EL MUNDO DE BLOQUES

Uno de los temas más espinosos de la IA, todavía no resuelto, es lo que los investigadores en este terreno llaman «proceso del lenguaje natural» o «comprensión del lenguaje natural», es decir, conseguir que los ordenadores comprendan el lenguaje humano.

Creo que todos podemos imaginar la complejidad que este propósito entraña, ya que nuestros idiomas están llenos de ambigüedades, frases hechas y coloquiales, etc, que nosotros mismos comprendemos sólo gracias al enorme banco de memoria que tenemos en nuestro cerebro, y a algo indefinible, por ahora al menos, que llamamos inteligencia.

Los computadores no tienen de eso, pero, como ya hemos aprendido en el curso de IA, pueden simularlo de manera bastante convincente (recuerden el programa Eliza).

En el caso del lenguaje, éste sólo puede ser comprendido dentro de un entorno predecible, esto es, que posea una serie de posibilidades o «movimientos» fijos.

Hubo un hombre, Terry Winnograd, que marcó un hito hasta ahora insuperado con un programa, escrito en «Planner», en la comprensión del lenguaje natural; podía mantener conversaciones con su «obra» de una increíble complejidad. Nosotros, con algunas lógicas limitaciones, hemos escrito en Basic una versión del Mundo de Bloques, y, lo que es mucho más decisivo, hemos explicado con todo detalle las técnicas que se usan para que una máquina comprenda nuestra lengua y actúe en consecuencia.

Sin más preámbulos, AMSTRAD Semanal presenta... El Mundo de Bloques.

Por José Antonio Morueco González

Inteligencia ARTIFICIAL

La inteligencia artificial es un área de la informática muy importante, que nos permite atacar tareas que no podían ser realizadas hasta hace pocos años. La inteligencia artificial tiene unas técnicas propias especiales y muy variadas según sea el objetivo a conseguir. Estos objetivos se dividen en tres grandes grupos:

— Los sistemas expertos.
— Los procesadores de lenguaje natural.
— La robótica.

Tanto la parte de los sistemas expertos como la de la robótica no tienen mucho que ver con el programa aquí presentado, aunque sí es bueno saber un poco sobre su objetivo primordial: ayudar al hombre a resolver situaciones y a tomar decisiones basándose en el conocimiento «inteligente» que tiene el ordenador en el área del saber en la que estamos. Este conocimiento inteligente, su presentación y su gestión, es el que distingue la inteligencia artificial del resto de las técnicas informáticas.

Lenguaje natural

Por otra parte, que nos atañe más en este artículo, está el procesamiento del lenguaje natural. Para una persona que no esté familizarizada con el mundo de la programación, este problema puede parecerle no muy compilado, porque a él le cuesta muy poco comprender lo que se dice en un escrito, o lo que se dice por la radio. Pero el problema de hacerle entender al ordenador nuestro lenguaje natural, aun teniendo ciertas estructuras, es muy poco sumiso a unas reglas de construcción severas. Esta flexibilidad del lenguaje humano es la que complica las cosas al ordenador.

Los lenguajes con los que el hombre se comunica con el ordenador son de unas estructuras muy severas, y en los que hay pocas palabras distintas, aunque desde fuera pueda parecer todo lo contrario. Cuando una persona hace un programa en BASIC, por ejemplo, tiene que andarse con mucho cuidado de utilizar exactamente las palabras reservadas que hagan falta; en caso contrario tendrá muchos errores. Cuando hablamos con un amigo no es necesaria tanta precisión, y de hecho no pensamos qué palabra usar en esta frase, o qué estructura sintáctica en aquella otra: simplemente decimos lo que se nos ocurra.

Todo esto nos conduce a reflexionar sobre cómo hacer que podamos conseguir que el ordenador comprenda castellano directamente. Para ello es necesario hacer un programa que sirva de interfase entre nuestras palabras y la acción que se quiera realizar. Esta interfase es la que traduce, a código más comprensible para el ordenador, dichas palabras.

Hasta ahora, no se han hecho procesadores de lenguaje natural que puedan considerarse completos, sino que se tiene un conjunto de palabras limitado o palabras claves, que son las que entiende el programa y mediante las cuales intenta comprender las frases dadas, como se verá luego más prácticamente con el programa que se presenta.

Claves del procesamiento del lenguaje natural

El procesamiento de lenguaje natural tiene varios puntos fundamentales:

— El diccionario, o conjunto de palabras que son conocidas y pueden traducirse.

— Las reglas sintácticas y estructuras del lenguaje natural que son analizadas por el procesador.

— Las herramientas que pueden usarse, etc.

Con respecto a las herramientas, es muy importante hacer notar que el BASIC no es precisamente muy adecuado para la inteligencia artificial, lo que produce en algunos casos ciertas limitaciones; se están desarrollando lenguajes, como LISP o PROLOG, más adecuados para estas cuestiones, y permiten unas representaciones de las estructuras más satisfactorias.

De todas formas esto no es un obstáculo insalvable; de hecho, pienso que es interesante ver, en un lenguaje de programación de gran extensión, algunas de estas técnicas, aunque la forma de representación no sea la más óptima.

En el caso que nos ocupa, tenemos un conjunto limitado de palabras que realmente traduce el programa, pero a la hora de usarse parece que hay más, debido a que puden usarse muchas más palabras, aunque sean ignoradas en la comprensión de la frase. Pasemos pues al programa.

Un procesador de lenguaje natural

¿Qué se puede hacer con el programa?

Antes de ver cómo funciona, veamos un poco cómo se usa. Cuando des al «run», para ejecutar el programa, te aparecerán varios cuadros y triángulos de distintos colores y tamaños, y debajo de éstos, unas líneas de texto. Ahí es donde debes ir dándole las órdenes al programa. Esta órdenes consiten en cuál de las figuras quieres que se mueva, y dónde. Por ejemplo puedes escribir:

pon el triángulo pequeño rojo encima del cuadro grande azul

Como puedes observar, tienes también unos números debajo de las figuras, del 1 al 12, que también pueden usarse; por ejemplo, después del movimiento anterior, la posición donde estaba el triángulo pequeño rojo, la número 10, queda vacía. Por ello, puedes ahora llevar allí otra figura; por ejemplo puedes dar la orden:

lleva el cuadro pequeño verde a la posición 10

De esta forma puedes ir realizando los movimientos que desees de una forma natural. Así, por ejemplo, el primer movimiento lo puedes expresar de muchas maneras diferentes en castellano, igualmente comprensible para el programa; algunas de estas formas podrían ser:

pon sobre el cuadro azul grande el triángulo rojo pequeño apila el triángulo rojo pequeño en el cuadro azul grande

sitúa debajo del triángulo pequeño rojo el cuadro grande azul

Como ves, hay bastantes variaciones ideomáticas para expresar una cosa tan sencilla como ésta. Este es uno de los problemas de traducir por ordenador el lenguaje natural, como decíamos en la introducción. Debido a ello hay ciertas limitaciones, pero bastante razonables. Por ejemplo, no intentes comprender frases que pueden ser ambiguas en su significado, ni frases tratadas de forma poética. — como podrían ser «pon el gran cuadro rojo sobre el triángulo cielo»— cosa que complicaría más todavía la lógica del programa, ya de por sí complicada sólo con un lenguaje normal.

Las leyes semánticas del programa «bloques»

Aparte de las cuestiones sintácticas, también he introducido algunas reglas semántias sencillas como son:

— No puedes poner figuras del mismo color apiladas juntas.

— No poner nada encima de un triángulo, porque, visto con naturalidad, se caería.

— Algo grande no pued ir encima de algo más pequeño, cosa también razonable, etc.

Estas reglas tienen por misión que se vea cómo se entiende la frase y, o bien se lleva a cabo el movimiento indicado, si es posible, o bien se indica el porqué no se puede hacer.

Para que lo veas, haz todas las combinaciones posibles de movimientos, e irás viendo que hay muchos posibles.

Por último notar que para acabar basta con decirle «adiós».

Visto ya cómo usarlo entremos un poco en su construcción. La idea general consiste en adivinar qué palabras clave están en la frase del usuario y cuál es la estructura de la frase.

Representación Del conocimiento

Cuando empiezas a pensar en cómo construir un programa así, lo primero que hay que limitar es el número de palabras y estructuras que van a ser comprendidas. En nuestro caso tenemos 5 estructuras generales, las cuales son bastante razonables.

Estas estructuras son:

1. Poner el objeto A encima del objeto B.
2. Poner el objeto B debajo del objeto A.
3. Poner encima del objeto B el objeto A.
4. Poner debajo del objeto A el objeto B.
5. Poner el objeto A en la posición n (con n entre 1 y 12).

donde el objeto puede ser un triángulo o un cuadrado, de un cierto tamaño (grande o pequeño), y un cierto color (verde, azul o rojo).

Cada objeto, junto con sus atributos, se considerará como un todo a la hora de analizar la frase. También hay que notar que llamo objeto A al que está encima, y objeto B al que va a estar debajo; esto se usará al hacer el programa.

En el programa hay varias partes, que iremos viendo:

— El programa principal, es la mayoría del programa, y donde se va analizando la estructura de la frase y, una vez hecho esto, se llevan a cabo las acciones semánticas (movimiento a hacer o mensaje de por qué no hacer el moviento).

— Las subrutinas 2500 y 3000, cuya misión es dibujar, o borrar, cuadrados y triángulos, respectivamente.

— La subrutina 2000 es clave y tiene por misión decir en qué posición está una palabra (pal$) en una frase (fr$), si está.

Vayamos al programa principal línea a línea. Para comprenderlo, veamos primero las variables más importantes y su función:

— px(i),py(i) —con i entre 1 y 12— tienen la posición horizontal y vertical de los objetos. Estos objetos están numerados del 1 al 12 y son:

1 = cuadrado azul grande.
2 = cuadrado azul pequeño.
3 = cuadrado rojo grande.
4 = cuadrado rojo pequeño.
5 = cuadrado verde grande.
6 = cuadrado verde pequeño.
7 = triángulo azul grande.
8 = triángulo azul pequeño.
9 = triángulo rojo grande.
10 = triángulo rojo pequeño.
11 = triángulo verde grande.
12 = triángulo verde pequeño.

En las primeras líneas se inicializan estas posiciones para la configuración inicial. También, asociada a éstas, está p(i), que mantiene siempre las posiciones iniciales de px(i), y sirve para los números que aparecen en pantalla del 1 al 12, y para cuando se indique que se ha de poner algo en dichas posiciones.

— db(i), indica qué figura está debajo de la i-esima; si no hay ninguna valdrá 0. Por ejemplo, si el triángulo rojo pequeño está encima del cuadrado azul grande entonces db(10) = 1.

— oc(j)— con j entre 1 y 6 —indica si el cuadrado número j está cubierto— 0c(j) = —1— o si está descubierto —oc(i) = 0.

— La variable p va a usarse para posiciones en frases donde está cierta palabra, según se calcula en la subrutina 2000. Además hay variables pr, pa y pb que indican posiciones en la frase de la preposición, el objeto A y el objeto B. Está también la varible n, que indica la posición de la frase donde se ha de empezar a buscar la palabra.

— na y nb contienen al final los números asociados a los objetos A y B respectivamente —números del 1 al 12.

— Otra variable importante es hn que indica si la estructura de la frase es la quinta, es decir, en la que se indica un movimiento a una posición numérica, en cuyo caso hn vale —1. En caso contrario valdrá 0.

— También está la variable cs, que vale 1 para las estructuras 1 y 4, en las cuales el objeto A está delante, en la frase, del objeto B; y vale 2 en las estructuras 2 y 3, que ocurre lo contrario.

— La variable ar indica si la estructura usa preposiciones como en, encima de, sobre, en cuyo caso ar vale —1, o si se usan preposiciones como bajo, debajo, en cuyo caso vale 0. Es decir, es otra variable para discernir la estructura.

— La variable aux indica si el objeto A es un triángulo —aux = 6— o es un cuadrado — aux = 0.

— La variable t indica si el objeto es grande o pequeño, según valga 32 ó 16; t es la apotema del objeto. Este valor depende de si el número del objeto es par —pequeño— o impar —grande.

— Para indicar el color están las variables ca y cb.

Hay alguna variable más, pero éstas son las más importantes.

Las demás las iremos viendo cuando aparezcan.

Veamos ya la realización del programa.

* Las primeras líneas, hasta la 280, es la inicialización de las variables, dibujar en pantalla el estado inicial, con el control de colores conveniente, la construcción de la ventana de texto.

Esta parte es sencilla y su única complicación es el cálculo de las posiciones en pantalla en donde se han de dibujar los objetos.

Después se escribe el texto inicial y, en la línea 300, se pide ya el movimiento deseado, que se mete en la variable fr$.

Se mira si la palabra es «adiós» para terminar.

A partir de aquí, se empieza ya el análisis de la frase con la intención de saber al final del análisis el valor de la na y nb, que representan el número del objeto de arriba y abajo.

Entre las líneas 330 y 370 se mira si hay en la frase algún número del 1 al 12. Si lo hay estamos en la quinta estructura y hacemos hn=—1. Si no la hay hecemos hn=0.

Si hemos encontrado el número, que será el ji, miramos —líneas 410 a la 450— si dicha posición ji está ocupada; esto ocurrirá cuando exista algún j con p(ji)=px(j), es decir, cuando algún objeto tenga su posición horizontal en la misma de ji. Si está ocupada, entonces se manda el mensaje correspondiente y se vuelve a empezar con otra orden. En caso contrario, se va a la línea 620, para ver qué objeto es el que ha de llevar a la posición ji.

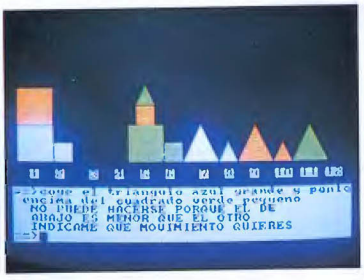
Si hn=0 entonces sigue por las líneas 460 hasta la 550, en donde se va viendo cuál es la preposición que se usa en la frase. Según cual sea, ar valdrá —1 ó 0, según vimos al ver las variables. Si no se encuentra ninguna de estas preposiciones se manda el mensaje correspondiente y se empieza de nuevo. En la 610 ponemos en pr la posición donde estaba dicha preposición.

En las 620-630 se comprueba si la palabra cuadrado está en la frase. Si no está y no estamos en la quinta estructura, entonces es imposible porque el objeto de abajo tiene que ser un cuadrado y se manda el mensaje conveniente de la línea 650.

En caso de la quinta estructura, si hn, es posible que no esté ningún cuadrado en la frase, y será un triángulo. Si no estamos en la quinta estructura, entonces hacemos pb=p, lo cual es provisional, y buscamos en la frase si hay, después de el cuadro encontrado, otro cuadro —por eso tomo n=pb+8, posición posterior a la de cuadrado. Si lo hay, no hace falta buscar si está el triángulo y nos saltamos las líneas 710-740; y variamos las posiciones pa y pb, suponiendo el caso 1. Las posiciones se cambiarán o se dejarán según la estructura que tengamos.

Si no hay dos cuadrados se mira si hay triángulo, como antes con los cuadrados; si no lo hay se manda el mensaje como pa, y aux=6, como se vio al ver las variables.

Así ya tenemos en pa y pb las posiciones de los dos objetos. En caso de la quinta estructura no es necesario y simplemente no influirá en su ejecución dichos parámetros.



Una parte fundamental está entre las líneas 750-760, en donde se analiza la estructura en la que nos encontramos. En las líneas 750 y 753, por medio de dos instrucciones condicionadas, se consigue saber cuál es el valor de cs:

— Si pr < pa y pr < pb y ar=1 entonces estaremos en la estructura 2 con lo que cs=2.

— Si pr < pa y pr < pb y ar=0 entonces estaremos en la estructura 4 con lo que cs=1.

— Si no se da la condición de las posiciones y ar=—1, estaremos en la estructura 3 y cs=2.

Tras esto sabremos ya el orden del objeto A y el objeto B:

cs=1: el objeto A delante del objeto B.

cs=2: el objeto B delante del objeto A.

Recordar que el objeto A es el de arriba y el objeto B es el de abajo, por lo cual es muy importante saber su orden. Todos estos cálculos se consiguen viendo todos los casos posibles que se pueden presentar en las extructuras, y, por ello, es lo más complicado de entender.

Después, en la línea 756, se comprueba si el objeto B, el de abajo, es un triangulo, por medio de un if un poco complicado, para sólo ve las posibilidades de que así sea; en dicho caso nos vamos a la línea 650, donde se da el nombre conveniente.

Por último, en la línea 758, si cs=2, es decir, si estamos en una estructura con el objeto B delante del objeto A, se intercambian pa y pb, para que tengan ya la posición de A y B de forma correcta, respectivamente.

Entre las líneas 760 y 950 se calcula el color que tiene cada objeto A y B actuales. Para ello, se usa un data con las palabras azul, rojo y verde, y sus números asociados, 1, 2 y 3. Si estamos con hn=—1 entonces es más sencillo porque sólo se busca un color.

Veamos qué pasa en los demás casos. Para ello se toman dos subpartes de la frase total:

— fr$(0) es la correspondiente al objeto A.

— fr$(1) es la correspondiente al objeto B.

Estas se calculan según estemos en unas estructuras o en otras, dependiendo del valor de cs y vía unos cálculos sencillos de las posiciones pb y pa. Después se van leyendo de los datas estos colores para cada subfrase adecuada, obteniendo en n(0) el color de A, y en n(1) el de B, valores que se traspasan a ca y cb respectivamente. Si no se encuentran estos colores se saca el mensaje adecuado, saltando en la línea 910 a la 720.

Si los colores coinciden, también se lanza el mensaje adecuado. En el caso de hn=—1 sólo tenemos, como es de esperar, un sólo color en ca.

Entre las líneas 960-1080, se calcula, de forma parecida, los tamaños de los objetos A y B, los cuales están al final en n(0) y n(1) respectivamente. El procedimiento es similar al anterior, incluido para hn=—1.

Ahora, calculo ya los valores de na y nb, que son los números asociados a los objetos A y B que se han calculado. Si estamos en la estructura 5, hn=—1, nb=0 y no hay problemas. En la línea 1110 se calcula, en caso de la estructura 5, si el objeto cabe en la posición en donde se le intenta colocar.

En estas líneas se comprueba también si el cuadrado de abajo, objeto B, estaba ya ocupado, oc(nb)=—1, o si el de arriba está ya cubierto. En ambos casos el movimiento no se hace y se manda el mensaje conveniente. También se ha comprobado si el objeto de arriba no es mayor que el de abajo.

Lo que queda de programa principal es borrar el objeto A de donde esté y colocarlo sobre el objeto B. Para borrarlo se va a la subrutina 2500 ó 3000, según sea cuadrado o triángulo, y con br=—1 se pinta con el color de fondo dicho objeto. Para dibujarlo en su sitio actual se calcula, según hn y según el objeto sea grande o pequeño, los valores de las nuevas posiciones; la variable xx se usa para saber, antes de borrar el objeto, si va a caber, en la nueva posición, dentro de la pantalla.

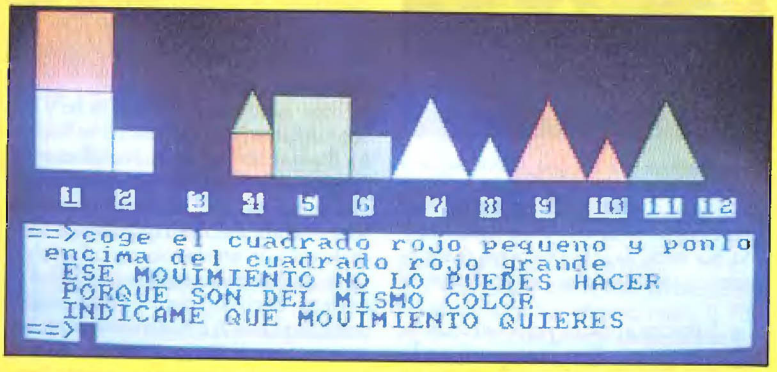
Por último poner de bajo del objeto A el objeto b —ob(na)=nb —y desocupar el anterior, lo cual se hace antes, como debe ser. Ya sólo decir que las subrutinas de dibujo son más sencillas. Solo indicar que la variable p indica el color.

Conclusión:

Espero que hayas entendido las ideas de la construcción del programas y que te sirva para que veas lo complicado que resulta llegar a comprender, digitalmente, el lenguaje humano.

También es importante notar que el BASIC no es muy adecuado, como ya quedó dicho y como se demuestra al seguir el programa paso a paso.

```
10 DIM PX(12),PY(12),p(12),DB(12)
20 MODE 1
30 INK 0,11:INK 1,0:INK 2,9:INK 3,6
40 BORDER 0
50 PAPER 1: CLS
60 PAPER 0:WINDOW #1,1,40,20,25
70 FOR I=1 TO 6:PRINT #1:NEXT
80 PX(1)=43:P(1)=43
90 FOR I=2 TO 12
100 PX(I)=PX(I-1)+52
110 P(I)=PX(I)
120 NEXT I
130 FOR I=1 TO 12
140 IF I MOD 2=0 THEN PY(I)=161 ELS
E PY(I)=177
150 NEXT I
160 FOR I=1 TO 6
170 GOSUB 2500
180 NEXT I
190 FOR I=7 TO 12
200 GOSUB 3000
210 NEXT I
220 FOR J=1 TO 12
230 K=K+3
240 IF K=9 OR K=22 THEN K=K+1
250 LOCATE K,18:PRINT MID$(STR$(J),
2,2)
260 NEXT J
```




```
270 PRINT #1,"  TIENES EN LA PANTAL
LA CUADRADOS Y       TRIANGULOS GRA
NDES Y PEQUENOS"
280 PRINT #1,"  PUEDES INDICAR SI Q
UIERES PONER ALGO    EN DONDE ESTAN
 LOS NUMEROS,DICIENDOLO"
290 PRINT #1,"  INDICAME QUE MOVIMI
ENTO QUIERES"
300 AUX=0
310 LINE INPUT #1,"==>",fr$
320 FR$=UPPER$(FR$)
330 N=1
335 PAL$="ADIOS":GOSUB 2000
336 IF P>0 THEN CLS:END
340 FOR JI=1 TO 12
350 PAL$=MID$(STR$(JI),2,2)
360 GOSUB 2000
370 IF P>0 THEN 400
380 NEXT JI
390 HN=0:GOTO 460
400 HN=-1
410 FOR J=1 TO 12
420 IF P(JI)=PX(J) THEN 450
430 NEXT J
440 GOTO 620
450 PRINT #1,"  NO PUEDES PONER NAD
A EN EL NUMERO       ";JI;" PORQUE E
STA OCUPADO":GOTO 290
460 PAL$="EN":GOSUB 2000
470 IF P>0 THEN 580
480 PAL$="ENCIMA":GOSUB 2000
490 IF P>0 THEN 580
500 PAL$="SOBRE":GOSUB 2000
510 IF P>0 THEN 580
520 PAL$="DEBAJO":GOSUB 2000
530 IF P>0 THEN 600
540 PAL$="BAJO":GOSUB 2000
550 IF P>0 THEN 600
560 PRINT #1,"  NECESITO QUE ME DIG
AS ALGO MAS          PARA SABER EL
MOVIMIENTO"
570 GOTO 310
580 AR=-1
590 GOTO 610
600 AR=0
610 PR=P
620 PAL$="CUADRADO":GOSUB 2000
630 IF P>0 THEN 660
640 IF HN THEN  700
650 PRINT #1,"  SI PONES ALGO ENCIM
A DE UN TRIANGULO    PUEDE CAERSE.R
EPITE":GOTO 290
660 IF HN THEN 740 ELSE PB=P
670 N=PB+8
680 GOSUB 2000
690 IF P<>0 THEN PA=PB:PB=P:GOTO  7
50
700 N=1
710 PAL$="TRIANGULO":GOSUB 2000
720 IF P=0 THEN PRINT #1,"  NO ME D
AS SUFICIENTE INFORMACION":GOTO 290
730 PA=P:AUX=6
740 IF HN THEN I=0: P=0:GOTO 860
750 IF PR<PA AND PR<PB THEN CS=2 EL
SE CS=1
753 IF NOT AR THEN CS=3-CS
755 IF AUX=0 THEN 758
756 IF (CS=1 AND PB<PA) OR (CS=2 AN
D PB>PA) THEN 650 ELSE 760
758 IF CS=2 THEN AB=PA:PA=PB:PB=AB
760 FT$=FR$
770 IF CS=2 THEN 810
780 FR$(0)=LEFT$(FT$,PB)
790 FR$(1)=RIGHT$(FT$,LEN(FT$)-PB)
800 GOTO 830
810 FR$(0)=RIGHT$(FT$,LEN(FT$)-PA)
820 FR$(1)=LEFT$(FT$,PA)
830 FOR I=0 TO 1
840 N=1:P=0
850 FR$=FR$(I)
860 WHILE PAL$<>"NO" AND P=0
870 READ PAL$,N(I)
880 GOSUB 2000
890 WEND
900 RESTORE
910 IF P=0 THEN 720
920 IF HN THEN CA=N(0):GOTO 960
930 NEXT I
940 CA=N(0):CB=N(1)
950 IF CA=CB THEN PRINT#1,"  ESE MO
VIMIENTO NO LO PUEDES HACER       P
ORQUE SON DEL MISMO COLOR":GOTO 290
960 PL$(0)="GRANDE":PL$(1)="PEQUENO
"
970 IF HN THEN I=0:GOTO 1010
980 FOR I=0 TO 1
990 FR$=FR$(I)
1000 ENC=0
1010 FOR J=0 TO 1
1020 PAL$=PL$(J)
1030 GOSUB 2000
1040 IF P<>0 THEN N(I)=J:ENC=-1
1050 NEXT J
1060 IF NOT ENC THEN 720
1070 IF HN THEN 1090
1080 NEXT I
1090 NA=2*CA-1+N(0)+AUX
1100 IF NOT HN THEN 1130 ELSE NB=0
1110 IF JI MOD 2<NA MOD 2 THEN PRIN
T #1,"  HAY NO CABE":GOTO 290
1120 GOTO 1200
1130 NB=2*CB-1+N(1)
1170 IF (NB MOD 2)<(NA MOD 2) THEN
PRINT #1,"  NO PUEDE HACERSE PORQUE
 EL DE          ABAJO ES MENOR QUE
 EL OTRO":GOTO 290
1190 IF OC(NB) THEN PRINT #1,"  ESE
 CUADRADO YA ESTABA CUBIERTO":GOTO
290
1200 IF NA<7 THEN IF OC(NA) THEN PR
INT #1,"  EL CUADRADO DE ABAJO ESTA
 CUBIERTO":GOTO 290
1210 IF HN THEN 1220 ELSE xx=PY(NB)
+20+16*((NB MOD 2)+1)+16*(NA MOD 2)
:GOTO 1230
1220 IF I MOD 2=0 THEN XX=161 ELSE
XX=177
1230 IF XX+T>400 THEN PRINT #1,"  T
E SALES DE LA PANTALLA":GOTO 290
1240 I=NA:BR=-1
1250 IF NA>6 THEN GOSUB 3000 ELSE G
OSUB 2500
1260 IF HN THEN PX(I)=P(JI) ELSE PX
(I)=PX(NB)
1270 IF NOT HN THEN OC(NB)=-1
1280 PY(NA)=XX
1290 BR=0
1300 IF NA>6 THEN GOSUB 3000 ELSE G
OSUB 2500
1310 OC(DB(NA))=0
1320 DB(NA)=NB
1330 GOTO 290
2000 REM posicion de fr$ donde esta
 pal$
2010 M=N
2020 p=INSTR(M,fr$,pal$)
2030 IF p=0 THEN 2120
2040 IF p=1 THEN 2070
2050 A$=MID$(fr$,p-1,1)
2060 IF A$<>" " AND A$<>"," AND A$<
>";" THEN 2100
2070 IF p+LEN(pal$)-1=LEN(fr$) THEN
 2120
2080 A$=MID$(fr$,p+LEN(pal$),1)
2090 IF A$<>" " AND A$<>"," AND A$<
>";" THEN 2100 ELSE 2120
2100 M=p+LEN(pal$)
2110 GOTO 2020
2120 RETURN
2500 REM cuadrado
2510 IF BR THEN P=1:GOTO 2540
2520 p=(i-1)\2
2530 IF p=1 THEN p=3
2540 GRAPHICS PEN p
2550 IF i MOD 2=0 THEN t=16 ELSE t=
32
2560 PLOT px(i)-t,py(i)-t
2570 DRAW px(i)+t-1,py(i)-t
2580 DRAW px(i)+t-1,py(i)+t-1
2590 DRAW px(i)-t,py(i)+t-1
2600 DRAW px(i)-t,py(i)-t
2610 MOVE px(i),py(i)
2620 FILL p
2630 RETURN
3000 REM triangulo
3010 IF BR THEN P=1:GOTO 3040
3020 p=(i-7)\2
3030 IF p=1 THEN p=3
3040 GRAPHICS PEN p
3050 IF i MOD 2=0 THEN t=16 ELSE t=
32
3060 PLOT px(i)-t,py(i)-t
3070 DRAW px(i)+t-1,py(i)-t
3080 DRAW px(i),py(i)+t-1
3090 DRAW px(i)-t,py(i)-t
3100 MOVE px(i),py(i)
3110 FILL p
3120 RETURN
3500 DATA "AZUL",1,"ROJO",2,"VERDE"
,3,"NO",0
```