CHADE

IKS

INFORMATYKA KOMPUTERY SYSTEMY

CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 3/1986 ISSN 0860-2794

W numerze:

Pióro świetlne

LOGO

BASIC

Program na:

— Commodore

— Spectrum

— Atari

Foto: Ryszard Rogoń

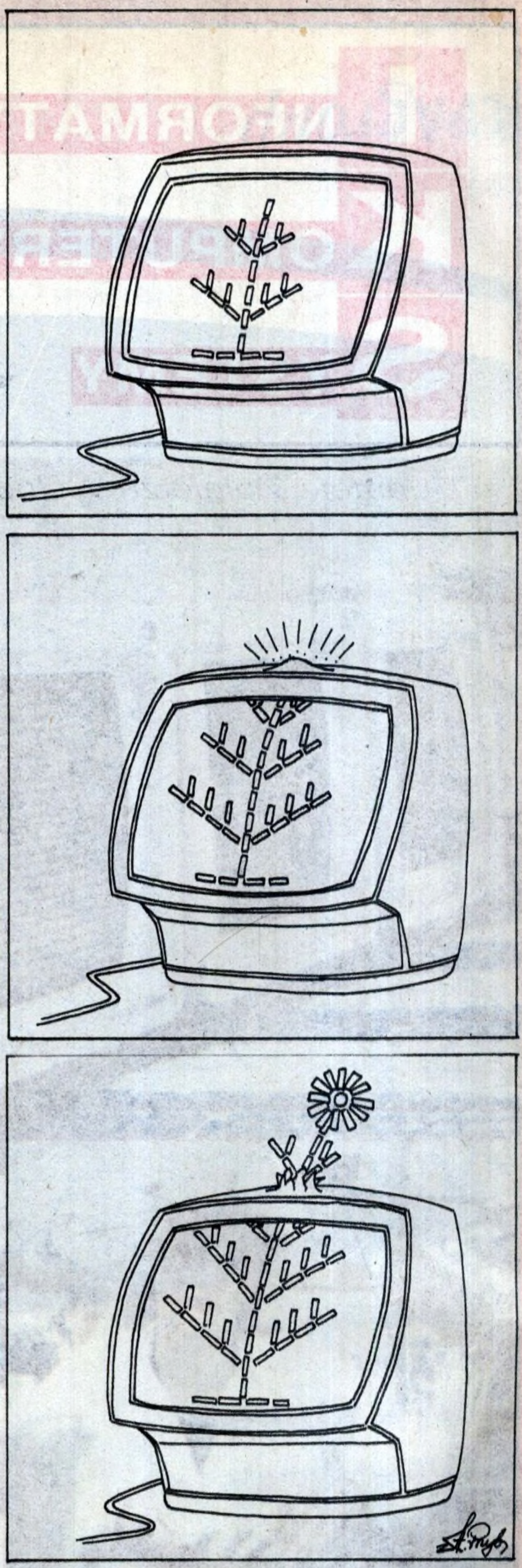
JUŻ wkrótce po ukazaniu się pierwszego numeru „IKS-a" dosłownie zasypani zostaliśmy listami. Ile ich przyszło? Nie liczyliśmy, ale codziennie poczta dostarczała nam ich ponad tysiąc — może jest ich 15, a może 20 tysięcy. Nie wszystkie możemy czytać natychmiast, ale z pewnością wszystkie przeczytamy. Dziękujemy za rady, uwagi i życzenia. Jeśli „IKS" ma być pismem dobrym, to każdy jego numer powstawać musi przy pomocy Czytelników. Stąd też tak niecierpliwie oczekujemy Waszego głosu.

Niezwykle ciekawa jest lektura nadesłanych ankiet. Prawie wszyscy domagacie się publikowania na naszych łamach programów na różne mikrokomputery, jedni proszą o programy gier, inni o programy użytkowe. Oczywiście będziemy je drukować. Ale prasa rządzi się własnymi prawami; nie możemy zamieszczać programów długich. To, co proponujemy, to krótkie konstrukcje programowe, sądzimy jednak, że okażą się ciekawe.

Ale prezentowanie programów nie może być naszym najważniejszym celem. Bo przecież bez względu na ich atrakcyjność wymagają od użytkowników jedynie biernych umiejętności : ciąg komend trzeba wczytać do komputera i potem jednym rozkazem uruchomić wprowadzony program. Nie potrzebna tu jest żadna wiedza — liczy się jedynie sprawność rąk. A gdy zdarza się pomyłka, bo przecież zdarzyć zawsze się może, okazuje się, że nie zawsze można sobie z jej usunięciem poradzić.

JEDNI uważają, że od użytkownika takich umiejętności wymagać nie można; wystarczy, że będzie znał programy, ich przeznaczenie i poradzi sobie z ich uruchomieniem. Zgoda. Dla wielu komputer pozostanie na zawsze czarną skrzynką, urządzeniem o nieznanej konstrukcji, ale poznanych możliwościach. Dociekliwi będą natomiast poszukiwać i im chcemy pomóc. Z „IKS-em" łatwiej będzie usiąść przed komputerem i rozpocząć z nim „rozmowę". Mamy nadzieję, że tak burzliwie rozwijające się kluby mikrokomputerowe sprawią, że dostęp do tych urządzeń będzie łatwiejszy, że komputery trafią „pod strzechy".

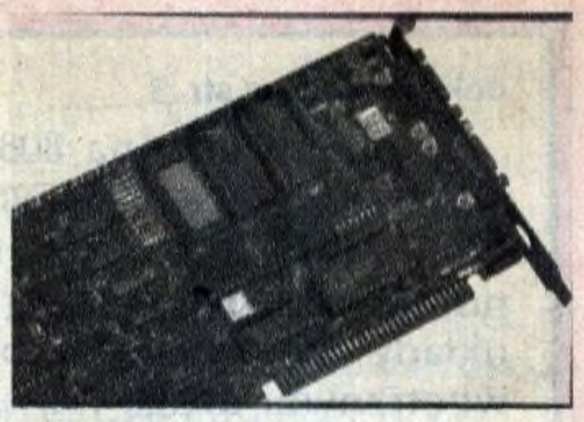
Redakcja



Uwaga! Błąd w programowaniu...

Rys. Michał Przybyłowski

Krótka historia mikroprocesorów



Niemal czterdzieści lat temu, bo w 1947 roku powstał pierwszy tranzystor. Po kilku latach doświadczeń, prawdopodobnie w celach oszczędnościowych, spróbowano umieścić kilka tych elementów w jednej obudowie. Rezultatem poszukiwań był (w 1959 roku) pierwszy układ scalony, tak zwanej *małej skali integracji* (SSI — small — scale integration). Z czasem w jednej obudowie „upakowywano" coraz więcej elementów a produkt finalny elektronicy nazwali „kością". Pojawiła się również *średnia skala integracji,* około 50 tranzystorów w jednej obudowie (MSI — medium — scale integration), następnie *duża skala integracji* — tysiące tranzystorów w jednej obudowie (LSI — large — scale integration). — Ostatnio króluje *bardzo duża skala integracji* — miliony tranzystorów w jednej obudowie (VLSI very large scale integration).

Istniejąca zaledwie od roku firma **Intel** na bazie osiągnięć „upakowania" zbudowała pierwszy układ pamięci półprzewodnikowej, o zawrotnej jak na ówczesne czasy (1969) pojemności — tysiąc (1K) bitów. Był to okres intensywnych prac nad kalkulatorami. Poświęcone im badania, w ciągu niecałych dwóch lat dają efekt w postaci pierwszego mikroprocesora 4004. Zgodnie z przyjętą nomenklaturą mikroprocesory zazwyczaj nie mają nazw literowych, jest to szereg cyfr (minimum czterech), czasami poprzedzonych symbolem literowym. Czterobitowa struktura nowego mikroprocesora wynikała z przeznaczenia — kalkulatory oraz obowiązujące standardy BCD (binary-coded decimal). Innymi słowy chodziło o to, by procesor przetwarzał cyfry dziesiętne — zapisanie ich wymaga właśnie czterech bitów. 4004 wykonuje sześćdziesiąt tysięcy operacji w czasie jednej sekundy.

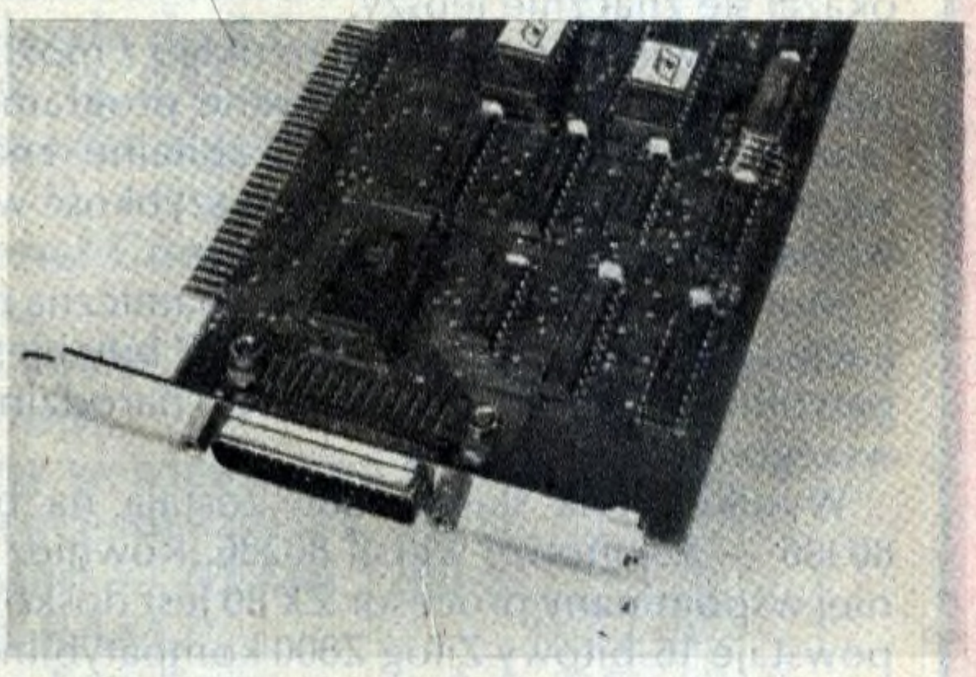
Między innymi konieczność dostosowania się do terminali wykorzystujących standard ASCII (American Standard Code for Information Interchange) przyczyniła się do rozpoczęcia badań nad mikroprocesorem ośmiobitowym. W kwietniu 1972 roku prace te zostają uwieńczone sukcesem — uruchomiono produkcję mikroprocesora 8008. Jego szybkość jest imponująca — ⅓ miliona instrukcji na jedną sekundę. Listę rozkazów tworzy aż 45 różnych instrukcji, które mogą operować na danych z pamięci o pojemności 16 000 (16K) słów.

Wydaje się, że osiągnięcie to stanowiło ostatni element fundamentów mikrokomputera, bowiem już na przełomie wiosny i lata 1974 roku pismo Radio — Elektronics prezentuje pierwszy mikrokomputer MARK-8. Jego twórcą jest Jonathan Titus.

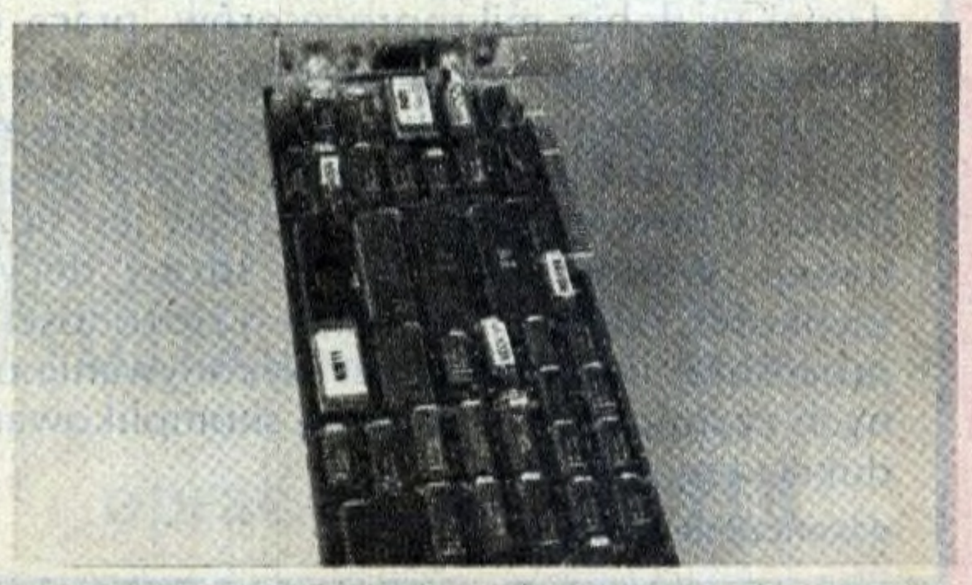
Kolejnym dzieckiem z rodziny mikroprocesorów jest 8080, który realizuje 75 instrukcji, a jego pamięć może mieć 64K słów.

W celach sondażowych firma MITS buduje mikrokomputer ALTAIR 8800 z pamięcią 256 słów (nie 256k!) za 395 $. Jest to cena stosunkowo niska, a producent, nie bez podstaw, zapewniał kupujących, że nabywają swój własny, prywatny, osobisty komputer. Firma MITS mogła sprzedać więcej w jeden dzień, niż planowała sprzedać przez okres całej produkcji ALTAIRa. Sukcesu należy upatrywać w tym, iż architektura tego komputera pozwalała dołączać urządzenia peryferyjne. Jednym z nich jest kontroler dysków zbudowany przez Digital Microsystems. W mikrokomputerze wykorzystano po raz pierwszy system operacyjny CP/M (Control Program for Microcomputers).

dokończenie na str. 4



Masowa produkcja układów scalonych — nowa perspektywa elektroniki



dokończenie ze str. 3

W odpowiedzi na 8080 firma Motorola rozpoczyna produkcję 6080 wraz z układami pozwalającymi łączyć mikroprocesor z peryferiami. Krótko potem rodzina tego procesora powiększa się o układy zapewniające kontakt z urządzeniami zewnętrznymi drogą transmisji równoległej — 6820 oraz szeregowej — 6850.

Jeden z filarów Intela — F. Faggin uznał, nie bez podstaw, iż bardziej kalkuluje się mu założyć własną firmę — Zilog. Wraz z Masatoshi Shima tworzą mikroprocesor „super 8080" — Z80 (4 MHz) 176 instrukcji! (Częstotliwość np.: 4 MHz, 2 MHz określa liczbę podstawowych operacji, jakie może wykonać procesor w czasie jednej sekundy). Był to bezwzględnie lepszy mikroprocesor od swego konkurenta 8080.

Intel „trzymający rękę na pulsie rozwoju mikroprocesorów" buduje w 1976 roku 8085 (3 MHz). Może ze względu na nieobecność F. Faggina, procesor te nie spełnił pokładanych w nim nadziei. Dopiero w 1978 roku prawie pełen sukces odniósł już szesnastobitowy 8086, ale tu wystąpiły kłopoty z oprogramowaniem.

Realizacja pomysłu tak zwanego „busa" dopełniła oczekiwanego sukcesu. Mikroprocesor z „busem" 8088 („młodszy brat" 8086) jest w swej istocie 16-bitowy, ale ma ośmiobitowe połączenie (bus) „do zewnątrz", tam gdzie niepodzielnie panuje 8-bitowy standard.

Podobnie Motorola w 1979 roku prezentuje „szesnastkę" — MC 68000. Mimo skorzystania z doświadczeń Intela w zakresie dobudowania „busa", czego wynikiem był 68008, mikroprocesor 8088 okazał się znacznie lepszy.

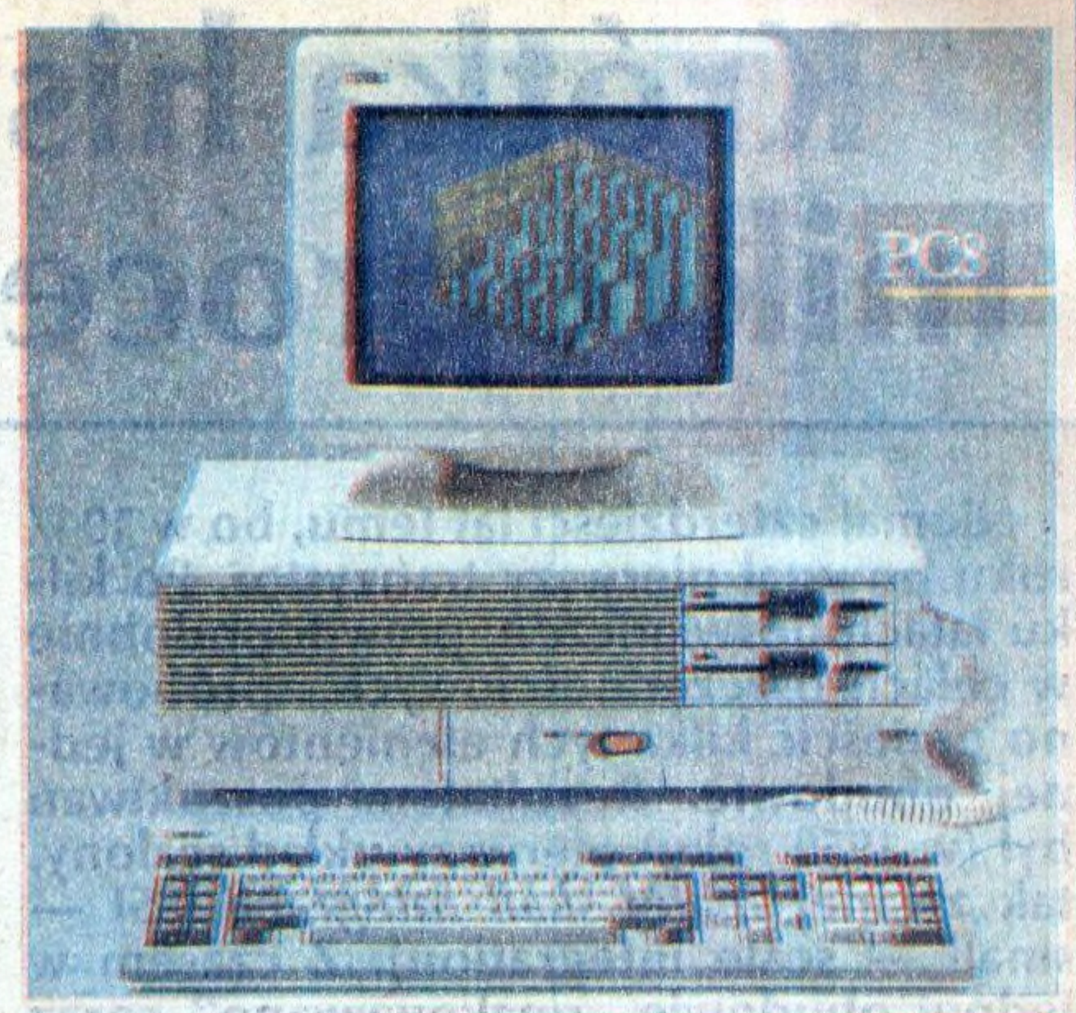
Warto wiedzieć, że 68000 to układ z możliwością mikroprogramowania (tak zwane programowanie „random — logic"). Ma to kapitalne znaczenie wszędzie tam, gdzie istotna jest szybkość wykonywania operacji.

Motorola 68020 pozwala na dynamiczną modyfikację rozmiaru „busa" 8, 16 lub 32 bity i jest kompatybilna z 68000. Zwiastuje to narodziny pierwszego mikroprocesora.

Wysiłki Intela skierowane zostają na budowę 80386 — 32-bitowej wersji 80286. Również wcześniej wspominany procesor ZX 80 jest doskonalony, powstaje 16-bitowy Zilog Z800 kompatybilny z Z80. Niestety prace te skończyły się niepowodzeniem. Jeden z głównych konstruktorów Z800 Masatoshi Shima odszedł, podobnie jak wcześniej jego szef z Intela, i nikt nie potrafił poprawnie dokończyć oprogramowania Z800. Zdarzenie to dowodzi, że i dzisiaj w dobie mikroprocesorów, praca i talent jednostki bywają niezastąpione.

Dotychczasowa praktyka konstruktorów wskazuje, iż jednym z podstawowych dylematów podczas budowy nowych mikroprocesorów jest liczba instrukcji. Wartość ta z jednej strony decyduje o szybkości, z drugiej natomiast o złożoności układu procesora; im krótszy czas wykonywania określonych czynności tym bardziej skomplikowana produkcja tego układu.

dokończenie na str. 5



Amerykańska firma NCR, której siedziba mieści się w Dayton, jest jeszcze mało znana na światowym rynku komputerowym. Oferowane przez nią komputery NCR PC4i, NCR PC6 i NCR PC8 są w pełni kompatybilne z komputerami firmy IBM a ponadto są od nich tańsze. Pracują w oparciu o 16-bitowy procesor 8088 (NCR PC6 i NCR PC4i) i 80286 (NCR PC8). Pojemność pamięci 256KB z możliwością rozszerzenia do 640 KB. Posiadają możliwość dołączenia stacji dysków miękkich i twardych co pozwala na zwiększenie pamięci do 20 MB.

O

dokończenie ze str. 4

Ostatnio podejmowane są starania, by procesory wykonywały bezpośrednio instrukcje języków wyższego poziomu. Przykładem jest procesor NOVIX, który wykonuje instrukcje języka FORTH.

Doskonalenie sprzętu mikro często jest źródłem nieoczekiwanych wytworów, jakim jest np. INMOS — transputer, układ pozwalający na równoległe przetwarzanie danych. Minie prawdopodobnie wiele czasu do jego użytkowego oprogramowania. Jednak perspektywa jest oszałamiająca. Ale już dzisiaj można kupić układ, RISC (reduced instruction set computer), „przyśpieszacz", który po dołączeniu do procesora znacznie skraca czas realizacji instrukcji.

Co dalej? Firma Hitachi proponuje pamięci, w niektórych tradycyjny bit stał się wielowartościowy. Ma cztery stany, a czas dostępu wynosi od 1 do 2 mikrosekund! Dla koneserów mikro Toschiba poleca pamięci RAM o pojemności 1 megabit każdy układ, z czasem dostępu 60 nanosekund.

Z historii mikroprocesorów

1948 Wynaleziono tranzystor.
1959 W Texas Instruments powstaje pierwszy zintegrowany obwód.
1964 J. G. Kemeny, T. E. Kurtz w Dortmouth College stworzyli BASIC
1971 Intel Corp. rozpoczyna sprzedaż 4-bitowego mikroprocesora 4004 za 200 $.
1972 W grudniu Intel proponuje 8-bitowy mikroprocesor 8008.
1973 Początek produkcji 8080.
1974 Gary Kidall opracowuje system operacyjny CP/M.
Zimą Zilog wprowadza do handlu mikroprocesor Z80.
1975 W październiku firma IBM oferuje mikrokomputer w teczce — IBM 5100, Basic, 16K pamięci, cena 9000 $.
1976 Pierwszy 16-bitowy mikroprocesor TM S9000 produkcji Texas Instruments.
W sierpniu pierwszy floppy-disk-drive, 8 cali (1195 $).
1977 Kwiecień — Commodore Business Machines Inc. produkuje komputer PET (procesor 6502, 4K RAM, 14K ROM, klawiatura, monitor, magnetofon.
Lipiec — Apple Computer (6502, 4K RAM, 16K ROM, klawiatura, interfejs do kaset, specjalizowane gniazda, kolorowy monitor.
1978 Grudzień — Zaprezentowana zostaje drukarka mozaikowa MX-80. Powstaje Atari 400 i 800, 8K RAM (do 48K), dwa gniazda do ROM.
1980 Powstaje do dzisiaj popularny na bazie Z80A, 1K RAM, 4K ROM.
Czerwiec — Shugart sprzedaje pierwsze „twarde dyski" typu Winchester — 5¼ cala.
1981 Wrzesień — dostępny w sklepach IBM Personal Computer (PC), mikroprocesor 8088, 64K RAM, 40K ROM, 5¼ cala dyski.
1982 Lato — Commodore 64, procesor 6510, 64K RAM, 20K ROM, dobrej jakości generator dźwięku, kolory, szeregowy interfejs.
LOGO — język programowania jest dostępny w większości komputerów.
Lipiec — Intel wprowadza na rynek mikroprocesory 80186, 80286.
1983 Styczeń — pojawia się doskonalsza wersja Atari — ATARI 1200 XL.
Luty — IBM PC XT, twardy dysk 10 MB, 128K RAM.
Wrzesień — dostępne są ekrany dotykowe — Touchsreen produkcji Hewlett-Packard Company — HP 150.
Październik — Shugart Comp. rozpoczyna produkcję optycznych dysków, tylko do odczytu, pojemność jeden gigabajt.
Dostępne są drukarki laserowe o zdolności nanoszenia 300 punktów na 2,5 cm.
1984 Styczeń — Apple Computer Inc. przedstawia mikrokomputer Macintosh.
Pierwszy zegarkowy komputer (produkcji Seiko Instruments), 10 znaków w każdej z 4 linii na mini-ekranie LCD, 2K CMOS RAM, 6K ROM.
Sinclair QL (procesor 68008, 128K RAM, 2 gniazda na mikrokasety).
Czerwiec — Motorola rozpoczyna produkcję 68020 — 32-bitowy procesor.
Sierpień — IBM PC AT, 80286, 256K RAM, 1,2 MB floppy-disk drive i szereg innych udoskonaleń, w cenie 5469 $.

cdn.

oprac. **W. GOGOŁEK**

ATARI

Plansza gry, przedstawiona na ekranie, składa się z dziewięciu pól. Część z nich, wybrana w sposób losowy jest jasna, pozostałe są ciemne. Zadaniem gracza jest zapalenie 8 pól brzegowych i wygaszenie środkowego. W celu przełączenia wybranego pola należy przesunąć w jego kierunku drążek manipulatora i nacisnąć czerwony przycisk (FIRE). Można wybierać tylko zapalone pola. Zmieniając wybrane pole zmienia się również stan jego sąsiadów stykających się z nim brzegami.

```
  1 REM « LAMIGLOWKA »
  5 GR.2:GOSUB 20000
 12 DIM A$(1),PZ(10)
100 MV=1:SET.4,0,0
110 GOSUB 6000:REM KRESLENIE PLAN
    SZY
115 GOSUB 7000:REM LOSOWANIE POL
116 GOSUB 7100:REM KRESLENIE POL
117 ?"CZY CHCESZ INSTRUKCJE GRY";
    :I.A$
118 IF A$="T"THEN GOSUB 3000:REM
    INSTRUKCJA
120 TRAP 120
121 ?"[CLEAR] RUCH NR "; MV;"?"
130 IF STRIG (JS)=0 THEN 130:REM
    OCZEKIWANIE NA NACISNIECIE
    PRZYCISKU
131 ST=STICK (JS):IF STRIG (JS)=1
    THEN 131
138 GOSUB 140+(ST-5)
139 GOTO 155
140 M=3:RETURN
141 M=9:RETURN
142 M=6:RETURN
144 M=1:RETURN
145 M=7:RETURN
146 M=4:RETURN
148 M=2:RETURN
149 M=8:RETURN
150 M=5:RETURN
155 IF PZ(M)=0 THEN ?"[CLEAR] WY
    BIERAJ TYLKO ZAPALONE POLA
    —TWOJ RUCH": GOTO 130
160 MV=MV+1
190 PZ(M)=0:REM WYLACZENIE WYBRA
    NEGO POLA
191 GOSUB 7100:REM AKTUALIZACJA
    PLANSZY
250 GOSUB 300+M*10
260 MN=0:GOSUB 7100
270 FOR Q=1 TO 9:WN=WN+PZ(Q):
    NEXT Q
280 IF WN=8 AND PZ(5)=0 THEN 7500:
    REM WYGRANA
290 IF WN=0 THEN 7600:REM PRZEGRANA
300 GOTO 120:REM NASTEPNY RUCH
310 PZ(2)=1-PZ(2):PZ(4)=1-PZ(4)
312 PZ(5)=1-PZ(5):RETURN
320 PZ(1)=1-PZ(1):PZ(3)=1-PZ(3)
322 RETURN
330 PZ(2)=1-PZ(2):PZ(5)=1-PZ(5)
332 PZ(6)=1-PZ(6):RETURN
340 PZ(7)=1-PZ(7):PZ(1)=1-PZ(1)
342 RETURN
350 PZ(8)=1-PZ(8):PZ(4)=1-PZ(4)
352 PZ(6)=1-PZ(6):PZ(2)=1-PZ(2)
354 RETURN
360 PZ(9)=1-PZ(9):PZ(3)=1-PZ(3)
362 RETURN
370 PZ(8)=1-PZ(8):PZ(4)=1-PZ(4)
372 PZ(5)=1-PZ(5):RETURN
380 PZ(7)=1-PZ(7):PZ(9)=1-PZ(9)
382 RETURN
390 PZ(8)=1-PZ(8):PZ(5)=1-PZ(5)
392 PZ(6)=1-PZ(6):RETURN
 999 END
1000 JS=99
1010 FOR X=0 TO 3
1020 IF STRIG(X)=0 THEN JS=X:GOSUB
     1100
1030 NEXT X
1040 IF JS>3 THEN 1010
1050 RETURN
1100 IF STRIG(JS)=0 THEN 1100:REM
     CZEKANIE NA ZWOLNIENIE PRZYCI
     SKU
1199 RETURN
```

```
6000 COLOR 3
6010 FOR Z=0 TO 12 STEP 6
6015 PLOT 10,Z:DR.28,Z
6020 FOR X=10 TO 28 STEP 6
6030 FOR Y=1 TO 5
6040 PLOT X,Y+Z
6050 NEXT Y:NEXT X:NEXT Z
6080 PLOT 10,18:DR.28,18
6099 RETURN
7000 FOR R=0 TO 10
7010 PZ(R)=INT(RND(0)+0.5)
7020 NEXT R
7099 RETURN
7100 PN=0
7110 FOR Z=12 TO 0 STEP -6
7120 FOR X=1 TO 13 STEP 6
7122 PN=PN+1:COL.PZ(PN)
7125 FOR Y=1 TO 5
7130 PLOT X+10,Y+Z:DR.X+14,Y+Z
7140 NEXT Y:NEXT X:NEXT Z
7190 RETURN
7500 GOSUB 8000:REM AKTUALIZACJA
7570 ?"[CLEAR] WYGRALES W ";MV;"RU
     CHACH"
7571 SET.4,4,4
7575 ?"NOWA GRA ";:I.A$
7590 IF A$="T" THEN 100
7595 GR.0:END
7600 GOSUB 8000:REM AKTUALIZACJA
7605 ?"[CLEAR] NIE MOZESZ TERAZ WY
     GRAC."
7606 SET.4,12,4
7610 GOTO 7575
8000 NG=NG+1
8010 AV=INT((AV*(NG-1)+MV)/NG)
8099 RETURN
20000 REM STRONA TYTULOWA
20010 ?"[CLEAR]":GR.2:POS.5,4
20020 ?#6;"LAMIGLOWKA"
20030 ?"  GRA WYMAGA MANIPULATORA!  "
20040 ?"nacisnij czerwony przycisk w celu "
20050 ?"   rozpoczecia gry   "
20060 JS=99
20070 FOR X=0 TO 3
20080 IF STRIG(X)=0 THEN JS=X
20090 NEXT X
20100 IF JS>3 THEN 20070
20110 GR.3:RETURN
```

```
30000 ?"[CLEAR] CELEM GRY JEST ZAPA
      LENIE WSZYSTKICH"
30010 ?"POL PLANSZY OPROCZ SRODKO
      WEGO."
30020 ?"(C.D. PO NACISNIECIU <FIRE>)...
      ":GOSUB 1000
30030 ?"[CLEAR] PRZELACZASZ POLE PO
      PRZESUNIECIU"
30040 ?"DRAZKA MANIPULATORA W JEGO
      KIERUNKU"
30050 ?"I NACISNIECIU CZERWONEGO
      PRZYCISKU":GOSUB 1000
30060 ?"[CLEAR] MOŻESZ WYBIERAC TYL
      KO ZAPALONE POLA."
30070 ?"ZMIENIAJAC WYBRANE POLE
      ZMIENIASZ"
30080 ?"STAN JEGO SASIADOW.":GOSUB
      1000
30090 ?"[CLEAR] PAMIETAJ: RUCH MANI
      PULATORA W GORE"
30100 ?"WYBIERA GORNY SRODEK, W
      DOL — DOLNY"
30110 ?"SRODEK; PODOBNIE INNE KIERU
      NKI:"GOSUB 1000
30120 ?"[CLEAR] PO DOKONANIU WYBO
      RU NACISNIJ"
30130 ?"CZERWONY PRZYCISK (TRZYMA
      JAC JEDNA"
30140 ?"REKA DRAZEK MANIPULATORA)."
      :GOSUB 1000
30150 ?"[CLEAR] JESLI WYLACZYSZ NARO
      ZNIK, TO"
30160 ?"ZMIENISZ TAKZE JEGO TRZECH
      SASIADOW.":GOSUB 1000
30170 ?"[CLEAR] JESLI WYLACZYSZ SRO
      DEK BOKU, TO"
30180 ?"ZMIENISZ TAKZE JEGO DWOCH
      SASIADOW NA TYM"
30190 ?"SAMYM BOKU.":GOSUB 1000
30200 ?"[CLEAR] JESLI WYLACZYSZ SRO
      DEK PLANSZY, TO"
30210 ?"ZMIENISZ TAKZE SRODKOWE
      KWADRATY"
30220 ?"NA KAZDYM BOKU.":GOSUB 1000
30230 RETURN
```

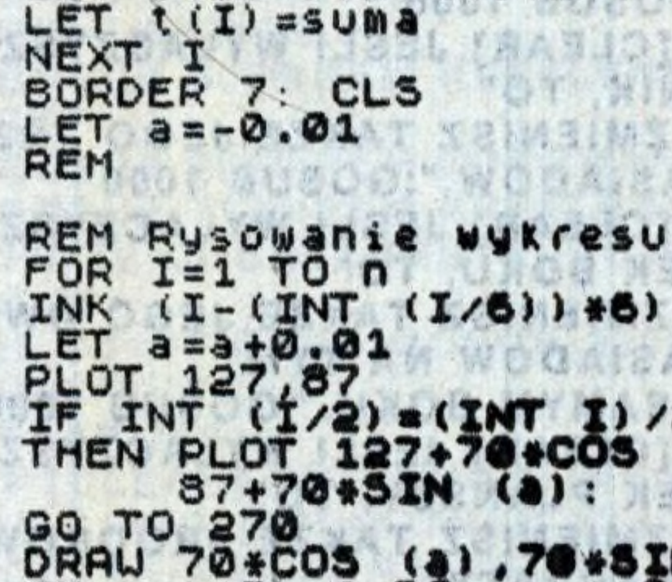
W następnym numerze gra losowa: „Pole minowe"

PROGRAM 12

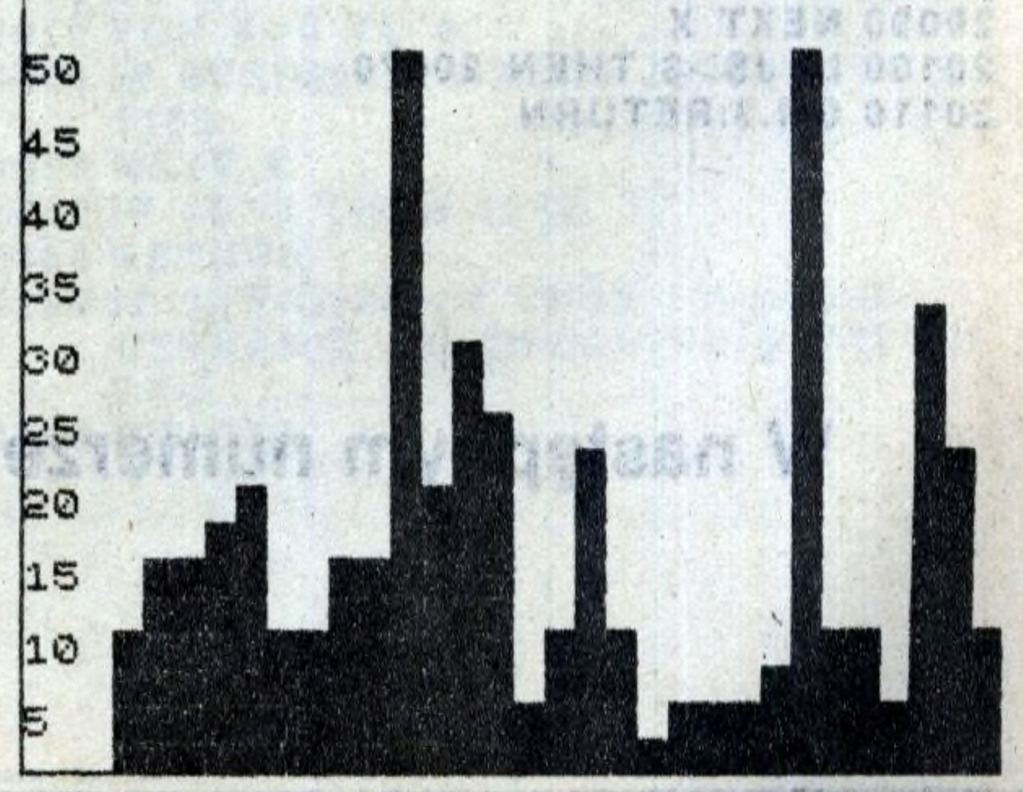
SPECTRUM

„Histogram" i „Wykres kołowy" są programami napisanymi na „Spectrum". Przeznaczone są do graficznego zobrazowania danych — dzięki nim wielkości liczbowe z łatwością przenoszone są na odpowiednie wykresy (mogą przydać się w szkole).

```
 10 REM WYKRES KOLOWY
 20 REM
 30 BORDER 1: PAPER 7:
    INK 1: CLS
 40 PRINT AT 9,8;
    "WYKRES KOLOWY"
 50 PRINT AT 18,4; FLASH 1;
    "nacisnij dowolny klawisz":
    PAUSE 0: CLS
 60 INPUT "Ile liczb wprowadzis
z? ";n
 70 DIM t(n)
 80 LET suma=0
 90 REM

100 REM Wczytanie kolejnych
       liczb
110 FOR I=1 TO n
120 PRINT AT 18,5;"Liczba nr ";
    I
130 BEEP .1,5
140 INPUT liczba
150 LET suma=suma+liczba:
    LET t(I)=suma
160 NEXT I
170 BORDER 7: CLS
180 LET a=-0.01
190 REM

200 REM Rysowanie wykresu
210 FOR I=1 TO n
220 INK (I-(INT (I/6))*6)/2
230 LET a=a+0.01
240 PLOT 127,87
250 IF INT (I/2)=(INT I)/2
    THEN PLOT 127+70*COS (a),
         87+70*SIN (a):
    GO TO 270
260 DRAW 70*COS (a),70*SIN (a)
270 IF a<t(I)*2*PI/suma-0.01
    THEN GO TO 230
280 NEXT I
290 INPUT "Nastepny wykres (Tak
,Nie)?";r$
300 IF r$(1)="T" OR r$(1)="t"
    THEN GO TO 170
310 STOP
```



```
 10 REM HISTOGRAM
 20 REM
 30 BORDER 7: PAPER 7:
    INK 1: CLS
 40 PRINT AT 3,10;"HISTOGRAM"
 50 PRINT AT 6,1;"Program wyswi
etlania danych w    postaci wykr
esu kolumnowego"
 60 PRINT AT 17,4; FLASH 1;
    "nacisnij dowolny klawisz"
 70 PAUSE 0: CLS
 80 INPUT "Ile liczb wprowadzis
z(max 29)?  ";illiczb
 90 IF illiczb>=30 THEN
       GO TO 80
100 IF illiczb<1 THEN GO TO 80
110 LET illiczb=INT (illiczb)
120 DIM v(illiczb)
130 LET max=0
140 REM
150 REM Wczytanie kolejnych
       liczb
160 FOR I=1 TO illiczb
170 PRINT AT 18,5; INK 2;
    "Liczba nr "
180 BEEP .1,5:
    PRINT AT 18,15;I
190 INPUT v(I)
200 IF v(I)>max THEN
       LET max=v(I)
210 NEXT I
220 CLS
230 REM

240 REM Przeskalowanie
250 LET skala=1
260 IF max>20 THEN
       LET skala=max/20
270 FOR I=1 TO illiczb
280 LET v(I)=INT (v(I)/skala)
290 NEXT I
300 REM

310 REM Rysowanie osi liczbowej
320 FOR I=1 TO 20 STEP 2
330 PRINT AT 21-I,0;
   INT (I*skala+.5)+INT (skala)
340 NEXT i
350 PLOT 0,0
360 DRAW 0,175
370 PLOT 255,0
380 DRAW -255,0
390 REM

400 REM Rysowanie histogramu
410 FOR I=1 TO illiczb
420 FOR J=1 TO v(I)
430 PRINT AT 22-J,2+I;"■"
440 NEXT J
450 NEXT I
460 STOP
```



PROGRAM 13 SPECTRUM

PROGRAM KORELACJE

UWAGA STATYSTYCY! Program ten pozwala na obliczenie wielkości statystycznej zależności między szeregami dwóch kolumn liczb. Wynikiem są wartości współczynników Spearmana i Pearsona.

Ze względów praktycznych adresy instrukcji wydzielają moduły programu. Ma to na celu ułatwienie ewentualnych udoskonaleń: kontrola zakresów danych wejściowych, zamierzone ich przekształcanie np.: $a = a^{-1}$, $a = a^2$, $a = \ln a$. Przekształcenia danych wejściowych pozwalają na stwierdzenie innych, poza liniowymi, zależności; wykładniczych, logarytmicznych itp.

```
  10 REM Wspolczynniki korelacji
       PEARSONA i SPEARMANA
  20 GO SUB 4000
  30 CLS : INPUT "Podaj liczbe d
anych" k
  40 DIM a(k)
  50 FOR n=1 TO k
  60 INPUT "Napisz wartosc a";VA
L "n";"=",a(n)
  70 NEXT n
  80 PRINT '''"Koniec wprowadzani
a danych typu a"''
  85 PRINT '''"Kontrolny list dan
ych a(n)"
  90 FOR n=1 TO k
 100 PRINT "a";n;"=";a(n)
 110 NEXT n
 120 LET t$="t"
 130 INPUT "Czy  chcesz poprawic
 dane wejs- ciowe?, jesli TAK na
cisnij kla- wisz litery t, jesli
 NIE - kla- wisz n",u$
 140 IF t$=u$ THEN GO TO 200
 150 GO TO 1030
 200 INPUT "Podaj numer danej kt
ora chcesz   zmienic"'n
 210 INPUT "Napisz poprawna wart
osc a";VAL "n",a(n)
 220 GO TO 85
1040 DIM b(k)
1050 FOR n=1 TO k
1060 INPUT "Napisz wartosc b";VA
L "n";"=",b(n)
1070 NEXT n
1080 PRINT "Koniec wprowadzania
b"''
1085 PRINT '''"Kontrolny list b(n
)"''
1090 FOR n=1 TO k
1100 PRINT "b";n;"=";b(n)
1110 NEXT n
1120 INPUT "Czy chcesz poprawiac
 dane wejs- ciowe typu b(n)?",u$
1130 IF t$=u$ THEN GO TO 1200
1140 GO TO 2000
1200 INPUT "Podaj numer danej kt
ora chcesz   zmienic",n
1210 INPUT "Napisz poprawna wart
osc b";VAL "n",b(n)
1220 GO TO 1085
2040 LET Nab=0: LET Sa=0: LET Sb
=0: LET ka=0: LET ksa=0: LET kb=
0: LET Ksb=0: LET r=0
2050 FOR n=1 TO k
2100 LET Nab=a(n)*b(n)+Nab
2110 LET Sa=a(n)+Sa
2120 LET Sb=b(n)+Sb
2130 LET ka=a(n)*a(n)+ka: LET ks
a=Sa*Sa: LET kb=b(n)*b(n)+kb: LE
T Ksb=Sb*Sb
2135 NEXT n
2140 REM   r  - wsp. korelacji
2150 LET r=(k*Nab-Sa*Sb)/SQR ((k
*ka-ksa)*(k*kb-Ksb))
2160 PRINT '''"Wspolczynnik kore
acji Pearsona r wynosi:",,,r
2165 GO SUB 4000
2167 PRINT '''"Obliczanie wspol
czynnika      SPEARMANA po nac
isnieciu      dowolnego klawis
za!": PAUSE 0: CLS
2170 PRINT "Ilustracja rang dla
korelacji   kolejnosci  Spearman
a:"''
2900 LET D=0
3000 FOR n=1 TO k
3002 LET l=k
3004 LET m=k
3010 FOR s=1 TO k
3020 IF a(n)<=a(s) THEN GO TO 30
40
3030 LET l=l-1
3040 IF b(n)<=b(s) THEN GO TO 30
60
3050 LET m=m-1
3060 NEXT s
3080 LET D=(l-m)*(l-m)+D
3085 PRINT "Ranga a(";n;")=";l,"
b(";n;")=";m
3090 NEXT n
3106 LET Sp=0
3110 LET Sp=1-((6*D)/(k*(k↑2-1))
)
3120 PRINT '"Wartosc wspolczynni
ka Spearmana wynosi:",,Sp
3125 GO SUB 4000
3130 PRINT '''"Po nacisnieciu do
wolnego        klawisza program
rozpocznie     prace od nowa!"
3140 PAUSE 0: GO TO 20
4000 BEEP .1,14: BEEP .1,14: BEE
P .2,5: BEEP .1,14: BEEP .1,14:
BEEP .2,5: BEEP .1,14: BEEP .1,1
4: BEEP .1,5: BEEP .1,6: BEEP .2
,7: BEEP .2,5: BEEP .1,15: BEEP
.2,6: BEEP .1,15: BEEP .1,14: BE
EP .2,5: BEEP .1,14: BEEP .1,4:
BEEP .1,15: BEEP .1,14: BEEP .1,
6: BEEP .2,5: BEEP .2,5
6010 RETURN
```

PROGRAM 14

COMMODORE C-64

Jest to propozycja na C-64 dla wszystkich, którzy chcą wykorzystać w swoich programach sprite, czyli ruchomy obiekt. Program ten pozwala opisać postać sprite-a na siatce projektowej, równocześnie wyświetlając jego obraz w prawym dolnym rogu ekranu; pozwala również na podstawie utworzonego obrazu sprite wygenerować wiersze programu w Basic-u zawierające jego zapis liczbowy zgodnie z wymaganiami mikrokomputera C-64.

Szczegółowy opis niektórych opcji (pozostałe wyczerpująco wyjaśnione są w programie):

M — wybór rodzaju grafiki sprite-a. Możliwe są dwa rodzaje grafiki: dwukolorowa i wielobarwna. W grafice dwukolorowej jedno okno siatki projektowej odpowiada jednemu punktowi świetlnemu w obrazie sprite-a. W grafice wielobarwnej para okien w poziomie interpretowana jest jako jeden punkt obrazu sprite. Naciskanie klawisza M powoduje przechodzenie od jednego rodzaju grafiki do drugiego.

1, 2, 3 — wybór kolorów obrazu sprite. Przy pracy w trybie dwukolorowym naciskając klawisz 2 możemy zmieniać kolor sprite, pozostałe klawisze (1 i 3) nie działają. Przy pracy w trybie wielokolorowym różnym parom okien siatki projektowej odpowiadają różne klawisze i naciskając klawisze 1, 2 lub 3 można zmieniać kolory odpowiednich punktów obrazu sprite.

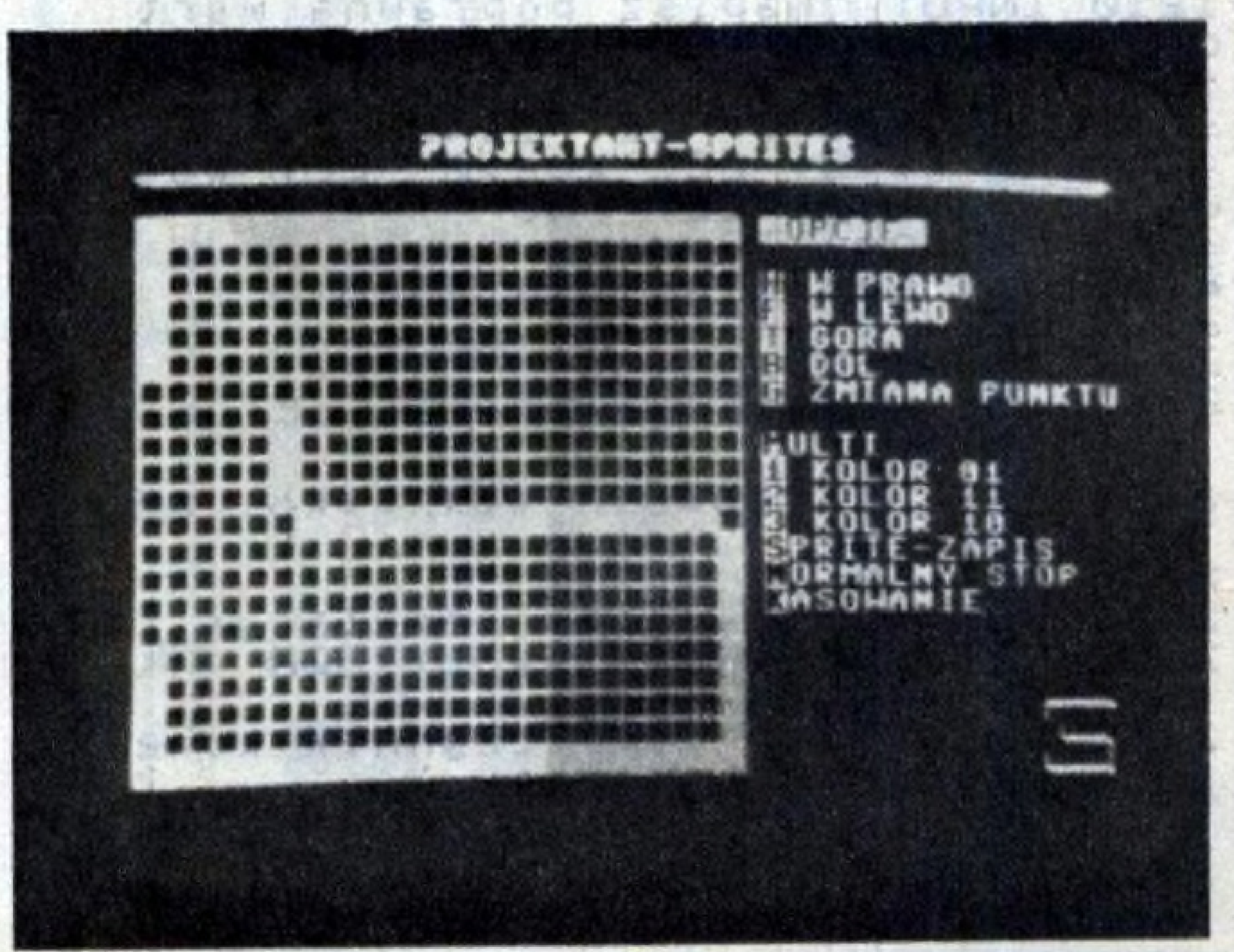
Z — zapis interpretacji liczbowej zaprojektowanego sprite do pamięci mikrokomputera, jako ciąg kolejnych wierszy programu. Numer pierwszego wiersza jest wprowadzany z klawiatury i nie może być większy od 994. Postać pojedyńczego wiersza:
nr wiersza DATA liczba, liczby, ..., liczba

K — zakończenie pracy programu poprzez usunięcie z pamięci wszystkich jego wierszy w Basic-u. Pozostają tylko wiersze utworzone opcją Z.

N — normalne zakończenie pracy programu (program pozostaje w pamięci).

UWAGA: Podczas uruchamiania programu należy posiadać jego kopię na taśmie lub dysku. Wersja rezydująca w pamięci mikrokomputera w przypadku wystąpienia błędów (powstałych w trakcie przepisywania kodu programu z „IKS"-a) może ulec uszkodzeniu.

```
1000 POKE53280,0:POKE53281,0
1001 PRINT"          PROJEKTANT-SPRIT
ES" FORI=0TO39:PRINT"-",:NEXT
1002 PRINT"
1003 FORI=1TO21:PRINT"
   ":NEXT
1004 POKE2040,14:V=53248:POKE650,255:FOR
I=896TO960:POKEI,0:NEXT:POKEV+21,1:POKEV
+16,1
1005 POKEV,60:POKEV+1,220:X=0:Y=0
1006 PRINT"    "TAB(25)" OPCJE ":PRIN
T
1007 PRINTTAB(25)" W PRAWO":PRINTTAB(
25)" W LEWO":PRINTTAB(25)" GORA"
1008 PRINTTAB(25)" DOL"
1009 PRINTTAB(25)" ZMIANA PUNKTU"
1009 PRINTTAB(25)"MULTI":PRINTTAB(25)"
1010 PRINTTAB(25)" KOLOR 11"
1  KOLOR 01":PRINTTAB(25)" KOLOR 11"
1011 PRINTTAB(25)" KOLOR 10":PRINTTAB
(25)"SPRITE-ZAPIS"
1012 PRINTTAB(25)"NORMALNY STOP":PRINT
TAB(25)"KASOWANIE"
1013 GETA$:IFA$=""THENGOSUB1049:GOTO1013
1014 IFA$="T"ORA$="B"ORA$="F"ORA$="H"ORA
$="G"THEN1020
1015 IFA$="S"THEN1028
1016 IFA$="M"ORA$="1"ORA$="2"ORA$="3"THE
N1044
1017 IFA$="K"THEN1054
1018 IFA$="N"THENPRINT"":POKEV+21,0:END
1019 GOTO1013
1020 X=X+(A$="F")-(A$="H"):Y=Y+(A$="T")-
(A$="B")
1021 IFX>23THENX=23
1022 IFY>20THENY=20
1023 IFX<0THENX=0
1024 IFY<0THENY=0
1025 Z=1144+X+Y*40:ZZ=PEEK(Z):YY=Y*3+INT
(X/8)+896:IFA$<>"G"THEN1013
1026 IFZZ=160THENPOKEZ,76:POKEYY,PEEK(YY
)AND255-(2↑(7-(X-INT(X/8)*8))):GOTO1013
1027 POKEZ,160:POKEYY,PEEK(YY)OR2↑(7-(X-
INT(X/8)*8)):GOTO1013
1028 PRINT"                    "TAB(25)"
NR WIERSZA":PRINTTAB(25):INPUTNR$
1029 PRINT"   ":PRINTTAB(25)"
":PRINTTAB(25)"
1030 L=VAL(NR$):IFL<0ORL>999THEN1028
1031 POKEV+28,0:POKEV,255:K=1020:GOSUB10
51:L=895:K=1022:GOSUB1051
1032 DEF FNLI(K)=PEEK(K)*256+PEEK(K+1):P
RINT""FNLI(1020)"D"":K=1022:FORI=0TO9
1033 GOSUB1052:PRINTPEEK(FNLI(1022))",";
:NEXT:GOSUB1052:PRINTPEEK(FNLI(1022))
1034 POKE198,9:POKE631,19:POKE632,13:POK
E633,71:POKE634,111:POKE635,49:POKE636,4
8
1035 POKE637,51:POKE638,54:POKE639,13:EN
D
1036 DEF FNLI(K)=PEEK(K)*256+PEEK(K+1):K
=1020:GOSUB1052
1037 IFFNLI(1020)=1000THENPRINT"PRZEKRO
CZONO ZAKRES NUMERACJI WIERSZY":END
1038 IFFNLI(1022)<949THEN1032
1039 PRINT""FNLI(1020)"D"":K=1022:FORI
=0TO7
1040 GOSUB1052:PRINTPEEK(FNLI(1022))",";
:NEXT:GOSUB1052:PRINTPEEK(FNLI(1022))
1041 POKE198,9:POKE631,19:POKE632,13:POK
E633,71:POKE634,111:POKE635,49:POKE636,4
8
1042 POKE637,52:POKE638,51:POKE639,13:EN
D
1043 GOTO1000
1044 IFA$<>"M"THEN1047
1045 IFPEEK(V+28)=1THENPOKEV+28,0:GOTO10
13
1046 POKEV+28,1:GOTO1013
1047 A=VAL(A$):I=PEEK(V+36+A)AND15:I=I+1
:IFI<16THENPOKEV+36+A,IOR240:GOTO1013
1048 POKEV+36+A,IOR240:GOTO1013
1049 Z=1144+X+Y*40:IFPEEK(Z)=160THENPOKE
Z,76:POKEZ,160:POKEZ,76:POKEZ,160:RETURN
1050 POKEZ,160:POKEZ,76:POKEZ,160:POKEZ,
76:RETURN
1051 POKEK,L/256:POKEK+1,L-(INT(L/256)*2
56):RETURN
1052 L=PEEK(K+1)+1:IFL>255THENPOKEK,PEEK
(K)+1:L=0
1053 POKEK+1,L:RETURN
1054 POKE1020,0
1055 PRINT""PEEK(1020)+1000:POKE198,9
:POKE631,19:POKE632,13:POKE633,71
1056 POKE634,111:POKE635,49:POKE636,48:P
OKE637,53:POKE638,55:POKE639,13:END
1057 POKE1020,PEEK(1020)+1:IFPEEK(1020)<
56THEN1055
1058 PRINT"1056     1057     1058":POKE
198,4:POKE631,19:POKE632,13:POKE633,13:P
OKE634,13:END
```

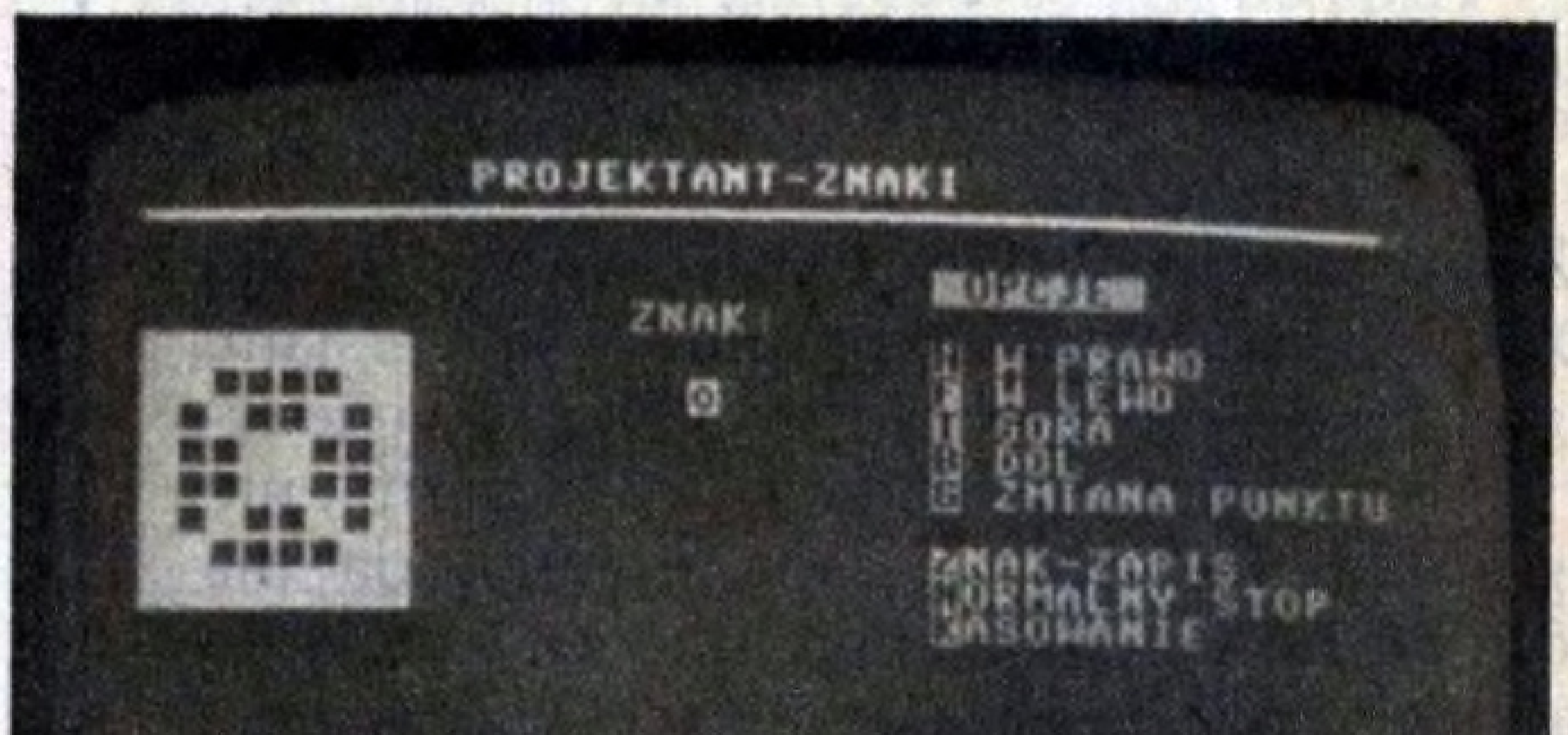


COMMODORE C-64

Projektant — znaki

Jest to program użytkowy na C-64 wspomagający projektowanie własnych znaków. Tworzony znak na siatce projektowej jest widoczny w środkowej części ekranu. Znaczenie opcji jest takie jak w programie „Projektant — sprites”.

```
1000 POKE53280,0:POKE53281,0
1001 PRINT"     PROJEKTANT-ZNAKI"
:FORI=0TO39:PRINT"-";:NEXT
1002 PRINT"      "
1003 FORI=1TO8:PRINT"        ":NEXT
1004 POKE2040,14:V=53248:POKE650,255:FOR
I=896TO960:POKEI,0:NEXT:POKEV+21,1
1005 POKEV,160:POKEV+1,100:X=0:Y=0
1006 PRINT"    "TAB(25)"  OPCJE  ":PRIN
TTAB(15)"ZNAK:"
1007 PRINTTAB(25)" H  W PRAWO":PRINTTAB(
25)" F  W LEWO":PRINTTAB(25)" T  GORA"
1008 PRINTTAB(25)" B  DOL"
1009 PRINTTAB(25)" G  ZMIANA PUNKTU"
1010 PRINTTAB(25)" Z NAK-ZAPIS"
1011 PRINTTAB(25)" N ORMALNY STOP":PRINT
TAB(25)" K ASOWANIE"
1012 GETA$:IFA$=""THENGOSUB1036:GOTO1012

1013 IFA$="T"ORA$="B"ORA$="F"ORA$="H"ORA
$="G"THEN1018
1014 IFA$="Z"THEN1027
1015 IFA$="K"THEN1038
1016 IFA$="N"THENPRINT" ":POKEV+21,0:END

1017 GOTO1012
1018 X=X+(A$="F")-(A$="H"):Y=Y+(A$="T")-
(A$="B")
1020 IFX>7THENX=7
1021 IFY>7THENY=7
1022 IFX<0THENX=0
1023 IFY<0THENY=0
1024 Z=1224+X+Y*40:ZZ=PEEK(Z):YY=Y*3+INT
(X/8)+896:IFA$<>"G"THEN1012
1025 IFZZ=160THENPOKEZ,76:POKEYY,PEEK(YY
)AND255-(2↑(7-(X-INT(X/8)*8))):GOTO1012
1026 POKEZ,160:POKEYY,PEEK(YY)OR2↑(7-(X-
INT(X/8)*8)):GOTO1012

1027 PRINT"                    "TAB(25)"NR
WIERSZA":PRINTTAB(25):INPUTNR$
1028 PRINT"   ":PRINTTAB(25)"
 ":PRINTTAB(25)"          "
1029 L=VAL(NR$):IFL<0ORL>999THEN1027
1030 POKEV+28,0:POKEV,255
1031 PRINT" "L"D↑";:FORI=896TO914STEP3
1032 PRINTPEEK(I)" ,";:NEXT:PRINTPEEK(91
7)
1033 POKE198,9:POKE631,19:POKE632,13:POK
E633,71:POKE634,111:POKE635,49:POKE636,4
8
1034 POKE637,51:POKE638,53:POKE639,13:EN
D
1035 GOTO1000
1036 Z=1224+X+Y*40:IFPEEK(Z)=160THENPOKE
Z,76:POKEZ,160:POKEZ,76:POKEZ,160:RETURN

1037 POKEZ,160:POKEZ,76:POKEZ,160:POKEZ,
76:RETURN

1038 POKE1020,0
1039 PRINT" "PEEK(1020)+1000;:POKE198,9
POKE631,19:POKE632,13:POKE633,71
1040 POKE634,111:POKE635,49:POKE636,48:P
OKE637,52:POKE638,49:POKE639,13:END
1041 POKE1020,PEEK(1020)+1:IFPEEK(1020)<
40THEN1039
1042 PRINT" 1040      1041      1042":POKE
198,4:POKE631,19:POKE632,13:POKE633,13:P
OKE634,13:END
```



1

LOGO

LOGO to język programowania, zasłużenie zdobywający sobie coraz większą popularność wśród użytkowników mikrokomputerów. Wszyscy interesujący się programowaniem słyszeli z pewnością o LOGO, a wielu podejmuje próby pisania programów w tym języku. LOGO został stworzony w latach siedemdziesiątych przez zespół kierowany przez Seymoura Paperta w Massachusetts Institute of Technology. Zaprojektowano go z myślą o początkujących programistach. LOGO doskonale nadaje się do celów edukacyjnych, a zarazem jest językiem o dużej sile (powstał na bazie LISP-u) i można w nim łatwo programować naprawdę ciekawe i skomplikowane problemy. Pozwala również na uzyskanie ciekawych efektów programistycznych już po krótkiej nauce. Wyrabia też właściwe nawyki na przyszłość.

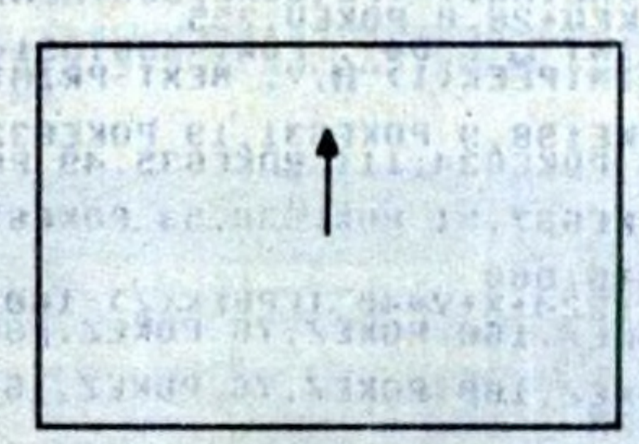
Można w nim w zwięzły i naturalny sposób zaprogramować wiele problemów, a jasna diagnostyka błędów i wbudowane w język komendy edytora pozwalają szybko znaleźć i usunąć pomyłki. Po prostu większość żmudnej pracy wykonuje za nas komputer. Człowiek koncentruje się tylko na celowym wysiłku intelektualnym.

Znane dotychczas w Polsce wersje języka LOGO na różne mikrokomputery są wersjami angielskimi — to znaczy słowa kluczowe języka i komunikaty o błędach mają brzmienie angielskie. Jest to nie lada przeszkoda dla osób, które nie znają angielskiego, a głównie dla dzieci i młodzieży. Dopiero LOGO w wersji narodowej nabiera pełnej wartości. Opracowano już standard polskiego LOGO, a niebawem będzie rozpowszechniany jego translator na ZX Spectrum — najpopularniejszy obecnie w Polsce komputer osobisty. Dlatego też, rozpoczynając w naszym piśmie kurs LOGO, chcemy od razu operować polskim słownictwem. Naturalnie obok polskich terminów będziemy podawali ich angielskie odpowiedniki, by ci, którzy dysponują angielskim translatorem, mogli sprawdzać na komputerze swoje programy. W szczegółach będziemy opierać się na wersji języka dla mikrokomputera ZX Spectrum.

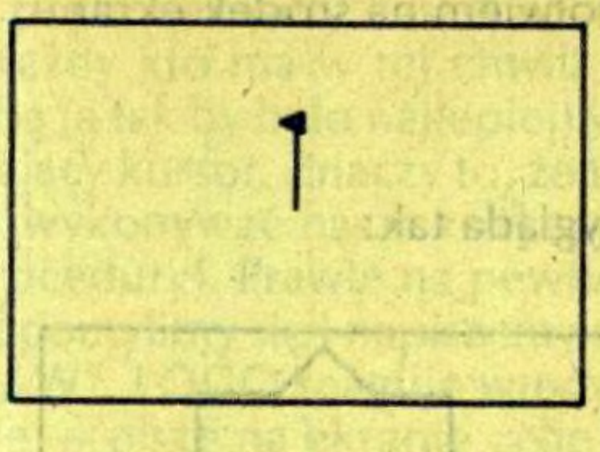
LOGO zawiera pewien zasób słów, które są dla niego zrozumiałe. Są to tak zwane słowa kluczowe. Posługując się tym zbiorem słów można pisać programy, które rysują na ekranie, manipulują słowami i listami, dokonują obliczeń. Liczbę słów rozumianych przez LOGO można zwiększyć, definiując nowe pojęcia przy użyciu już znanych — pisząc procedury o określonych nazwach.

Rysunki wykonuje w LOGO żółw. Jest to abstrakcyjny obiekt, wyobrażony na ekranie komputera w postaci małego trójkąta. Żółwiowi można wydawać różne rozkazy, każąc mu poruszać się po ekranie. Żółw rysuje ciągnąc za sobą kreskę. Aktualny stan żółwia opisują: jego pozycja na ekranie określona w układzie współrzędnych, którego środek pokrywa się ze środkiem ekranu, i kierunek, w którym jest zwrócona głowa żółwia. Żółw może być widoczny albo nie. Gdy napiszemy PŻ (od pokaż żółwia) ang. SHOWTURTLE i odpowiednio skrót ST — trójkącik pojawi się na ekranie. Komenda SŻ „schowaj żółwia", ang. HIDETURTLE HT, sprawi, że zniknie. Na początku pracy z LOGO komenda PŻ spowoduje, że trójkąt pokaże się na środku ekranu i będzie skierowany do góry. Jest to początkowe położenie żółwia. Ma on wtedy pozycję (0, 0) w układzie współrzędnych i przypisany kąt 0 stopni. Jeśli kiedykolwiek później będziemy chcieli, by żółw znalazł się znów w tym miejscu, wystarczy podać komendę WRÓĆ (HOME).

NAPRZÓD (NP FORWARD FD) to komenda przesuwająca żółwia do przodu. Trzeba określić jak daleko ma on się posunąć. Dlatego też, po słowie NAPRZÓD, podajemy parametr. Na przykład NAPRZÓD 30 przesunie żółwia o 30 kroków w kierunku, w jakim jest on ustawiony:



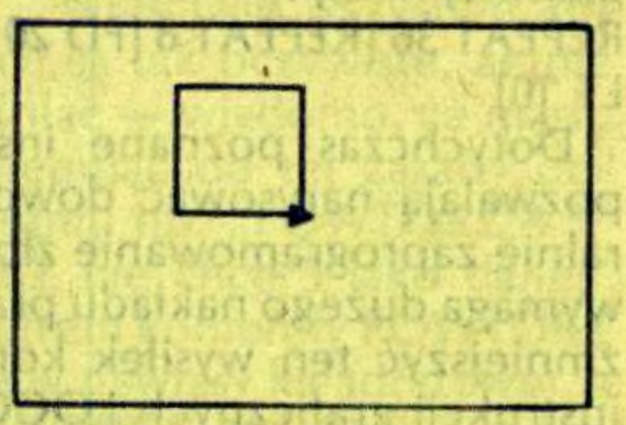
Żółw zmienił swoje położenie, ale w dalszym ciągu „patrzy" w tę samą stronę — w górę. Gdy chcemy, aby żółw skręcił w bok, musimy kazać mu się obrócić w lewo bądź w prawo. Robimy to przy pomocy komend PRAWO (PW RIGHT RT) i LEWO (LW LEFT LT), których parametrami są liczby stopni, o jakie żółw powinien się obrócić. LEWO 90 zmieni kierunek żółwia tak:



Posługując się poznanymi komendami, możemy już rysować różne figury geometryczne. Zanim to zrobimy, warto przygotować sobie ekran do rysowania — zmazać to, co było na nim poprzednio narysowane. Robimy to podając komendę CZYŚĆ (CS CLEARSCREEN CS). Po wykonaniu tej komendy żółw wraca do położenia początkowego, pośrodku ekranu. Gdybyśmy chcieli wyczyścić ekran, nie zmieniając pozycji żółwia, musimy podać komendę ZMAŻ (CLEAN). Jak łatwo zauważyć, napisanie kolejno WRÓĆ i ZMAŻ daje ten sam efekt co pojedyncza komenda CZYŚĆ.

Na czystym ekranie narysujmy kwadrat:

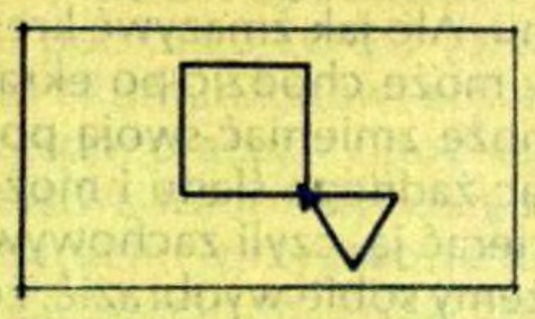
```
NAPRZÓD 50
LEWO 90
NAPRZÓD 50
LEWO 90
NAPRZÓD 50
LEWO 90
NAPRZÓD 50
```



Jeśli teraz narysujemy trójkąt równoboczny, pisząc kolejno komendy:

```
NAPRZÓD 30
PRAWO 120
NAPRZÓD 30
PRAWO 120
NAPRZÓD 30
```

uzyskamy taki efekt na ekranie:



Przy rysowaniu kwadratu i trójkąta powtarzaliśmy kilkakrotnie te same komendy z tymi samymi parametrami. Można zapisać to krócej, używając komendy POWTÓRZ (REPEAT). Po słowie POWTÓRZ podajemy, ile razy ma nastąpić powtórzenie, a dalej listę instrukcji do wielokrotnego wykonania, ujętą w nawiasy kwadratowe. Kwadrat możemy zatem narysować w ten sposób (posługując się dla jeszcze większego skrótu skróconymi nazwami komend):

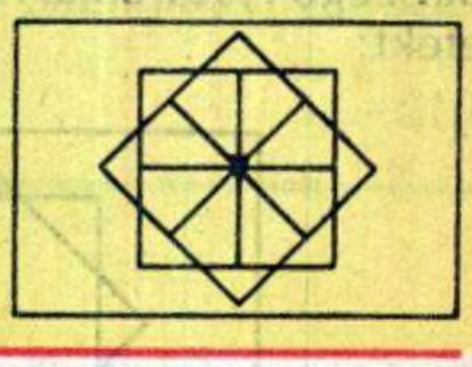
```
POWTÓRZ 4 [NP 50 LW 90]
```

Zauważmy, że po wykonaniu tej instrukcji żółw będzie skierowany do góry, a nie w prawo, tak jak przy poprzednim rysowaniu kwadratu, gdzie po ostatnim NAPRZÓD nie podaliśmy już komendy LEWO 90. Najlepiej jest przyjąć jako zasadę, że po narysowaniu figury żółw znajduje się w takiej pozycji, jak przed rozpoczęciem rysowania. Zaoszczędzi nam to czasu na zastanawianie się, jak jest ustawiony żółw i błędów wynikających z niewłaściwego określenia jego pozycji.

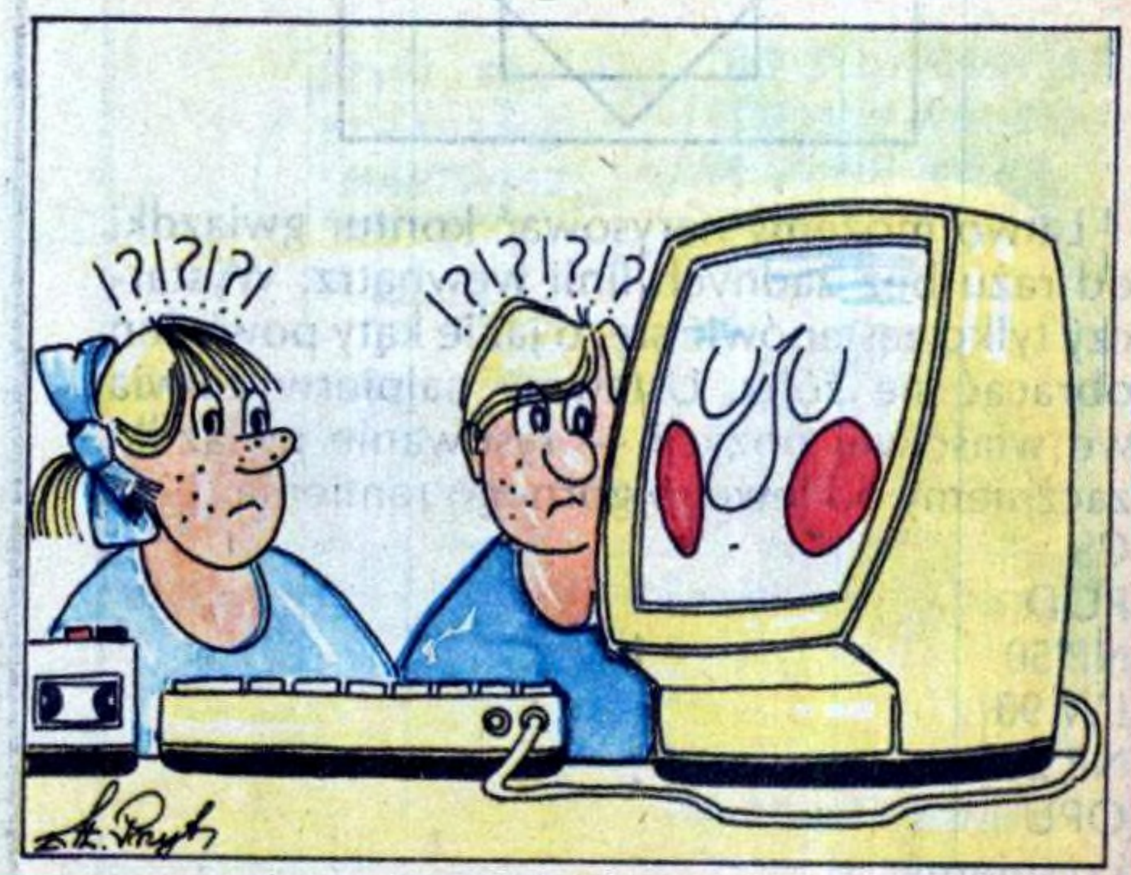
Spróbujmy teraz wykonać nieco bardziej skomplikowany rysunek: rozetę złożoną z kwadratów, której środek będzie się znajdował pośrodku ekranu.

```
POWTÓRZ 8 [POWTÓRZ 4 [NAPRZÓD 50
LEWO 90] LEWO 45]
```

Uzyskamy taki rysunek:



dokończenie na str. 14



dokończenie ze str. 13

Chcemy teraz z rozety zrobić gwiazdkę — usunąć linie rozchodzące się promieniście ze środka ekranu. Ale jak zmazywć kreski?

Otóż żółw może chodzić po ekranie, rysując kreski, może zmieniać swoją pozycję, nie pozostawiając żadnego śladu i może też idąc po kresce ścierać ją, czyli zachowywać się jak gumka. Możemy sobie wyobrazić, że żółw ma pióro, którym rysuje po ekranie. Jeśli trzyma to pióro uniesione w górę, idzie, ale nie zostawia śladu. Jeśli je opuści, rysuje kreski, a jeśli zamieni pióro na gumkę, ściera je. Odpowiednie komendy określają, co ma robić żółw ze swoim piórem.

POD (od podnieś, ang. PENUP i skrót PU) podnosi pióro. Od chwili wydania tej komendy żółw przemieszczając się nic nie rysuje.

OPU (od opuść ang. PENDOWN skrót PD) opuszcza pióro — żółw zaczyna rysować.

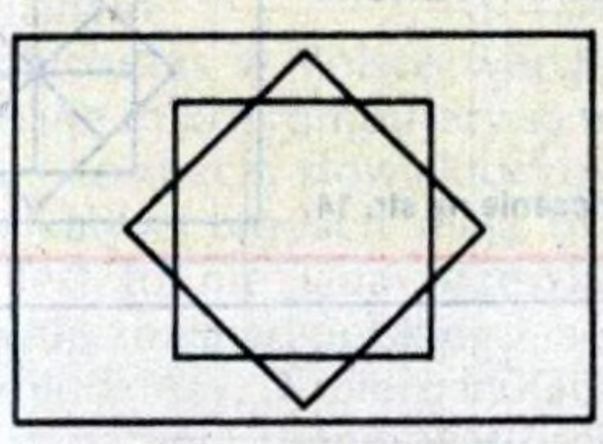
ŚCIERANIE (ang. PENERASE skrót PE) — sprawia, że żółw zmazuje kreski, które napotyka na swej drodze.

Dla narysowania gwiazdki przyda się także komenda cofająca żółwia, tak jak gdyby szedł on tyłem. Rozkaz WSTECZ (skrót WS, ang. BACK i skrót BK) cofa żółwia z powrotem, nie zmieniając kierunku, w jakim jest on ustawiony. Żółw wraca tyle kroków, ile podaje parametr (tak jak w komendzie NAPRZÓD).

Następująca sekwencja instrukcji przekształci naszą rozetkę w gwiazdkę:

```
ŚCIERANIE
POWTÓRZ 8 [NAPRZÓD 50 WSTECZ 50 LEWO 45]
OPU
```

Ostatnia instrukcja przygotowuje żółwia do dalszego rysowania. Na ekranie uzyskamy taki efekt:



Łatwo możemy narysować kontur gwiazdki od razu bez żadnych linii wewnątrz. Wystarczy tylko zastanowić się, o jakie kąty powinien obracać się żółw. Ustawmy najpierw żółwia we właściwej pozycji — rysowanie gwiazdki zaczniemy od lewego górnego ramienia.

```
CS
POD
NP 50
LW 90
NP 20
OPU
```

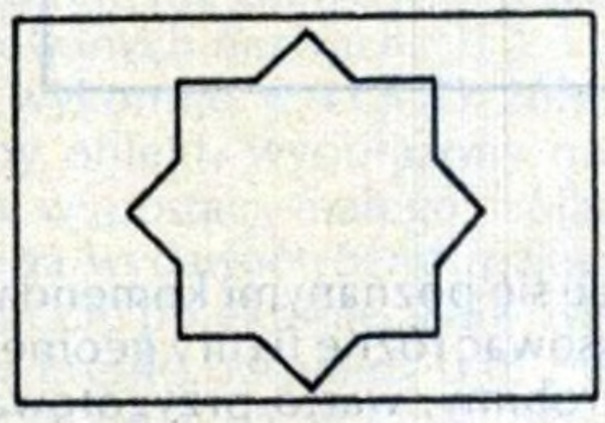
i rysujemy:

```
POWTÓRZ 8 [NP 30 LW 90 NP 30 PW 45]
```

Wracamy żółwiem na środek ekranu:

```
POD
WRÓĆ
OPU
```

Rysunek wygląda tak:



Bardzo ciekawe efekty można uzyskiwać, rysując proste figury wielokrotnie w różnych miejscach ekranu. Na przykład naszą ośmioramienną gwiazdkę można powtórzyć 36 razy, obracając ją za każdym razem o 10 stopni. Zaczynamy od środka ekranu i rysujemy przy użyciu takich instrukcji:

```
POWTÓRZ 36 [POWTÓRZ 8 [NP 20 LW 90 NP 20 PW 45] LW 10]
```

Czytelnicy, którzy mają komputer i translator LOGO, niech sprawdzą ten program w angielskiej wersji:

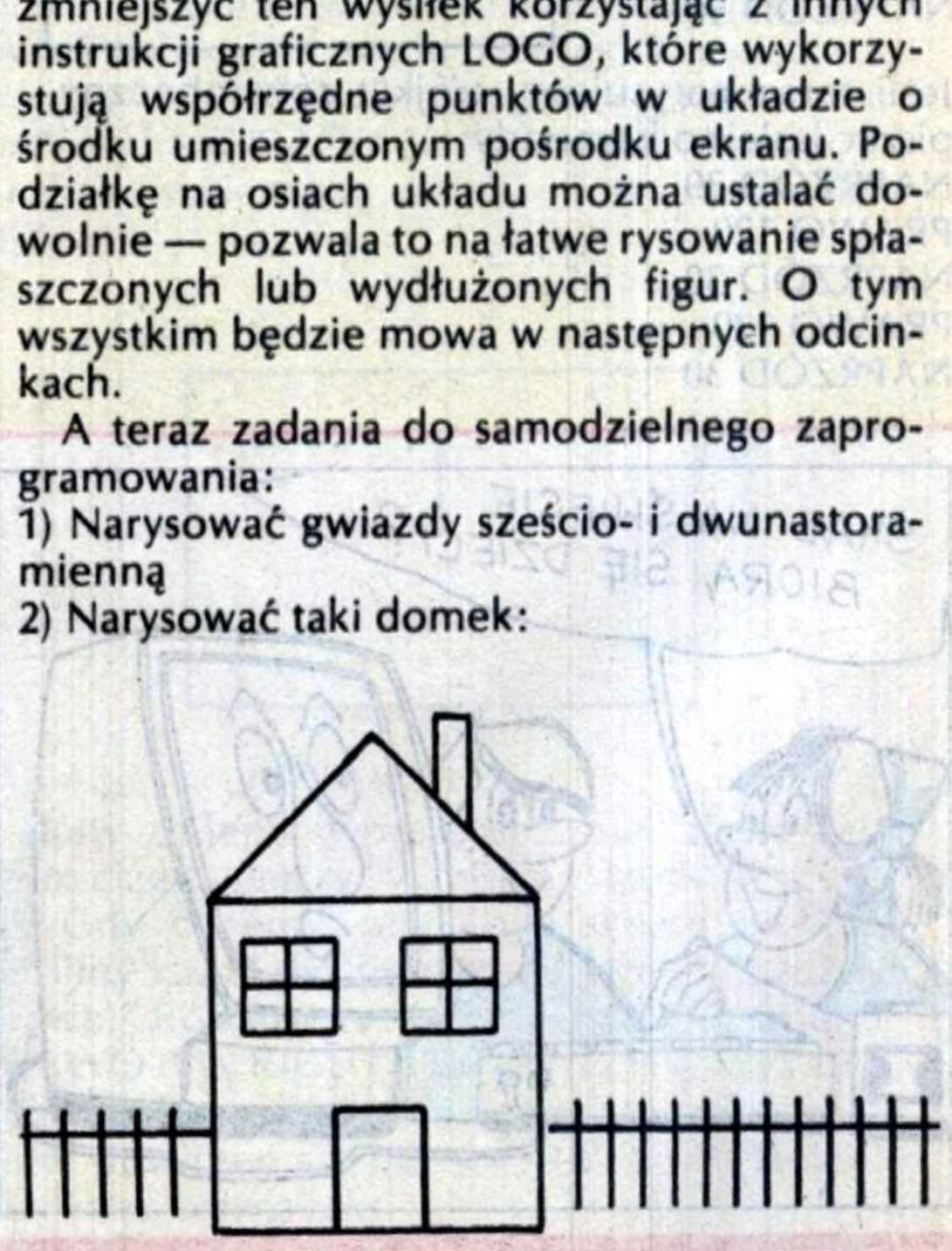
```
REPEAT 36 [REPEAT 8 [FD 20 LT 90 FD 20 RT 45] LT 10]
```

Dotychczas poznane instrukcje graficzne pozwalają narysować dowolną figurę. Naturalnie zaprogramowanie złożonych rysunków wymaga dużego nakładu pracy. Często można zmniejszyć ten wysiłek korzystając z innych instrukcji graficznych LOGO, które wykorzystują współrzędne punktów w układzie o środku umieszczonym pośrodku ekranu. Podziałkę na osiach układu można ustalać dowolnie — pozwala to na łatwe rysowanie spłaszczonych lub wydłużonych figur. O tym wszystkim będzie mowa w następnych odcinkach.

A teraz zadania do samodzielnego zaprogramowania:

1) Narysować gwiazdy sześcio- i dwunastoramienną

2) Narysować taki domek:



Nie pisaliśmy nic na temat błędów, jakie możemy popełniać przy naszej konwersacji z LOGO. Każdy kto ma w tej chwili komputer przed sobą (a tak by było najlepiej) widzi znak „?" i migający kursor. Znaczy to, że LOGO gotowe jest wykonywać nasze rozkazy (komendy lub procedury). Prawie na pewno przynajmniej raz pomylimy się i napiszemy np. „LLW" zamiast „LW". LOGO reaguje wtedy zupełnie naturalnie — pisze na ekranie „Nie wiem, jak zrobić LLW". Informuje nas, że nie rozumie komendy (popełniliśmy błąd) i czyni to w sposób najzupełniej dla nas zrozumiały (angielskie LOGO sygnalizuje — „I dońt know how to..."). Drugi rodzaj błędu, jaki mogliśmy do tej pory popełnić, to napisanie dla przykładu „NP 30 20...". W takim wypadku LOGO oczywiście wykonuje komendę NAPRZÓD 30 i dalej pisze: „Nie powiedziałeś co zrobić z 20". Ta reakcja LOGO nie wymaga chyba komentarza. Wreszcie mogliśmy napisać „NP LW 30", co nie ma sensu, gdyż wprawdzie wiadomo, że trzeba obrócić się o 30 stopni w lewo ale nie wiadomo ile kroków iść do przodu. No i LOGO zasygnalizuje: „LW nie daje wyniku potrzebnego NP". Gdy natomiast napiszemy: „NP" i wciśniemy ENTER, zobaczymy: „za mało danych dla NP". Właściwie wszystkie komunikaty o błędach są tak oczywiste, że nie ma sensu ich wyjaśniać — wiadomo, że masło jest maślane. Odstępując od tej zasady dodajmy, że po napisaniu np. „POWTÓRZ 4 5" będzie: „POWTÓRZ nie chce 5 jako danej" (bo drugim parametrem komendy POWTÓRZ musi być lista ujęta w nawiasy kwadratowe), a po napisaniu „POWTÓRZ 10 NP 20" zobaczymy „NP nie daje wyniku potrzebnego powtórz" (bo nie daje listy — powód ten sam). Na koniec zauważmy jeszcze, że jeśli coś rysujemy, na napisy (konwersacja z komputerem) przeznaczone są tylko dwie linie u dołu ekranu. Zdarza się wtedy czasem, że komputerowi brakuje miejsca na tekst i wówczas w prawym dolnym rogu miga pionowa strzałka, co oznacza: przeczytaj to, co napisałem, ale to jeszcze nie koniec — wciśnij coś, to zobaczysz resztę. Wszystkie te praktyczne wskazówki są tak naprawdę w tym miejscu niepotrzebne. Wystarczy dłuższa chwila samodzielnej zabawy (na razie zabawy) by wszystko stało się oczywiste. LOGO zostało zaprojektowane tak, by samodzielnie uczyć się i ograniczyć do minimum kontakt z nauczycielem. Dlatego też nauka LOGO ma sens, kiedy mamy rzeczywisty kontakt z komputerem, nie tylko kartkę i ołówek. Optymistyczny jest fakt, że jak na razie (wynika to ze statystyk) liczba tych drugich przypadków systematycznie maleje.

Uwagi o polskim translatorze LOGO na ZX Spectrum

Standard polskiego LOGO (brzmienie słów kluczowych i komunikatów) został opracowany przez zespół pracujący pod egidą Polskiego Towarzystwa Informatycznego. Zgodnie z tym standardem przygotowano translator LOGO na ZX Spectrum. Polski translator (jest to tzw. interpreter) jest już gotowy i będzie niebawem (albo już jest, gdy niniejszy tekst ukaże się w druku) rozpowszechniany w szkołach oraz wśród prywatnych użytkowników mikrokomputerów. Jest on obiektywnie lepszy niż angielski — na którego bazie powstał (firmy software'owej SOLI—LCSI), gdyż jego autorzy oprócz spolszczenia słów kluczowych i komunikatów usunęli kilka poważnych błędów, które powodowały nieprawidłowe działanie paru instrukcji. Wprowadzono możliwość posługiwania się literami charakterystycznymi dla języka polskiego (ą, ę, ł itp.), które mogą być używane w nazwach określanych przez użytkownika, a w słowach kluczowych np. powtórz, są wręcz niezbędne (wszak ma to być po polsku). Litery te otrzymuje się na ekranie po wciśnięciu klawisza „graph" i odpowiedniego klawisza litery: np. ą to „graph" i a, ł to „grahp" i l itd. Ponadto polskie LOGO ma nowe (nie istniejące w angielskiej wersji) atrakcyjne komendy i operacje. Są to m.in. funkcje rzeczywiste ln i exp oraz np. komenda graficzna ZAMALUJ (ZAM) pozwalająca kolorować powierzchnie rysunków. O tej ostatniej komendzie — w następnym odcinku, gdzie również spróbujemy napisać pierwsze programy — czyli nauczymy LOGO rozumieć nowe słowa — zdefiniujemy procedury.

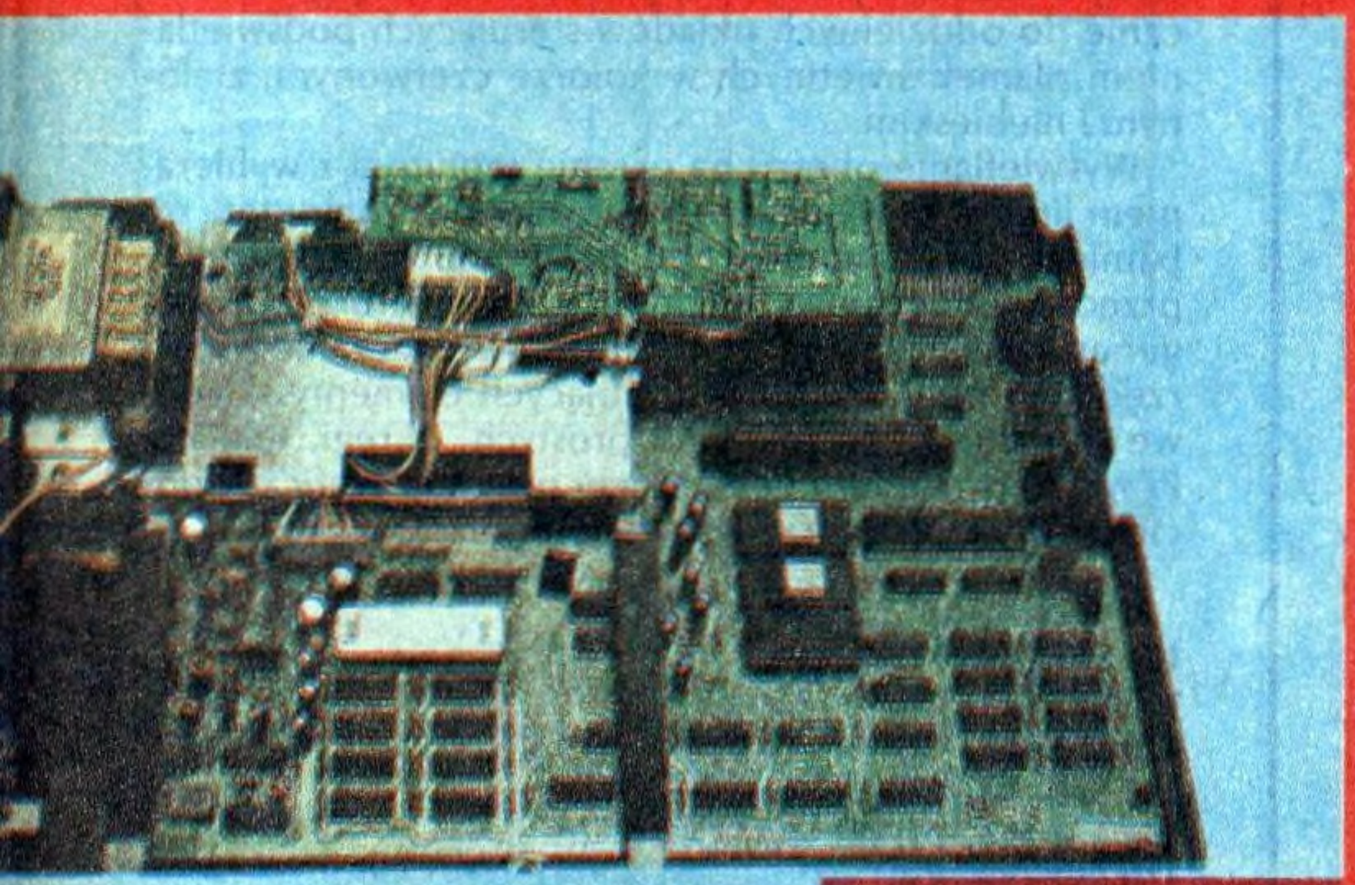
SIS

Rys. Michał Przybyłowski

Mikroprocesory odmierzyły kolejny krok w rozwoju techniki. W chwili obecnej nie ma już praktycznie takiej dziedziny życia, w której wprost lub pośrednio nie byłyby one stosowane. Są wszędzie tam, gdzie mowa jest o elektronicznych, twórczych rozwiązaniach.

Wszystko, co nowe, co stwarza szansę rozwoju — w wojsku preferuje się szczególnie. W nim, jak w soczewce zbiegają się wszystkie nici postępu. Każda nowatorska myśl jest starannie analizowana, rozwijana, wartościowe projekty są szybko wdrażane — przede wszystkim z myślą o sprawności, skuteczności i niezawodności sił zbrojnych.

Współczesne konstrukcje sprzętu bojowego, wykorzystywanego na lądzie, morzu i w powietrzu oparte są na najnowszych zdobyczach techniki. Prym wiedzie — co zrozumiałe — elektronika, wymagająca od żołnierzy obsługujących ów sprzęt, bezpośrednich użytkowników, odpowiedniego stopnia wtajemniczenia. Budowy i zasad działania tych skomplikowanych urządzeń uczyć ich musi kadra zawodowa, wysokiej klasy specjaliści. Dziś przyszli dowódcy zdobywają wiedzę w nowoczesnych, doskonale wyposażonych pracowniach i labolatoriach. A potem praktyka: intensywne poligonowe ćwiczenia.



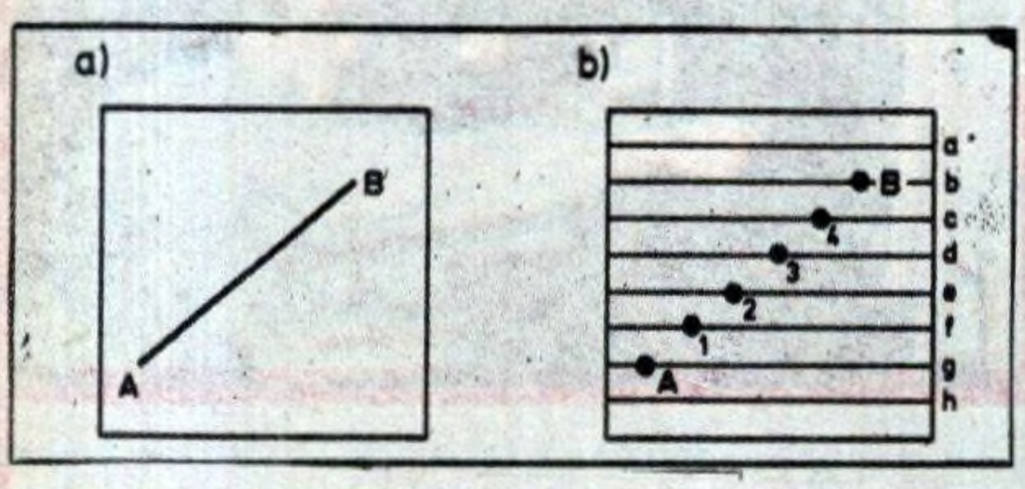
GRAFIKA KOMPUTEROWA

Jerzy Chmurzyński

Program graficzny

Program graficzny jest to ciąg kodów sterujących pracą procesora graficznego w zakresie kreślenia obrazu. Program graficzny generowany jest w komputerze i przesyłany do pamięci obrazu grafoskopu. W zależności od przyjętej metody kreślenia obrazu na ekranie lampy elektronopromieniowej różne są postacie programów graficznych.

W przypadku metody kreślenia konturowego program graficzny złożony jest z ciągu rozkazów opisujących pozycjonowanie i przemieszczanie strumienia elektronów oraz sterujących jasnością i kolorem świecenia plamki świetlnej na ekranie lampy. Rozkazy te są w procesorze graficznym dekodowane i wykorzystane w układach odchylania i podświetlania do bezpośredniego sterowania strumieniem elektronów. Przykładowy program graficzny kreślenia odcinka AB z rys. 1a przedstawiono poniżej:



POSITION (XA, YA)
VECTOR (XB,YB)

Rozmieszczenie programu graficznego w pamięci obrazu grafoskopów kreślących metodą konturową nie jest orientowane geometrycznie względem wymiaru rastrowego ekranu. Orientacja geometryczna następuje w końcowej fazie kreślenia obrazu, to znaczy w układach odchylania pionowego i poziomego, gdzie wykorzystuje się przechowywane w pamięci obrazu argumenty rozkazów programu graficznego, np. współrzędne XA, YA, XB, YB. Inaczej zagadnienie to przedstawia się w grafoskopach kreślących metodą rastrową z wybieraniem liniowym. W tym przypadku rozmieszczenie programu graficznego w pamięci obrazu jest jednoznacznie zorientowane geometrycznie względem wymiaru rastrowego ekranu. Przedstawia to rys. 3.

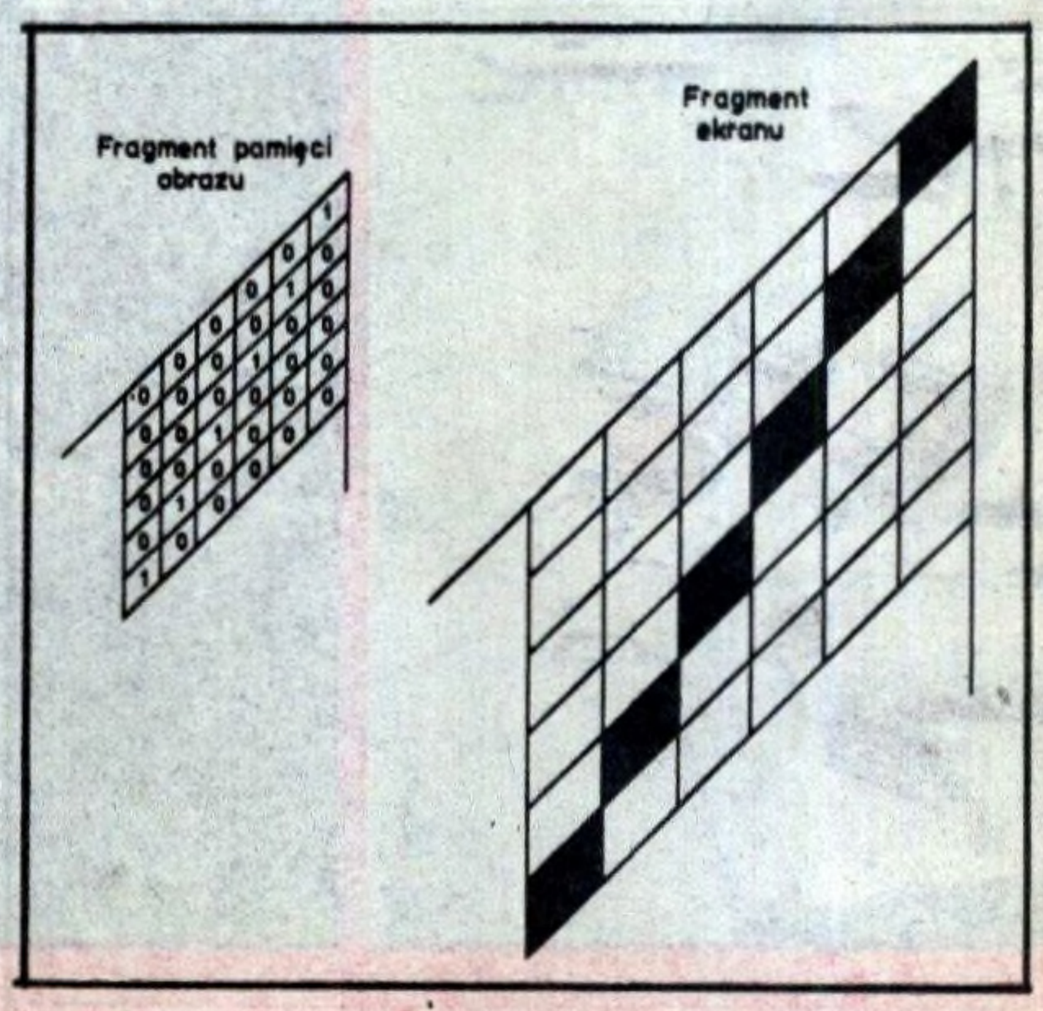
Program graficzny złożony jest z kodów określających rodzaj wyświetlanej informacji dla każdego adresowalnego punktu ekranu. Kody te uporządkowane są w kolejności linii wybieranych przez układ odchylania pionowego strumienia elektronów. W ramach każdej linii uporządkowanie kodów jest zgodne z kolejnością punktów wybieranych przez układ odchylania poziomego. Kod wyświetlanej informacji w najprostszym przypadku wartością 1 oznacza żądanie wyświetlenia punktu na ekranie, natomiast wartością 0 — zakaz wyświetlania. Kody zero — jedynkowe przesyłane są wierszami do układu sterującego podświetlaniem plamki świetlnej, synchronicznie z przemieszczaniem się strumienia elektronów wzdłuż wybranej linii poziomej. W punktach oznaczonych kodem 1 nastąpi rozjaśnienie plamki, natomiast w punktach oznaczonych kodem 0 — obraz nie będzie wyświetlany. Pamięć obrazu zawiera zatem dokładne odwzorowanie obrazu wyświetlanego na ekranie monitora. Pojedynczy płat pamięci może odwzorowywać obraz kreślony w jednym kolorze. Do wysterowania monitora kolorowego, dla każdego z trzech podstawowych kolorów potrzebne są oddzielne płaty pamięci. Kody z każdego płatu pamięci kierowane są synchronicznie do oddzielnych układów sterujących podświetlaniem plamek świetlnych w kolorze czerwonym, zielonym i niebieskim.

Wyświetlanie obrazu na ekranie monitora z wybieraniem liniowym wymaga wprowadzenia z komputera do pamięci obrazu grafoskopu programu graficznego w przedstawionej powyżej postaci. Zadaniem procedury generującej programy graficzne jest wyznaczanie współrzędnych punktów aproksymujących elementy składowe obrazu, tj. odcinki linii prostych, okręgi, łuki itp. Przykład procedury generującej program graficzny odcinka AB z rys. 1b przedstawiono poniżej.

```
100 SET (XA, YA)
110 SET (X1, Y1)
120 SET (X2, Y2)
130 SET (X3, Y3)
140 SET (X4, Y4)
150 SET (XB, YB)
160 RETURN
```

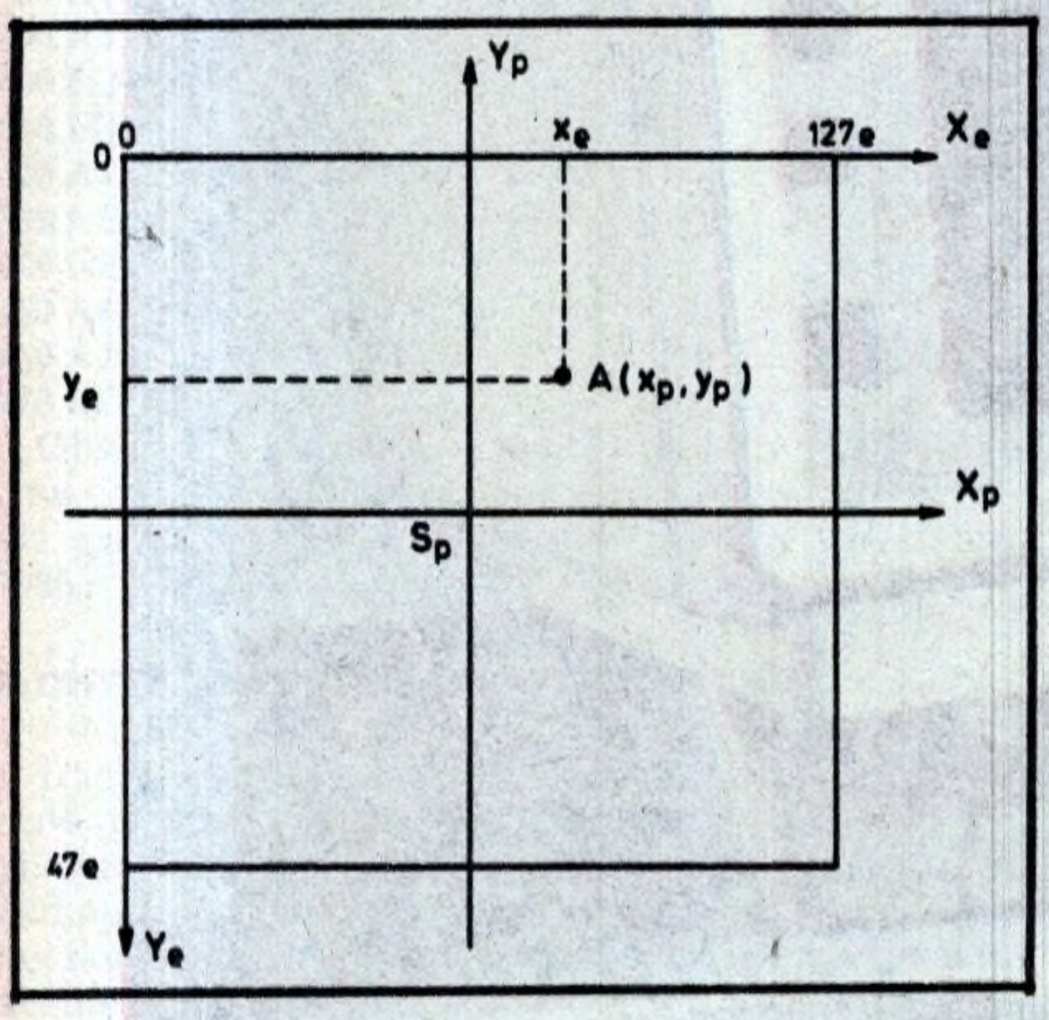
Procedura ta zapisana została w języku BASIC dla mikrokomputera MERITUM I. Instrukcja SET (X, Y) wyświetla punkt na ekranie w miejscu określonym współrzędnymi X, Y. Początek układu współrzędnych tego mikrokomputera znajduje się w lewym górnym rogu ekranu.

Rys. 3



Zmiana układu współrzędnych

Położenie środka układu współrzędnych prostokątnych dla danego monitora ekranowego określane jest na poziomie oprogramowania systemowego. Położenie to może być zatem zmieniane programowo na poziomie oprogramowania systemowego lub użytkowego. Na rys. 4 przedstawiono ekranowy układ współrzędnych prostokątnych X_e, Y_e.



Rys. 4

Środek S_p przykładowego, programowo zdefiniowanego układu współrzędnych X_p, Y_p znajduje się w punkcie o współrzędnych (64e, 24e) ekranowego układu współrzędnych. Aby wyświetlić na ekranie monitora dowolny punkt A o współrzędnych (x_p, y_p) należy dokonać następującego prostego przeliczenia współrzędnych prostokątnych tego punktu z układu X_p, Y_p na układ X_e, Y_e:

$$x = 64 + x_p \qquad y_e = 24 - y_p$$

Stąd wyświetlenie punktu $A(x_p,y_p)$ następuje zgodnie z instrukcją SET(64 + XP, 24 — YP).

Zobrazowanie odcinków

Odcinki linii prostych należą do podstawowych składników obrazów. Zobrazowanie odcinka metodą rastrową z wybieraniem liniowym wymaga wyznaczenia współrzędnych zbioru punktów aproksymujących dany odcinek. W grafoskopach, które nie mają procesów graficznych z możliwościami generowania punktów aproksymujących, należy to wykonywać programowo. Poniżej przedstawiona jest procedura zobrazowania odcinka ukośnego AB z rys. 1b.

```
300 D = (YB — YA) / (XB — XA)
310 FOR X = XA TO XB
320 Y = YA + D * (X — XA)
330 SET (X, Y)
340 NEXT X
350 RETURN
```

Transformacje obrazów na płaszczyźnie

Położenie punktu P na płaszczyźnie określamy za pomocą współrzędnych (x,y). Współrzędne te w komputerze mogą być pamiętane jako elementy macierzy [x,y]. Podstawowe transformacje obrazów na płaszczyźnie obejmują następujące operacje:

— zmiana skali obrazu,
— przesunięcie (translacja) obrazu,
— obrót obrazu wokół środka układu współrzędnych,
— obrót obrazu wokół dowolnego punktu.

W wyniku dokonania transformacji punktu o współrzędnych [x,y] otrzymujemy współrzędne transformowane [x',y'] tego punktu. Transformacje odcinków sprowadzają się do transformacji punktów początkowych i końcowych tych odcinków.

Transformacje obrazów na płaszczyźnie dokonywane są w przestrzeni trójwymiarowej, z wykorzystaniem współrzędnych jednorodnych. Oznacza to przedstawienie macierzy współrzędnych punktu [x,y] za pomocą macierzy [kx, ky, k] dla $k \neq 0$. Współczynniki transformacji zebrane są w macierzy transformacji T o wymiarach 3 x 3:

$$T = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & s \end{bmatrix}$$

Poszczególne elementy macierzy T dokonują następujących transformacji składowych:

a — zmiana skali współrzędnej X,
b, c — obrót i skręcenie względem środka układu,
d — zmiana skali współrzędnej Y,
e — przesunięcie względem osi X,
f — przesunięcie względem osi Y,
s — zmiana skali współrzędnej X i Y

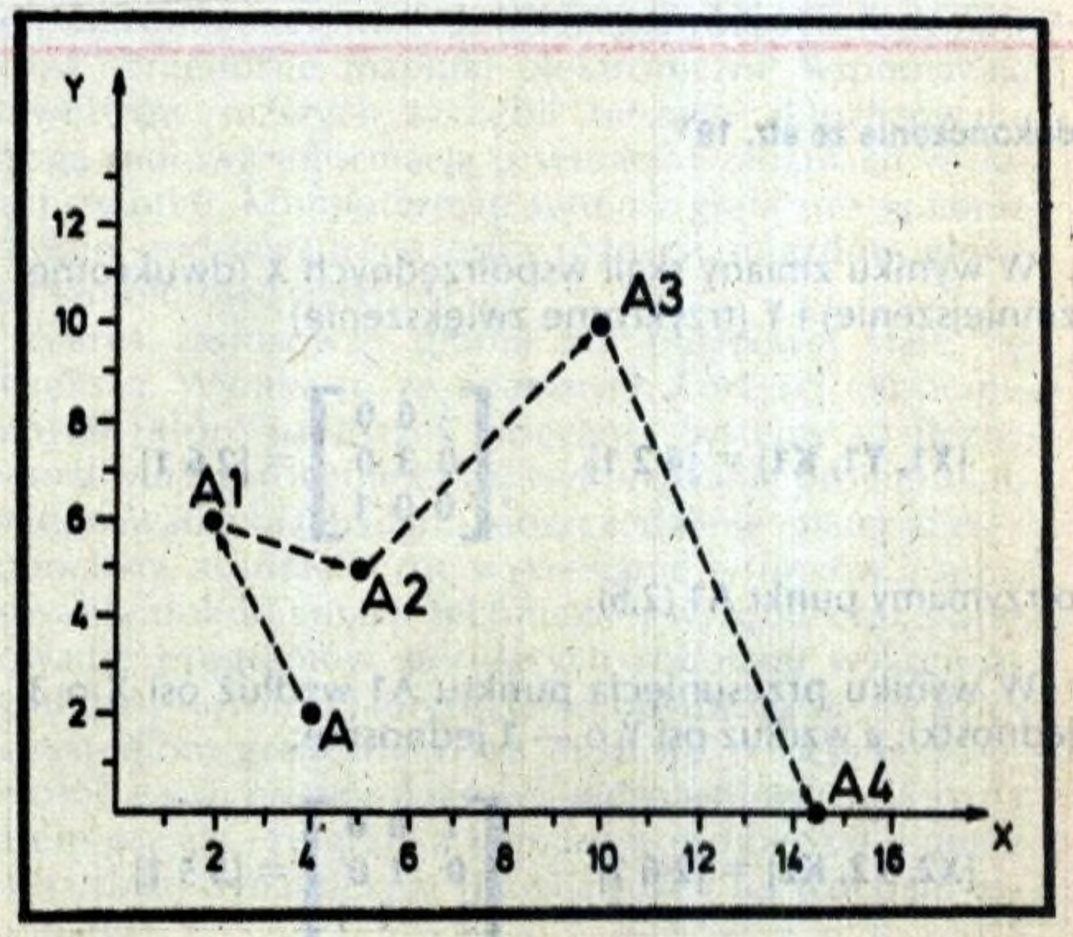
Współrzędne transformowane [x', y']punktu [x,y] otrzymuje się w wyniku wymnożenia współrzędnych jednorodnych punktu [x,y,1] przez macierz transformacji T:

$$[X, Y, K] = [x, y, 1] \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & s \end{bmatrix}$$

i przekształcenie transformowanych współrzędnych jednorodnych do postaci:

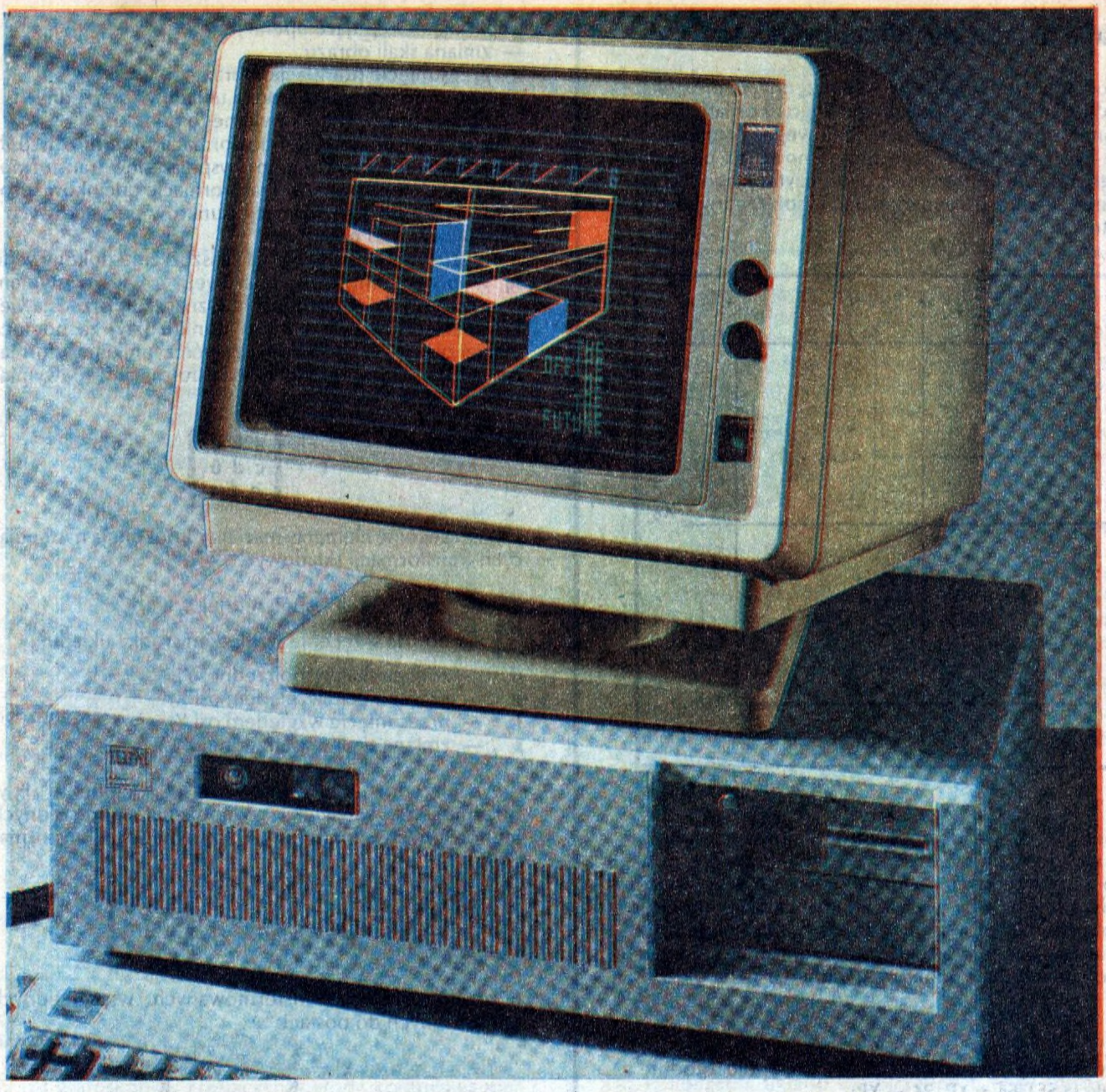
$$x' = \frac{X}{K} \qquad y' = \frac{Y}{K}$$

Rozważmy następujący ciąg transformacji punktu A (4,2) (rys. 5)



Rys. 5

dokończenie na str. 20



dokończenie ze str. 19

W wyniku zmiany skali współrzędnych X (dwukrotne zmniejszenie) i Y (trzykrotne zwiększenie)

$$[X1, Y1, K1] = [4\ 2\ 1] \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [2\ 6\ 1]$$

otrzymamy punkt A1 (2,6).

W wyniku przesunięcia punktu A1 wzdłuż osi X o 3 jednostki, a wzdłuż osi Y o — 1 jednostkę:

$$[X2, Y2, K2] = [2\ 6\ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & -1 & 1 \end{bmatrix} = [5\ 5\ 1]$$

otrzymamy punkt A2 (5,5).

Po dokonaniu dwukrotnego zwiększenia skali współrzędnych X i Y punktu A2 (5,5):

$$[X3, Y3, K3] = [5\ 5\ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} = [5\ 5\ \frac{1}{2}]$$

otrzymamy punkt A3 (10,10).

Aby dokonać obrotu punktu A3 wokół środka układu o kąt 45°, zgodnie z ruchem wskazówek zegara, należy dokonać następującej transformacji:

$$[X4, Y4, K4] = [10\ 10\ 1]$$

$$\begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} = [14.1\ 0\ 1]$$

W wyniku otrzymamy punkt A4 (14.1,0).

Poniżej przedstawiono w języku BASIC procedurę wyznaczania współrzędnych transformowanych punktu na płaszczyźnie. Założono, że współrzędne punktu transformowanego zawarte są w tablicy A, współrzędne transformowane w tablicy A1, zaś współczynniki macierzy transformacji zawarte są w tablicy T.

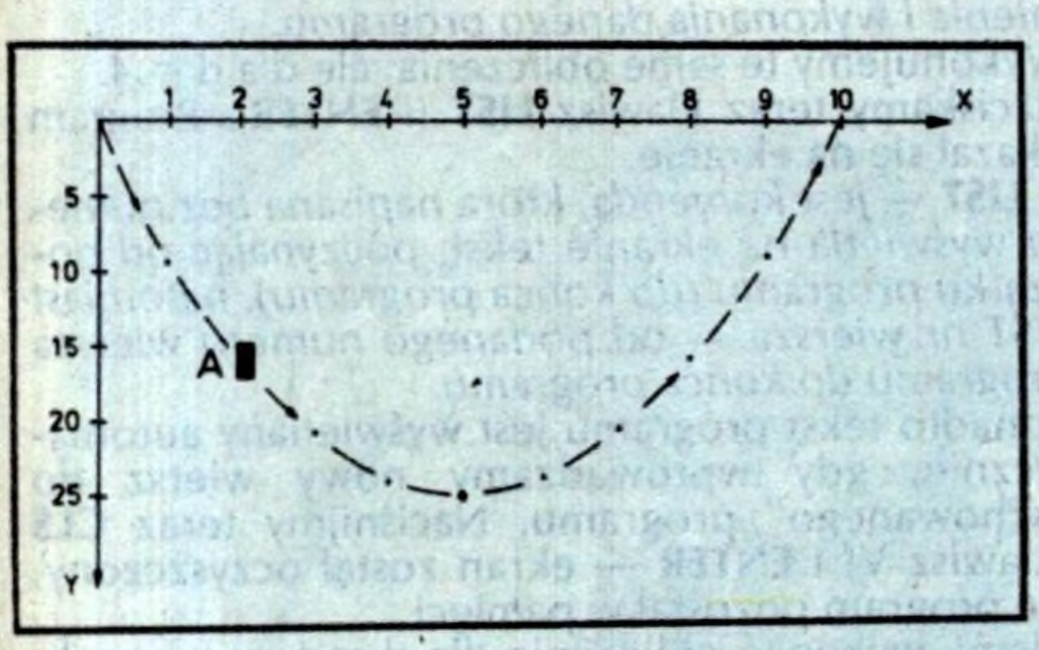
```
 10 DIM A (2)
 20 DIM A1 (2)
 30 DIM T (2,2)
370 A1 (0) = 0
380 A1 (1) = 0
390 A1 (2) = 0
400 FOR I = 0 TO 2
410 FOR J = 0 TO 2
420 A1 (I) = A1 (I) + A (J) * T (J,I)
430 NEXT J
440 NEXT I
450 A1 (0) = A1 (0) / A1 (2)
460 A1 (1) = A1 (1) / A1 (2)
470 RETURN
```

Obszerne informacje na temat transformowania obrazów Czytelnik znajdzie w pracy T. Parlielisa: Algorytmy dla grafiki i przetwarzania obrazów WNT. Warszawa 1986 r.

Animacja komputerowa

Transformacje obrazów stają się bezpośrednio użyteczne podczas badań symulacyjnych projektowanych obiektów. Animacja komputerowa pozwala śledzić na ekranie monitora przebieg określonego procesu w różnej skali czasu. Badania takie mogą być poświęcone np. ocenie skutków zderzenia samolotu z przeszkodą, ocenie metody lądowania pojazdu na Księżycu, szacowaniu skutków wybuchu jądrowego itp. Ruch obrazu na ekranie monitora można uzyskać poprzez wygaszenie i wyświetlenie kadrów kolejnych faz ruchu z częstotliwością ok. 25 Hz. Kadry wyświetlane na ekranie mogą być filmowane i wykorzystane następnie do produkcji filmów animowanych. Na ekranie monitora filmy takie mogą być wielokrotnie powtarzane dla różnych parametrów badanego procesu. Animacja komputerowa w istotny sposób przyspiesza proces projektowania. Ruchomy obraz modelujący badany proces ułatwia projektantom wybór parametrów optymalnych dla danej konstrukcji czy procesu.

Poniżej przedstawiono procedurę przemieszczania punktu A (rys. 6) po paraboli $y = -x^2 + 10 \cdot x$ w przedziale $x = [0,10]$.



Rys. 6

```
500 FOR X = 0 TO 10
510 Y = 10 * X — X * X
520 SET (X,Y)
530 GOSUB 600
540 RESET (X,Y)
550 NEXT X
560 RETURN
600 P = 2
610 FOR I = O TO P * X
620 FOR J = 1 TO 1000
630 Z = 0
640 NEXT J
650 NEXT I
660 RETURN.
```

Podprogram 600 — 660 symuluje zmianę parametru P w funkcji drogi przebytej wzdłuż osi X. Procedura RESET (X,Y) wygasza punkt świecący na ekranie o współrzędnych X,Y.

Zastosowania

Zakres zastosowań grafiki komputerowej jest obszerny. Głównym obszarem zastosowań jest projektowanie wspomagane komputerem (CAD). Nowoczesne systemy komputerowe dzięki grafice dostarczają wyniki prac projektowych w postaci wykresów lub rysunków technicznych projektowanej konstrukcji. W zakresie projektowania wspomaganego komputerem główne dziedziny zastosowań grafiki komputerowej to:

— przemysł elektroniczny — 55 %
— kreślenia — 15 %
— kartografia — 10 %
— architektura i inne dziedziny techniki i nauki — 20 %

W przemyśle elektronicznym komputerowe systemy graficzne stosowane są głównie do projektowania układów elektronicznych, płytek drukowanych i układów scalonych. W dziedzinie budownictwa grafika komputerowa znalazła zastosowanie w projektowaniu autostrad i konstrukcji architektonicznych. Stosowane systemy graficzne umożliwiają prowadzenie symulacyjnych badań bezpieczeństwa jazdy na projektowanych fragmentach dróg.

W przemysłach lotniczym i motoryzacyjnym oraz w badaniach kosmicznych systemy graficzne stosowane są na wszystkich etapach projektowania. Systemy graficzne umożliwiają zwiększenie efektywności procesu projektowania i produkcji oraz dokonywanie symulacyjnych badań obiektu bez konieczności budowania wielu kosztownych modeli. W wypadku badań kosmicznych wiele badań eksploatacyjnych bez systemów graficznych byłoby niemożliwych.

Grafika komputerowa znalazła również wielostronne zastosowanie w wojsku. Komputerowe urządzenia zobrazowania danych w postaci graficznej znajdują się w wyposażeniu stanowisk dowodzenia różnych rodzajów wojsk. Przenośne mapniki elektroniczne wspomagają dowódców niższych szczebli bieżąco aktualizowaną drogą radiową informacją o sytuacji i zadaniach w danym rejonie. Komputerowe systemy graficzne są coraz częściej podstawą trenażerów różnych pojazdów, głównie samolotów i śmigłowców.

Zakres zastosowań grafiki komputerowej stale się zwiększa. Wynika to ze znacznych korzyści ekonomicznych, które są z tym związane. Systemy graficzne umożliwiają skrócenie czasu wykonywania projektu lub przeprowadzenia badań, zaoszczędzenie materiałów i robocizny, automatyczne wykreślanie rysunków i opracowanie dokumentacji technicznej wyrobu oraz opracowanie programów sterujących robotami wykonującymi dany wyrób w fabryce. Obecnie uważa się, że dzięki systemom graficznym jest możliwe skrócenie 5—8-krotne czasu prac projektowo-wdrożeniowych. Można zatem przyjąć, że grafika komputerowa będzie jedną z najbardziej dynamicznie rozwijających się dziedzin informatyki.

Jerzy CHMURZYŃSKI

2

BASIC

(Podstawy programowania)

W poprzednim numerze „IKS-a" rozpoczęliśmy wykład z programowania w języku BASIC na mikrokomputer ZX Spectrum. Kontynuujemy więc pracę, włączamy komputer i zaczynamy.

Wykład drugi: program w BASIC-u edytor na ZX Spectrum
elementy BASIC-u: stałe zmienne nazwy,
instrukcje: REM, LET, PRINT, INPUT
komendy: RUN, LIST, EDIT, NEW

Język BASIC (**B**eginnerś **A**ll Purpose **S**ymbolic **I**nstruction **C**ode) jest najpopularniejszym uniwersalnym językiem programowania wyższego rzędu. Stosują go zarówno użytkownicy posługujący się komputerami do przeprowadzania dość prostych obliczeń, jak i ci, którzy posługują się nimi przy rozwiązywaniu bardzo skomplikowanych problemów naukowo-technicznych, organizacyjnych lub ekonomicznych. Zaletą BASIC-u jest z jednej strony prostota, umożliwiająca szybkie opanowanie podstaw programowania, z drugiej strony istnienie w nim elementów pozwalających na efektywne zaprogramowanie wielu, nawet skomplikowanych problemów.

BASIC jest językiem konwersacyjnym, co pozwala programującemu na nadzorowanie i bieżącą korektę programu podczas jego uruchamiania.

Program w BASIC-u. Edytor na ZS Spectrum

Program w języku BASIC składa się z wierszy (linii), które zawierają instrukcje opisujące wykonanie poszczególnych kroków algorytmu. Włączamy komputer i zaczynamy pracę od przykładu.

Przykład 2.1

Obliczyć wartość sumy n pierwszych wyrazów postępu arytmetycznego według wzoru:

$$s_n = \left[a_1 + \frac{(n-1)\,d}{2} \right] n$$

gdzie: a_1 — pierwszy wyraz, d — różnica
dla $a_1 = 5$, $n = 11$, $d = 1$

Program realizujący te obliczenia ma postać:

```
10 REM Przyklad 2.1
30 LET d=1
40 LET n=11
20 LET a1=5
50 LET sn=(a1+(n-1)*d/2)*n
60 PRINT "SUMA =";sn
```

Wprowadźmy ten program. Załóżmy, że wprowadziliśmy 3 wiersze, na ekranie telewizora otrzymamy:

```
10 REM Przyklad 2.1
30 LET d=1
40 LET n=11
```

Wprowadźmy następny wiersz, a teraz ekran wygląda tak:

```
10 REM Przyklad 2.1
20 LET a1=5
30 LET d=1
40 LET n=11
```

Wprowadzając zauważamy, że wiersze programu zaczynają się od numerów i nie pojawiają się od razu, lecz zostają „odłożone" na później jako program, ponadto numery wierszy decydują o ich kolejności w programie (jest to także istotne przy wykonywaniu programu). Numer wiersza jest liczbą całkowitą, maksymalnie czterocyfrową (1 ÷ 9999). Wiersz w ZX Spectrum może składać się z kilku linii opatrzonych jedynie numerem w linii pierwszej.

Wprowadzając następny wiersz, na ekranie otrzymamy:

```
10 REM Przyklad 2.1
20 LET a1=5
30 LET d=1
40 LET n=11
50 PRINT "SUMA =";sn
```

Sprawdzamy program, okazuje się, że popełniliśmy błąd. Z numerem 50 został wprowadzony wiersz 60 naszego programu, tzn. chcemy wydrukować wartość S_n, która nie została obliczona. Wobec tego wprowadzamy wiersz:

45 **LET** $s_n = (a1 + (n - 1) * d/2) * n$

i na ekranie otrzymamy:

```
10 REM Przyklad 2.1
20 LET a1=5
30 LET d=1
40 LET n=11
45 LET sn=(a1+(n-1)*d/2)*n
50 PRINT "SUMA =";sn
```

Niemożliwe byłoby wstawienie wiersza między poprzednie, gdyby wiersze były numerowane kolejno np: 41, 42, 43 itd. Przy pisaniu programów i ich konwersacyjnym wprowadzaniu do komputera zaleca się numerować wiersze np. co 5 lub 10.

Naciskamy teraz klawisz **RUN** (i **ENTER**), w lewym górnym rogu ekranu pokaże się nam napis:

SUMA = 110

jest on wynikiem wykonania tego programu. Program „schował" się w pamięci komputera.

RUN — *jest komendą, która służy do uruchomienia i wykonania danego programu.*

Wykonujemy te same obliczenia, ale dla d = 4. Naciskamy teraz klawisz **LIST** (i **ENTER**). Program ukazał się na ekranie.

LIST — *jest komendą, która napisana bez numeru wyświetla na ekranie tekst, poczynając od początku programu (do końca programu), natomiast LIST nr wiersza — od podanego numeru wiersza programu do końca programu.*

Ponadto tekst programu jest wyświetlany automatycznie, gdy wprowadzamy nowy wiersz do „schowanego" programu. Naciśnijmy teraz **CLS** (klawisz V) i **ENTER** — ekran został oczyszczony, ale program pozostał w pamięci.

Mamy wykonać obliczenia dla d = 4, w tym celu wprowadzamy

30 **LET** d = 4 (i **ENTER**)

Program ukazał się na ekranie, poprzedni wiersz o numerze 30 został zastąpiony nowo wprowadzonym.

Błędny wiersz z danym numerem można usunąć z programu przez wprowadzenie nowego wiersza z tym samym numerem. Takie poprawianie jest nie-

wygodne, gdy liczba znaków w wierszu jest duża. Usunięcie wiersza z programu wykonujemy przez napisanie jego numeru i naciśnięcie klawisza ENTER. Napiszmy teraz 29 (i **ENTER**), *w tym wypadku zniknął kursor wiersza* ⊠.

„Schował" się on między linie w numerach 20 i 30. Naciśnijmy teraz jednocześnie klawisze **CAPS SHIFT** i **EDIT** *(klawisz z cyfrą 1). W lewym dolnym rogu ekranu pokazał się nam wiersz o numerze 30. (30* **LET** *d = 4). Naciśnijmy* **ENTER** *— wiersz 30 powrócił na swoje miejsce w programie. Wykonajmy ten program* **RUN** *(i* **ENTER**).

Otrzymamy wynik:

SUMA = 275

Wykonajmy teraz kolejno następujące czynności:

LIST (klawisz K i ENTER)

EDIT (klawisze CAPS SHIFT i 1)

Na ekranie pokazał się program, a w lewym dolnym rogu wiersz o numerze 10. Naciśnijmy klawisz **ENTER**, wiersz o numerze 10 powrócił na swoje miejsce i kursor wiersza znajduje się w tym wierszu. (10 ⊠REM Przykład 2.1) Kursor wiersza można przenosić z wiersza do wiersza, w dół naciskając ↓ (klawisz **CAPS SHIFT** i 6), a w górę ↑ (klawisz **CAPS SHIFT** i 7). Przesuńmy kursor (wiersza) ⊠ do wiersza o numerze 40, następnie naciśnijmy **EDIT**. Wiersz 40 pojawił się w lewym dolnym rogu ekranu. Wykorzystując ←, → lub **DELETE** wprowadzamy na n liczbę 21. Poprawiać w wierszu (gdy jest on na dole ekranu) uczyliśmy się w pierwszym wykładzie. Po poprawkach wiersz powinien wyglądać tak:

40 **LET** n = 21

Naciskamy teraz kolejno: **ENTER**, **RUN** i **ENTER**. W lewym górnym rogu otrzymamy wynik:

SUMA = 945

Naciśnijmy teraz klawisz **NEW** (i **ENTER**)

NEW *— jest komendą, która usuwa z pamięci komputera program i czyści ekran.*

Po jej wykonaniu możemy wprowadzać nowy program.

Stałe, zmienne, nazwy, instrukcje: REM, LET, PRINT, INPUT

Program podany w przykładzie 2.1 zapiszmy w następującej postaci:

```
10 REM Przyklad 2.1.1
15 REM to co jest napisane po
   slowie REM to jest komentarz
20 LET a1=5
30 LET d=1
40 LET n=11
45 REM liczby 5,1,11 sa stalymi
   mi numerycznymi
50 LET sn=(a1+(n-1)*d/2)*n
52 REM napisy a1,d,n,sn sa
   nazwami zmiennych prostych
54 REM napis (a1+(n-1)*d/2)*n
   jest wyrazeniem arytmetycz-
   nym
56 REM LET nazwa = wyrazenie
   arytmetyczne jest to aryt-
   metyczna instrukcja podsta-
   wienia
60 PRINT "a1=";a1;"    ";"d=";
   d;"    ";"n=";n
65 PRINT
70 PRINT "SUMA =";sn
75 REM kazdy ciag znakow napi-
   sany w cudzyslowie nazywa
   sie stala alfanumeryczna
   (lub lancuchem)
80 REM dzialanie instrukcji
   PRINT bylo omawiane, gdy
   traktowalismy komputer
   jako kalkulator
```

wykonajmy ten program, tzn. naciśnijmy **RUN** (i **ENTER**). Na ekranie telewizora w lewym górnym rogu otrzymamy:

```
a1=5    d=1    n=11

SUMA =110
```

Komputer zignorował wszystkie teksty napisane po

dokończenie na str. 24

Komputery... fascynacja dla wszystkich **Foto. S. Iwan**

dokończenie ze str. 23

słowie kluczowym **REM**. Instrukcja **REM** ma postać:

n **REM** dowolny tekst

n — numer wiersza

REM — *napisane w dowolnym wierszu jest komentarzem, który został umieszczony w celu opisania naszego programu.*

W *ZX* Spectrum po słowie **REM** napisany tekst może zajmować kilka linii. W trakcie wykonywania programu komputer pominie tekst aż do końca ostatniej linii.

Aby otrzymać na ekranie (w wyniku wykonania programu) wszystkie teksty, które zostały napisane po słowie REM, należy zamienić słowo **REM** — słowem kluczowym **PRINT** i każdy tekst zamknąć w cudzysłów.

Wykonajmy te poprawki, będzie to dobrym sprawdzianem omówionych wcześniej czynności wykonywanych przy poprawianiu wierszy. Po poprawkach program powinien wyglądać tak:

```
  10 REM Przyklad 2.1.2
  15 PRINT "to co jest napisane
po slowie   REM to jest komentar
z"
  20 LET a1=5
  30 LET d=1
  40 LET n=11
  45 PRINT '"liczby 5,1,11 sa st
alymi          numerycznymi"
  50 LET sn=(a1+(n-1)*d/2)*n
  52 PRINT "napisy a1,d,n,sn sa
 nazwami     zmiennych prostych"
  54 PRINT '"napis (a1+(n-1)*d/2
)*n jest      wyrazeniem arytmety
cznym"
  56 PRINT '"LET nazwa = wyrazen
ie arytme-    tyczne jest to aryt
metyczna      instrukcja podstawi
enia"
  60 PRINT '"a1=";a1;"     ";"d="
;     d;"     ";"n=";n
  65 PRINT
  70 PRINT "SUMA =";sn
  75 PRINT '"kazdy ciag znakow n
apisany       w cudzyslowie nazyw
a sie stala   alfanumeryczna (lub
 lancuchem)"
  80 REM dzialanie instrukcji
     PRINT bylo omawiane, gdy
     traktowalismy komputer
     jako kalkulator
```

a po wykonaniu na ekranie otrzymamy:

```
to co jest napisane po slowie
REM to jest komentarz

liczby 5,1,11 sa stalymi
numerycznymi
napisy a1,d,n,sn sa  nazwami
zmiennych prostych

napis (a1+(n-1)*d/2)*n jest
wyrazeniem arytmetycznym

LET nazwa = wyrazenie arytme-
tyczne jest to arytmetyczna
instrukcja podstawienia

a1=5     d=1     n=11

SUMA =110

kazdy ciag znakow napisany
w cudzyslowie nazywa sie stala
alfanumeryczna (lub lancuchem)
```

Instrukcja **PRINT** ma postać:

n **PRINT** lista lub n **PRINT**

n — numer wiersza

lista — może składać się z nazw zmiennych, wyrażeń arytmetycznych, stałych numerycznych, stałych alfanumerycznych i znaków specjalnych (' , ;)

Znaki specjalne sterują postacią wyprowadzanych wyników, zapoznaliśmy się z ich działaniem, gdy traktowaliśmy komputer jako kalkulator. W wyniku wykonania drugiej postaci instrukcji **PRINT** na ekranie otrzymujemy pustą linię.

Omawialiśmy też wyrażenie arytmetyczne i kolejność wykonywania działań w celu obliczenia wartości wyrażenia, w którym występowały stałe numeryczne (liczby) połączone operatorami arytmetycznymi. *W naszym przykładzie napis:*

$$(a1 + (n - 1) * d / 2) * n$$

jest wyrażeniem arytmetycznym, w którym występują proste zmienne numeryczne, stała numeryczna (liczba 2), operatory arytmetyczne (+, −, /, *) oraz nawiasy ( ). W trakcie obliczania wartości takiego wyrażenia według obowiązujących reguł (podaliśmy je w przykładzie 1) komputer pobiera

Rys. Michał Przybyłowski

wartości odpowiednich zmiennych i na tych wartościach wykonuje obliczenia.

Zmienna prosta numeryczna przyjmuje wartości będące liczbami rzeczywistymi. Nazwa tej zmiennej może zawierać dowolne litery lub cyfry, ale musi zaczynać się od litery. Stała alfanumeryczna (tekstowa) ma postać:

"ciąg znaków"

— ciąg znaków: dowolny ciąg znaków
— znaki cudzysłowu ograniczają jej początek i koniec
— tekst "" bez żadnego znaku zwany jest tekstem pustym
— tekst " " nie jest tekstem pustym, oznacza jedną spację, spacje wewnątrz cudzysłowu są znakami tekstu

Instrukcja **LET** wyznacza wartość zmiennej podczas działania programu i ma postać:

n **LET** V = e

n — numer wiersza
V — nazwa zmiennej
e — wyrażenie arytmetyczne

Wykonanie tej instrukcji polega na obliczeniu wartości wyrażenia arytmetycznego stojącego po prawej stronie znaku = (równa się) i podstawienie obliczonej wartości na miejsce zmiennej występującej po lewej stronie znaku = (równa się).

Przykład 2.2.

Dany jest program:

```
 10 REM Przyklad 2.2
 20 PRINT "cwiczenia"
 25 PRINT
 30 LET a=2.718281
 40 PRINT "a=";a
 45 PRINT
 50 LET b=2.71828182
 60 PRINT "b=";b
 70 LET c=2.7182818285
 75 PRINT
 80 PRINT "To jest przyblizona
wartosc    stalej e :";c
```

Wprowadźmy i wykonajmy ten program. Na ekranie otrzymamy:

```
cwiczenia

a=2.718281

b=2.7182818

To jest przyblizona  wartosc
   stalej e :2.7182818
```

Napiszmy teraz:

70 **LET** c = 0.27182818285 e^{1}

i wykonajmy program, następnie napiszmy:

70 **LET** c = 27182818285 e^{-10}

i wykonajmy program. Kolejne realizacje programu dla różnych postaci zapisu stałej w wierszu 70 dają ten sam wynik:

2.2182818

Zapamiętajmy:
— litera e oznacza mnożnik dziesiętny np. $2.5e^2$ czytamy jako $2,5 \cdot 10^2 = 250$
— zapis liczby postaci $2,5e^2$ nazywa się zapisem wykładniczym,
— kropka jest używana w miejsce przecinka oddzielającego część całkowitą od części ułamkowej,
— PRINT drukuje najwyżej 8 cyfr znaczących,
— do zapisania liczby nie może być użytych więcej niż 14 znaków,
— liczby są pamiętane z dokładnością do około dziewięciu i pół cyfry,
— największa liczba całkowita, która może być zapamiętana dokładnie, to $2^{32} - 1 = 4294967295$

Analogicznie do pojęcia zmiennej prostej numerycznej istnieje pojęcie zmiennej prostej alfanumerycznej, której „wartością" jest tekst. Nazwa takiej zmiennej składa się z litery (tylko jednej), po której musi następować znak dolara ($). Nazwom zmiennych alfanumerycznych można nadawać „wartości", używając instrukcji LET, np:

20 **LET** p **$** = „zadanie"

dokończenie na str. 26

dokończenie ze str. 25

Napisanie w programie instrukcji postaci:

30 **PRINT** p **$**

spowoduje w trakcie jej wykonania wypisanie na ekranie słowa: zadanie.

Przykład 2.3.

Napisać program w BASIC-u na obliczenie pola powierzchni całkowitej stożka obrotowego ściętego według wzoru:

$$P = \pi * R * (R + l) + \pi * r * (r + l)$$

gdy: tworząca $l = 30{,}25$
promień duży $R = 12{,}75$
promień mały $r = 7{,}25$

W programie należy zastosować zmienne alfanumeryczne.

Rozwiązanie:
BASIC (ZX Spectrum) nie rozróżnia liter małych i dużych, tzn. nazwy R i r oznaczają tę samą nazwę, przyjmujemy więc:
rd — promień duży
rm — promień mały

```
  10 REM Przyklad 2.3
  20 PRINT "Jest to przykladowy
program,     jezeli chcesz to nap
isz go sam   w inny sposob"
  25 PRINT
  30 LET rd=12.75
  35 LET rm= 7.25
  40 LET l =20.75
  45 LET r$="promien"
  50 LET d$=" duzy"
  55 LET m$=" maly"
  60 PRINT "obliczanie pola powi
erzchni      stozka scietego,gdy
dane sa :"
  65 PRINT
  70 PRINT "tworzaca =";l'r$;d$;
" =";rd'r$;m$;" =";rm
  75 PRINT
  77 REM PI znajduje sie nad kla
wiszem M
  80 LET P=PI*rd*(rd+l)+PI*rm*(r
m+l)
  90 PRINT '"pole pow.calk. stoz
ka scietego"'"          P =";P
```

Wprowadźmy i wykonajmy ten program. Sprawdźmy otrzymany wynik z wynikiem podanym niżej:

```
Jest to przykladowy program,
jezeli chcesz to napisz go sam
w inny sposob

obliczanie pola powierzchni
stozka scietego,gdy dane sa :

tworzaca =20.75
promien duzy =12.75
promien maly =7.25

pole pow.calk. stozka scietego
          P =1979.5961
```

Przerabiając podane przykłady programów zauważyliśmy, że zmieniając jakąś wartość danej, należało wprowadzić całą instrukcję od nowa, np. gdy w programie było 30 **LET** d = 1, to zmieniając wartość d na 4, wprowadziliśmy 30 **LET** d = 4.

Napisane dotychczas programy wykonywały obliczenia na tych samych danych. Każda zmiana wartości dowolnej danej wymagała jej wprowadzenia do programu przez zmianę odpowiedniej instrukcji.

Napiszemy teraz program z Przykładu 2.3, tak aby mógł być wykonywany dla dowolnych wartości l, R i r.

```
  10 REM Przyklad 2.3.1
  20 PRINT "Jest to przykladowy
program,     jezeli chcesz to nap
isz go sam   w inny sposob"
  25 PRINT
  30 INPUT "rd=";rd,"rm=";rm
  40 INPUT "l=";l,"r$=";r$
  50 INPUT "d$=";d$,"m$=";m$
  60 PRINT "obliczanie pola powi
erzchni      stozka scietego,gdy
dane sa :"
  65 PRINT
  70 PRINT "tworzaca =";l'r$;d$;
" =";rd'r$;m$;" =";rm
  75 PRINT
  77 REM PI znajduje sie nad kla
wiszem M
  80 LET P=PI*rd*(rd+l)+PI*rm*(r
m+l)
  90 PRINT '"pole pow.calk. stoz
ka scietego"'"          P =";P
```

Wprowadźmy i wykonajmy ten program.

Omówimy teraz kolejność działań wykonywanych przez komputer i wyświetlanych na ekranie.

W górnej części ekranu ukaże się napis:
Jest to przykładowy program, jeżeli chcesz to napisz go sam w inny sposób.

Następnie w lewym dolnym rogu wyświetlone zostaną kolejno odpowiednie teksty i kursor L, należy wtedy wprowadzić dane. A więc:

rd = [L]

Komputer czeka na wprowadzenie liczby — wprowadzamy 12.75 (i **ENTER**), a komputer w tym samym wierszu w połowie ekranu wypisze:

rm = [L]

wprowadzamy 7.25 (i **ENTER**). Oba napisy zniknęły, a komputer wypisał (w lewym dolnym rogu ekranu):

l = [L]

wprowadzamy 20.75 (i **ENTER**), a komputer w tym samym wierszu w połowie ekranu wypisze:

r $ = ["L"]

wprowadzamy słowo promień (i **ENTER**). Oba napisy zniknęły, a komputer wypisał:

d $ = ["L"]

wprowadzamy słowo duży (i **ENTER**), a komputer w tym samym wierszu w połowie ekranu wypisze:

m $ = "L"

wprowadzamy słowo mały (i **ENTER**). Oba napisy zniknęły, a komputer wypisał w górnej części ekranu wyniki, które już znamy.

Instrukcja **INPUT** *umożliwia nadawanie wartości zmiennym w czasie działania programu.* Możliwość ta jest szczególnie ważna przy pracy konwersacyjnej.

Instrukcja **INPUT** ma postać:

n **INPUT** lista

n — numer wiersza

lista — może składać się z nazw zmiennych, stałych alfanumerycznych i znaków specjalnych.

Działanie znaków specjalnych jest takie samo jak w instrukcji **PRINT**.

W czasie wykonywania instrukcji **INPUT** na jej zlecenie wprowadzamy z klawiatury wartości stałych numerycznych i alfanumerycznych, które są następnie przenoszone pod kolejne zmienne. Stałe te nazywają się danymi.

Stałe alfanumeryczne (tekstowe) wprowadzamy w cudzysłowie, komputer sam to sygnalizuje podając: "L" (kursor L). Umieszczanie stałych tekstowych opisujących nazwy zmiennych (na liście INPUT) którym mają być nadane wartości, ułatwia wprowadzanie tych wartości, a tym samym zmniejsza liczbę pomyłek.

Instrukcja (na ZX Spectrum) postaci:

30 **INPUT** "rd = "; rd

w wyniku wykonania daje to samo, co poniższe instrukcje:

30 **PRINT** "rd =";
35 **INPUT** rd

A teraz wykonajmy jeszcze raz ostatni program dla danych np. rd = 75,2, rm = 55, l = 100,25

Zadania

1. Napisać w BASIC-u program obliczania wartości y danej wzorem:

$$y = \frac{a^4 - b^4}{2(a^2 + b^2)} + \frac{a^2 - b^2}{a^2 + b^2} - \frac{a^2 + b^2}{2}$$

dla dowolnych dodatnich wartości a i b.
Uwaga: $a^4 - b^4 = (a^2 - b^2) * (a^2 + b^2)$, wprowadzić zmienne pomocnicze: $c = a^2$, $d = b^2$, $w = c + d$, $z = c - d$ np. dla a = 2 i b = 3 $y = -9.3846154$

2. Napisać w BASIC-u program obliczania wartości z danej wzorem:

$$z = \frac{x^2+y^2}{2xy} - 6(x^2 - 2xy + y^2)$$

gdy: $x = 5u^2 + 2v$, $y = 3{,}7 + v^2$
u, v dowolne wartości, ale $u > 0$ i $v > 0$.

3. Napisać w BASIC-u program, który dla dowolnej osoby wyprowadza na ekranie:

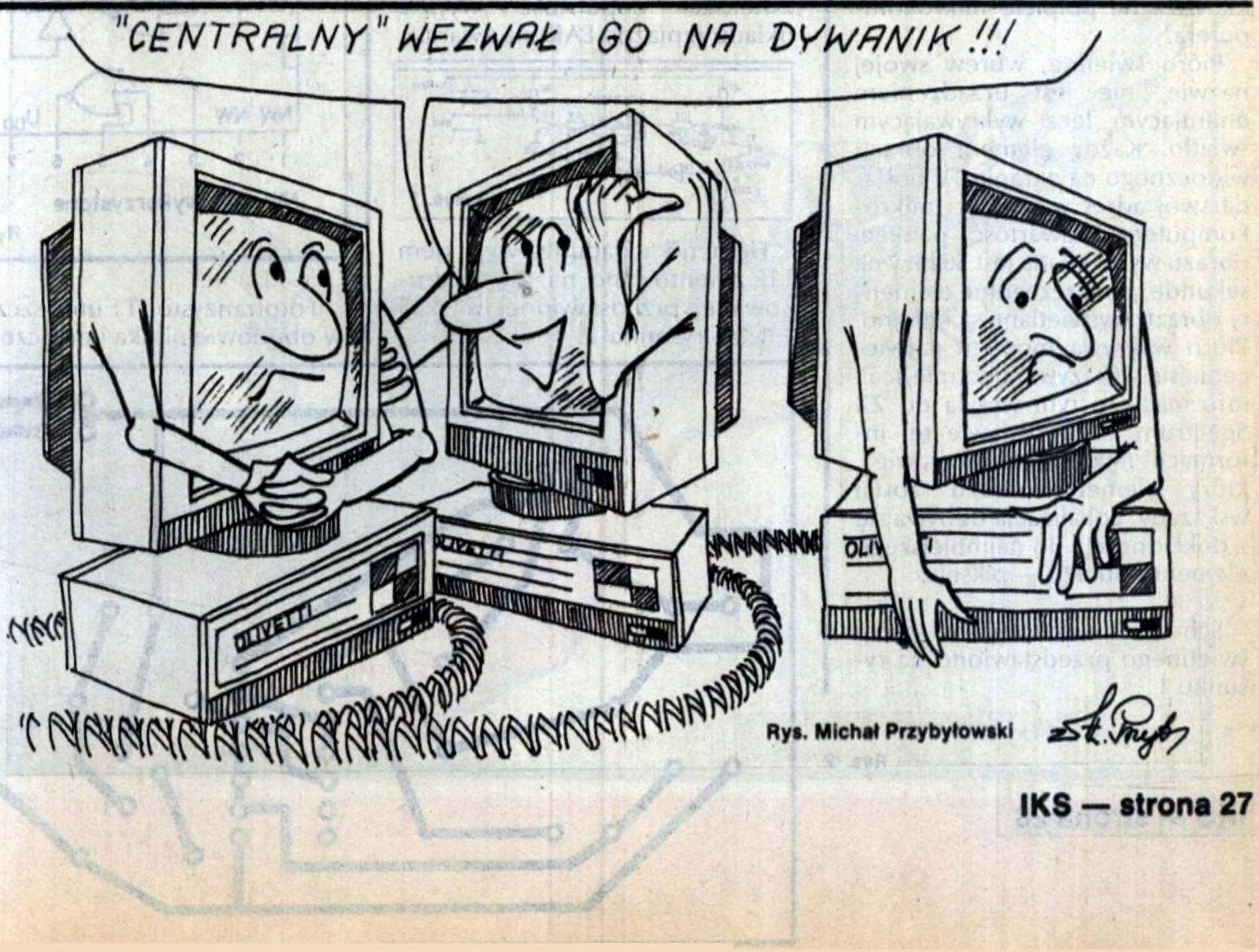
NAZWISKO I IMIĘ
ADRES:
KOD MIEJSCOWOŚĆ
ULICA NR DOMU NR MIESZK.
NR TELEFONU
ZAWÓD

} w tej części dane dotyczące dowolnej osoby

pierwsza strefa — druga strefa

ekranu

Z.W.



Rys. Michał Przybyłowski

PIÓRO ŚWIETLNE DO ZX SPECTRUM

Jednym z ciekawszych urządzeń zewnętrznych do mikrokomputera ZX Spectrum jest pióro świetlne. Jest ono urządzeniem, które samodzielnie może zbudować każdy posiadacz mikrokomputera. Wymagana jest jedynie umiejętność lutowania, a do uruchomienia pióra nie jest potrzebny żaden przyrząd pomiarowy. Do budowy wykorzystano części produkcji krajowej, których koszt nie przekracza 600 zł!

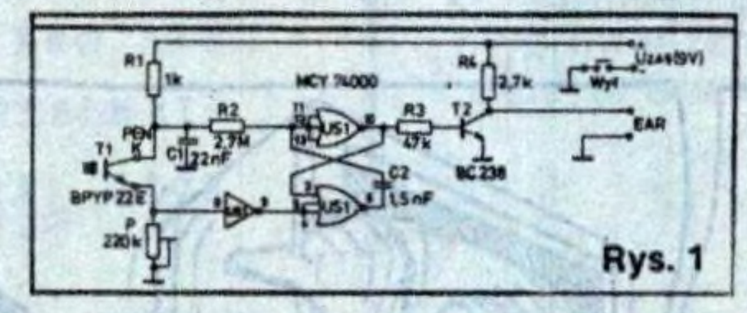
Zastosowanie odpowiedniego oprogramowania umożliwia rysowanie piórem na ekranie telewizora; kreślenie linii, prostokątów, okręgów, kolorowanie wskazanego obszaru, umieszczanie napisów, wybór koloru atramentu i papieru. Istnieją również programy wykorzystujące pióro przy wykonywaniu całego szeregu obliczeń matematycznych, np. pakiet programów „Library" umożliwia wykonywanie rachunku macierzowego, całkowania funkcji, obliczeń statystycznych.

Posługiwanie się piórem polega na dotknięciu wybranego miejsca ekranu (np. wskazanie opcji programu) i naciśnięciu dowolnego klawisza na pulpicie mikrokomputera.

Pióro świetlne, wbrew swojej nazwie, nie jest urządzeniem emitującym lecz wykrywającym światło. Każdy element obrazu widocznego na ekranie TV posiada swój adres w pamięci mikrokomputera. Zawartość pamięci obrazu wyświetlana jest 50 razy na sekundę, a poszczególne elementy obrazu wyświetlane są kolejno. Pióro wykrywa moment zaświecenia się wskazywanego miejsca i informację o tym wysyła do ZX Spectrum. Na podstawie tej informacji mikrokomputer „wie", który element obrazu został wskazany. Lokalizacja odbywa się z dokładnością do najmniejszego elementu obrazu — piksela.

Schemat elektroniczny pióra świetlnego przedstawiono na rysunku 1.

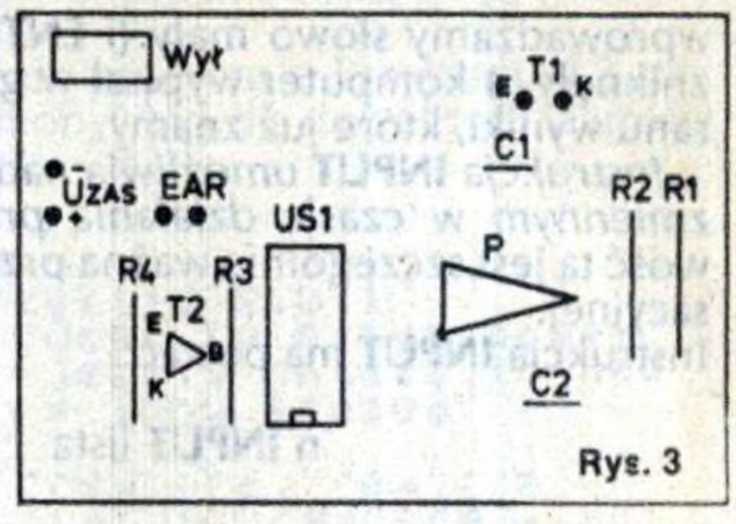
Czujnikiem wykrywającym moment zaświecenia się wskazywanego przez pióro miejsca jest fototranzystor T1. W wyniku oświetlenia fototranzystora zostaje wyzwolony przerzutnik monostabilny, który generuje krótki dodatni impuls. Impuls ten jest potwierdzeniem wykrycia momentu zaświecenia się wskazywanego punktu ekranu. Układ przerzutnika tworzą bramki NOR i NOT (ukł. scalony US1) wraz z rezystorem R2 i kondensatorem C2. ZX Spectrum wymaga jednak sygnału o polaryzacji ujemnej. Zmianę fazy sygnału z przerzutnika realizuje tranzystor T2. Zastosowanie w tej roli tranzystora zwiększa odporność wyjścia układu (gniazdo EAR) na zwarcia.



Rys. 1

Elementy układu (za wyjątkiem T1) zmontowano na płytce drukowanej przedstawionej w skali 1:1 na rysunku 2.

Na rysunku 3 przedstawiono rozmieszczenie elementów na płytce.



Rys. 3

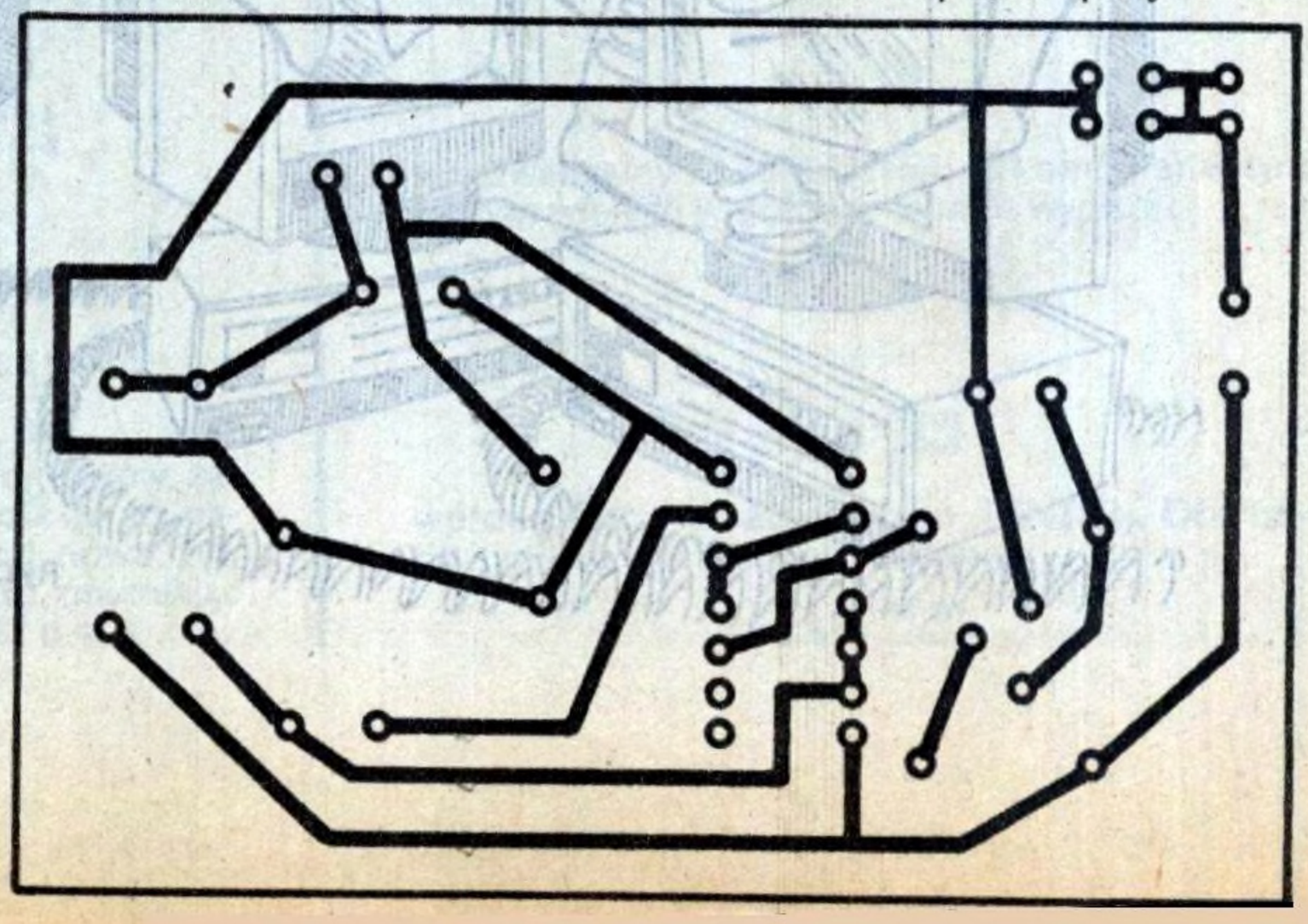
W pierwszej kolejności lutujemy elementy dyskretne, natomiast układ scalony US1 montujemy na końcu. Zastosowany układ scalony jest wykonany w technologii CMOS i należy obchodzić się z nim ostrożnie przestrzegając następujących zasad:

— nie dotykać palcami końcówek układu,
— lutować uziemioną lutownicą.

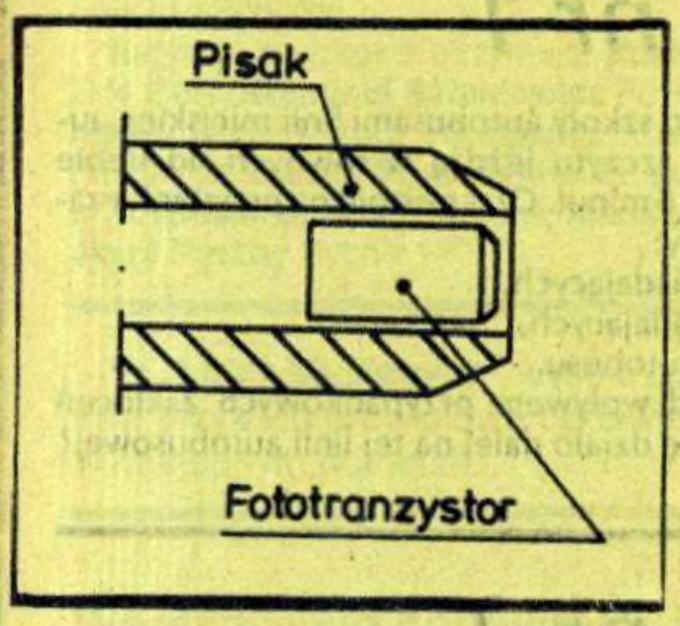
Nieprzestrzeganie zaleceń może spowodować nieodwracalne uszkodzenie układu. Na rysunku 4 pokazano rozmieszczenie wyprowadzeń zastosowanego układu scalonego.

Rys. 4

Fototranzystor T1 umieszczono w obudowie pisaka i połączono z


Rys. 2

resztą układu za pośrednictwem długiego przewodu zakończonego wtyczką „mini Jack". Należy zwrócić uwagę na sposób umieszczenia tego tranzystora — jego koniec powinien być osłonięty (rys. 5).



Rys. 5

Zmontowaną płytkę drukowaną umieszczono w obudowie wspólnie z baterią 6F22. Na jednej z bocznych ścianek obudowy umocowano dwa gniazdka. Gniazdko oznaczone EAR służy do połączenia pióra z gniazdkiem EAR ZX Spectrum. Gniazdko oznaczone PEN służy do podłączenia czujnika pióra (fototranzystor T1).

Tak wykonany i sprawdzony pod względem poprawności montażu układ możemy podłączyć do mikrokomputera.

UWAGA

Podłączenie do ZX Spectrum należy wykonywać przy odłączonym zasilaniu pióra. Najpierw łączymy fototranzystor z gniazdkiem PEN, następnie łączymy gniazdka EAR pióra i mikrokomputera (firmowym kabelkiem). Dopiero teraz włączamy zasilanie pióra.

W czasie tych czynności ZX Spectrum jest włączony i ma wczytany program obsługi pióra. Niektóre egzemplarze ZX Spectrum lepiej pracują, gdy podłączymy pióro do gniazdka MIC.

Jedyną regulacją, jakiej wymaga pióro jest kalibracja, do przeprowadzenia której służy potencjometr montażowy P (rys. 1). Należy zaznaczyć, że kalibracji dokonuje się w czasie pracy z programem obsługi pióra. Ustawiamy suwak potencjometra w położeniu środkowym i postępując według instrukcji zawartej w programie próbujemy pióro skalibrować. Jeżeli próba nie powiodła się, zmieniamy położenie suwaka (w górę lub w dół) i próbujemy skalibrować pióro ponownie. Czynności te powtarzamy aż do skutku.

Raz skalibrowane pióro praktycznie nie wymaga już regulacji.

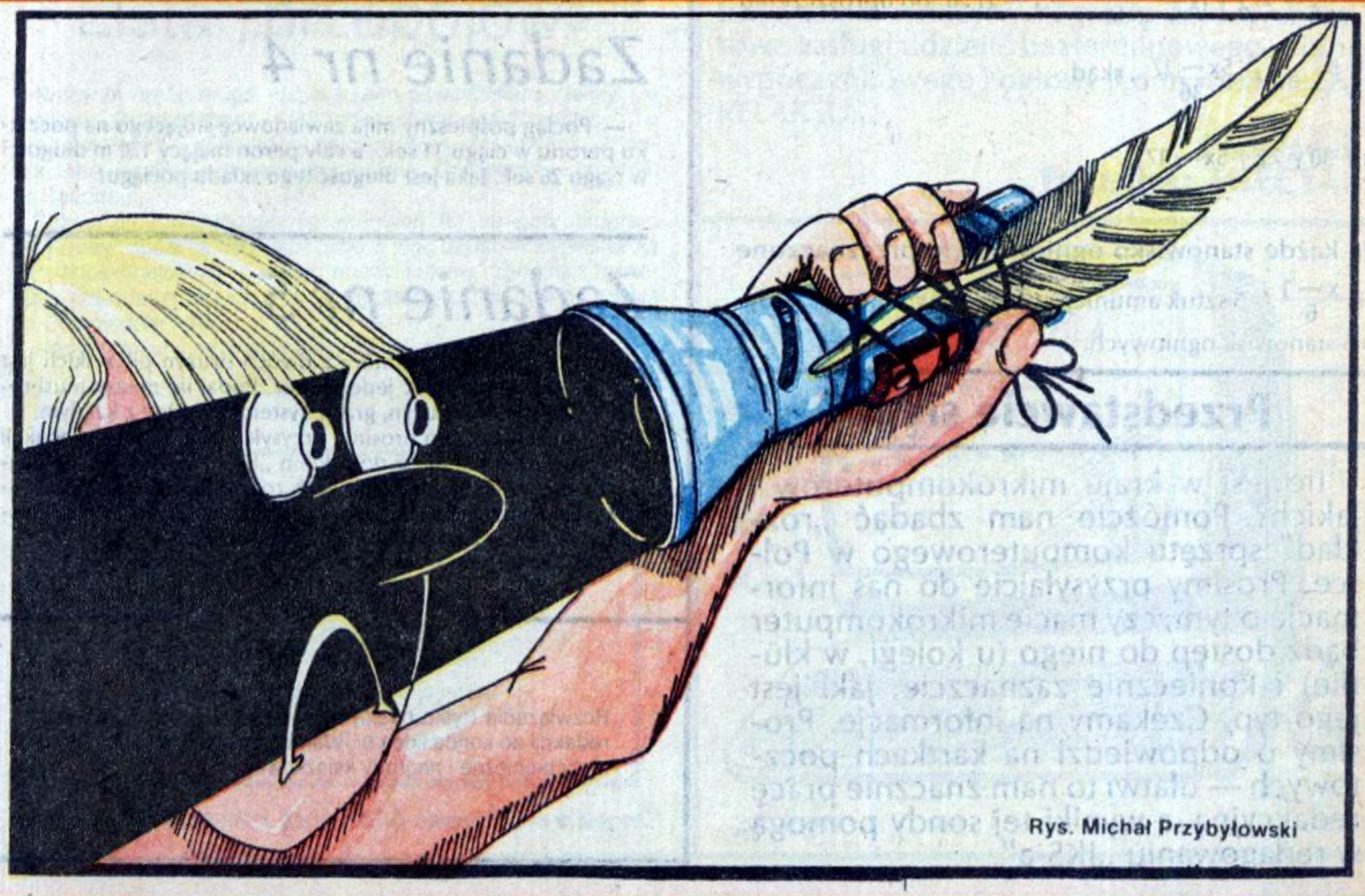
Spis części

Kondensatory (dowolnego typu)
- C1 — 22 nF
- C2 — 1,5 nF

Rezystory (najlepiej typu MŁT 0,125 W lub 0,25 W)
- R1 — 1 kΩ
- R2 — 2,7 MΩ
- R3 — 47 kΩ
- R4 — 2,7 kΩ

Półprzewodniki
- US1 — MCX74000 N
- T1 — BPYP 22
- T2 — BC 238, BC 237

Inne
- P — miniaturowy potencjometr montażowy 220 kΩ
- 2 gniazdka typu „mini Jack"
- 2 wtyczki typu „mini Jack"
- Miniaturowy wyłącznik zasilania



Rys. Michał Przybyłowski

Rozwiązania zadań z nr. 1

Zadanie 1: Aby dowiedzieć się, jakie kule są w poszczególnych koszach wystarczy tylko jedno ciągnienie. Jeżeli np. z kosza oznaczonego kartką czarno-białą (o którym wiemy, że nie ma w nim kul białych i czarnych) wyciągniemy kulę białą, możemy powiedzieć, że są tam 2 kule białe, w konsekwencji — w tym, na którym jest kartka biała-biała, muszą być 2 kule czarne, a w trzecim koszu: biała-czarna, analogiczne rozumowanie trzeba przeprowadzić gdy wyciągniemy kulę czarną.

Zadanie 2: Jest to najbardziej znane zadanie w teorii grafów nazywane „problemem mostów królewskich". Został on rozwiązany przez Leonarda Eulera (1707—1783) w roku 1736. Dwie wyspy, A i B, na rzece Pregole w Królewcu (ówczesna stolica Prus Wschodnich, obecnie Kaliningrad w zachodniej części ZSRR) były połączone ze sobą i z brzegami C i D siedmioma mostami, tak jak to pokazano na rys. 1.1. Euler przedstawił tę sytuację za pomocą grafu (rys. 1.2.).

Punkty A, B, C, D, zwane w teorii grafów wierzchołkami, reprezentują części lądowe, a linie łączące te punkty, w teorii grafów — krawędzie reprezentują mosty. Euler udowodnił, że rozwiązanie tego zagadnienia nie istnieje, a więc nie jest możliwe przejście przez wszystkie mosty dokładnie jeden raz, tak, aby powrócić do punktu wyjścia.

Zadanie 3: To typowe zadanie z kombinatoryki. Jest to 5-elementowa permutacja określona na 5-elementowym zbiorze i wyraża się wzorem $5! = 1 \times 2 \times 3 \times 4 \times 5 = 125$, a więc na pięciu krzesłach pięcioosobowa komisja może usiąść na 125 różnych sposobów.

Zadanie 4: W pudełku było x sztuk amunicji. Wiedząc, że liczba amunicji na wszystkich stanowiskach jest równa, możemy ułożyć równanie:

$$1+\frac{x-1}{6}=2+(x-\frac{x-1}{6}-1-2):6;$$ po uproszczeniu

$$\frac{x+5}{6}=2+\frac{5x-17}{36},$$ skąd

$$6x+30=72+5x-17$$

$$x=25$$

Na każde stanowisko ogniowe było przeznaczone $1+\frac{x-1}{6}=5$ sztuk amunicji, a stąd otrzymujemy, że było 5 stanowisk ogniowych.

Przedstawcie się...

Ile jest w kraju mikrokomputerów i jakich? Pomóżcie nam zbadać „rozkład" sprzętu komputerowego w Polsce. Prosimy przysyłajcie do nas informacje o tym, czy macie mikrokomputer bądź dostęp do niego (u kolegi, w klubie) i koniecznie zaznaczcie, jaki jest jego typ. Czekamy na informacje. Prosimy o odpowiedzi na kartkach pocztowych — ułatwi to nam znacznie pracę redakcyjną, a wyniki tej sondy pomogą w redagowaniu „IKS-a".

LIGA MYŚLĄCYCH

Zadanie nr 1

— Roman dojeżdża do szkoły autobusami linii miejskiej, autobusy te w godzinach szczytu jeżdżą w równych od siebie odstępach czasowych, co 6 minut. Czas postoju na przystanku zależy od trzech czynników:
— liczby pasażerów wysiadających,
— liczby pasażerów wsiadających,
— stopnia napełnienia autobusu.
Jeden z autobusów pod wpływem przypadkowych zakłóceń opóźnił się. Co będzie się działo dalej na tej linii autobusowej?

Zadanie nr 2

— Mapa Polski jest podzielona na 49 województw. Na mapie administracyjnej każda para sąsiadujących ze sobą województw powinna mieć inne barwy. Ile najmniej barw trzeba użyć do prawidłowego sporządzenia takiej mapy Polski?

Zadanie nr 3

— Proszę podać czynności Andrzeja przy próbie telefonowania do Piotra w kolejności ich wykonania. Czynności te proszę narysować w dowolnym schemacie rysunkowym, uwzględniając wszystkie okoliczności towarzyszące próbie telefonowania, przy czym schemat rysunkowy powinien zaczynać się jedną czynnością — kończyć też jedną czynnością, a wszystkie one powinny być połączone ze sobą w sposób logiczny.

Zadanie nr 4

— Pociąg pośpieszny mija zawiadowcę stojącego na początku peronu w ciągu 11 sek., a cały peron mający 120 m długości w ciągu 26 sek. Jaka jest długość tego składu pociągu?

Zadanie nr 5

— Aby rozstrzygnąć, która z dwóch drużyn piłkarskich jest lepsza, należy rozegrać jeden mecz. Podaj ile meczy musi rozegrać dwanaście drużyn, grając systemem każdy z każdym.

Rozwiązania zadań prosimy przysyłać pod adresem redakcji do końca czerwca br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji do końca lipca br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

Pocztowa giełda

Rozwiązanie krzyżówki nr 1. Hasło brzmi: **„IKS MOIM PISMEM"**. Bony pieniężne 1000 zł wylosowali: **Przemysław Murawski** Radom, **Elżbieta Bieńkowska** Szczecin, **Ewa Chruściel** Rzeszów, **Zenon Swierczyński** Ogrodzieniec, **Adam Mazurczak** Sokołów Podlaski.

Nagrody książkowe otrzymują: **Zdzisław Woźniczko** Tarnowskie Góry, **Romuald Sztukiewicz** Poznań, **Praveen kant Sharama** Wrocław, **Witold Turkiewicz** Kołobrzeg, **Mariusz Jaworski** Pabianice, **Leszek Szydełko** Kalisz, **Dariusz Ruiski** Piekary Śląskie, **K. Kulikiewicz** Wrocław, **Mikołaj Kmita** Gdynia Karwiny I, **Jerzy Pyszny** Rybnik 12.

W **„Lidze Myślących"** wysoki poziom. Kilkuset uczestników przysłało prawidłowe odpowiedzi. „Liderów" ogłosimy w następnym numerze.

Warszawskie notowania „komputerowe":

ZX Spectrum 48K + magnetofon — 100 tys.
ZX Spectrum 128K — 200 tys.

Atari 600XL + magnetofon — 120 tys.
Atari 800XL + magnetofon — 120 tys.
Atari 130XE — 130 tys.

Commodore C 64 + magnetofon + 2 joystiki — 300 tys.
Commodore VC 20 — 55 tys.
Commodore 16 + magnetofon — 65 tys.

Sharp 700 — 190 tys.
Schneider Amstrad CPC 464 — 250 tys.

Lista „przebojów"

Notowań część druga. Największym powodzeniem cieszył się program **„Odwracanka"** — tak zadecydowali nasi Czytelnicy. „Tu trzeba myśleć" — słyszeliśmy najczęściej w uzasadnieniu. W naszej tabeli na drugie miejsce wysunął się **„Compect"** — program na Spectrum.

Prowadzący w naszych notowaniach do tej pory program **„Wykresy"** spadł na trzecią pozycję, a jego użytkownicy wskazują na niedoskonałości — program zmodyfikujemy i zgodnie z życzeniem posiadaczy innych mikrokomputerów przedstawimy wersję na Commodore i Atari.

Nadal czekamy na Wasze propozycje i opinie.

Ogłoszenia

W naszym komputerlandzie

Tak się jakoś złożyło, że w naszym komputerlandzie wypoczywały ostatnio prawie wszystkie roboty i jeśli nawet któryś miał ochotę, choćby dla draki, od rana coś zrobić, to i tak okazywało się, że jest kolejne święto, dzień wolny, dzień już odpracowany, dzień, który będzie odpracowany, wolna sobota, niedziela itp. Ponieważ zespół robotów w świętowaniu i odpoczynku osiągnął już prawdziwe mistrzostwo, postanowił przetestować następujące sytuacje:

Teza: Robot nie jest stworzony do pracy.

Analiza możliwości: Jak wiadomo z ogólnych doświadczeń praca robota męczy. Po pracy, podobnie zresztą jak przed pracą, robot czuje się zmęczony, wypluty, skołowany oraz oczekuje zbawczego dzwonka na fajrant.

Wniosek: Gdyby robot był stworzony do pracy, to by go praca nie męczyła c.b.d.u.

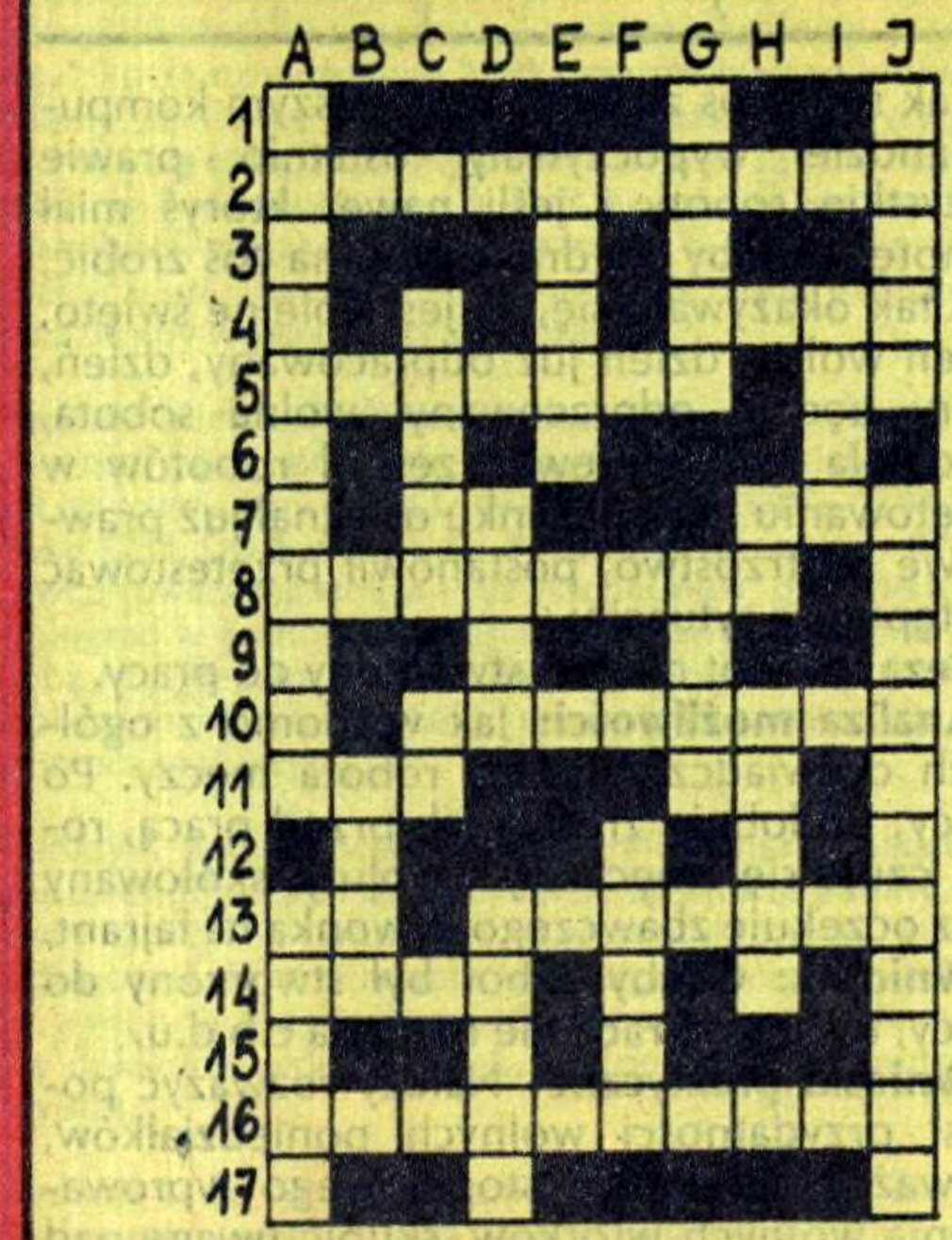
Wnioski praktyczne: Należy rozważyć pomysł przydatności wolnych poniedziałków, rozważyć koncepcję stopniowego wprowadzania wolnych wtorków, skupić uwagę nad projektem zarządzeń w sprawie wolnych środ, przeanalizować ideę ustalenia wolnych czwartków i dokonać wstępnych przymiarek do projektu wolnych piątków.

Gdyby jednak, wśród ogólnego luzu i bimbania znaleźli się ekstremiści i zaprzysięgli malkontenci (jak — nie wytykając — Spektruś), takim należy w nagrodę za dotychczasowe zasługi udzielić bezterminowego urlopu wypoczynkowego i ogłosić jeden wielki CZAS RELAKSU...

Podglądał:
Eugeniusz MLECZAK

Rys. Michał Przybyłowski

Krzyżówka Nr 3



POZIOMO: 2-A zajmuje się elektroniką, 4-G napój alkoholowy, 5-A bizantyjski okręt wiosłowy, 5-I 100 m^2, 7-H popularny kwiat rosnący w zbożu, 8-A reda, 10-C 220 V, 11-A władca wiatrów, 13-A posiada, 13-D bieg na..., 14-A mała Urszula, 16-A np. odnoszenie się referatu lub ustawy do czegoś.

PIONOWO: A-1 przeciwieństwo kodowania, G-1 starsza siostra pompki, I-1 występuje na scenie, E-2 sakwa, kiesa, C-4 atomowa lub sezonu, I-4 może być cukrowa, D-7 imię żeńskie, J-7 zbieracz, G-8 Hg, H-10 wóz pancerny, B-11 kamień szlachetny, F-12 przedmiot, A-13 rzeka przepływająca przez Dessan, D-13 kijek.

HASŁO: 2-I, 14-I, 7-H, 8-E, 14-A, 2-E, 16-G, 13-E, 6-I, 4-I, 8-F, 10-A, 5-D, 10-E, 8-G, 16-F, 16-F, 16-D, 13-I, 5-G 10-H, 2-I, 2-A, 11-C, 11-A, 6-C.

Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji do końca lipca br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

Mikrociekawostki

Komputery serii GRiDCase wyprodukowane przez firmę GRiD Systems, Co. są utrzymane w standardzie IBM. Ich wielkość odpowiada rozmiarom małej teczki. Poszczególne typy różnią się między sobą zastosowanymi monitorami od zwykłych, ciekłokrystalicznych w cenie 2975 $ do gazowo-plazmowych za 4350 $. Pracują opierając się na 16-bitowym mikroprocesorze 80C86. Standardowy akumulator umożliwia ciągłą pracę przez 5 godzin dla komputera z ciekłokrystalicznym monitorem i 1 godzinę dla komputera z monitorem gazowo-plazmowym. GRiDCase wyposażony jest w stacje dysków dla 3,5-calowych dyskietek, z których każda może zawierać 720 KB. (c.)

Rys. Michał Przybyłowski

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera, Ryszard Rogoń. Adres redakcji: 00—950 Warszawa ul. Grzybowska 77, telefon 20-21-27 i 359-34. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 7728. Nr ind. 382809 P- 77