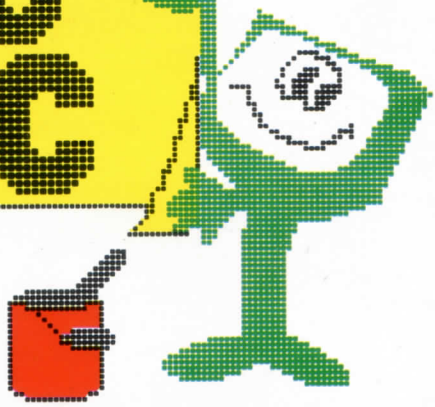
VIDEO BASIC



20 LIÇÕES DE BASIC
PARA APRENDER COM O SPECTRUM

EDIÇÕES LATINAS

JACKSON

- Como funciona o teclado
- O código ASCII
- O jogo de caracteres do SPECTRUM
- CODE, CHR$
- INKEY$
- FOR, TO, STEP, NEXT
- Os ciclos automáticos
- Exercícios vídeo
- Jogo vídeo n.º 5

5

Spectrum

16K/48K/PLUS

TIMEX COMPUTER 2048

VIDEO BASIC
Uma publicação de:
EDIÇÕES LATINAS-JACKSON
Director editor:
Manuel A. Lopes
Director editor da JACKSON ESPANHA:
Lorenzo Bertagnolio
Director de produção:
Vicente Robles
Autor:
Softidea
Redacção Software:
Francesco Franceschini
Stefano Cremonesi
Desenho gráfico:
Studio Nuovaidea
Ilustrações:
Cinzia Ferrari, Silvano Scolari,
Equipo Galata
Edições Latinas, Lda.
Direcção, redacção e administração, números atrasados e assinaturas:
Av. Almirante Reis, 219, 3.º-Esq.
1000 Lisboa
Fotocomposição e impressão:
Antunes & Amílcar, Lda.

ISBN do tomo 1: 84-85831-12-8
ISBN do fascículo: 84-85831-11-X
ISBN da obra completa: 84-85831-10-1
Depósito Legal N.º: 9962/85
Plano geral da obra:
20 fascículos e 20 cassetes, de publicação quinzenal.
Edições Latinas-Jackson garante a publicação de todos os fascículos e cassetes que compõem esta obra e o fornecimento de qualquer número atrasado, até 1 ano, depois de terminada a sua publicação.
Está reservado ao editor, o direito de modificar o preço de venda dos fascículos durante a sua saída, se o mercado assim o exigir.

1.ª Quinzena – Jan. 1986

SUMÁRIO

Introdução

Nesta lição aprofundaremos os nossos conhecimentos acerca do teclado, que é o dispositivo principal para a entrada de dados. Nem todos os teclados são iguais e os seus mecanismos e características variam de um tipo para outro, assim como o seu modo de funcionamento.
Relacionados directamente com o teclado, veremos também o código ASCII e o jogo de caracteres.
Em seguida, conheceremos e aprenderemos a usar: CODE, CHR$, INKEY$ e FOR-TO-STEP-NEXT, o que te permitirá repetir, todas as vezes que desejares, um conjunto de instruções.
E para terminar, uma técnica indispensável para o programador: os ciclos controlados.

Funcionamento e esquema dos tipos de teclado

O teclado constitui o dispositivo principal para a entrada de informações com que está dotado o teu computador.
O teclado é o elemento que te permite comunicar todas as instruções e os dados que pretendes executar ou memorizar na unidade central.
Um computador sem teclado seria como um automóvel sem volante: poderias pô-lo em marcha ou pará-lo, mas não o poderias controlar nem utilizar.
Assim, é importante que compreendas, além do simples e habitual uso diário, a estrutura e os princípios de funcionamento do teclado de um computador.
Mas antes de abordar este tema, é necessário precisar e aclarar com exactidão o que se entende por "teclado". Esta palavra refere-se única e exclusivamente ao dispositivo utilizado na entrada de dados.

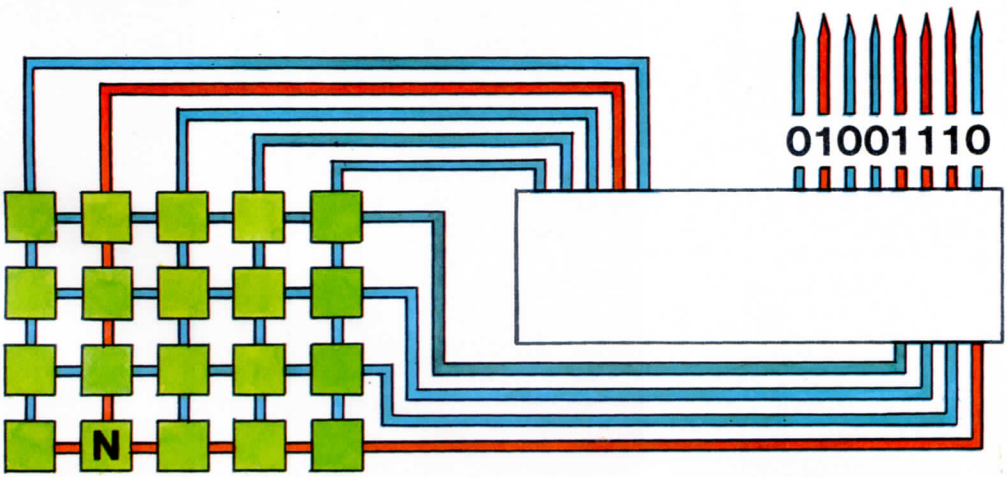
Chamar "teclado" a todo um computador (como fazem algumas pessoas menos esclarecidas) é incorrecto e está absolutamente errado.
Como já terás reparado, o teclado do teu Spectrum é bastante idêntico, tanto no aspecto como no funcionamento, ao de uma simples máquina de escrever; também neste caso será suficiente carregar na tecla correspondente ao caracter escolhido.
Nalgumas ocasiões, através da combinação de duas ou três teclas que se carregam ao mesmo tempo, podem obter-se outras letras, símbolos ou instruções que normalmente não estão disponíveis nas máquinas de escrever.
Um pormenor que talvez te tenha escapado, é o da colocação das teclas: estão ordenadas segundo o padrão norte-americano chamado QWERTY. Este nome atribuído aos teclados do tipo norte-americano, resulta da colocação das primeiras seis teclas alfabéticas da segunda fila.

Pelo contrário, nos teclados ditos europeus, a tecla Z ocupa a segunda posição no lugar do W; daqui o nome QZERTY.
Outras diferenças dizem respeito à posição da tecla M e à disposição da quase totalidade dos símbolos e sinais de pontuação.

De qualquer modo, na prática, nada muda; ambos os teclados (norte-americano e europeu) se comportam de forma absolutamente idêntica e fiel do ponto de vista de funcionamento.
Agora procuraremos compreender como funciona na realidade um teclado, ou seja, o que acontece quando carregas numa tecla do teu Spectrum. Todas as teclas existentes no teclado estão ligadas electricamente (ou seja, através de fios eléctricos) a um circuito integrado especial, que comprova qual das teclas foi carregada, e que emite um único e distinto código numérico binário de 8 bits para cada uma das teclas. Assim, a unidade central, quando recebe uma destas combinações, é imediatamente capaz (graças a um programa memorizado na ROM ou noutro circuito electrónico) de distinguir e localizar o carácter batido, para poder assim realizar as operações requeridas.

Por exemplo, quando carregas a letra A, na saída do *circuito codificador* (este é o termo técnico utilizado para denominar este componente) aparece o código binário 01000001, cujo número décimal correspondente é o 65.
A este código, e unicamente a ele corresponde-lhe na CPU, o caracter A; assim, não há nenhuma hipótese de que surjam, na máquina, erros e confusões.

Um circuito integrado reconhece e localiza qual a tecla carregada e emite o seu código binário correspondente.



O código ASCII

Os códigos numéricos que se atribuem a cada um dos caracteres de uso mais frequente não são escolhidos arbitrariamente pela marca fabricante, mas são o resultado da cooperação entre utilizadores de aparelhos e indústrias estabelecidas no campo da elaboração de dados. Em princípio, tais códigos foram escolhidos com o objectivo de simplificar e uniformizar as comunicações entre os diversos computadores eliminando assim todos os problemas relacionados com as diferentes representações de dados e informações.

Decimal	ASCII	Decimal	ASCII	Decimal	ASCII
0	NUL	43	+	86	V
1	SOH	44	,	87	W
2	STX	45	–	88	X
3	ETX	46	.	89	Y
4	EOT	47	/	90	Z
5	ENQ	48	0	91	[
6	ACK	49	1	92	\
7	BEL	50	2	93	]
8	BS	51	3	94	^
9	HT	52	4	95	_
10	LF	53	5	96	`
11	VT	54	6	97	a
12	FF	55	7	98	b
13	CR	56	8	99	c
14	SO	57	9	100	d
15	SI	58	:	101	e
16	DLE	59	;	102	f
17	DC1	60	<	103	g
18	DC2	61	=	104	h
19	DC3	62	>	105	i
20	DC4	63	?	106	j
21	NAK	64	@	107	k
22	SYN	65	A	108	l
23	ETB	66	B	109	m
24	CAN	67	C	110	n
25	EM	68	D	111	o
26	SUB	69	E	112	p
27	ESC	70	F	113	q
28	FS	71	G	114	r
29	GS	72	H	115	s
30	RS	73	I	116	t
31	US	74	J	117	u
32	espaço	75	K	118	v
33	!	76	L	119	w
34	"	77	M	120	x
35	#	78	N	121	y
36	$	79	O	122	z
37	%	80	P	123	{
38	&	81	Q	124	|
39	'	82	R	125	}
40	(	83	S	126	~
41	)	84	T	127	DEL
42	*	85	U		

A difusão cada vez maior dos computadores pessoais fez com que a referida codificação, chamada ASCII (abreviatura de American Standard Codes for Information Interchange, ou seja Códigos Estandardizados Americanos para o Intercâmbio de Informações), se tenha convertido de facto no "padrão" utilizado na totalidade dos computadores. Portanto, aos caracteres alfabéticos e de pontuação presentes no teu Spectrum, correspondem as mesmas combinações de bits codificadas nos outros computadores, quer dizer, todos obtêm idênticos códigos com as mesmas letras.

Teclas e teclados

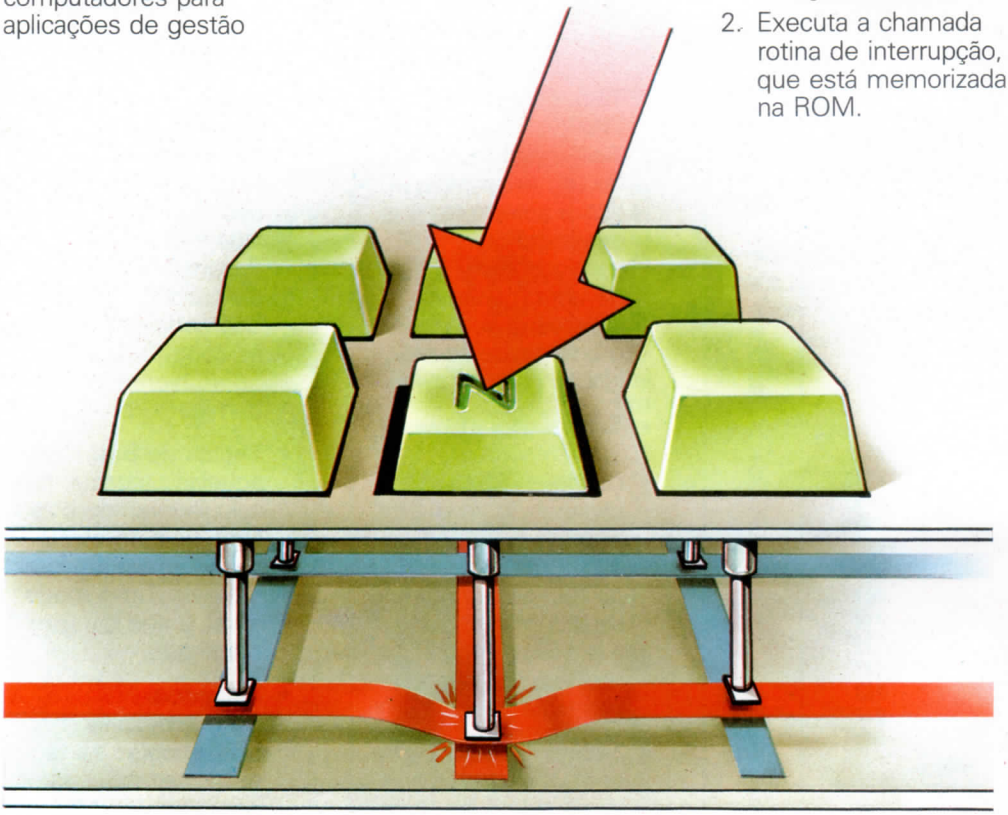
O teclado está submetido a um constante trabalho mecânico. A sua vida e duração dependem em grande parte da qualidade das teclas; esta pode oscilar entre algumas dezenas de milhar de batidas, nos teclados de inferior qualidade, e muitas dezenas de milhões.
Vejamos rapidamente os diferentes tipos de teclas, começando pelas melhores:

- **Teclas de efeito "Hall":** Aproveitam a acção de um íman móvel sobre a corrente que atravessa um semicondutor. Dotados de uma mecânica simples, a sua duração média mede-se em biliões de batidas.
- **Teclas capacitivas:** Tratam-se de condensadores cuja capacidade varia com a batida da tecla. Têm uma duração de muitas dezenas de milhões de batidas e empregam-se nos melhores computadores pessoais.
- **Teclas "reed":** Um contacto situado no interior de uma ampola de vidro (o contacto *reed*) fecha-se pela acção de um íman colocado à volta. A duração é de algumas dezenas de milhões de batidas. Devido à eficiência dos teclados capacitativos, que são mais duros, o seu uso reduziu-se muito durante os últimos anos.
- **Teclas mecânicas estandardizadas:** Utilizam-se em muitos computadores pessoais: a sua vida depende da qualidade de fabrico. Na melhor das hipóteses, a sua vida é de algumas dezenas de milhões de batidas. São sensíveis às condições do ambiente: humidade, pó.
- **Teclas de bola:** São aquelas utilizadas nas calculadoras de bolso. Uma bola de metal volta-se pela pressão da tecla. Dum modo geral, a sua duração é limitada.
- **Teclados de membrana ou película:** As teclas estão constituídas por duas películas

condutoras, tensas e separadas por uma lâmina isolante perfurada de forma adequada. Pressionando a película superior, as duas folhas tocam-se, estabelecendo o contacto. Normalmente, os teclados de membrana empregam-se nos computadores para aplicações de gestão ou industriais, visto que garantem um excelente isolamento daqueles factores externos que podem prejudicar o seu funcionamento. A sua duração depende da qualidade de fabrico.

Pondo de parte as diferenças de construção, podemos explicar numa única sequência as operações que o teu computador, ou melhor, a sua CPU, executa, para analisar o teclado e localizar a tecla batida. Vejamos, de forma muito simplificada, o que é que acontece:

1. Periodicamente, e em intervalos estabelecidos, a CPU interrompe o que está a fazer para prestar atenção ao teclado.
2. Executa a chamada rotina de interrupção, que está memorizada na ROM.



3. Controla (na gíria, "scanning") a situação do teclado, ou seja, se alguma tecla foi batida.
4. No caso de alguma tecla ter sido batida, a CPU obtém a posição da dita tecla.
5. Uma vez verificado o ponto 4, volta a comprová-lo um instante depois (recorda-se que a velocidade da CPU é da ordem do Megahertz, ou seja, de milhões de vezes por segundo), para poder excluir possíveis interferências.
6. Agora a CPU volta ao seu trabalho, para retomar mais adiante a sequência já descrita.

Para concluir: o teclado é o teu principal meio de comunicação com o computador; trata-o com cuidado.
Por ser um instrumento mecânico está sujeito a desgaste e a sua duração depende do uso que faças dele. Evita portanto movimentos bruscos, pressões excessivas das teclas e sobretudo, quando um programa não funciona... evita descarregar a tua raiva no teclado.

O jogo dos caracteres

Como bem sabes (perdoa o pedantismo, mas vem a propósito), o teu SPECTRUM, como qualquer outro computador, conhece, sabe usar e memoriza unicamente números e para além disso, somente binários.
Como consegue então compreender e visualizar no ecrã a interrogação (?), a inscrição "adeus", o asterisco (*)? É muito simples. O teu *SPECTRUM* converte em números todos aqueles caracteres alfabéticos, numéricos e especiais (o jogo de caracteres da máquina) que é capaz de gerir, enviar e receber.
Para efectuar esta codificação utiliza um código muito parecido ao usado por todos os outros computadores pessoais: o já citado código ASCII.
Por esta razão, o teu computador também sabe "manipular" cadeias; interpreta-as como uma sucessão de números codificados que correspondem a uma tabela de caracteres presente na memória.

O importante é que a cada código corresponda um único caracter, para que assim o computador nunca encontre ambiguidades de interpretação.

Assim como o ASCII, também o jogo de caracteres do teu SPECTRUM é um código de 7 bits; pelo que são possíveis 128 (ou seja, 2^7) combinações, rápida e facilmente reconhecíveis com uma primeira observação,

sendo de uso frequente para qualquer: letras do alfabeto, símbolos de pontuação, valores numéricos.

Outros, pelo contrário, antes de tomarem um significado, requerem alguma reflexão; e ainda outros, tornam-se absolutamente alheios aos símbolos normalmente utilizados pelo homem. Estes são os chamados caracteres especiais (ou caracteres de controle); através deles podes dar ordens especiais que se adaptam às características de funcionamento da máquina.

Se bem que necessitando de correspondência com a linguagem humana, os caracteres de controle

são importantíssimos para o teu SPECTRUM; graças a eles podes, por exemplo, indicar o final de uma linha de programas (com a tecla ENTER), deslocar o cursor para qualquer lugar do ecrã (com as setas ← , →), ou apagar um caracter do ecrã (DELETE).
Dissemos que o jogo de caracteres do teu SPECTRUM emprega códigos de 7 bits, mas o teu SPECTRUM é um computador de 8 bits; por consequência, as combinações possíveis são 256 (2^8).

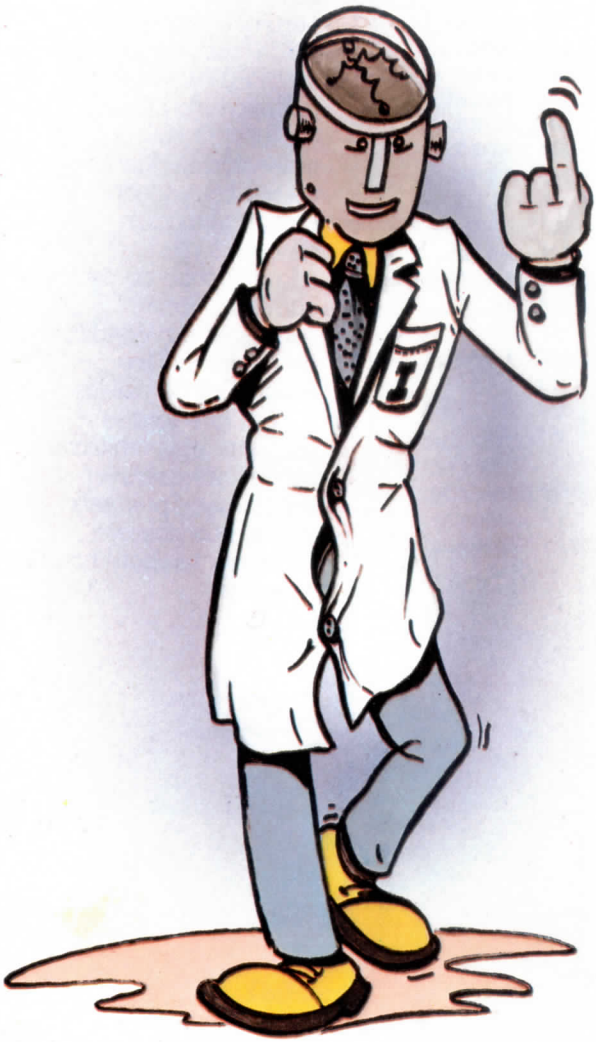
Código caracter		*Código caracter*		*Código caracter*	
0	não utilizados	44	,	91	[
1		45	—	92	/
2		46	.	93	]
3		47	/	94	↑
4		48	0	95	—
5		49	1	96	£
6	vírgula do **PRINT**	50	2	97	a
7	**EDIT**	51	3	98	b
8	cursor esquerdo	52	4	99	c
9	cursor direito	53	5	100	d
10	cursor abaixo	54	6	101	e
11	cursor acima	55	7	102	f
12	**DELETE**	56	8	103	g
13	**ENTER**	57	9	104	h
14	numero	58	:	105	i
15	não utilizado	59	;	106	j
16	car. control **INK**	60	<	107	k
17	car. control **PAPER**	61	=	108	l
18	car. control **FLASH**	62	>	109	m
19	car. control **BRIGHT**	63	?	110	n
		64	@	111	o
20	car. control **INVERSE**	65	A	112	p
		66	B	113	q
21	car. control **INVERSE**	67	C	114	r
		68	D	115	s
22	car. control **OVER**	69	E	116	t
23	car. control **AT**	70	F	117	u
24	car. control **TAB**	71	G	118	v
25	não utilizados	72	H	119	w
26		73	I	120	x
27		74	J	121	y
28		75	K	122	z
29		76	L	123	{
30		77	M	124	|
31		78	N	125	}
32	espaço	79	O	126	~
33	!	80	P	127	©
34	"	81	Q	128	□
35	#	82	R	129	▗
36	$	83	S	130	▖
37	%	84	T	131	▄
38	&	85	U	132	▝
39	'	86	V	133	▐
40	(	87	W	134	▚
41	)	88	X	135	▜
42	*	89	Y	136	▘
43	+	90	Z	137	▞

Código caracter			Código caracter		Código caracter	
138	▘		185	EXP	232	CONTINUE
139	▝		186	INT	233	DIM
140	▀		187	SQR	234	REM
141	▖		188	SGN	235	FOR
142	▌		189	ABS	236	GO TO
143	▐		190	PEEK	237	GO SUB
144	(a)	caracteres gráficos definidos pelo utilizador	191	IN	238	INPUT
145	(b)		192	USR	239	LOAD
146	(c)		193	STR$	240	LIST
147	(d)		194	CHR$	241	LET
148	(e)		195	NOT	242	PAUSE
149	(f)	caracteres gráficos definidos pelo utilizador	196	BIN	243	NEXT
150	(g)		197	OR	244	POKE
151	(h)		198	AND	245	PRINT
152	(i)		199	<=	246	PLOT
153	(j)		200	>=	247	RUN
154	(k)		201	<>	248	SAVE
155	(l)		202	LINE	249	RANDOMIZE
156	(m)		203	THEN	250	IF
157	(n)		204	TO	251	CLS
158	(o)		205	STEP	252	DRAW
159	(p)		206	DEF FN	253	CLEAR
160	(q)		207	CAT	254	RETURN
161	(r)		208	FORMAT	255	COPY
162	(s)		209	MOVE		
163	(t)		210	ERASE		
164	(u)		211	OPEN#		
165	RND		212	CLOSE#		
166	INKEY$		213	MERGE		
167	PI		214	VERIFY		
168	FN		215	BEEP		
169	POINT		216	CIRCLE		
170	SCREEN$		217	INK		
171	ATTR		218	PAPER		
172	AT		219	FLASH		
173	TAB		220	BRIGHT		
174	VAL$		221	INVERSE		
175	CODE		222	OVER		
176	VAL		223	OUT		
177	LEN		224	LPRINT		
178	SIN		225	LLIST		
179	COS		226	STOP		
180	TAN		227	READ		
181	ASN		228	DATA		
182	ACS		229	RESTORE		
183	ATN		230	NEW		
184	LN		231	BORDER		

Porquê desperdiçar semelhante capacidade? As restantes 256 – 128 = 128 combinações foram utilizadas pela casa Sinclair para definir outros caracteres que não pertencem ao padrão ASCII, mas unicamente ao SPECTRUM, como por exemplo os caracteres gráficos. O seu uso fica reservado exclusivamente ao teu computador; os computadores de outras marcas, não dispondo do carácter correspondente, não estão em condições de reconhecê-los e portanto de usá-los. É bom que tenhas presente esta limitação se desejas escrever programas "portáteis", isto é, que possam funcionar também noutras máquinas.

CODE

Às vezes pode ser útil conhecer o código ASCII de um carácter determinado, ou pelo contrário, produzir e imprimir um carácter a partir do seu código. O BASIC põe à tua disposição duas palavras com as quais podes converter os caracteres em códigos e os códigos em caracteres, que são: CODE e CHR$. Ambas são funções e portanto será necessário que a cada uma delas acrescentes o argumento sobre o qual vão operar. CODE produz o valor do código ASCII, ou seja, um número correspondente ao primeiro carácter de uma cadeia. Portanto o símbolo deverá ser uma cadeia, sob a forma de constante (limitada naturalmente por aspas), ou de uma variável. CODE proporciona como resultado um número compreendido entre 0 e 255, pois que os caracteres disponíveis no teu computador, como já vimos, são no total 256. Se o argumento for uma cadeia vazia, CODE devolve o valor 0.



Vejamos alguns exemplos:

```
PRINT CODE ("A")
```

Obterás 65, o código numérico ASCII do caracter "A".

```
PRINT CODE ("ABCD")
```

- Obterás 65, o primeiro caracter do argumento é "A".

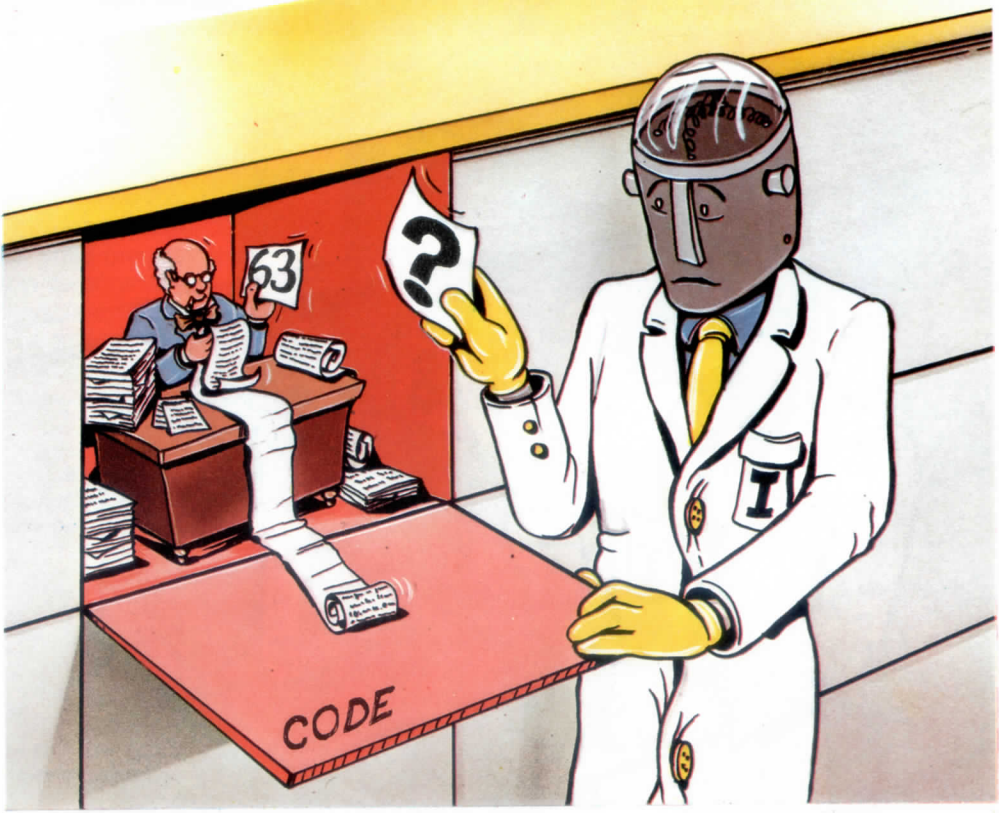
```
50 C$ = "TABELA"
60 PRINT CODE (C$)
```

Obterás 84, código numérico do caracter "T"

Sintaxe da instrução

CODE (cadeia)

O argumento de CODE é um caracter. O resultado é o seu código numérico.



CHR$

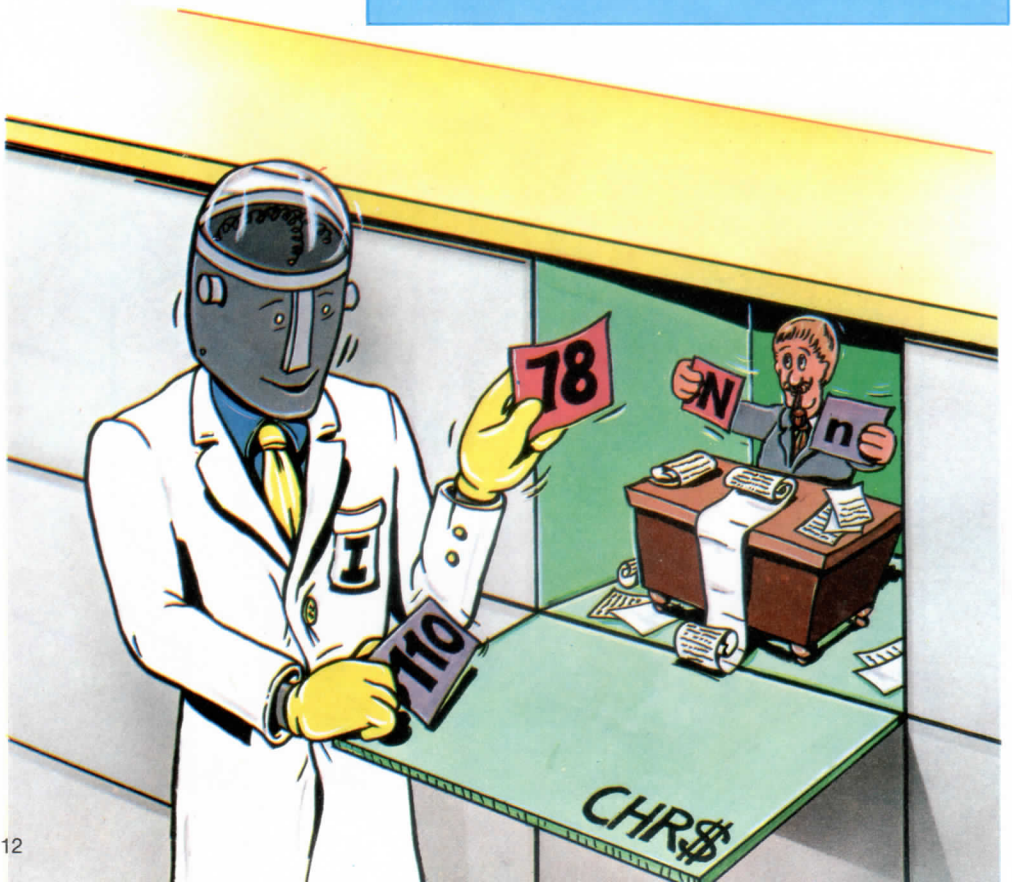
CHR$ é, em certo sentido, a função oposta de CODE: devolve-te o caracter correspondente ao número que tenhas utilizado como argumento. Este número pode especificar-se mediante uma variável ou uma expressão.

O argumento deverá ficar compreendido entre 0 e 255; em caso contrário, o computador visualizará a mensagem de erro INTEGER OUT OF RANGE.
Quando o valor do argumento for um número decimal, CHR$ arredonda-o ao valor inteiro obtido somando 0,5.

```
15 PRINT CHR$ (151/2)
```

151/2 = 75,5; o número inteiro que se obtém ao somar 0,5 é 76. A este argumento corresponderá o carácter L.
O programa seguinte imprime o jogo completo de caracteres do teu SPECTRUM.

```
10 FOR A=0 TO 255: PRINT CHR$ (A): NEXT A
```



Alguns destes caracteres (os chamados caracteres de controle), ainda que fazendo parte do jogo de caracteres, não são visualisáveis; portanto não aparecerão no ecrã. As funções CODE e CHR$ tornam-se úteis se se pretende transformar uma letra maiúscula no caracter seu correspondente em minúscula, e vice-versa. Estudando a tabela ASCII poderás dar-te conta rapidamente de que o código de uma letra minúscula é igual ao código da sua maiúscula correspondente, somando-lhe 32.
O programa seguinte, aproveita precisamente esta característica para imprimir em minúscula o caracter correspondente à maiúscula que se tenha batido.

O argumento de CHR$ é um código numérico cujo resultado é o caracter correspondente.

```
10 CLS
20 INPUT A$: IF A$="*" THEN GOTO 90: REM
   espera o batimento de uma tecla.
30 IF CODE (A$)<65 THEN GOTO 20
40 IF CODE (A$)>90 THEN IF CODE (A$)<97
   THEN GOTO 20
50 IF CODE (A$)>122 THEN GOTO: REM esta
   concatenação de IF faz que se aceitem unicamente
   os caracteres correspondentes às letras
   maiúsculas, entre 65 e 90 e das minúsculas entre
   97 e 122, ignorando-se os restantes.
60 IF CODE (A$)<91 THEN PRINT A$, CHR$ (CODE
   (A$)+32): REM se o caracter está em
   maiúsculas, então imprime também
   o correspondente caracter em minúscula.
70 IF CODE (A$)>96 THEN PRINT A$, CHR$ (CODE
   (A$)-32): REM se, pelo contrário, é minúscula,
   então imprime também a correspondente
   maiúscula.
80 GOTO 20
90 REM FIM
```

Exemplos:

```
PRINT CHR$ (65)
```

O código 65 corresponde ao caracter «A».

```
PRINT CHR$ (18 * 5)
```

O argumento da função CHR$ pode ser uma expressão cujo valor esteja compreendido entre 0 e 255.

```
S=(100-1)
PRINT CHR$ (S/3)
```

Se observares a tabela ASCII publicada algumas páginas atrás, verás que o código 33 corresponde à exclamação (!).

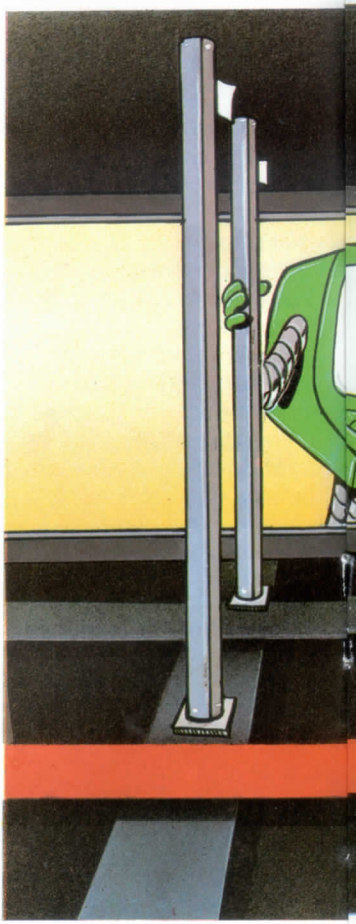
```
PRINT CHR$ (65.91)
```

De acordo com o afirmado anteriormente, o valor inteiro que se obtém somando 0,5 é 66. Obtém-se portanto a impressão do caracter B.

Sintaxe da instrução

CHR$ (número)
Portanto, NUMERO pode ser um número, uma variável ou uma expressão numérica.

INKEY$

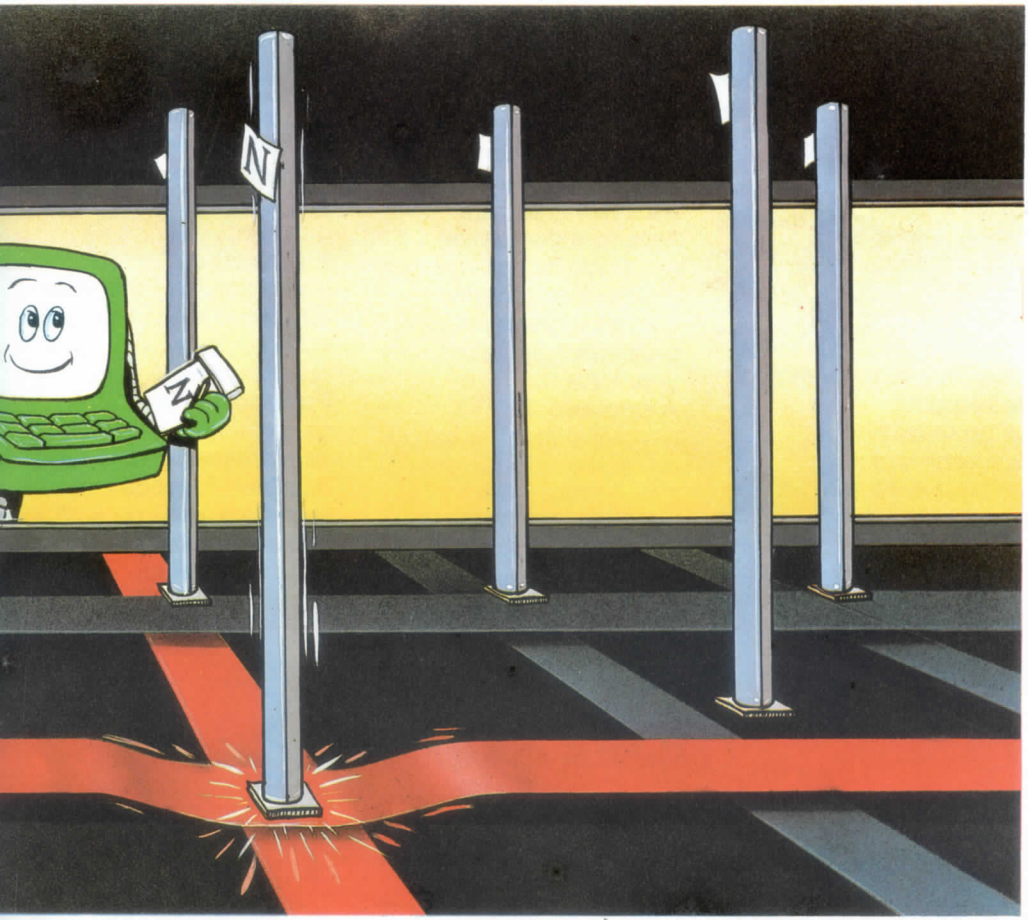


LINGUAGEM

A função INKEY$ permite a entrada no teclado do teu SPECTRUM de um único caracter, mas sem ter que bater depois a tecla ENTER. Além disso, o caracter correspondente à tecla batida não se visualisa no écran.

Por que não utilizar então o conhecido e familiar INPUT? Não é certo que com INPUT também se pode introduzir um único caracter? A diferença é subtil mas importante. Como já se viu, falando de INPUT, se te enganares no dado a introduzir, as consequências podem ser catastróficas, os resultados errados, ou ainda pior, pode bloquear-se o programa. Com INKEY$ não acontece nada disto: parece feito

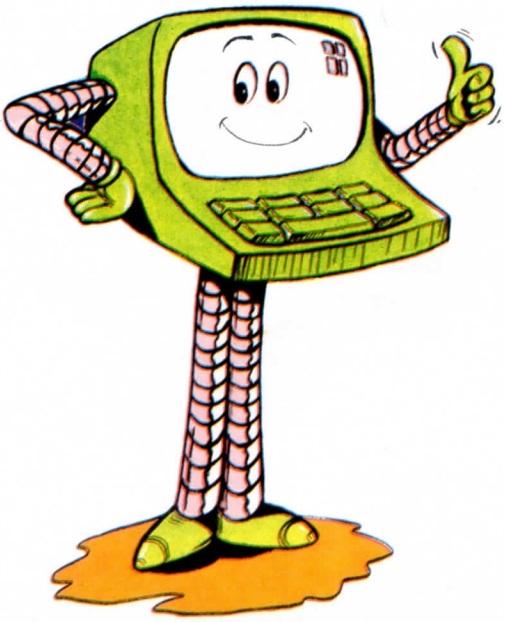
O teu Spectrum, quando encontra INKEY$, anota o caracter que estás batendo nesse momento.



expressamente para que possas enganar-te (INPUT controlado) sem que aconteça nada. O teu SPECTRUM espera tranquilamente que carregues a tecla correcta. O termo «espera» merece umas considerações à parte. Com INPUT, o programa interrompe-se para esperar os dados de entrada, pelo contrário, INKEY$ executa-se como uma função normal do BASIC, com a velocidade de que é capaz o intérprete, muito superior à de qualquer ser humano. Portanto será necessário incluir a função INKEY$ dentro de um ciclo de espera que torne mais "humano" o tempo de resposta.
Se quiseres podes descobrir sózinho (naturalmente sem ler a listagem) quando os programas, por exemplo, os de VÍDEOBASIC, utilizam a função INKEY$ ou um INPUT. Pensa... cada vez que o programa requeira a pressão de uma tecla seguida de ENTER significa que tem escondido um INPUT; se ENTER não é necessário, tratar-se à de INKEY$, cujo argumento pode ser uma ou mais variáveis de cadeia. Logicamente se pretendes que o caracter reconhecido seja 2 ou *, ou mesmo o !, será imprescindível que o coloques entre aspas. Em resumo, INKEY$ utiliza-se habitualmente para esperar e comprovar a entrada de um caracter qualquer no teclado, ou para esperar até que o utilizador pressione uma tecla.
A instrução INKEY$ devolve o caracter batido no teclado tão rapidamente como se recorre a ela. Quando não está pressionada (ou não foi pressionada) nenhuma tecla, INKEY$ nomeia a variável de valor nulo e o programa continua com toda a normalidade.
O exemplo seguinte aclarar-te-á as ideias:

```
10 LET A$=INKEY$
20 IF A$=" " THEN 10
30 PRINT A$
```

A linha 10 atribui à variável A$ o valor da tecla que pressionaste no teu SPECTRUM; se nenhuma tecla foi pressionada, A$ tomará um valor nulo. Tanto num caso como no outro, o programa não sofrerá nenhuma paragem ou interrupção.
A linha 20 comprova o valor de A$: se o valor for nulo, a execução voltará à linha 10, recomeçando o ciclo. Portanto a única forma de terminar o programa será pressionar uma tecla qualquer. Fazendo isto, aparecerá ainda no ecrã o carácter batido no teclado (linha 30). É fácil



encontrar uma utilização típica de INKEY$ naqueles programas que submetem o utilizador a opções deste tipo:

```
Queres continuar? (S/N)
```

Estas perguntas requerem como resposta o batimento das teclas S ou N, construindo-se uma estrutura cíclica semelhante à anteriormente explicada, sendo assim possível eliminar automaticamente as contestações que não estejam entre as admissíveis, e evitando que o ecrã possa ficar cheio de caracteres inúteis e inestéticos.

```
10 CLS
20 PRINT "QUERES CONTINUAR? (S/N)"
30 LET R$=INKEY$
40 IF R$<>"S" THEN IF R$<>"N"
   THEN GOTO 30
50 PRINT R$
60 IF R$="S" THEN GOTO 20
70 REM FIM
```

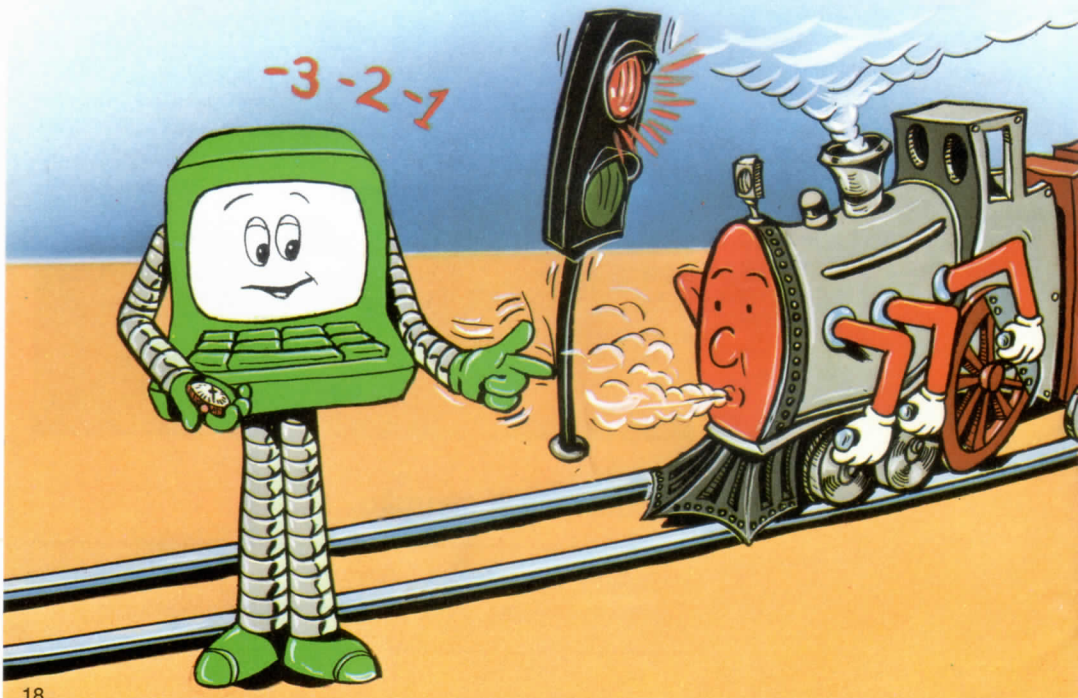
Sintaxe da instrução

VARIÁVEL DE CADEIA = INKEY$
Como observarás, INKEY$ necessita de argumento.

PAUSE

Às vezes, durante a execução de um programa, pode ser útil dispôr da possibilidade de parar momentaneamente a execução das instruções. Nalguns casos tornam-se necessárias pausas que permitam ao utilizador ler e apreciar uma determinada visualização ou tomar qualquer decisão.

Com este objectivo, o teu *Sinclair* dispõe de uma ordem especial, PAUSE. PAUSE n, para a execução do programa durante um tempo tanto maior quanto maior fôr o valor numérico atribuído a n, e como máximo 65535, valor a que corresponde uma paragem de aproximadamente 22 minutos.
Se indicares n = 0, o programa parará enquanto não carregares numa tecla. Valores intermédios, compreendidos entre 0 e 65535 pararão a execução durante tempos inferiores.
Por exemplo, para n = 50 segundos é necessário indicar n = 250.
PAUSE dispõe além disso de outra particularidade: se durante o decorrer

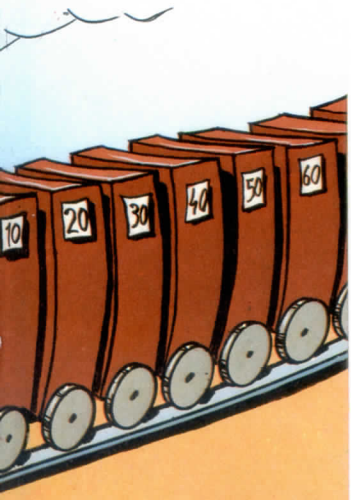


da pausa, se pressiona uma tecla qualquer, o programa começa de novo imediatamente a trabalhar, encurtando a pausa inicialmente prevista.
Assim esta instrução pode usar-se para adequar o tempo de permanência no ecrã das várias visualizações à capacidade de leitura do utilizador do programa.

Sintaxe da instrução

PAUSE n
n é um valor numérico compreendido entre 0 e 65535

FOR, TO, STEP, NEXT



Acontece com frequência num programa ser necessário repetir uma instrução ou um conjunto de instruções. Supõe, a título de exemplo, que desejas escrever um programa que multiplique a variável A pelos valores 1, 2, 3, 4 e 5.
Uma solução possível seria esta:

```
10 LET I=1:REM I
   é o número
   a multiplicar por A
20 LET A=A*I
30 LET I=I+1
40 IF I<6 GOTO 20
```

Trata-se de um exemplo típico de ciclo, ou seja, uma sequência de instruções executadas um certo número de vezes. Existe, em BASIC, uma instrução especial que te permite realizar estes ciclos sem recorrer a estruturas complexas. A instrução é composta pelas palavras FOR... NEXT. Imagina que desejas repetir uma série de instruções um determinado número de vezes, e exactamente até que um contador C (cujo valor inicial é S) alcance o valor A. Empregando a instrução FOR... NEXT, poderás escrever:

```
FOR C=S TO A
```

Faz alguma coisa.

```
NEXT C
```

A primeira vez que se executa o ciclo, C estabelece-se com um valor igual a S. Em seguida executam-se todas as instruções. Ao chegar a NEXT, o valor de C é incrementado e comparado automáticamente com A. Se C for menor que A, o ciclo repete-se, se não continua-se com a instrução que se seguir ao NEXT.

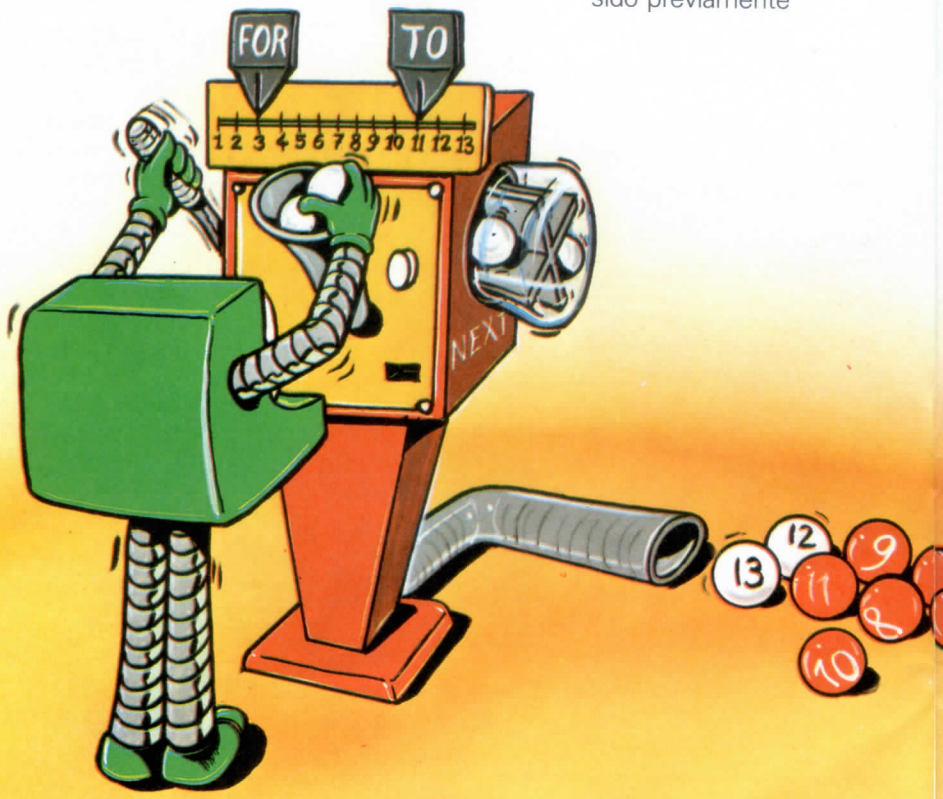
Portanto, o procedimento repete-se até que o contador C alcance o valor de A. Vendo de novo o problema da multiplicação, poderia dar-se-lhe então outra solução;

```
10 FOR C=1 TO 5
20 LET A=A*C
30 PRINT A
40 NEXT C
```

C chama-se variável de controle do ciclo (ou contador) e pode tomar qualquer nome legal permitido. Terás notado, no exemplo, que 1 é o seu valor inicial e 5 o valor final ou de teste. Na instrução FOR também se podem usar expressões, por exemplo:

```
FOR A=M+5 TO B/5
```

e também variáveis, sempre que os seus valores tenham sido previamente



estabelecidos.
Por exemplo:

```
10 LET DA=5:LET A=
   15
20 FOR C=DA TO A
30 ...
40 NEXT C
```

Além disso é possível incrementar o valor da variável de contador com um passo diferente de 1, utilizando simplesmente a palavra STEP (passo) seguida do valor com que se deseja incrementar de cada vez o contador:

```
FOR I=2 TO 10
STEP 2
```

I assumirá então os valores 2, 4, 6, 8, 10. Também é possível escrever programas nos quais os ciclos contenham no seu interior outros ciclos:

```
10 FOR I=0 TO 10
     STEP 3
  20 FOR J=3 TO 9
  30 ...
  40 ...
  50 ...
  60 NEXT J
70 NEXT I
```

Diz-se então que os ciclos estão concatenados (recorda-te de IF... THEN... GOTO). Naturalmente, os contadores dos diferentes ciclos deverão ser diferentes. O ciclo compreendido noutro ciclo deverá além disso ficar totalmente contido no primeiro; portanto, não é possível sobrepor partes dos ciclos:

```
100 FOR I=13 TO 20
  110 FOR J=0 TO 10
  120 ...
  130 ...
  140 ...
  150 ...
  160 NEXT I
170 NEXT J
```

Está errado! Os dois ciclos estão sobrepostos. É muito perigoso! Se a variável de índice for inicialmente maior que o valor final, o ciclo executar-se-á uma só vez. Por exemplo:

```
50 FOR I = 20 TO 5
60 PRINT I
70 NEXT I
```

será praticamente ignorado.
O passo do ciclo também pode ser negativo:

```
30 FOR X = 13 TO 10
   STEP -2
40 ...
50 ...
60 ...
70 NEXT X
```

Neste caso, o ciclo executa-se até que X, decrescendo 2 de cada vez, se torne menor que 10.
Se o valor do passo se estabelece igual a 0, o ciclo repete-se indefinidamente.

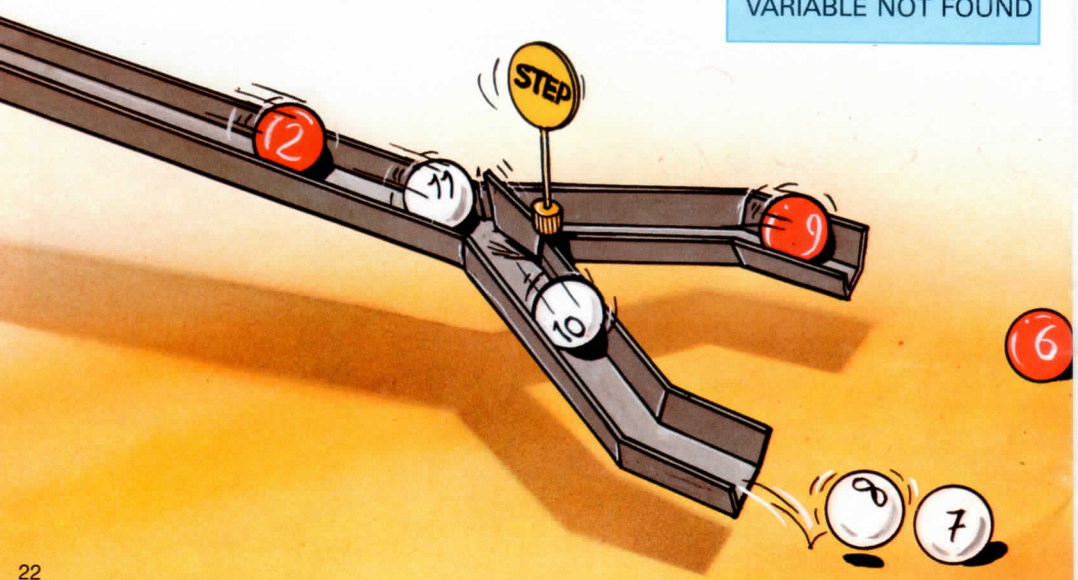
```
10 FOR K = 1 TO 10
   STEP 0
20 PRINT K
30 NEXT K
```

O único resultado deste programa será, portanto, a visualização contínua no ecrã do valor inicial de K, ou seja, 0. Além disso, deverás prestar atenção ao facto de que sair do interior do ciclo antes de a variável de controle ter alcançado o valor final, fará com que o computador espere o encerramento de um ciclo que não encontrará nunca. Nalguns casos isto poderia chegar a provocar mensagens deste tipo:

```
NEXT WITHOUT FOR
```

ou melhor,

```
VARIABLE NOT FOUND
```



Portanto, é uma prática aconselhável evitar incluir num ciclo instruções de salto (ou seja, GOTO); isto será também vantajoso para a legibilidade do programa. Outro erro muito comum é o de empregar um número de NEXT superior ao dos FOR: Neste caso o teu SPECTRUM contestará com:

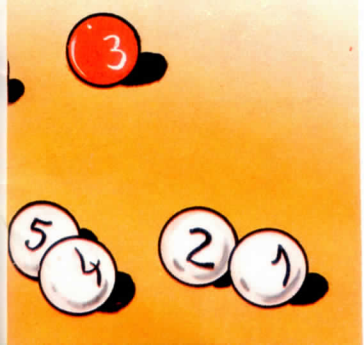
NEXT WITHOUT FOR

Uma última recomendação: o nome da variável de controle deverá ser sempre uma só letra, como i, j, k, x...

Sintaxe da instrução

FOR índice = valor inicial TO valor final [STEP passo]
NEXT índice

Na qual:
- Índice é o nome de uma variável numérica utilizada como contador.
- Valor inicial é o valor de saída do índice.
- Valor final é o valor que o índice tem que igualar ou superar.
- Passo é o incremento que o índice experimenta com cada repetição. Se não foi especificado estabelece-se como 1. Pode ser negativo.



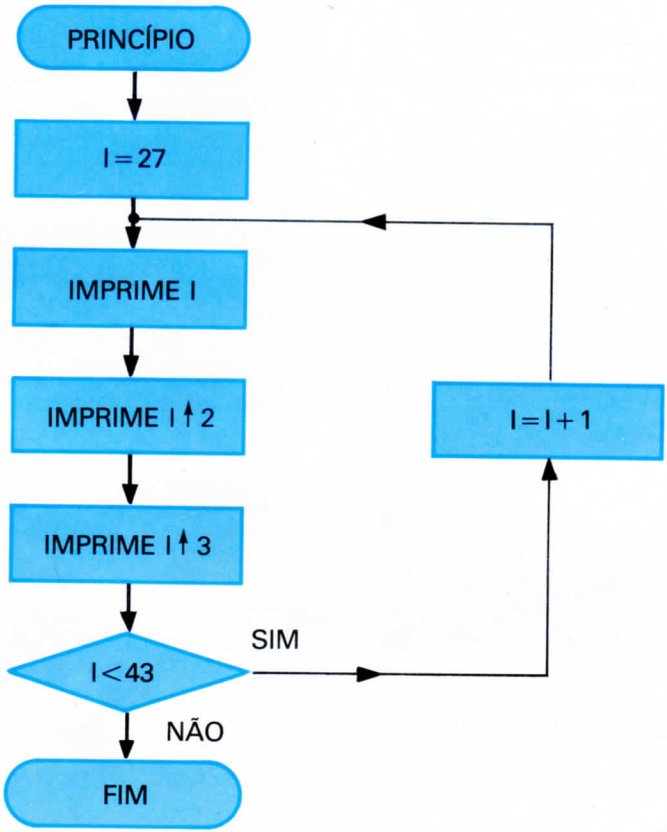
Os ciclos automáticos

Chamamos ciclos automáticos a todos aqueles ciclos que executam repetidamente uma sequência de instruções, utilizando a instrução FOR... NEXT. Na maior parte dos programas, os ciclos automáticos são tão comuns e úteis que constituem um instrumento importantíssimo nas mãos de qualquer utilizador.
Os exemplos de aplicações que vêm a seguir, ajudar-te-ão a aclarar e a aprofundar os conceitos teóricos que já conheces, tanto no que respeita ao uso dos ciclos como à instrução FOR... NEXT.

Quadrados e cubos

Como primeiro exemplo estudaremos este problema: encontrar os quadrados e os cubos dos números compreendidos entre 27 e 43. Uma possível solução poderia ser:

A este diagrama de fluxo correspondem os dois programas BASIC seguintes: o primeiro utiliza um ciclo controlado (quer dizer, criado "artificialmente" pelo programador); o segundo usa um ciclo automático:



```
10 LET I=27
20 PRINT I ↑2,
30 PRINT I ↑3
40 IF I<43 THEN I=I+1:GOTO 20
50 REM FIM
```

```
10 FOR I=27 TO 43
20 PRINT I ↑2,
30 PRINT I ↑3
40 NEXT I
50 REM FIM
```

Saltará imediatamente à vista do leitor, a maior legibilidade do programa com o ciclo automático; distinguem-se em seguida nele, o princípio e o fim do ciclo, o que não ocorre na primeira listagem. Também para o teu SPECTRUM é preferível a segunda solução, visto que a execução do ciclo fica confiada a uma instrução concebida especialmente para resolver este tipo de problemas, e portanto específica para eles. Nos casos em que seja possível (e são a maioria) será sempre melhor recorrer a um ciclo automático: o programa – e o programador – serão beneficiados sob todos os pontos de vista.

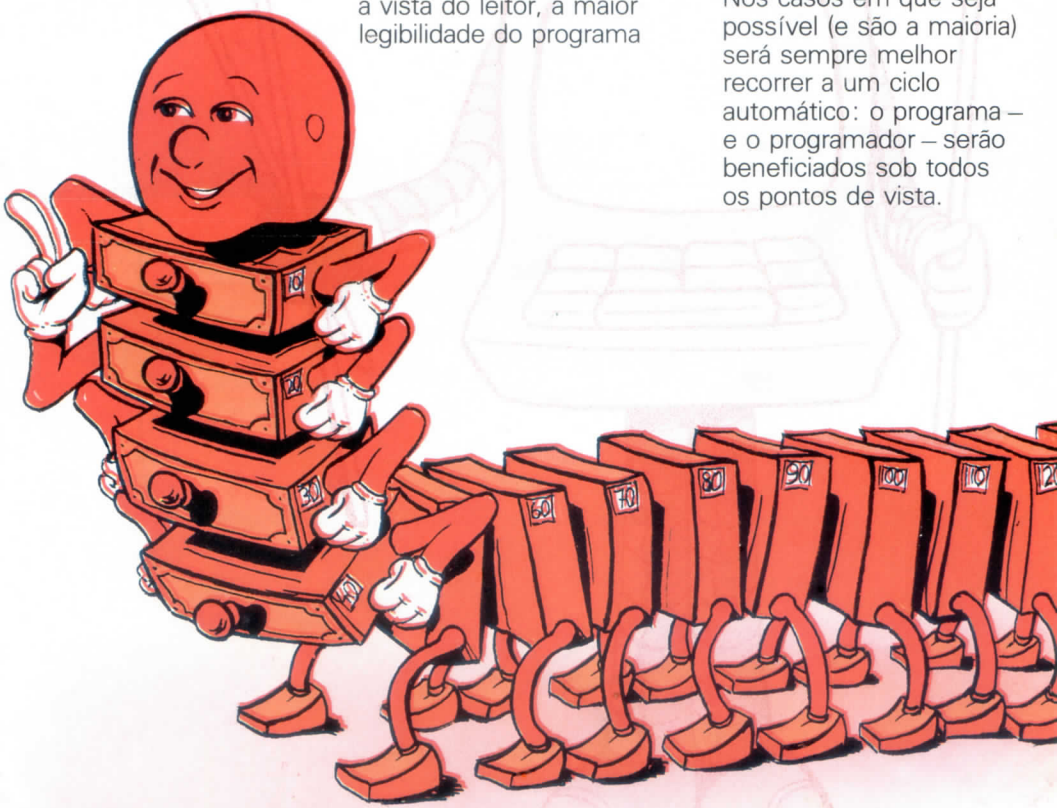
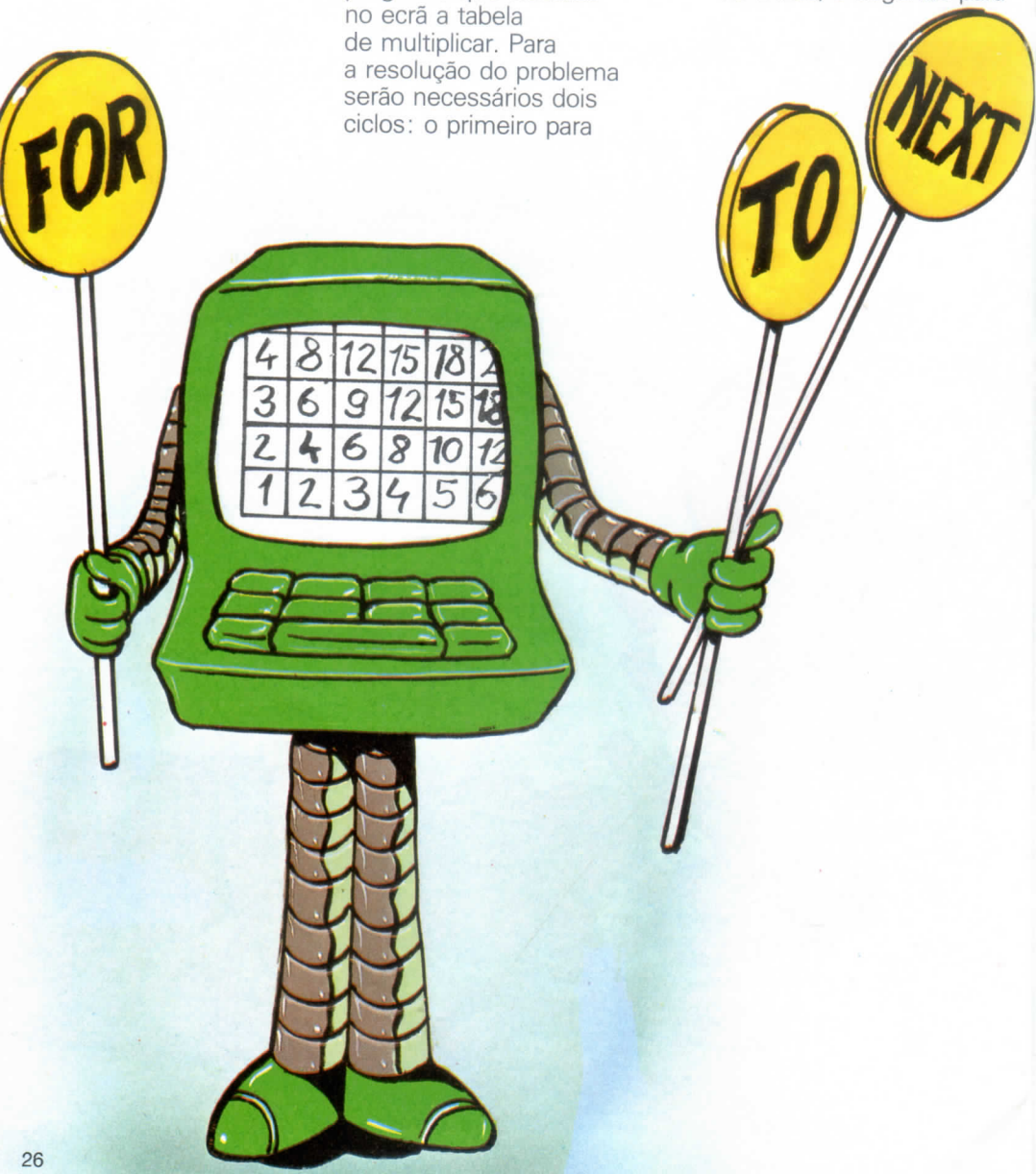
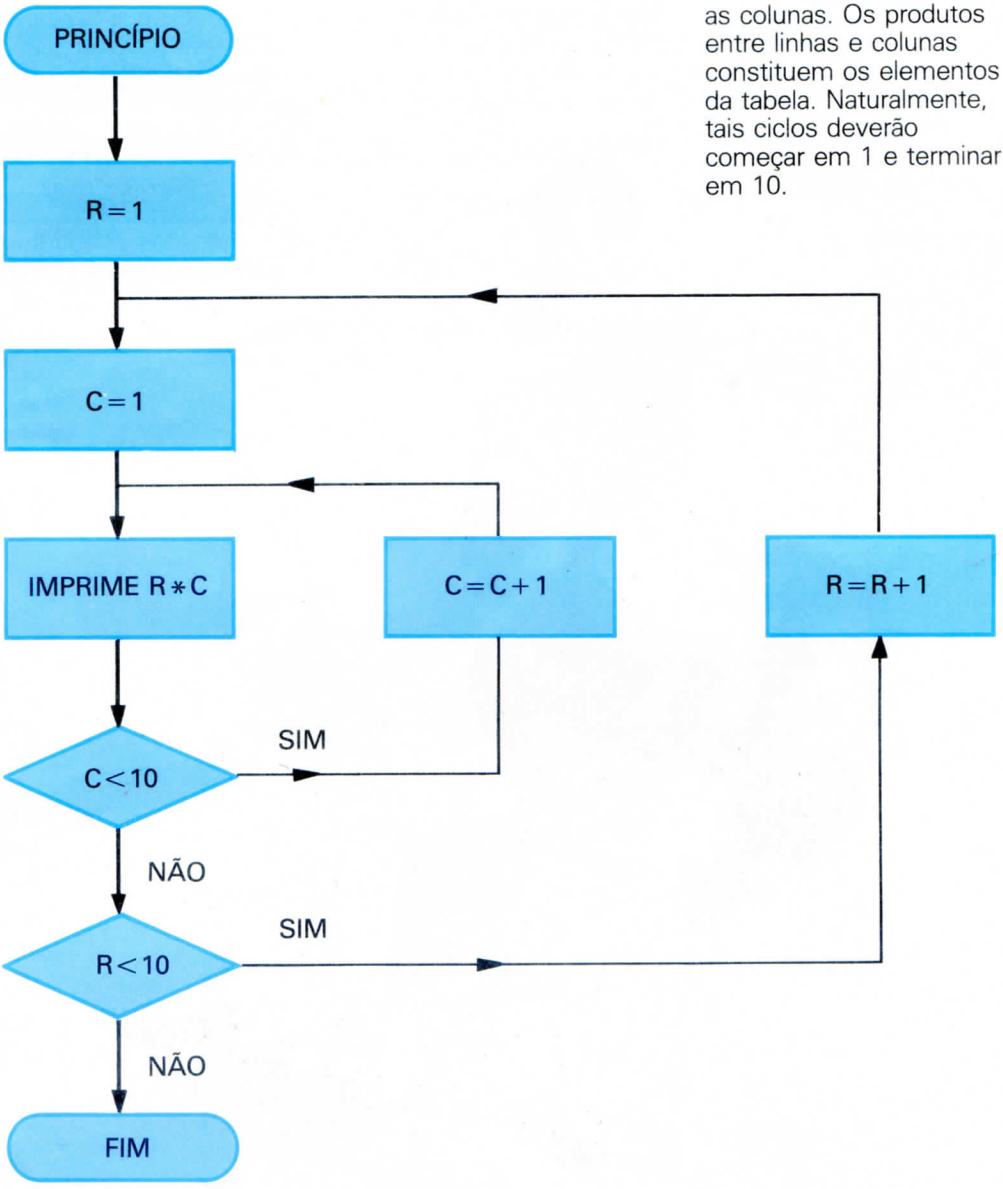


Tabela de multiplicar

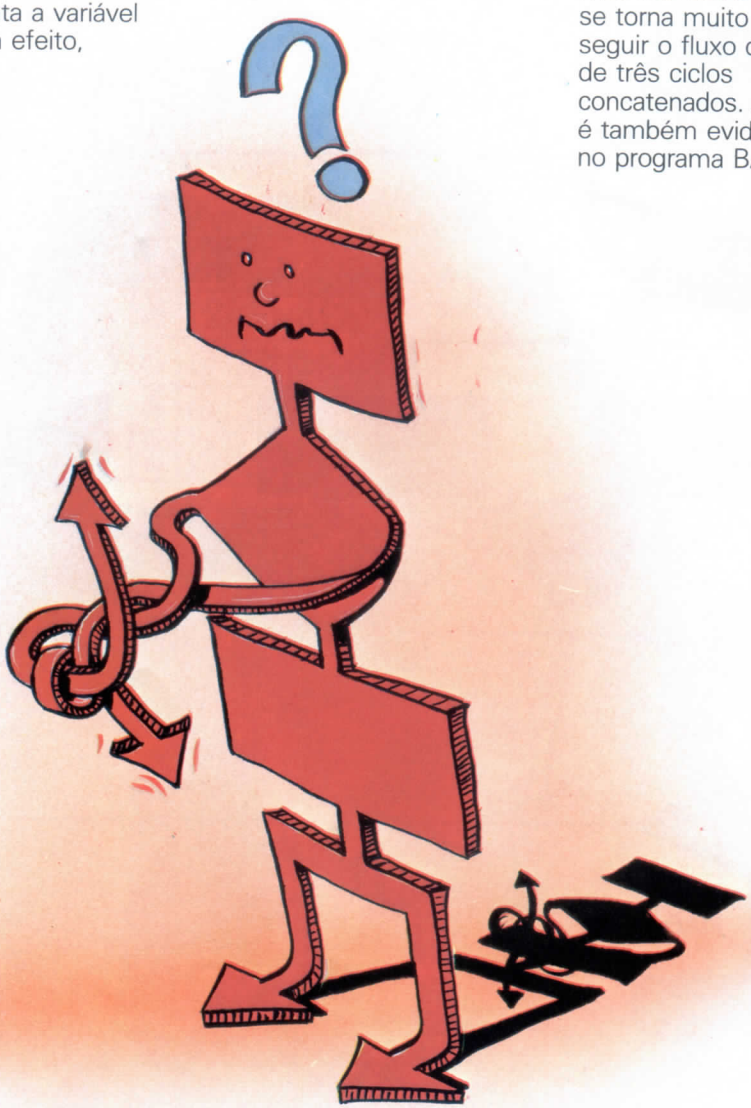
Como segundo exemplo, escreveremos um programa que visualiza no ecrã a tabela de multiplicar. Para a resolução do problema serão necessários dois ciclos: o primeiro para produzir os números correspondentes às linhas, o segundo para



as colunas. Os produtos entre linhas e colunas constituem os elementos da tabela. Naturalmente, tais ciclos deverão começar em 1 e terminar em 10.



O diagrama de fluxo ilustra perfeitamente o conceito de "ciclos concatenados"; o ciclo que modifica e incrementa a variável C fica, com efeito, completamente incluído no da variável R. No teu SPECTRUM os ciclos FOR podem concatenar-se, isto é, situar-se uns dentro dos outros, até um máximo de 10 vezes. Mas não abuses, porque se torna muito difícil seguir o fluxo de mais de três ciclos concatenados. A estrutura é também evidente no programa BASIC:



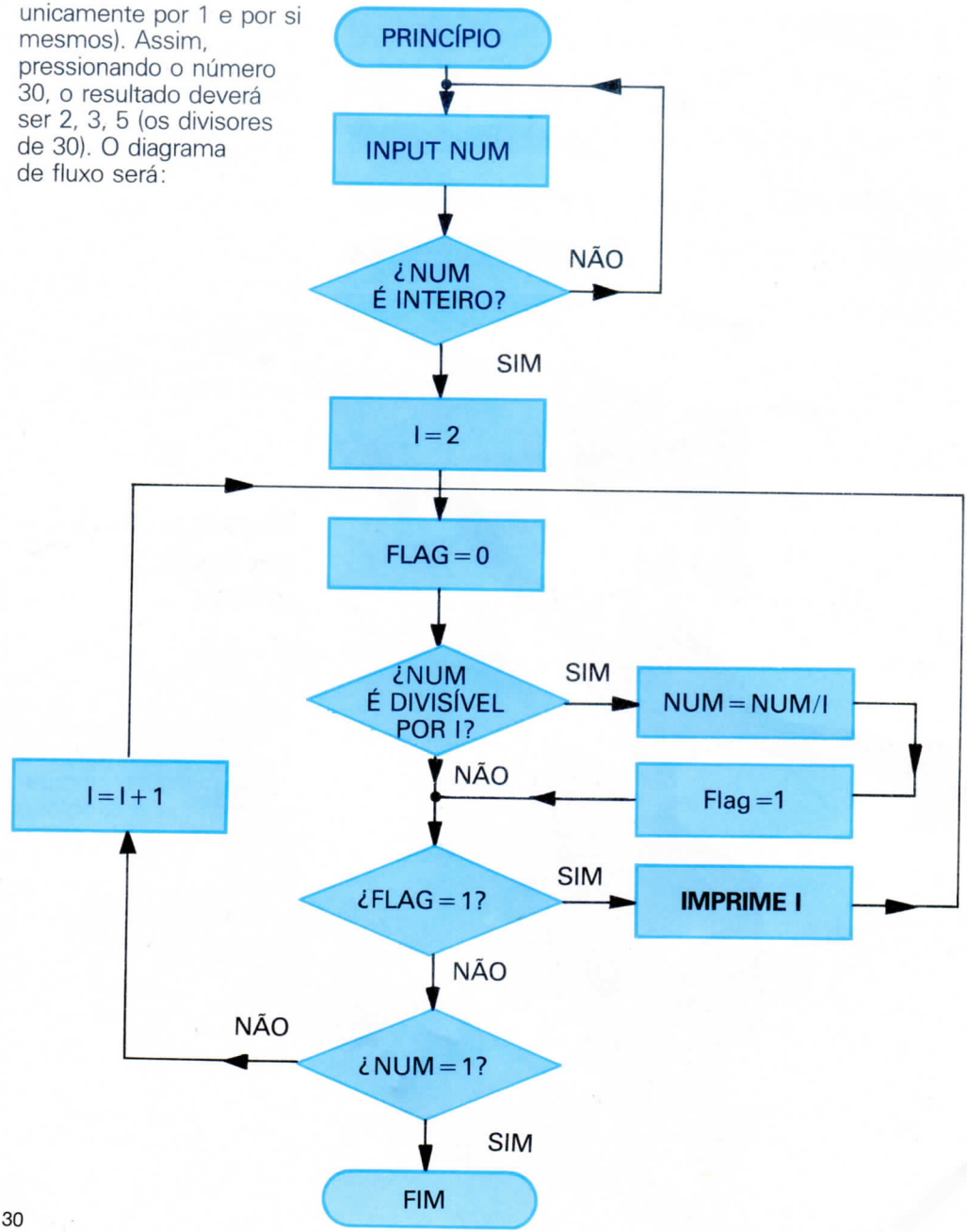
```
10 FOR R=1 TO 10
20 FOR C=1 TO 10
30 PRINT R*C; " ";
40 IF R*C<THEN PRINT " ";: REM se o produto
   é composto por um só valor então imprime um
   espaço.
50 NEXT C
60 PRINT
70 NEXT R
80 REM FIM
```

As linhas 40 e 60 foram incluídas no programa para alinhar as tabelas, incluindo os espaços adequados entre as diferentes linhas e colunas. Para compreender qual é o seu efeito experimenta eliminá-las (começa pela linha 40 e depois experimenta com a 60): pelo que aparece no ecrã serás capaz de compreender imediatamente a sua função e eficácia.

Decomposição em factores primos

Como último exemplo, vejamos o problema seguinte. Escrever um programa que, aceitando como entrada um número inteiro qualquer, produza na saída todos os valores deste número que sejam seu factores primos (recorda que os números primos são aqueles números divisíveis

unicamente por 1 e por si mesmos). Assim, pressionando o número 30, o resultado deverá ser 2, 3, 5 (os divisores de 30). O diagrama de fluxo será:



E aqui a correspondente listagem BASIC:

```
 5 INPUT "NÚMERO = "; NUM
10 CLS: PRINT "FACTORES PRIMOS DE"; NUM
20 IF NUM< >INT (NUM) GOTO 5: REM NUM É UM NÚMERO INTEIRO?
30 FOR I = 2 TO NUM
40 LET FLAG = 0: REM FLAG< >0 QUANDO NUM É DIVISÍVEL POR I
50 IF NUM/I = INT (NUM/I) THEM LET NUM = NUM/I: LET FLAG = 1
60 IF FLAG = 1 THEN PRINT I: GOTO 40
70 IF NUM = 1 THEN GOTO 90
80 NEXT I
90 REM FIM
```

O programa começa por verificar que o valor do número batido não tem parte decimal, caso contrário, solicita a entrada de um novo número.
A partir da linha 30 começa a autêntica fase de execução e resolução do problema: se NUM (o número pressionado como dado) é divisível por I (linha 50), então FLAG toma o valor 1. FLAG é uma variável utilizada como indicador: quando toma o valor 1 significa que I é um divisor de NUM e portanto deve imprimir-se. Se, pelo contrário, vale 0, I não é divisor de NUM, e portanto há que incrementar 1. À medida que o ciclo continua, NUM torna-se sucessivamente mais pequeno; no final tomará o valor 1. Neste momento o problema terá sido resolvido: terão aparecido no ecrã todos aqueles números que multiplicados entre si formavam o valor inicial de NUM.

EXERCÍCIOS

Recorrendo à função CODE dispõe em ordem crescente segundo os códigos, as seguintes cadeias:

"RUN"
"O TEU MICRO"
"SOFTWARE"
CHRS (13) + "VIDEOJ".
"VIDEOBASIC"

CÓDIGO	CADEIA

Recorrendo à tabela ASCII determina e escreve a saída do seguinte programa:

```
10 PRINT CHR$ (86); CHR$ (73); CHR$ (68); CHR$ (69); CHR$ (79);
20 PRINT CHR$ (66); CHR$ (65);
30 PRINT CHR$ (83); CHR$ (73); CHR$ (67)
```

A) Cronometra e anota a duração do programa.
B) Procura, anota e depois elimina a linha de pausa.
C) Cronometra novamente e anota o tempo.

```
10 CLS
20 FOR D = 10 TO 90 STEP 8
30 PRINT "DATO"; D
40 FOR P = 1 TO 3000: NEXT P
50 NEXT D
```

A	TEMPO
B	N.º DA LINHA DE PAUSA
C	TEMPO

Substitui agora a linha 40 do programa anterior com a seguinte:
40 PAUSE 150

Cronometra novamente e procura um valor de pausa análogo ao obtido com FOR P = 1 TO 3000: NEXT P.

Regulamento do Concurso de VIDEOBASIC

Para participar neste concurso, deverá enviar um "slogan" criativo que defina o Curso VIDEOBASIC. O "slogan" deverá ser o mais pequeno possível e escrito num postal onde colocará os 5 selos dos fascículos n.os 1 a 5, (os selos recortam-se da contra-capa do VIDEOBASIC de 1 a 5). Serão considerados participantes todos os "slogans" enviados até ao dia 31 de Janeiro de 1986. Pode participar com quantos "slogans" quiser, separadamente, desde que mande cada "slogan" com 5 selos. Todos os "slogans", classificados ou não, passarão a pertencer às Edições Latinas.

Os funcionários das Edições Latinas, não poderão participar no concurso.

Os trabalhos de selecção e classificação serão realizados por uma comissão julgadora, constituída por elementos a designar pelas Edições Latinas, cuja decisão será soberana e irrevogável. Os "slogans" serão seleccionados pela sua criatividade, clareza e adequação à publicação. Serão seleccionados e classificados os 5 melhores "slogans", cabendo a cada criador do "slogan" seleccionado um **TIMEX COMPUTER 2048 PERSONAL COLOR.**

- Escrita automática de instruções de comando
- Verificação imediata de erros
- Memória de 48K (RAM) e 16K (ROM)
- Linguagem integrada de programação BASIC
- Ficha de expansão para periféricos
- Joystick Interface Kempstone
- Saída Monitor Video composto
- Compatível Spectrum

A apuração será efectuada no dia 1 de Março de 1986, na sede das Edições Latinas, Lda., na Av. Almirante Reis, 219, 3.º-Esq. – 1000 Lisboa, onde os prémios se encontram.

O resultado será divulgado nos fascículos do VIDEOBASIC, e os contemplados serão notificados pelas Edições Latinas, Lda.

Enviar o postal a:

EDIÇÕES LATINAS, LDA.
Av. Almirante Reis, 219, 3.º-Esq.
1000 Lisboa

BOLETIM DE ASSINATURA

Envie a **EDIÇÕES LATINAS – VIDEOBASIC**
Av. Almirante Reis, 219, 3.º-Esq. • 1000 LISBOA

Desejo subscrever o curso completo de VIDEOBASIC por 7.500$00.

Junto envio cheque n.º_____ Banco_____

Nome

Morada

Cód. Postal Cidade

(Em vez de cortar o cupão tire uma fotocópia)

Timex Computer 2068 Personal Color

CARACTERÍSTICAS

- Escrita automática de instruções de comando
- Verificação imediata de erros
- Memória de 72 K expansível através de Cartridge com 56 K (Rom)
- Utiliza TIMEX COMMAND Cartridges
- Linguagem de programação BASIC
- Apresentação de Textos e Gráficos
- Capacidades em tratamentos de Texto, Cor e Som
- Ficha de expansão para periféricos
- 42 Teclas tipo Máquina de Escrever

Sensacional!...
...dois computadores num só!

Importante:
...a cartridge emuladora permite-lhe utilizar todo o software existente para o ZX Spectrum.®

TIMEX COMPUTER 2068

® ZX Spectrum é uma marca registada de Sinclair Research, Lda.