD>"
6060 INPUT "NOM: ";AG$(R,1)
6070 INPUT "DIRECCIO: ";AG$(R,2)
6080 INPUT "POBLACIO: ";AG$(R,3)
6090 IMPUT "TELEFON: ";AG$(R,4)
6100 PRINT "<3CRSRD>DADES CORRECTES (S/N)"
6110 GET T$ : IF T$ <> "S" AND T$ <> "N" THEN
     GOTO 6110
6120 IF T$ = "N" THEN GOTO 6040
6200 GOTO 160
7000 REM ********** ACABAR EL PROGRAMA **************
7010 PRINT "<CLR>"
7015 PRINT "<1CRSRD><CTRL-RVSON>OPCIO 7: ACABAR EL
     PROGRAMA<CTRL-RVSOFF>"
7020 PRINT "<5CRSRD>ABANS D'ACABAR EL PROGRAMA RECORDEU"
7030 PRINT "QUE HEU DE GRAVAR ELS REGISTRES, EN ALTRE CAS,"
7040 PRINT "QUAN DESCONECTEU EL MICRO S'ESBORRARAN"
7050 PRINT "<3CRSRD>VOLEU ACABAR (S/N)"
7060 GET T$ : IF T$ <> "S" AND T$ <> "N" THEN
     GOTO 7060
7070 IF T$ = "N" THEN GOTO 160
7080 PRINT "<1CRSRD>MOLT BE, ADEU" : END
```

Taula d'equivalències

	ZX Spectrum	Dragon	IBM-PC	TRS-80	COMMODORE-64
Esborrar pantalla	CLS	CLS	CLS	CLS	CLS
Color de pantalla	PAPER num: CLS	CLS (num)	COLOR num: CLS	–	POKE 53280, num
Condició	IF cond. THEN	IF cond. THEN (... ELSE)	IF cond. THEN (... ELSE)	IF cond. THEN (... ELSE)	IF cond. THEN
Manipular una cadena	nom$ (x TO y)	MID$ (nom$, x, y)	MID$ (nom$,x,y)	MID$ (nom$, x, y)	MID$ (nom$, x, y)
Exploració del teclat	INKEY$	INKEY$	INKEY$	INKEY$	GET nom$
Generar un n. aleatori entre 0 i 1	RND	RND (0)	RND (1)	RND (0)	RND (1)
Codi ASCII d'un caràcter	CODE (car)	ASC (car)	ASC (car)	ASC (car)	ASC (car)
Mirar una posició de pantalla	SCREEN $ (X,Y)	POINT (x, y)	SCREEN (x, y, 0)	–	PEEK (1024 + x * 0 + y)
Posicionament a pantalla	PRINT AT x, y	PRINT & 32 x + y	LOCATE x, y	PRINT & 64 * x + y)	A$= "<Pitjar cursor avall 19 cops>" B$= "<Pitjar cursor a la dreta 39 cops>" PRINT "<HOME>" + LEFT$ (A$, x) + LEFT$ (B$, y)

Els autors volen agraïr al Centre de Càlcul de la Universitat Politècnica de Barcelona, tot el material tècnic i gràfic que de manera ben desinteressada ens han facilitat.

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

CURS DE BASIC

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, pica l'start» és una iniciativa conjunta de Caixa de Pensions «la Caixa» i TV3

275 Ptes.

8 480002 006859 00003