# 2 1987 (10)

# INFORMATYKA KOMPUTERY SYSTEMY

Dodatek „Żołnierza Wolności" ISSN0860-2794 Cena 50 zł

## Grafika Commodore 64

## Sztuki i sztuczki Spectrum

**NIE prowadziliśmy dokładnej buchalterii nadchodzącej do redakcji korespondencji. Wiele odpowiedzi na pytania kierowane do nas przez Czytelników znajdowało się, bądź w najbliższym czasie znajdzie się na łamach „IKS-a". Z otrzymywanych listów wnioskować można o stanie naszej mikroinformatyki — jeżeli pod tym pojęciem rozumieć będziemy stan posiadania, ilość i jakość sprzętu mikrokomputerowego oraz poziom wiedzy na temat jego wykorzystania.**

**Co zatem wiemy? Otóż największą grupą użytkowników są posiadacze poczciwego** *Spectrum* **i wszystkich jego wersji. Niewiele mniejsza grupa to właściciele** *Atari 800 XL;* **systematycznie przybywa nowszych modeli** *Atari 130 XE.* **Trudno jednak powiedzieć, ile komputerów tego typu jest w Polsce. Na nasz apel o przedstawienie się odpowiedziało kilkanaście tysięcy Czytelników — wśród nich najwięcej było użytkowników zdobywającego obecnie polski rynek** *Atari.* **To im właśnie jeszcze niedawno dokuczał brak oprogramowania — szczególnie w mniejszych miastach, na warszawskich giełdach bowiem, programów na ten właśnie komputer jest pod dostatkiem (tak jest również w Krakowie). Podobna sytuacja panuje na rynku oprogramowania** *Spectrum* **— ilość egzemplarzy tego komputera również stale rośnie. Oprogramowanie jest, pojawiają się także programy użytkowe, choć raczej jeszcze nieśmiało torują sobie drogę do indywidualnych użytkowników. Od niedawna przybywa** *Amstradów. Amstrad 464* **jest już co prawda przeszłością konstrukcyjną — do nas jednak nadal napływają jego egzemplarze. Znacznie zmniejszyła się cena tego modelu, a w niektórych komisach komputer ten (wyposażony w zielony monitor) można kupić już za 250—300 tysięcy. Spadła również cena** *Amstrada 6128* **— choć 500—600 tysięcy dla prywatnych właścicieli, a 800 tysięcy do miliona (z zielonym monitorem) dla odbiorców uspołecznionych, to jednak nadal dużo. Niewiele przybywa natomiast** *Commodorów.* **Nowe** *C-128* **są nadal drogie i nie wytrzymują konkurencji z innymi komputerami tej klasy. Mamy też nieco sprzętu egzotycznego:** *Sharpy, Orici, Casio* **— stosunkowo tanie, ale za to z ubogim oprogramowaniem, szczególnie z niewielką ilością gier.**

**Posiadamy zatem przede wszystkim sprzęt zagraniczny. Te niewielkie ilości** *Meritum* **czy** *MK* **są bowiem mniej niż kroplą w morzu potrzeb — pracują głównie w klubach, choć i tam nikt ich nie wita z entuzjazmem. Nie należy się spodziewać rychłej poprawy sytuacji — produkcja z wrocławskiej ELWRO przeznaczona dla szkół (***Elwro 800 Junior***) nie wypełni półek w Składnicy Harcerskiej, nie zaspokoi nawet komputerowego głodu w szkołach. Na taką sytuację będziemy musieli jeszcze poczekać. Tymczasem komputer klasy** *Juniora* **już jest na świecie konstrukcją przestarzałą z co najmniej pięcioletnią historią, a to w przypadku mikrokomputera jest już archeologią!!!**

**Mamy w Polsce jeszcze prototypy** *Mazovii* **i** *Kraka 86* **— jest to sprzęt profesjonalny, ale perspektywy masowej produkcji są raczej mgliste — choć z pewnością jego obecność uzdrowiłaby kaleki rynek komputerowy.**

**Obecna sytuacja sprzyja udanemu wydawaniu dużej ilości pieniędzy, a chcemy przecież udostępnić komputery młodzieży. Nie każdy może jednak wydać co najmniej sto tysięcy, aby rozwijać swoją osobowość. Pozostają jedynie kluby komputerowe, które też potrzebują niemało grosza na rozruch, ceny obowiązujące przy zakupie na rachunek są bowiem u nas znacznie wyższe niż na giełdach, choć sprzęt bywa ten sam! Zakłady pracy również wydają miliony, kilkakrotnie przepłacając prawdziwą wartość komputera — taka jest cena rachunku i frycowe za obowiązujące przepisy finansowe.**

**Lepiej jest z wiedzą. Niczym grzyby po deszczu rosną domorośli informatycy, przed którymi komputer nie ma tajemnic. I jeśli byłoby u nas można włamać się do jakiegoś komputerowego banku danych, to z pewnością już by się włamali — na szczęście nam to nie grozi.**

**Grupa dociekliwych jest zdecydowanie większa od tej, w której poszukuje się rozwiązań gotowych, najchętniej poprawnie działających programów.** *„Dlaczego drukujecie program Sharpa, a nie Spectrum?"* **— pytano mnie w liceum ogólnokształcącym, na południu Polski, w którym informatyka jest już przedmiotem nauczania! Kochani, jeśli jeszcze nikt Wam tego nie powiedział, to pomyślcie sami. Ileż to roboty, aby dialekt Basica obowiązujący w** *Sharpie* **przerobić na** *Spectrum?* **Przecież algorytm działania jest ten sam, inne jedynie są instrukcje we/wy. Ale jeśli nie wszyscy potrafią, to słownik podstawowych instrukcji Basica różniących wersję obowiązującą na** *Amstradzie* **i** *Sharpie* **od tej, którą dysponuje** *Spectrum,* **podajemy już w tym numerze.**

**A teraz życzymy przyjemnej lektury. Czekają na Was programy, nauka Basica, a dla tych, którzy lubią pogrzebać się w ROM-ie kolejny odcinek „Sztuk i sztuczek".**

**WIESŁAW CETERA**

# Jak to zrobić (4)

Dziś przedstawiam „Edytor znaków" — program pozwalający na przygotowanie ich według własnego życzenia. W czasie pracy na ekranie widoczne jest pole 8 x 8, na którym za pomocą joysticka (powoduje ruch) i przycisku fire (zmienia kolor punktu wskazującego położenie kursora — pozwalając na zamalowywanie lub czyszczenie punktów) możemy utworzyć rysunek własnego znaku. Dodatkowo, na wysokości każdego wiersza rysunku podawana jest liczba, w znany nam sposób kodująca ten wiersz. W trakcie tej twórczej pracy możemy na bieżąco obserwować, jak nasz znak będzie wyglądał w grafice Ø, 1 i 2.

Program ten jest dodatkowo wzbogacony o :
— odczytywanie stanu klawiszy funkcyjnych,
— elementy samomodyfikacji programu.

Dla lepszego zrozumienia tych zagadnień niezbędne są jednak pewne dodatkowe informacje.

I tak komórka o adresie 53279 pozwala nam stwierdzić, jaki klawisz funkcyjny został wciśnięty. Może w niej znajdować się wartość z przedziału Ø — 7 oznaczająca użycie:

Ø — OPTION, SELECT, START
1 — OPTION, SELECT
2 — OPTION, START
3 — OPTION
4 — SELECT, START
5 — SELECT
6 — START
7 — żaden klawisz nie został wciśnięty

W programie wykorzystane mogą być następujące klawisze:
SELECT — w zależności od postaci

punktu wskazującego położenie kursora, zamienia go na biały prostokąt zaciemniający zajęte pole lub pozostawia wolne miejsce (pozwala spokojnie przyjrzeć się utworzonemu znakowi); jest sygnałem do analizy klawiszy funkcyjnych,
SELECT/START naciśnięte jednocześnie ponawiają proces edycji po wciśnięciu SELECT
START — rozpoczyna od początku edycje znaku
OPTION/START naciśnięte jednocześnie kończą działanie programu,
OPTION — w treści programu zapamiętuje dane do utworzenia przygotowanego znaku; przed użyciem tego klawisza należy wiedzieć, jaki znak chcemy przedefiniować.

W ten sposób doszliśmy do dziwnie brzmiącego, a w gruncie rzeczy bardzo prostego do realizacji hasła samomodyfikacji programu. To dziwne zjawisko polega na kontrolowanej ingerencji programu w swoją treść, w rezultacie czego zmienić możemy jego działanie, a nawet otrzymać całkiem nowy program.

Uwagi na ten temat zacznijmy od pokazania... jak to zrobić. Otóż wstawienie do komórki o adresie 842 wartości 13 powoduje ustawienie naszego komputera w tryb czytania z ekranu. Umiejętne wykorzystanie tej sytuacji zasymuluje proste wciśnięcie klawisza BREAK, wprowadzenie zadanych poprawek i następnie instrukcji CONT. Cały pomysł opierać się musi na tym, aby na ekranie wypisać właściwe instrukcje i zmusić komputer do ich realizacji. Spójrzmy zatem, jak wygląda nic nie robiący schemat rozwiązania naszego problemu:

```
10 GRAPHICS 0
20 POSITION 2,4
30 REM WYPISANIE INSTRUKCJI OD
DRUGIEJ KOLUMNY
40 PRINT „CONT"
50 POSITION 2,2 : REM Instrukcja
                  STOP powodu-
                  je wypisanie na
                  ekranie dwóch
                  linii, musimy
                  ustawić kursor
                  o te dwie linie
                  wyżej
60 POKE 842,13 : REM Czytanie ins-
                  trukcji z ekranu
70 STOP
80 POKE 842,12 : REM Powrót do nor-
                  malnego trybu
                  pracy
```

W programie powyższe instrukcje mają za zadanie umieścić w odpowiednim jego miejscu instrukcje DATA z danymi potrzebnymi do zdefiniowania aktualnie stworzonego znaku.

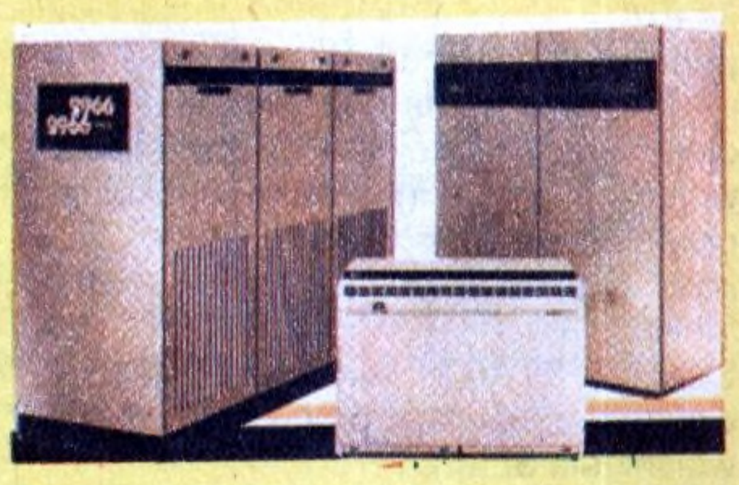
Czytelnikom, którym prezentowany program może pomóc w pracy proponuję wykorzystanie tej możliwości do przygotowania procedury zmiany znaków. Po zakończeniu działania programu można go zapisać np. na kasecie (LIST "C:") i umieścić w każdym ze swoich programów (ENTER "C:"). Pierwszy krok jest już dokonany, w przypadku użycia klawiszy OPTION/START, do zakończenia pracy programu usuwana jest jedna z jego linii, by po napisie READY otrzymać zadaną procedurę należy wymazać jeszcze kilka linii.
Ale jak to zrobić...?

**Grześ**

```
10 REM Program demonstruje metode
15 REM tworzenia wlasnych znakow
20 REM
25 REM Rownoczesnie pokazany jest
30 REM sposob wykorzystania klawiszy
       funkcyjnych
35 GRAPHICS 0
40 DIM W(8)
45 MEM=PEEK(106)-8:POKE 106,MEM:POKE 7
56,MEM
50 CHADR=256*MEM:POKE 752,2
55 ZN=ASC("!"):ZN=CHADR+(ZN-32)*8
60 REM Kopiowanie zbioru znakow
65 GOSUB 1000:LAB=1100
69 REM
70 REM Zmiany w  Display List
71 REM
75 DL=PEEK(560)+256*PEEK(561)
80 POKE DL+23,6
85 POKE DL+25,7
89 REM
90 REM Zmiana koloru ekranu
91 REM
95 FOR K=0 TO 2:POKE 710+K,144:NEXT K
99 REM
100 REM Tworzenie obrazu na ekranie
101 REM
105 POSITION 5,0:PRINT "E D Y T O R
Znakow"
110 POSITION 5,2:PRINT ""
115 FOR I=3 TO 10
120 POSITION 4,I:PRINT "
  0    "
125 W(I-2)=0:POKE ZN+I-3,0
130 NEXT I
135 POSITION 5,11:PRINT ""
140 X=5:Y=3:C0=32:C1=32:C2=20
149 REM
150 REM Petla glowna
151 REM
160 S=STICK(0):ST=STRIG(0)
165 L=0:I=0
170 IF ST=0 THEN C1=192-C1:C2=168-C2
175 POSITION X,Y:PRINT CHR$(C2)
180 OX=X:OY=Y:OC=C0
185 IF (S>8) AND (S<12) AND (X>5) THEN
 X=X-1:L=1
190 IF (S>4) AND (S<8) AND (X<12) THEN
 X=X+1:L=1
195 IF (S=10 OR S=14 OR S=6) AND (Y>3)
 THEN Y=Y-1:L=1
200 IF (S=13 OR S=9 OR S=5) AND (Y<10)
 THEN Y=Y+1:L=1
205 IF L=0 THEN GOTO 275
210 POSITION OX,OY:PRINT CHR$(C1)
215 IF C0=C1 THEN GOTO 250
220 REM Zmiana znaku
225 IF C1=160 THEN I=1
230 IF C1<>160 THEN I=-1
235 OX=12-OX:OY=OY-2
240 W(OY)=W(OY)+I*2^OX
245 POKE ZN+OY-1,W(OY):POSITION 20,OY+
2:PRINT W(OY);"      "
250 LOCATE X,Y,C0:POSITION X,Y:PRINT C
HR$(C2)
255 REM Wypisanie wzoru znaku
260 POSITION 6,16:PRINT "!  !  !  !  !
265 POSITION 3,18:PRINT "! ! ! !"
270 POSITION 23,19:PRINT "! ! ! !"
275 IF PEEK(53279)=6 THEN GOTO 105
279 REM Analiza stanu klawiszy
       funkcyjnych
280 POSITION X,Y:PRINT CHR$(C1):IF PEE
K(53279)<>5 THEN GOTO 160
285 WY=W(Y-2):WW=2^(12-X):IF (C1=160 A
ND C0<>160) THEN W(Y-2)=WY+WW
290 IF (C1<>160 AND C0=160) THEN W(Y-2
)=WY-WW:POSITION X,Y:PRINT CHR$(C1)
295 POKE ZN+Y-3,W(Y-2):POSITION 20,Y:P
RINT W(Y-2);"      "
300 REM Osladanie znaku
305 IF PEEK(53279)<=3 THEN GOTO 350
310 IF PEEK(53279)<>4 THEN GOTO 305
```

dokonczenie na str. 23

*Z dużym zainteresowaniem przeczytałem w BAJTKU refleksje na temat „Staruszki ODRY" (S. Polak, R. Wojciechowski, Nie tylko małe, BAJTEK 10/86.) Jestem jednym z wielu, którzy poznawali informatykę, dzięki tej, wówczas bardzo nowoczesnej, maszynie. Mimo upływu lat jeszcze dzisiaj komputery z ELWRO (obecnie MERAELWRO) dostarczają podstawowych mocy obliczeniowych do profesjonalnych zastosowań komputerów w kraju.*



# ICL 1900... to już historia

Szkoda tylko, że w bajtkowych refleksjach „zabrano": przetwornicy koło zamachowe, każdemu ze sterowników (kontrolerów lub po prostu PDSów) po dwa napędy dyskowe, a dyskom trochę (300%) pamięci (ostatnio nie 8, a 32 Mega wynosi pojemność każdej stacji).

Najważniejsze jest jednak, że w myśl zasady: „szanujmy wspomnienia, panowie " — temat zasłużonego komputera rodzimej produkcji znalazł trochę miejsca w naszej bardzo poczytnej prasie informatycznej.

ODRA jest w pełni kompatybilna z ICL 1900. Ale... to już niemal historia. Co słychać teraz w firmie ICL?

Okazuje się, że proponowany przez ICL sprzęt i oprogramowanie są nowoczesne i konkurencyjne wobec innych firm zachodnich, także IBMa. Przykładem jest nowa generacja systemów komputerowych „SERIA 39". Są to dwie całkowicie nowe maszyny: LEVEL 80 i LEVEL 30 — pierwszy owoc współpracy ICL z FUJITSU.

Komputery systemu LEVEL 80 wykonują ponad 13 milionów operacji na sekundę przy pojemności pamięci RAM od 16 do 32 Mbajtów. Natomiast LEVEL 30 jest nieco skromniejszy i jest przeznaczony do typowych biur. Komputer ten wykonuje dwa miliony operacji na sekundę, a jego pamięć RAM wynosi od 4 do 16 Mega. Imponującym uzupełnieniem pamięci RAM tych komputerów są dyski o pojemności od 1000 do 10 tysięcy Mega!

Rewelacją w skali światowej są proponowane od ponad roku przez ICL sieci typu MACROLAN. Połączenia sieci — to kable zbudowane ze światłowodów. Przekazują one informacje z ponad 1250 stron maszynopisu w czasie jednej sekundy, to jest ok. 50 Megabitów na sekundę. Technikę tę wykorzystano do sprzężenia szybkich urządzeń peryferyjnych, takich jak na przykład dyski, a także do łączenia zestawów komputerów w duże systemy — sieci komputerowe. Odległość między sprzęganymi elementami nie powinna przekraczać 1500 m.

Kabel światłowodu tworzą dwa włókna: jedno do transmisji informacji, drugie do sterowania transmisją. Łączna średnica wynosi zaledwie 5 mm. Dołączane są one do urządzeń za pośrednictwem przystawki wielkości teczki (Port Switch Unit). Producent radzi, by „teczkę" tę wieszać na ścianie. Łączy ona sześć linii i może wchodzić w skład systemu do 15 stacji MARCOLANa.

Nieco tańszym rozwiązaniem jest OSLAN (Open Systems Local Area Network). Łączy terminale, drukarki, i inne urządzenia komputerowych zestawów. Wykorzystano tu tradycyjny pośrednik przesyłania informacji — kabel koncentryczny. Kabel wewnętrznej sieci OSLANa ma średnicę 10 mm, odcinki po 500 m, co 1500 m wzmacniacz. Nie byłoby w tym nic nadzwyczajnego, gdyby nie wartości parametrów tej sieci — szybkość transmisji wynosząca 10 Megabitów/sek. Jest to ekwiwalentem około 1000 szybkich linii transmisji lub inaczej równoznaczne z przesyłaniem 250 stron A4 w czasie jednej sekundy. OSLAN może „jednocześnie obsłużyć" 250 szybkich drukarek.

Obecnie system OSLAN stosowany jest do łączenia komputerów Serii 39 z DRS 20, System 25, PERQ i SERIA 2900, a także sprzęgania z siecią telefoniczną całego świata.

Nowoczesne komputery firmy ICL wyposażone są w szybkie kontrolery dysków. Ostatnio dostępne są napędy: FDS 300 z dyskami o pojemności 337 Mega i szybkości transmisji 2,46 Mb/s oraz FDS 2500 o pojemności 2521 Mbajty i szybkości transmisji 2,86 Mb/s. Zatem każdy kontroler łączący cztery napędy FDS 2500 stanowi dla komputera pamięć dyskową o pojemności ponad 10 GIGA!

Prezentowane parametry, bardzo atrakcyjne parametry, „wnuków, może prawnuków, stryjecznych" ODRY zachęcają do stosowania ich w wielu sytuacjach, gdzie szybkość przetwarzania dużych zbiorów informacji jest celem nadrzędnym. Wiąże się to zazwyczaj z koniecznością zapewnienia bezawaryjnej pracy (elektrownie, fabryki, koleje itp.) maszyn. Osiąga się to przez systemowe dublowanie komputerów lub taniej (w nieco ograniczonym zakresie) przez tak zwany system wczesnego ostrzegania.

Komputery Serii 39 mogą być połączone za pośrednictwem zwykłej telefonicznej linii z centrum diagnozy i kontroli — ICL Support Centres. Linia ta pośredniczy w bezustannej kontroli poprawności pracy sprzętu i oprogramowania komputera bez udziału ludzi. Diagnoza zestawu komputera dotyczy wszystkich elementów: pamięci taśmowych, dyskowych, RAMowskich, logiki, działania systemu operacyjnego itp. W razie, gdy kontrola stanu lub działania komputera wypadnie niepomyślnie, według oceny komputera centralnego zainstalowanego w Support Centres, natychmiast powiadamiany jest o tym użytkownik ze wskazaniem miejsca ewentualnej lub zaistniałej awarii.

Oferta programowa ICLa jest nie mniej atrakcyjna niż sprzęt (bazy danych, edytory, języki itp.). Także UNIX uznawany jest jako przyszłościowy system operacyjny potomków naszej ODRY.

**WŁODZIMIERZ GOGOŁEK**

# PROGRAM COMMODORE 5

**Oprogramowanie dydaktyczne nie nadąża za rozwojem oprogramowania użytkowego, a w porównaniu z tak popularnymi grami to praktycznie margines zastosowań.**

Przedstawiony program jest przykładem wykorzystania mikrokomputera właśnie w dydaktyce, zarówno przez uczących się, jak i wykładowców. Jest testem z wyborem wariantu odpowiedzi. Nie dotyczy konkretnego, tematycznego zastosowania. Własne pytania, warianty odpowiedzi, a także dane do oceny (prawidłowe odpowiedzi, liczba dopuszczalnych błędów) koduje użytkownik. Ilość pytań zależy tylko od pojemności pamięci komputera.

Zakodowane pytania zostają zapisane na dyskietce, w zbiorze DANE. Z jednolitą nazwą zbioru zapisaną w wierszach 2010 i 3020 poradzi sobie każdy. Można używać różnych dyskietek lub rozbudować program, wprowadzając alternatywne nazwy zbioru. Można także składować kolejne wersje programu ze zmienioną nazwą (zarówno programu, jak i zbioru danych).

Po wykonaniu tych operacji TEST gotowy jest do zapisu kolejnych pytań (tematów).

Program zapewnia wykonanie następujących operacji;

Z — zapisanie przygotowanych pytań i wariantów odpowiedzi (wiersze 300-500), prawidłowych odpowiedzi dla każdego pytania (510--560) oraz dopuszczalnej liczby błędów dla uzyskania ocen: b.db., db., dst. (600-790). Zatrzymanie programu w wierszu 700 umożliwia przyłączenie dysków. Po wciśnięciu dowolnego klawisza następuje zapis danych (pytań) na dyskietce.

T — realizację testu. Wyprowadzone są kolejne pytania wraz z wariantami możliwych odpowiedzi (800-920). Należy każdorazowo wcisnąć klawisz numeryczny odpowiadając przyjętej (za prawidłową) odpowiedzi. Za każdą prawidłową odpowiedź uzyskujemy jeden punkt.

Wyprowadzenie ilości poprawnych i błędnych odpowiedzi oraz uzyskanej oceny kończy test.

K — korekta zapisów. Wyprowadzane są kolejno wszystkie wprowadzone na etapie „Z" dane. Jeżeli wiersz zapisany jest prawidłowo należy wcisnąć klawisz RETURN, gdy nie — wpisać prawidłowy tekst.

Każde uruchomienie programu wymusza podanie ilości pytań i wariantów odpowiedzi, a operacje „T" i „K" poprzedzane są zawsze wczytaniem zbioru danych z dyskietki.

Program może być wykorzystany w praktyce, ale powstał w zasadzie w celu pokazania idei (jednego ze sposobów) wykorzystania mikrokomputera w dydaktyce. Stąd proste (łatwość adaptacji na innych komputerach) instrukcje oraz sprowadzone do minimum procedury kontroli i korekty danych.

Nowy program można rozbudować o nowe elementy, takie jak: wprowadzenie nazwiska, wyprowadzenie wyników na drukarkę, różną ilość punktów za poszczególne odpowiedzi (w tym punkty ujemne za błędne odpowiedzi).

**KRZYSZTOF ŚWIĘTOŃ**

```
10 PRINT"□"
20 PRINT:PRINT:PRINT:PRINT:PRINT"     T E S T"
30 PRINT:PRINT:PRINT:PRINT:PRINT
40 INPUT "PODAJ ILOSC PYTAN    ";A
50 INPUT "PODAJ ILOSC WARIANTOW";B
60 PRINT"□"
100 PRINT:PRINT:PRINT:PRINT"   K O D Y   P R A C Y"
110 PRINT"  ----------------------":PRINT:PRINT:PRINT
120 PRINT"Z  - ZAPIS PYTAN":PRINT:PRINT
130 PRINT"T  - TEST":PRINT:PRINT
140 PRINT"K  - KOREKTA":PRINT:PRINT
150 INPUT "PISZ KOD";A$
200 IF A$="Z" THEN 300
210 IF A$="T" THEN 800
220 IF A$="K" THEN 1300
230 PRINT:PRINT:PRINT:PRINT"BLEDNY KOD PRACY"
240 INPUT A$
250 GOTO 60
300 DIM Z$(A*(B+3)+3)
310 I=0
320 FOR K=1 TO A
330 PRINT"□":PRINT:PRINT"PYTANIE NR - ";K
340 PRINT:PRINT:PRINT:INPUT Z$(I)
350 FOR J=1 TO B
360 I=I+1
370 PRINT:PRINT:PRINT"WARIANT ODPOWIEDZI NR - ";J
380 INPUT Z$(I)
390 NEXT J
400 I=I+1
410 NEXT K
490 J=1
500 FOR I=A*(B+1) TO A*(B+2)-1
510 PRINT"□"
520 PRINT:PRINT:PRINT"WPROWADZ PRAWIDLOWE ODPOWIEDZI"
530 PRINT:PRINT J;" - "
540 INPUT Z$(I)
550 J=J+1
560 NEXT I
```

```
600 PRINT"□":PRINT:PRINT:PRINT"DOPUSZCZALNA ILOSC BLEDOW DLA OCEN
610 PRINT:PRINT
620 INPUT "B. DB. - ";Z$(A*(B+3))
630 PRINT:PRINT
640 INPUT "DOBRY  - ";Z$(A*(B+3)+1)
650 PRINT:PRINT
660 INPUT "DOST.  - ";Z$(A*(B+3)+2)
700 PRINT"□":PRINT:PRINT
710 INPUT"ZAPISZ DANE";B$
720 GOSUB 2000
730 PRINT:PRINT:PRINT:PRINT:PRINT
790 STOP
800 I=0:GOSUB 3000
810 L=A*(B+2)
820 FOR K=1 TO A
830 PRINT"□":PRINT:PRINT:PRINTK;" - ";Z$(I)
850 FOR J=1 TO B
860 I=I+1
870 PRINT:PRINT:PRINT"   ";J;" - ";Z$(I)
880 NEXT J
890 I=I+1:PRINT:PRINT
900 INPUT "ODPOWIEDZ ";Z$(L)
910 L=L+1
920 NEXT K
940 S=0:K=A*(B+2)
950 FOR L=A*(B+1) TO A*(B+2)-1
960 IF Z$(K)<>Z$(I) THEN 980
970 S=S+1
980 K=K+1
990 NEXT I
1000 PRINT"□":Z=A-S
1010 PRINT:PRINT:PRINT"ILOSC PYTAN          - ";A
1020 PRINT:PRINT:PRINT"POPRAWNYCH ODPOWIEDZI - ";S
1030 PRINT:PRINT:PRINT"BLEDNYCH ODPOWIEDZI   - ";Z
1040 PRINT:PRINT:PRINT"-------------------------"
1100 T=VAL(Z$(A*(B+3)))+1
1110 U=VAL(Z$(A*(B+3)+1))+1
1120 V=VAL(Z$(A*(B+3)+2))+1
1130 S$="NDST.  (2)"
1150 IF Z<V THEN S$="DST.   (3)"
1160 IF Z<U THEN S$="DOBRY  (4)"
1170 IF Z<T THEN S$="B. DB. (5)"
1200 PRINT:PRINT:PRINT"   O C E N A       - ";S$
1210 PRINT:PRINT:PRINT:PRINT:PRINT
1290 STOP
1300 GOSUB3000
1310 I=0:L=1
1320 PRINT"□":PRINT:PRINT"PYTANIA"
1330 FOR K=1 TO 2*A+3
1335 PRINT:PRINT:PRINT:PRINTL;" - ";Z$(I)
1340 B$="":INPUT B$:IF B$="" THEN 1360
1350 Z$(I)=B$
```

dokończenie na str. 6

dokończenie ze str. 5

```
1360 I=I+1:L=L+1
1370 IF K>A THEN1500
1380 FOR J=1TOB
1390 PRINT"   ";J;". - ";Z$(I)
1400 B$="":INPUT B$
1410 IF B$="" THEN 1430
1420 Z$(I)=B$
1430 I=I+1
1440 NEXT J
1500 IF K=A THEN GOSUB 1800
1520 IF K=2*A THEN GOSUB 1900
1530 NEXT K
1540 GOSUB2000
1550 PRINT:PRINT:PRINT:STOP
1800 L=1
1810 PRINT"□":PRINT:PRINT:PRINT"ODPOWIEDZI"
1820 RETURN
1900 L=1
1910 PRINT"□":PRINT:PRINT:PRINT"DOPUSZCZALNA ILOSC BLEDOW"
1920 PRINT"DLA OCEN : B,DB. ,DB. ,DST.":PRINT:PRINT
1930 RETURN
2000 OPEN15,8,15
2010 OPEN3,8,3,"@0:DANE,S,W"
2020 FOR K=0 TO A*(B+3)+3
2030 PRINT#3,Z$(K)
2040 NEXT K
2050 A$="****"
2060 PRINT#3,A$,CHR$(13);
2070 CLOSE 3:CLOSE15
2080 RETURN
2420 Z$(I)=B$
3000 OPEN 15,8,15
3010 DIMZ$(A*(B+3)+3)
3020 OPEN3,8,3,"0:DANE,S,R"
3030 FOR K=0 TO A*(B+3)+3
3040 INPUT#3,A$
3045 Z$(K)=A$
3050 IF A$="****" THEN 3070
3060 NEXT K
3070 CLOSE3:CLOSE15
3080 RETURN
```

# UWAGA! ATARI

## Komputerowy teleexpress

Zapewne nieraz widzieliście efektowny napis świetlny biegnący wzdłuż telewizyjnego ekranu. Bardzo często efekt ten wykorzystuje się w celach reklamowych. Może on mieć również zastosowanie w redagowaniu gazety świetlnej lub też możecie go wykorzystać np. do przesłania bardzo atrakcyjnych edytorsko życzeń koleżance lub koledze, którzy dysponują minikomputerem „Atari". Program ten jest nadzwyczaj prosty i łatwo można go rozbudować. A więc, przystępujemy do zredagowania naszej pierwszej telegazety. A może... na naszym domowym ekranie wyświetlimy komputerowe życzenia noworoczne?... Spróbujcie!

```
10 DIM TEKST$ (200)
20 GRAPHICS 2
30 PRINT "PROSZĘ WPROWADZIĆ TEKST"
40 DIM SP$ (200)
50 INPUT TEKST$
60 SP$= " "
70 SP$ (LEN (SP$) + 1) = TEKST$
80 TEKST$ = SP$
90 TEKST$ (LEN (TEKST$) + 1) = " "
100 ML=LEN (TEKST$)
110 POKE 752,1
120 PRINT " ) "
130 SETCOLOR 2,0,0
140 FOR P=1 TO ML
150 EL=P+19:IF EL>ML THEN EL=ML
160 POSITION 0,5
170 PRINT #6; TEKST$ (P, EL)
180 FOR W=1 TO 99:NEXT W
190 NEXT P
200 GOTO 140
```

## „IKS UCZY": Projektujemy stronę tytułową programu

Każdy z Was zdążył już zebrać sporo programów na „Atari" — chociażby ułożonych na podstawie iksowych listingów. Warto więc zadbać i o to, by miały one ładną stronę tytułową. Prezentowany tu krótki program jest jedynie wstępem do tej pożytecznej zabawy. Można go wzbogacić np. o miłe dla ucha efekty dźwiękowe, czy też swój własny piktogram lub hasło; datę; można wpisać jakąś dedykację itp. Przyjemnej zabawy!

Atari 260ST

```
10 GRAPHICS 2
20 SETCOLOR 2,0,0
30 POSITION 6,4
40 PRINT #6; „IKS UCZY"
50 FLIP=224
60 PRINT „NACIŚNIJ START"
70 SETCOLOR 0,0,0
80 FOR WAIT=1 TO 500: NEXT WAIT
90 SETCOLOR 0, HUE,6
100 HUE=HUE+1
110 POKE 756,FLIP:FLIP=450—FLIP
120 FOR WAIT=1 TO 500:NEXT WAIT
130 IF PEEK (53279) <> 6 THEN GOTO 70
140 POKE 756,224
```

## „Atari" udaje, że kreśli koła

Użytkownicy „Atari" — szczególnie najmłodsi, nie bardzo wiedzą, jak posługując się wyłącznie instrukcjami „BASIC-u" wykreślić koło. Podsuwamy więc pomysł na wykreślenie kilku kół koncentrycznych względem siebie. Wprawdzie do elegancji daleko im, jednak zawsze to już coś.

```
10 GRAPHICS 7
20 DEG
30 FOR J=1 TO 360
40 COLOR J
50 FOR I=1 TO 360
60 PLOT 80+INT (J * 10 * COS (I)), 40 + INT (J * 10 *
SIN (I))
70 NEXT I
80 NEXT J
```

**opracowała ANITA FRYSZKIEWICZ**

GRAFIKA

# KRESKA

## (Basic CPC 464) (1)

**Każdy z nas w mniejszym lub większym stopniu potrafi posługiwać się instrukcjami interpretera BASIC rezydującego w komputerze. Istniejące rozkazy interpretera, umożliwiają w praktyce narysowanie każdej płaskiej figury geometrycznej, a także realizację nieskomplikowanej grafiki przestrzennej.**

Zamieszczony i opisany niżej program KRESKA pozwala na tworzenie prostych rysunków na ekranie monitora. Jest jednocześnie ilustracją określonych rozwiązań możliwych do zrealizowania przy użyciu języka BASIC. W niniejszym numerze „IKS-a" zamieszczona została podstawowa część tego programu. W kolejnych odcinkach będzie się on rozrastał o procedury realizujące kolejne funkcje. Chciałbym, aby jego końcowa postać była przydatnym narzędziem do tworzenia rysunków. Nie będzie to jednak grafika komputerowa, gdyż nie program będzie jej autorem. W naszym przypadku ekran monitora zastąpi papier i farby, a klawisze kursora lub joystick — pędzel i kredki. Ponieważ w kolejnych artykułach zamieścimy fragmenty programu, które dołączać będziemy do części podstawowej, mamy więc szansę wspólnie tworzyć, przez kilka kolejnych miesięcy, program graficzny. Jego jakość będzie zależała od naszych pomysłów i rozwiązań. Przygotowałem dwa pierwsze artykuły. Mam koncepcję odnośnie dalszej rozbudowy programu. Chciałbym jednak, abyśmy wspólnie spróbowali dokończyć „dzieła" — oczekuję Waszych propozycji i uwag.

* * *

Podstawowym parametrem decydującym o jakości widzianego przez nas obrazu jest jego rozdzielczość, czyli maksymalna ilość punktów świetlnych, na jakie jest podzielony ekran. Niezależnie od trybu pracy każdy punkt możemy opisać parą współrzędnych x, y. Pozycję punktu wzdłuż osi Y można określić za pomocą liczb od 0 do 399, przy czym dwie kolejne wartości odpowiadają jednej pozycji na ekranie. Oś X posiada numerację od 0 do 639. Zatem ekran jest podzielony na 640 punktów w poziomie i 400 punktów w pionie. Rozwiązanie takie przyjęto dla lepszego zachowania proporcji obrazu. CPC 464 może wyświetlać znaki w trzech trybach:

MODE 0 — tryb wielobarwny. Mamy do dyspozycji jedynie 20 kolumn i 25 wierszy tekstu oraz 160 x 200 punktów w przypadku grafiki. Możliwe jest jednoczesne wykorzystywanie 16 kolorów z palety 27 barw ponumerowanych do 0 do 26.

MODE 1 — tryb standardowy. Mieści on 40 kolumn i 25 wierszy tekstu, co odpowiada 320 x 200 punktom dla grafiki. Ale do dyspozycji mamy już tylko cztery kolory z 27.

MODE 2 — tryb wysokiej rozdzielczości. Pozwala na równoczesne użycie jedynie dwóch z 27 barw, lecz wynagradza nam to dostępność 80 kolumn i 25 wierszy tekstu, lub 640 x 200 punktów w przypadku obrazów.

Jak widać parametrem związanym ze zmianą trybu pracy jest wielkość punktów kreślonych wzdłuż osi X. Ponieważ numeracja nie zależy od trybu i wynosi, jak już wspomniałem, od 0 do 639, tak więc, jedna pozycja na ekranie w poziomie odpowiada kolejnym trzem, dwóm lub jednej wartości.

BORDER (ramka) to obszar otaczający PAPER (tło). Znaki wyświetlane na ekranie mogą ukazywać się tylko wewnątrz ramki. PEN (pióro) pisze sam znak. INK (atrament) możemy sobie wyobrazić jako zbiorniczek wypełniony atramentem o wybranym kolorze. Relacje między komendami PAPER, PEN, INK są ustawiane standardowo w chwili włączenia komputera. Możemy je zmieniać poprzez użycie instrukcji INK.

O miejscu zapalenia punktu na ekranie decyduje położenie kursora. Zazwyczaj w programach wykorzystujemy dwa kursory: znaku i graficzny. Różnią się one przeznaczeniem oraz wielkością — działają w sposób rozłączny. Kursor znaku ma wielkość matrycy znaku i dopóki nie użyjemy rozkazu LOCATE zaczyna on ruch od lewego górnego rogu ekranu co odpowiada współrzędnym (x,y)=(1,1), gdzie x oznacza numer kolejny, a y numer wiersza. Wielkość maksymalna współrzędnej x determinowana jest przez tryb pracy (MODE) i przybiera wartości 20,40 lub 80. W przeciwieństwie do kursora znaku współrzędne kursora graficznego odnoszą się do lewego dolnego rogu ekranu (x,y)=(0,0). Współrzędne opisujące położenie tego kursora nie zależą od trybu. Kursor graficzny nie jest wyświetlany na ekranie. Do pokazania jego położenia może być użyty rozkaz PLOT.

Pewne instrukcje języka BASIC wprowadzające wyniki na określone urządzenie wyjścia używają pojęcia stream — strumień, przy czym stream opisywany jest cyfrą od 0 do 9. Pojęcie stream określa, jakie urządzenie wyjścia powinno być wykorzystane. Znak # to znacznik strumienia. Skierowanie danych do strumieni o numerach od # 0 do # 7 powoduje wyświetlenie ich na ekranie monitora. Strumienie o powyższych numerach są przypisane do całego ekranu lub tzw. okienek (agn. WINDOWS), o ile zostaną one zdefiniowane. Skierowanie danych do strumienia # 8 powoduje ich przesłanie na drukarkę. Strumień # 9 związany jest natomiast z pamięcią kasetową.

Po tych ogólnych uwagach przejdźmy do prezentowanego przykładu.

Ekran w naszym programie został zorganizowany w trybie standardowym (MODE 1). Podzieliłem go na dwie części. Górna, wiersze 1 i 2, w trybie znakowym, spełniają funkcję informacyjną. W lewym górnym rogu (WINDOW # 5) wyświetlane są bieżące współrzędne kursora graficznego. Część środkowa (WINDOW # 2) przeznaczona została do prezentacji funkcji możliwych do zrealizowania w danej chwili (tzw. menu), zaś górny prawy róg (WINDOW # 3) wykorzystałem do ukazania wybranego koloru pióra. Pozostała część ekranu (WINDOW # 1) — wiersze 3 do 25 — służyć nam będzie do tworzenia rysunków. Kreślenia dokonuje się za pomocą klawiszy sterujących ruchem kursora lub joystickiem.

Program KRESKA realizuje trzy podstawowe funkcje:

— kreślenie — opcja wybierana klawiszem z literą K;
— czyszczenie części ekranu przeznaczonej do rysowania — opcja C;
— zakończenie pracy — opcja Z.

**dokończenie na str. 8**

**dokończenie ze str. 7**

Wybór opcji K — kreślenie, daje nam następujące możliwości:
— przemieszczanie kursora graficznego bez śladu — opcja P;
— wybór koloru pióra — opcja K;
— rysowanie „punkt po punkcie" opcja W;
— wycieranie niepotrzebnych fragmentów rysunku — opcja G.

Podstawowymi komendami języka BASIC wykorzystanymi w programie KRESKA są:

BORDER <nr INK> — określa kolor ramki

CLS [#<stream>] — wypełnia określony zakres ekranu odpowiednim dla tła kolorem.

DRAW x,y, <nr INK> — rysuje linię od aktualnej pozycji kursora graficznego, (x',y') do nowego punktu (x,y).

INK <nr atr>,<nr> — przypisuje atramentowi o numerze nr atr kolor o numerze nr, jaki zostanie użyty przez rozkazy PEN lub PAPER.

LOCATE [#<stream>],x,y — określa miejsce położenia kursora znakowego na ekranie w okienku określonym numerem stream.

MOVE x,y — przesuwa kursor graficzny do punktu określonego współrzędnymi (x,y).

MOVER a,b — przesuwa kursor graficzny z punktu (x,y) do punktu (x',y') = (x+a, y+b).

PAPER [#<stream>],<nr INK> — ustawia kolor tła w okienku ekranu o numerze # stream, zadeklarowanym komendą WINDOW.

PEN [#<stream>],<nr INK> — określa kolor atramentu pióra, który jest używany do pisania w okienku o numerze # stream.

PLOT x,y, <nr INK> — rysuje punkt o współrzędnych (x,y). Kolor punktu określony jest numerem atramentu.

PLOTR a,b, <nr INK> — rysuje punkt względem bieżącego położenia kursora graficznego (x,y), o współrzędnych (x+a, y+b) kolorem określonym przez numer atramentu.

WINDOW [#<stream>],l,p,g,d — określa rozmiary okienka tekstowego nadając mu numer strumienia — # stream. Parametry l — numer pierwszej — lewej kolumny, p — ostatniej — prawej kolumny, g — numer górnego wiersza i d — dolnego wiersza definiują punkty graniczne okienka zgodnie z zadanym trybem (MODE).

TEST x,y — bada numer atramentu punktu o współrzędnych (x,y).

XPOS YPOS — podają współrzędne odpowiednio x i y kursora graficznego.

Powyższe komendy związane były bezpośrednio z tworzeniem rysunku. W programie używam także innych instrukcji:

CALL adres — w naszym przypadku umożliwia wywołanie podprogramu znajdującego się w pamięci ROM.

INKEY (nr) — funkcja przyjmuje wartość 0, jeśli zostanie wciśnięty na klawiaturze klawisz o numerze nr.

INKEY $ — przyjmuje wartość pierwszego znaku wprowadzonego z klawiatury lub tzw. łańcucha pustego — "", gdy żaden klawisz nie został wciśnięty.

GOSUB nr linii — skok do podprogramu określonego numerem linii. Koniec podprogramu jest oznaczony instrukcją RETURN. Po zakończeniu jego realizacji następuje powrót do programu głównego, do następnego rozkazu GOSUB.

GOTO nr linii — skok do określonej numerem linii programu.

FOR wsk = a TO b
⋮
NEXT b — realizuje fragment programu między FOR a NEXT — b-a razy.

IF wyr. log. THEN w1 ELSE w2 — sprawdza czy wyrażenie logiczne jest prawdziwe. Jeżeli tak, to realizowany jest ciąg instrukcji w1 zapisany w tej samej linii programu. Jeżeli jest fałszywe, realizowana jest druga część wyrażenia.

Omówię dalej funkcje realizowane przez poszczególne fragmenty programu KRESKA. Linie od — do zawierają:

50 — 100 zdefiniowanie okienek, określenie kolorów tła i pióra. Wyznaczają punkt startowy kreślenia.

140 — 170 wybór sposobu kreślenia z jednoczesnym początkowym nadaniem wartości parametrom G%, D%,L%, P%,F%. Przyporządkowanie tym parametrom numerów klawiszy kursora i klawisza COPY (odpowiednio: 0,2,8, 1,9), ewentualnie styków joysticka i przycisku FIRE (72,73,74,75,76).

360 linia ta zawiera rozkaz CALL B1A9. Wykonanie tego polecenia powoduje wyzerowanie oraz wystartowanie procedur inicjujących pracę (wynik analogiczny, jak przy równoczesnym naciśnięciu SHIFT, CONTROL i ESC).

420 — 440 wypełnienie fragmentu ekranu przeznaczonego do tworzenia rysunków kolorem tła. Jednocześnie zostaje narysowany punkt o współrzędnych (320, 200). W ten sposób zostaje określone nowe położenie kursora graficznego.

480 — 590 w liniach tych tworzone jest menu związane z wyborem opcji K — kreślenie. Po dokonaniu wyboru sterowanie przenoszone jest do odpowiedniej linii związanej z wybraną opcją. Akceptowane znaki to: M,P,K,G,W,↑,↓.

630 — 700 wybór koloru pióra. Możliwy jest wybór jednego z trzech kolorów (kolor tła jest niedostępny). W wypadku naciśnięcia klawisza kursora następuje zmiana numeru atramentu (nr INK) związanego z piórem (PEN). Zmiana powyższa jest sygnalizowana w górnym prawym rogu ekranu (WINDOW#3).

750 — 820 w liniach tych zawarte zostało sterowanie przesunięciem kursora graficznego (bez kreślenia). Samo przemieszczenie odbywa się przez odwołanie do odpowiednich procedur.

870 — 920 ustalenie numeru atramentu (INK) punktu o współrzędnych (x%,y%) i podstawienie jego wartości pod zmienną p.

930 — 970 wypełnienie punktu (x%, y%) kolorem atramentu (INK) o numerze p.

980 — 1020 sprawdzenie czy kursor graficzny mieści się w obszarze przeznaczonym do kreślenia. Punkty graniczne tego obszaru to: dół (0,0) i (639,0), góra (0,366) i (639,366). W wypadku, gdy nowe współrzędne wynikające z przesunięcia wychodzą poza obszar dozwolony, punkt jest cofany na granicę okna. Próby wyjścia poza ekran są sygnalizowane dźwiękiem (CHR$ 7).

1070 — 1120 nadanie nowych wartości współrzędnym (x%,y%) w zależności od naciśnięcia

klawisza kursora lub wychylenia joysticka.

1160—1300 kreślenie „punkt po punkcie" - z jednoczesnym sprawdzeniem czy kursor nie opuszcza dozwolonego obszaru.

1340—1480 wycieranie fragmentów rysunku. Mechanizm wycierania polega na wypełnianiu obszaru ograniczonego rozmiarami gumki kolorem tła (INK 0).

1530—1550 wyświetlanie aktualnych współrzędnych kursora graficznego. Należy podkreślić, że w języku BASIC wszystkie wyprowadzane liczby uzupełniane są spacją kończącą. Stąd np. liczba trzycyfrowa zajmuje cztery pozycje na ekranie. W naszym przypadku wymusza to deklarację okna tekstowego o szerokości czterech znaków (3+1).

1600—1610 zapamiętanie numeru atramentu (INK) i bieżących współrzędnych kursora graficznego.

W programie używam następujących zmiennych:

x%,y% — współrzędne bieżące kursora graficznego;
xp%,yp% — współrzędne pamiętanego punktu;
x,y — współrzędne służące do wydruku;
G% — numer klawisza lub styku joysticka wywołującego ruch kursora graficznego w górę
D% — j.w w dół;
L% — j.w. w lewo;
P% — j.w. w prawo;
F% — numer klawisza COPY lub przycisku FIRE;
sk% — informuje, czy punkt przykryty kursorem graficznym został zamalowany;
pr% — informuje, czy nastąpiła zmiana współrzędnych kursora graficznego;
p — pamiętany numer atramentu (INK);
rb% — oznacza stany gumki (Ø--wyłączona 1-włączona).

```
10 REM**************************************
20 REM    PROGRAM GRAFICZNY - KRESKA
30 REM autor: Jan Romanowski          1986
40 REM**************************************
50 MODE 1:BORDER 13:INK 0,26:INK 1,0:INK
 2,6:INK 3,14:kolor=1:x%=320:y%=200
60 WINDOW #0,1,5,1,2:PAPER #0,2:PEN #0,1
:CLS #0:PRINT #0,"X=320";
70 LOCATE #0,1,2:PRINT #0,"Y=200";
80 WINDOW #2,6,6,1,2:PAPER #2,2:PEN #2,1
:PRINT #2,CHR$(211)+CHR$(211);
90 WINDOW #2,7,40,1,2:PAPER #2,2:CLS #2:
PEN #2,1
100 WINDOW #1,1,40,3,25:PAPER #1,0:CLS #
1:PLOT x%,y%,0:x=XPOS:y=YPOS:GOSUB 1530
110 REM**************************************
120 REM          MENU GLOWNE
130 REM**************************************
140 PRINT #2,"   (K)LAWIATURA CZY (J)OYS
TICK."
150 IF INKEY(37)=0 THEN G%=0:D%=2:L%=8:P
%=1:F%=9:GOTO 180
160 IF INKEY(45)=0 THEN G%=72:D%=73:L%=7
4:P%=75:F%=76:GOTO 180
170 GOTO 150
180 CLS #2:PRINT #2," (Z)AKONCZENIE PRAC
Y (K)RESLENIE";
190 LOCATE #2,2,2:PRINT #2,"(C)LS ";
200 REM**************************************
210 REM            KRESLENIE
220 REM**************************************
230 IF INKEY(37)=0 THEN CLS #2:GOTO 480
240 REM**************************************
250 REM    CLS - CZYSZCZENIE EKRANU
260 REM**************************************
270 IF INKEY(62)=0 THEN CLS#2:PRINT #2,"
CLS - CZY JESTES PEWIEN  T/N":GOTO 420
280 REM**************************************
290 REM       ZAKONCZENIE PRACY
300 REM**************************************
310 IF INKEY(71)=0 THEN CLS #2:PRINT #2,
" CZY JESTES PEWIEN ?    T/N":GOTO 360
320 GOTO 230
330 REM**************************************
340 REM     ZAKONCZENIE PRACY  T/N
350 REM**************************************
360 IF INKEY(51)=0 THEN MODE 0:PRINT "K
O N I E C":FOR n=1 TO 2000:NEXT n:CALL B
1A9
370 IF INKEY(46)=0 THEN GOTO 180
380 GOTO 360
390 REM**************************************
400 REM       CZYSZCZENIE EKRANU
410 REM**************************************
420 IF INKEY(51)=0 THEN CLS #1:x%=320:y%
=200:PLOT x%,y%,0:x=XPOS:y=YPOS:GOSUB 15
30:GOTO 180
430 IF INKEY(46)=0 THEN GOTO 180
440 GOTO 420
450 REM**************************************
460 REM     KRESLENIE - MENU GLOWNE
470 REM**************************************
480 WINDOW #2,7,38,1,2:PRINT #2,"(M)ENU
GLOWNE (P)RZEMIESZCZENIE (K)OLOR
        LUB ("+CHR$(241)+")" ;
490 WINDOW #3,39,40,1,2:PAPER #3,kolor:C
LS #3:PEN #3,1:PRINT #3,CHR$(211)+CHR$(3
2)+CHR$(211);
500 IF INKEY(38)=0 THEN WINDOW #2,7,40,1
,2:GOTO 180
510 IF INKEY(2)=0 THEN GOTO 550
520 IF INKEY(37)=0 THEN GOSUB 630
530 IF INKEY(27)=0 THEN GOSUB 1600:GOTO
750
540 GOTO 500
550 CLS #2:PRINT #2,"(G)UMKA (W)OLNE KRE
SLENIE LUB("+CHR$(240)+")"
560 IF INKEY(0)=0 THEN GOTO 480
570 IF INKEY(59)=0 THEN GOTO 1160
580 IF INKEY(52)=0 THEN GOSUB 1340
590 GOTO 560
600 REM**************************************
610 REM          WYBOR KOLORU
620 REM**************************************
630 CLS #2:PRINT #2,"WYBIERAJAC KOLOR UZ
YWAJ - "+CHR$(241)+"     POWROT DO MENU
       - P"
640 1$=INKEY$:IF 1$="" THEN 640
650 IF INKEY(2)=0 THEN kolor=kolor+1:SOU
ND 1,100,10,7,3,5,1
660 IF INKEY(27)=0 THEN GOTO 480
670 IF kolor<4 THEN PAPER #3,kolor:CLS #
3:PEN #3,1:PRINT #3,CHR$(211)+CHR$(32)+C
HR$(211)+CHR$(32);
680 IF kolor<4 THEN FOR i=1 TO 100:NEXT
i:GOTO 640
690 IF kolor>3 THEN kolor=1:PAPER #3,kol
or:CLS #3:FOR i=1 TO 100:NEXT i:GOTO 640
700 GOTO 650
710 REM**************************************
720 REM     PRZEMIESZCZANIE KURSORA
730              GRAFICZNEGO
740 REM**************************************
750 CLS #2:PRINT #2," UZYWAJ :"+CHR$(240
)+CHR$(242)+CHR$(243)+CHR$(241)+" LUB JO
YSTICK"
760 GOSUB 1600
770 LOCATE #2,1,2:PRINT #2," POWROT DO M
ENU - P"
780 IF INKEY(27)=0 THEN 480
790 GOSUB 930:GOSUB 1070:GOSUB 870:GOSUB
 930:GOSUB 980:GOSUB 870
800 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp%
,kol:MOVE x%,y%:sk%=1
810 GOTO 780
820 RETURN
830 REM**************************************
840 REM          ANALIZA KURSORA
850 REM            GRAFICZNEGO
860 REM**************************************
870 IF TEST(x%,y%)=1 THEN p=1
880 IF TEST(x%,y%)=2 THEN p=2
890 IF TEST(x%,y%)=3 THEN p=3
900 IF TEST(x%,y%)=0 THEN p=0
910 PLOT x%,y%,1
920 RETURN
930 IF p=1 THEN PLOT x%,y%,1
940 IF p=2 THEN PLOT x%,y%,2
950 IF p=3 THEN PLOT x%,y%,3
960 IF p<>1 AND p<>2 AND p<>3 THEN PLOT
x%,y%,0
970 RETURN
980 IF y%<1 THEN y%=1:PRINT #3,CHR$(7)
```

dokończenie na str. 11

# Zmienne systemowe ROM Spectrum

## EKRAN ZNAKOWY

| | |
|---|---|
| S-POSN | 23688/9 |
| SPOSNL | 23690/1 |
| DF-CC | 23684/5 |
| DF-CCL | 23686/7 |
| SCR-CT | 23692 |
| DF-SZ | 23659 |
| CHARS | 23606/7 |
| K-CUR | 23643 |
| MODE | 23617 |

Układ bajtów w obszarze ekranu nie jest sekwencyjny, co najlepiej sprawdzić obserwując pojawianie się obrazka z dowolnej gry. Adresy dostosowane są do wydruku znaków. Każdy standardowy znak ASCII na ekranie składa się z 64 punktów w konfiguracji 8 × 8. Definiuje to ekran znakowy na 24 linie i 32 kolumny. 22 linie zawiera ekran główny, a dwie część edycyjna. Te ustalenia można oczywiście zmienić. Rozszerzenie edytora nastąpi po: POKE 23659, a gdy a > 2, co jednocześnie zmniejszy ekran główny do 24-a linii. Ustalając DF-SZ na zero pozbywamy się edytora, choć jest to pomysł trochę niebezpieczny. Nie można wtedy stosować komend INPUT i SAVE w progamie, wymuszać pytanie scroll? i pieczołowicie omijać BREAK (z drugiej strony to niezła blokada). Do dyspozycji są 24 linie ekranu głównego, ale o ile TAB w instrukcjach z PRINT działa normalnie, to AT nie wolno używać dla linii ostatniej (23). Poniżej przykład wydruku w liniach edycji:

```
10 BORDER 2: PAPER 1:CLS
20 POKE 23659,0: REM brak linii edycji
30 PRINT PAPER 2;AT 22,2;"WYNIK";TAB 10;"PALIWO";
   TAB 20;"CZAS";TAB 4;0;TAB 12;100;TAB 20;"0:00"
40 POKE 23659,2: REM dwie linie edycji
50 GOTO 50
```

Zmienna SCR-CT jest wskaźnikiem przewijania ekranu. Zawiera liczbę o 1 większą, niż liczba linii, o jaką przesunie się obraz ku górze.

```
10 POKE 23692,10
20 PRINT AT 10,1;"TEKST"
30 PRINT AT 21,31;''
40 IF PEEK 23692=1 THEN STOP
50 PAUSE 4
60 GOTO 30
```

Proszę zauważyć, że przesuwanie zostaje wymuszone w linii 30 przez symulację wydruku poza ekranem, co normalnie spowodowałoby reakcję scroll? Efekt podobny można osiągnąć przez bezpośrednie wywołanie procedur ROM w adresach 3190 lub 3582:

```
10 CLS:PRINT AT 21,0;
20 IF LEN INKEY$ THEN GOTO 20
30 IF NOT LEN INKEY$ THEN GOTO 30
40 IF CODE INKEY$=12 THEN PRINT CHR$ 8;" ";CHR$ 8;:GOTO 60
50 PRINT INKEY$;
60 BEEP .02,-10
70 IF PEEK 23689=3 AND PEEK 23688=1 THEN RANDOMIZE USR 3190:
   POKE 23688,33
80 GOTO 20
```

Przy okazji wykorzystaliśmy zmienną S-POSN. Określa ona pozycję wydruku kolejnego znaku. Oznacza to, że jeśli wydruk był w polu np. (x,y), to następny znak pojawi się w polu (24-PEEK 23689, 33-PEEK 23688). Podobnie odczytujemy pozycję w liniach edycyjnych wykorzystując SPOSNL. Poniższy program „nie lubi" przenosić części wyrazu do kolejnej linii.

```
10 LET a$="Najlepszym sposobem wydruku na ekranie jest
          pojawianie się kolejnych liter od lewej do prawej
          strony z szybkością czytania wzrokiem."
20 LET k=1
30 PRINT a$(k);
40 BEEP .05,20
50 IF NOT (k-LEN a$) THEN STOP
60 LET k=k+1
70 IF PEEK 23688=1 AND a$(k-1)<>" " THEN GOSUB 200
80 GOTO 30
200 IF a$(k)=" " THEN GOTO 250
210 PRINT CHR$ 8;: REM cofacz
220 LET k=k-1
230 IF a$(k)<>" "THEN GOTO 210
240 PRINT TAB 0;
250 LET k=k+1: RETURN
```

Wymuszaliśmy już przesuw ekranu, więc spróbujmy go teraz uniknąć.

```
10 PRINT "mikrokomputer"
20 IF PEEK 23689=2 THEN POKE 23689,3: REM ekran "stoi"
30 GOTO 10
```

Innym sposobem wydruku jest bezpośrednie korzystanie z adresów obszaru ekranu. Zmienne DF-CC i DF-CCL odpowiednio dla ekranu głównego i edycyjnego wskazują adres kolejnego wydruku. Wykorzystujemy to do zrobienia konkurencji grafikom użytkownika:

```
10 CLS: PRINT "H";
20 GOSUB 60
30 PRINT "O";
40 STOP
50 DATA 0,0,0,28,4,24,32,60
60 LET adres=PEEK 23684+256*PEEK 23685
70 LET licz=0
80 READ k
90 POKE adres,k
100 LET adres=adres+256
110 LET licz=licz+1
120 IF licz<8 THEN GOTO 80
130 POKE 23684,PEEK 23684+1
140 RETURN
```

Znaki, które ukazują się na ekranie zdefiniowane są w ROM-ie. W zmiennej CHARS znajduje się ich adres początkowy 15616 pomniejszony o 256. Własny zbiór grafik dla znaków o kodach 32..127 można wprowadzić do RAM najlepiej pod adres modulo 256 pamiętając, że każdą grafikę określa 8 bajtów. Przejście odbywa się przez modyfikację CHARS:

```
POKE 23606,0
POKE 23607,adres/256-1
```

Oto dwa przykłady wykorzystania CHARS:

```
 5 LET chars=PEEK 23606+256*PEEK 23607
10 INPUT "znak? ";LINE a$
20 IF a$="" OR LEN a$>1 OR CODE a$<32 THEN GOTO 10
30 LET n=CODE a$
40 FOR a=0 TO 7
50 LET x=PEEK (chars+8*n+a)
60 LET t$=""
70 FOR b=1 TO 8
80 LET c=INT(x/2):LET d=x-2*c
90 LET t$=("*" AND d)+(":" AND NOT d)+p$
100 LET x=c: NEXT b
110 PRINT p$: NEXT a
```

```
10 DIM c(8,8)
20 LET chars=PEEK 23606+256*PEEK 23607
30 INPUT LINE a$
40 LET s=LEN a$: IF s>8 THEN LET s=8
50 FOR r=1 TO s
```

```
 60 LET i=chars+8*CODE a$(r)
 70 FOR t=0 TO 7: LET m=PEEK (i+t)
 80 FOR p=0 TO 7
 90 LET c(t+1,8-p)=m-2*INT(m/2)
100 LET m=INT(m/2)
110 NEXT p:NEXT t
120 FOR u=1 TO 4: LET t$=""
130 FOR z=1 TO 4
140 LET k=c(2*u-1,2*z)+2*c(2*u-1,2*z-1)+4*c(2*u,2*z)+
    8*c(2*u,2*z-1)
150 LET t$=t$+ CHR$(128+k)
160 NEXT z: PRINT AT u,4*r-4;t$
170 NEXT u: NEXT r
```

Bajt 23606 można wykorzystać do innych efektów:

```
 10 DATA "Zmieniając","wartość","bajtu","23606",
"osiągamy","bardziej","spektakularny","sposób",
"wydruku","na","ekranie","$"
 20 RESTORE: PRINT AT 0,0;
 30 READ a$
 40 IF a$="$" THEN GOTO 20
 50 FOR a=7 TO 0 STEP -1
 60 POKE 23606,a
 70 PRINT a$
 80 POKE 23689,PEEK 23689+1
 90 NEXT a: PRINT
100 GOTO 30
```

Szczególnym znakiem drukowanym na ekranie jest kursor edycyjny. Miejsce jego aktualnego pobytu w obszarze między E-LINE, a WORKSP przechowuje zmienna K-CUR, natomiast znak, jaki będzie drukowany zależy od bajtu MODE. Manipulując tą ostatnią zmienną można otrzymać różne rozwiązania graficzne kursora w trakcie egzekucji komendy INPUT, gdyż jest to jedyna sposobność, gdy bufor edycji przedstawiany jest na ekranie. Tu zajmę się tylko wymuszaniem trybów pracy podczas INPUT. Tryb graficzny osiągamy przez:

```
10 POKE 23617,2
20 INPUT LINE i$
30 IF CODE i$>164 THEN GOTO 10
```

Tryb EXTENDED będzie obowiązywał tylko dla pierwszego znaku, co może posłużyć do napisania podręcznego kalkulatora obliczającego wartości zaimplementowanych funkcji:

```
10 POKE 23617,1
20 INPUT LINE i$
30 PRINT VAL i$
```

Po wczytaniu np. BIN 10 otrzymamy 2, a po SIN (PI/2) zobaczymy 1.
Możliwe jest również wymuszenie trybu K:

```
10 PRINT "Która z komend służy do
   niszczenia obszaru zmiennych
   BASIC ?"
20 POKE 23611,128: POKE 23617,158
30 INPUT LINE i$
40 IF CODE i$<>253 THEN GOTO 20
50 PRINT i$(1) 'INVERSE 1;"DOBRZE"
```

**W następnym numerze o atrybutach ekranu.**

**Krzysztof MAMCARZ**

---

dokończenie ze str. 9

```
990 IF y%>366 THEN y%=366:PRINT #3,CHR$(
7)
1000 IF x%<1 THEN x%=1:PRINT #3,CHR$(7)
1010 IF x%>639 THEN x%=639:PRINT #3,CHR$
(7)
1020 RETURN
1030 REM*********************************
1040 REM       ZMIANA  WSPOLRZEDNYCH
1050 REM        KURSORA GRAFICZNEGO
1060 REM*********************************
1070 IF INKEY(G%)=0 THEN y%=y%+4:pr%=1
1080 IF INKEY(D%)=0 THEN y%=y%-4:pr%=1
1090 IF INKEY(L%)=0 THEN x%=x%-4:pr%=1
1100 IF INKEY(P%)=0 THEN x%=x%+4:pr%=1
1110 x=XPOS:y=YPOS:GOSUB 1530
1120 RETURN
1130 REM*********************************
1140 REM          WOLNE KRESLENIE
1150 REM*********************************
1160 CLS #2:PRINT #2,"UZYWAJ :"+CHR$(240
)+CHR$(242)+CHR$(243)+CHR$(241)+" LUB JO
YSTICK"
1170 LOCATE #2,1,2:PRINT #2,"POWROT DO M
ENU - P"
1180 IF INKEY(27)=0 THEN GOTO 480
1190 IF INKEY(G%)=0 THEN PLOTR 0,1,kolor
1200 IF INKEY(D%)=0 THEN PLOTR 0,-1,kolo
r
1210 IF INKEY(L%)=0 THEN PLOTR -1,0,kolo
r
1220 IF INKEY(P%)=0 THEN PLOTR 1,0,kolor
1230 x=XPOS:y=YPOS:GOSUB 1530
1240 x%=x:y%=y
1250 IF x<1 THEN MOVER 1,0:PRINT #3,CHR$
(7)
1260 IF x>639 THEN MOVER -1,0:PRINT #3,C
HR$(7)
1270 IF y>366 THEN MOVER 0,-1:PRINT #3,C
HR$(7)
1280 IF y<1 THEN MOVER 0,1:PRINT #3,CHR$
(7)
1290 GOTO 1180
1300 RETURN
1310 REM*********************************
1320 REM                GUMKA
1330 REM*********************************
1340 CLS #2:PRINT #2,"GUMKA - WYLACZANA
  P - POWROT"
1350 LOCATE #2,1,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+"
 COPY LUB JOYSTICK"
1360 rb%=0
1370 GOSUB 930:GOSUB 1070:GOSUB 980:GOSU
B 870:GOSUB 930
1380 IF INKEY(27)=0 THEN GOTO 480
1390 IF INKEY(F%)=0 AND rb%=0 THEN rb%=1
:FOR i=1 TO 200:NEXT i:LOCATE #2,9,1:PRI
NT #2,"WLACZONA ":GOTO 1400 ELSE IF INKE
Y(F%)=0 AND rb%=1 THEN rb%=0:LOCATE #2,9
,1:PRINT #2,"WYLACZONA":FOR i=1 TO 200:N
EXT i:GOTO 1400
1400 IF rb%=1 THEN GOTO 1410 ELSE GOSUB
870:GOTO 1350
1410 FOR v%=-2 TO 2
1420 MOVE x%-3,y%+v%:DRAW x%+3,y%+v%,1
1430 NEXT v%
1440 FOR v%=-3 TO 3
1450 MOVE x%-4,y%+v%:DRAW x%+4,y%+v%,0
1460 NEXT v%
1470 GOTO 1370
1480 RETURN
1490 REM*********************************
1500 REM   WYSWIETLANIE WSPOLRZEDNYCH
1510 REM      KURSORA GRAFICZNEGO
1520 REM*********************************
1530 WINDOW #5,3,6,1,2:PAPER #5,2:PEN #5
,1:PRINT #5,USING"###";x
1540 PRINT #5,USING"###";y
1550 RETURN
1560 REM*********************************
1570 REM     CHARAKTERYSTYKA PUNKTU
1580 REM          BIEZACEGO
1590 REM*********************************
1600 xp%=XPOS:yp%=YPOS:kol=TEST(xp%,yp%)
:sk%=0:pr%=0
1610 RETURN
```

W następnym numerze będziemy rozbudowywać prezentowany program o szereg nowych funkcji, w tym m.in.: kreślenie linii, okręgu, czworokąta, składowanie pamięci ekranu na kasecie magnetofonowej, wczytywanie rysunku z kasety do pamięci obrazu.

**Janek**

GRAFIKA

# Możliwości Commodore 64

**Commodore 64 jest jednym z mikrokomputerów, które oferują znaczne możliwości graficzne. Niestety dostarczana użytkownikowi wraz z mikrokomputerem instrukcja obsługi nie omawia wszystkich szczegółów związanych ze sposobem wykorzystania C-64 do tworzenia grafiki komputerowej. Zacznijmy zatem od początku.**

## Jak wygląda pamięć Commodore 64?

Mikrokomputer posiada dwa rodzaje pamięci półprzewodnikowej: pamięć stałą (ROM) i pamięć o dostępie swobodnym (RAM).

Po włączeniu mikrokomputera C-64 wyświetlany jest komunikat:

64 RAM SYSTEM  38911 BASIC BYTES FREE

Informuje on nas o tym, że system dysponuje pamięcią 64Kbajtów, lecz dla progamu w Basicu dostępne jest tylko 38911 bajtów.

| ADRES | | OPIS |
|---|---|---|
| od | do | |
| 0 | 1 | rejestry mikroprocesora |
| 2 | 1023 | pamięć wykorzystywana przez system operacyjny |
| 1024 | 2039 | pamięć (matryca) obrazu wyświetlanego za pomocą znaków — PAMIĘĆ EKRANU |
| 2040 | 2047 | wskaźniki na bloki pamięci, w których zdefiniowano postacie sprite |
| 2048 | 40959 | pamięć wykorzystywana przez program w Basicu, dane lub w inny sposób; jest to nasza dyspozycyjna pamięć |
| 40960 | 49151 | interpreter języka Basic |
| 49152 | 53247 | dodatkowy obszar pamięci RAM do wykorzystania przez użytkownika |
| 53248 | 53249 | rejestry układu kontroli obrazu (VIC-a) |
| 54272 | 55295 | rejestry syntetyzera dźwięków (SID-a) |
| 55296 | 56319 | pamięć (matryca) kolorów wyświetlanych znaków — PAMIĘĆ KOLORU |
| 56320 | 57343 | rejestry wejścia/wyjścia |
| 57344 | 65535 | procedury systemu operacyjnego (KERNAL) |

Rys. 1. Jedna z możliwych (podstawowa) konfiguracji pamięci C-64.

Uwaga: Generator znaków ROM (zbiór standardowych postaci znaków C-64) można uzyskać w obszarze od 53248 do 57343 bajtu po ustawieniu odpowiednich przełączników.

Na rys. 1. przedstawiono jedną z możliwych, najczęściej wykorzystywaną konfigurację pamięci.

## Jak powstaje obraz?

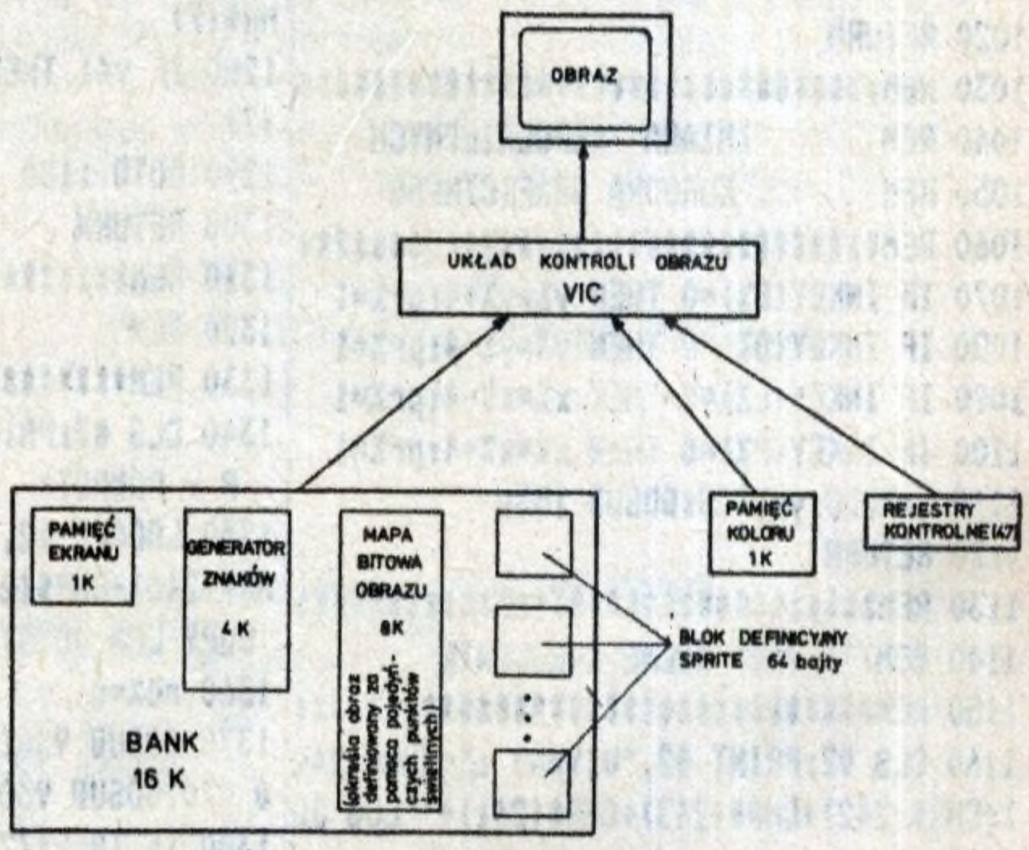
Na ekranie monitora widzimy różne obiekty (np. litery, znaki graficzne, sprite) wytwarzane przez mikrokomputer. Każdy obiekt powstaje po zinterpretowaniu określonych obszarów pamięci definiujących jego postać, kolor, położenie na ekranie. Można powiedzieć, że mikrokomputer wyświetla swoją pamięć.

Całą grafiką mikrokomputera C-64 steruje układ scalony zwany układem kontroli obrazu. W dalszej części będziemy posługiwać się jego skrótową nazwą VIC (ang. Video Interface Controller). Przetwarza on określone obszary pamięci mikrokomputera w obiekty oglądane na ekranie. VIC posiada następujące podstawowe tryby pracy:

a) tryb znakowy — możliwość wyświetlania znaków w 40 kolumnach i 25 wierszach;
b) tryb graficzny — możliwość wyświetlania obrazów graficznych o rozdzielczości 320 na 200 punktów świetlnych;
c) sprite — możliwość wyświetlania do 8 ruchomych obrazków graficznych (24 na 21 punktów świetlnych każdy).

Wymienione tryby pracy mogą być realizowane w grafice wielobarwnej lub standardowej. Można je również łączyć konstruując bardzo barwne obrazy graficzne.

Ważną własnością VIC jest niezatrudnianie do swojej pracy mikroprocesora. VIC jest sam specjalizowanym procesorem mającym możliwość niezależnej adresacji pamięci. Na formę obrazu oglądanego na ekranie, a wytwarzanego przez VIC wpływa zawartość wielu obszarów pamięci. W schematyczny sposób pokazano to na rys. 2.




Rys. 2. Podstawowe elementy biorące udział przy tworzeniu obrazu graficznego przez C-64.

Do sterowania pracą i komunikowania się z VIC służy 47 rejestrów rozmieszczonych między 53248 i 53294 bajtem pamięci. Ustalając odpowiednie zawartości tych rejestrów możemy wpływać na grafikę uzyskiwaną na ekranie. Niektóre z rejestrów pełnią kilka funkcji. Realizację wybranej funkcji uzyskuje się zapalając odpowiednie bity (ustawiając na wartość 1), bądź też gasząc odpowiednie bity (ustawiając na wartość 0).

Wykonajmy następującą instrukcję w trybie bezpośrednim:

POKE 53265, PEEK(53265)AND 239

W rejestrze zlokalizowanym w 53265 bajcie zgasiliśmy bit 4. Spowodowało to zniknięcie obrazu na ekranie. Cały ekran jest w kolorze obrzeża, a znaki nie są wyświetlane. Przyciśnijmy dowolny klawisz. Na ekranie nie pojawiają się żadne znaki. Naciśnijmy klawisz RETURN, a następnie wprowadźmy kolejną instrukcję (ostrożnie, gdyż wprowadzane znaki są niewidoczne):

POKE 53265, PEEK(53265)OR 16

W tym samym rejestrze zapaliliśmy obecnie bit 4 i otrzymaliśmy normalny obraz na ekranie.

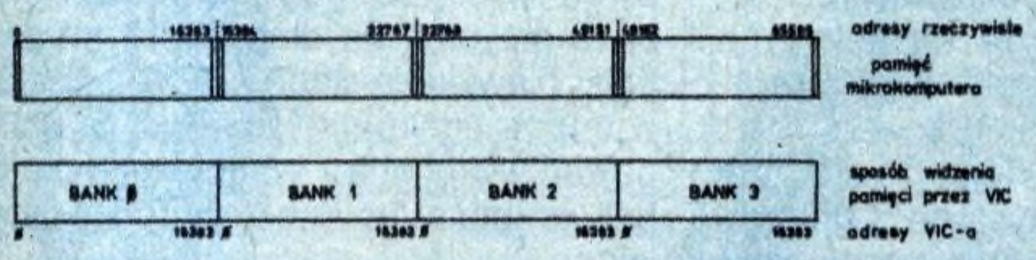
Do tworzenia obrazu wykorzystywany jest również obszar pamięci zwany PAMIĘCIĄ KOLORU zlokalizowany między 55296 i 56319 bajtem. PAMIĘĆ KOLORU określa kolor znaku wyświetlanego w każdej pozycji na ekranie.

Oprócz wymienionych obszarów pamięci (rejestry i PAMIĘĆ KOLORU) VIC wykorzystuje do tworzenia obrazu blok pamięci wielkości 16Kbajtów, zwany BANKIEM. W BANKU VIC posiada formy wszystkich obiektów wyświetlanych na ekranie monitora:

- PAMIĘĆ EKRANU zawierającą kody znaków wyświetlanych w 40 kolumnach i 25 wierszach;
- GENERATOR ZNAKÓW zawierający zbiór form wyświetlanych znaków; każdy kod znaku ma odpowiadającą mu dokładnie jedną formę;
- bloki definiujące postać każdego sprite'a wyświetlanego na ekranie.

Rozmieszczenie tych obiektów wewnątrz BANKU może ulegać zmianie. Sterujemy tym za pomocą rejestrów VIC.

C-64 dysponuje pamięcią 64K. Można wyodrębnić w niej cztery bloki o wielkości 16K (rys. 3). Każdy z wydzielonych bloków może zostać interpretowany przez VIC jako BANK. VIC widzi tylko 16K naraz. Każdy z BANKÓW jest adresowany od 0 do 16383 w procesie sterowania pracą VIC, choć w rzeczywistości mogą to być inne adresy.




Rys. 3. Koncepcja BANKU.

## Jak powstają znaki na ekranie?

Każdy znak wyświetlony na ekranie monitora można scharakteryzować następującymi składowymi: forma, kolor i umiejscowienie. Z punktu widzenia mikrokomputera dodatkowym i niezbędnym elementem charakteryzującym znak staje się jego kod. Przyspiesza bowiem realizację wielu operacji na znakach np. porównanie.

Spróbujmy w bardzo ogólnym zarysie prześledzić procedurę tworzenia znaku na ekranie monitora. Naciskając klawisz z dowolnym znakiem powodujemy generację odpowiedniego kodu dla tego znaku. Na podstawie zawartości 648 bajtu pamięci system operacyjny określa rzeczywiste położenie PAMIĘCI EKRANU i przesyła do odpowiedniej pozycji tej pamięci kod znaku. Kolor znaku (kod koloru) jest zapisywany do odpowiedniej pozycji PAMIĘCI KOLORU. Aktualny kolor znaków wprowadzanych jest zawarty w 646 bajcie. Mając te dane VIC pobiera z GENERATORA ZNAKÓW na podstawie kodu znaku jego formę, a z PAMIĘCI KOLORU identyfikuje jego kolor i wyświetla odpowiedni znak w odpowiednim miejscu i kolorze na ekranie monitora.

GENERATOR ZNAKÓW standardowych pochodzących z pamięci ROM jest dostępny dla VIC tylko w BANKU 0 i 2. Znajduje się on między 4096 i 8191 bajtem wg adresacji VIC. Zawartość generatora nie zmienia zawartości pamięci RAM w obu odpowiadających mu rzeczywistych obszarach. Z punktu widzenia VIC pamięć ROM przesłania pamięć RAM.

## Jak gospodarować pamięcią?

Nie licząc rejestrów VIC i PAMIĘCI KOLORU wszystkie pozostałe obszary pamięci definiujące formę obrazu są ruchome tzn. można je tworzyć w różnych rejonach pamięci RAM.

Stwarza to duże możliwości graficzne np.:

- możliwość definiowania własnych znaków graficznych;
- możliwość tworzenia animacji;
- możliwość optymalnego wykorzystania posiadanej pamięci.

Dla tych, którzy pragną korzystać z tych możliwości zamieszczamy tabelę, która w prosty sposób pozwala gospodarować wybranymi obszarami pamięci. Za pomocą tej tabeli możemy wybrać BANK, a wewnątrz BANKU określić położenie PAMIĘCI EKRANU i GENERATORA ZNAKÓW. Wykonujemy te czynności ustawiając odpowiednie zawartości trzech bajtów:

- 56576 — najmłodsze dwa bity określają BANK dołączany do VIC;
- 53272 — rejestr VIC-a określający położenie GENERATORA ZNAKÓW (3 najmłodsze bity) i PAMIĘCI EKRANU (4 najstarsze bity);
- 648 — wskaźnik położenia PAMIĘCI EKRANU używany przez system operacyjny; oblicza się go według wzoru:

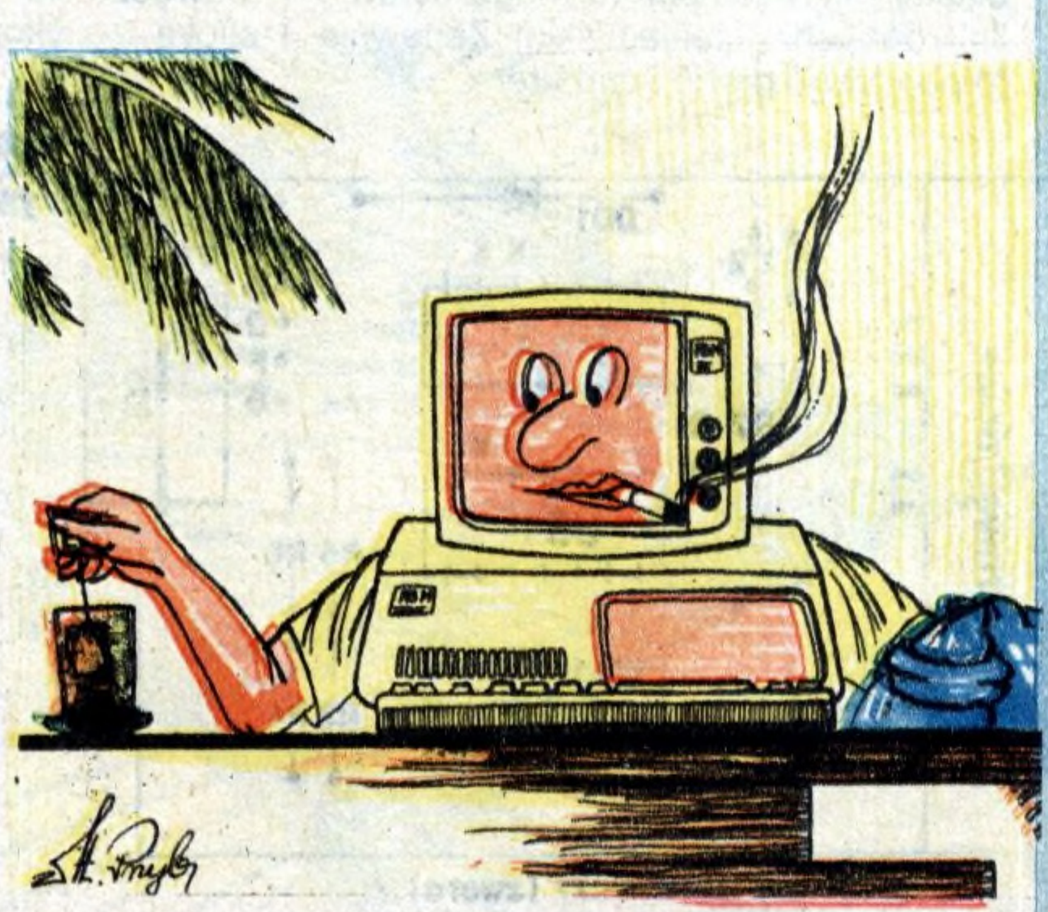
$$\text{wskaźnik} = \frac{\text{adres początkowy PAMIĘCI EKRANU}}{256}$$

Przykłady użycia tabeli:

a) Wykonując sekwencję instrukcji:

POKE 56576,149: POKE 648,140: POKE 53272,53

**dokończenie na str. 28**



**Rys. Michał Przybyłowski**

**Najwygodniejszym sposobem sterowania grami jest zastosowanie joysticka. Najpopularniejszy w naszym kraju mikrokomputer ZX Spectrum nie posiada gniazda, dającego taką możliwość. W tym celu musimy zastosować odpowiedni interfejs.**

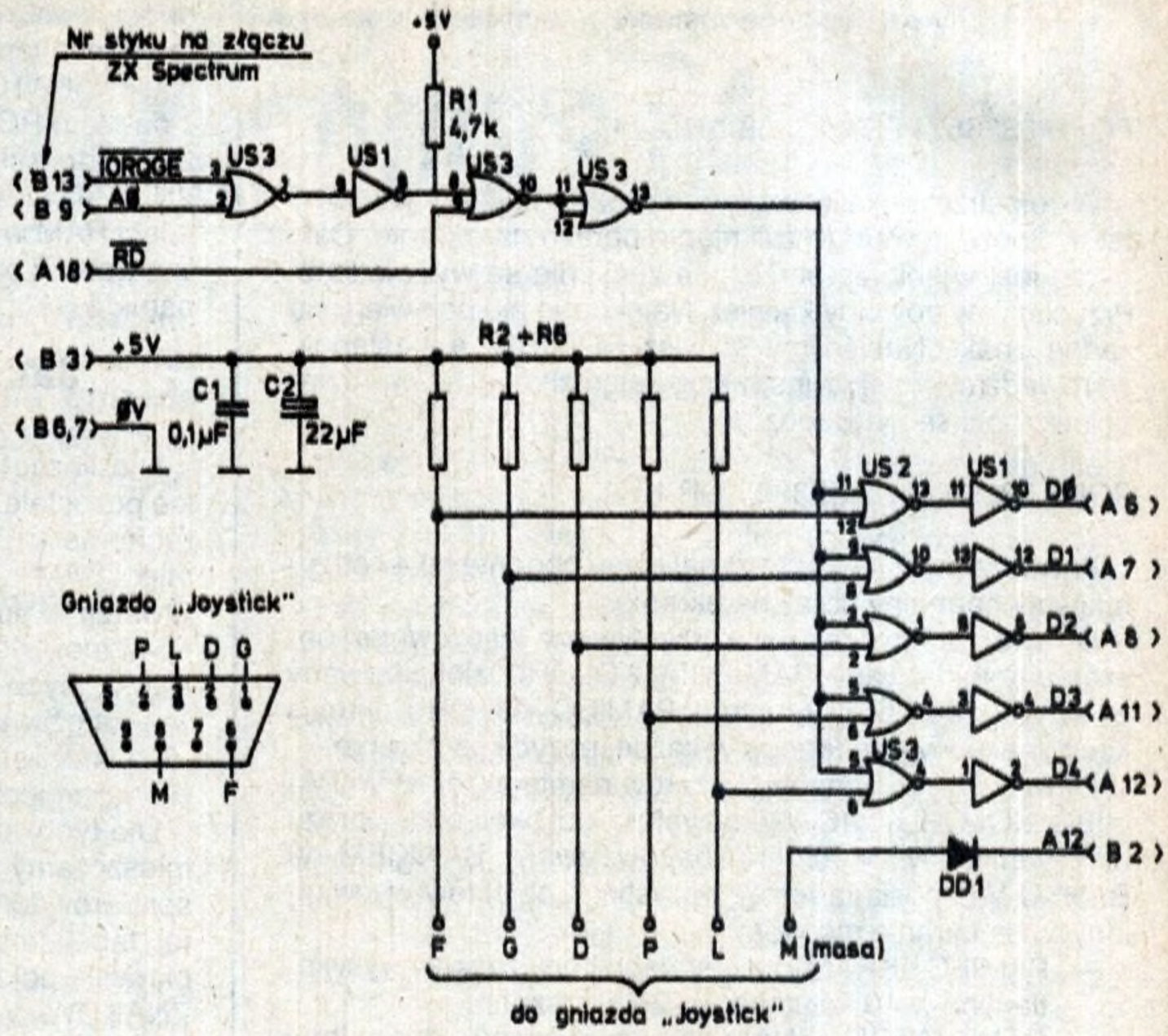
W artykule opisano konstrukcję interfejsu pracującego w standardzie SINCLAIR umożliwiającego dołączenie jednego joysticka. Wybrano ten standard ze względu na możliwość wykorzystania do budowy interfejsu elementów tylko krajowej produkcji.

Schemat ideowy urządzenia przedstawiono na rys. 1. Zastosowanie cyfrowych układów scalonych rodziny TTL-LS pozwala na zasilanie interfejsu i zasilacza +5V mikrokomputera.

Z punktu widzenia oprogramowania odczyt funkcji joysticka jest odczytem stanu klawiszy: 6, 7, 8, 9, 0.

Odczyt stanu manipulatora następuje w momencie wybierania układu ULA do odczytu (sygnały IORQGE, A0, RD mają niski stan) oraz po podaniu na linię adresową A 12 niskiego stanu. Niski poziom napięcia z tej linii adresowej może być podany, poprzez styki joysticka, na jedną z linii F, G, D, P, L. Stan zwarcia (niski stan) podawany jest poprzez bramki NOR i NOT na linie danych D0 + D43. Informacja o funkcji manipulatora pojawia się na odpowiednich bitach tej linii w postaci niskiego stanu. W położeniu neutralnym joysticka wszystkie bity linii danych mają (muszą mieć) wysoki stan.

Osiągnięto to przez zastosowanie rezystorów R2 ÷ R6 utrzymujących wyjście bramek NOR w stanie niskim. Po zanegowaniu przez bramki NOT układu US1, daje nam to wysoki stan na bitach D0 ÷ D4 linii danych mikrokomputera. Układ US1 ma wyjścia z otwartym kolektorem. Dzięki temu układy mikrokomputera mogą ustawić linie danych w stanie niskim. Zapewnia to poprawną pracę komputera.




Rys. 1 Schemat interfejsu

# INTERFEJS

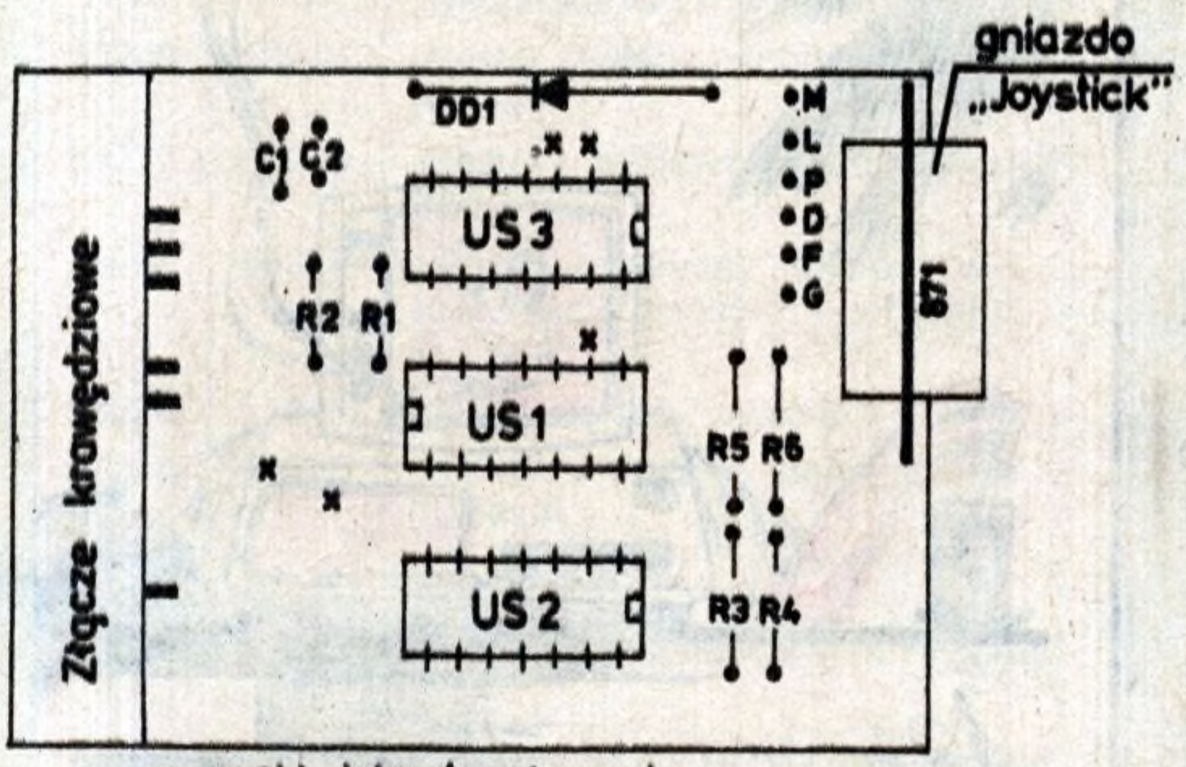
Szerszą informację o wykorzystywanych sygnałach możemy znaleźć w pracach [1] i [2].

Układ zmontowano na dwustronnej płytce drukowanej o wymiarach 62×54 mm. Na rysunku 2 pokazano rozmieszczenie elementów, a na kolejnych rysunkach (3 i 4) widok mozaiki połączeń po obu stronach płytki.

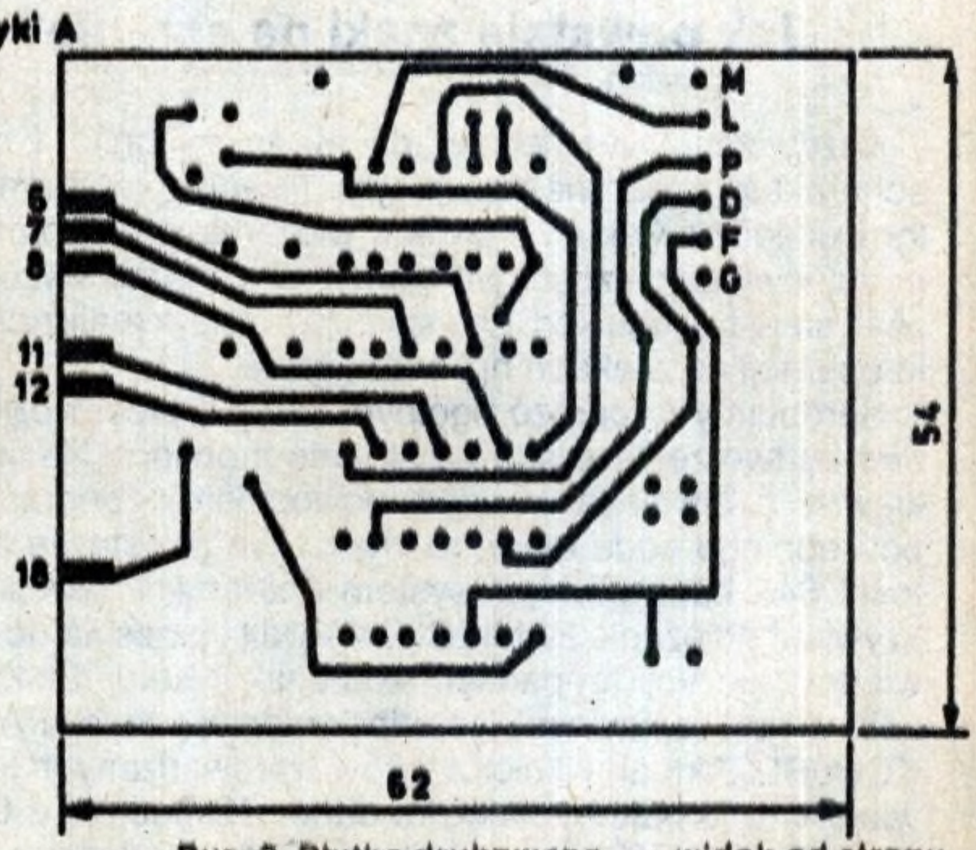
Do połączenia interfejsu ze złączem komputera, wykorzystano gniazdo krawędziowe, a do podłączenia joysticka 9-stykowe gniazdo szufladowe. Oba gniazda umocowano do płytki drukowanej. Gniazdo krawędziowe przylutowano (rys. 5), a gniazdo manipulatora przykręcono za pośrednictwem dwóch wsporników (kątowników).

Gniazdo krawędziowe wymaga przed montażem przygotowania. Należy je przyciąć na długość 17 styków. Styki niewykorzystane należy usunąć i w odpowiednim miejscu (rys. 6) umocować wykonaną z izolacyjnego materiału grubości 1,5 mm przegródkę ustalającą.

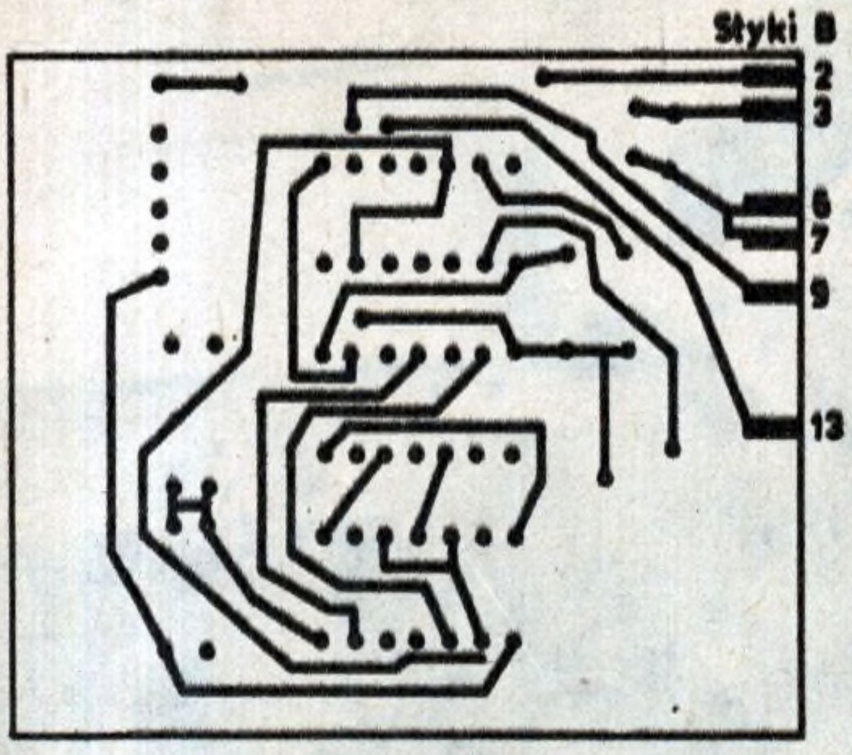
Po umocowaniu gniazd lutujemy kolejno elementy dyskretne i układy sca-



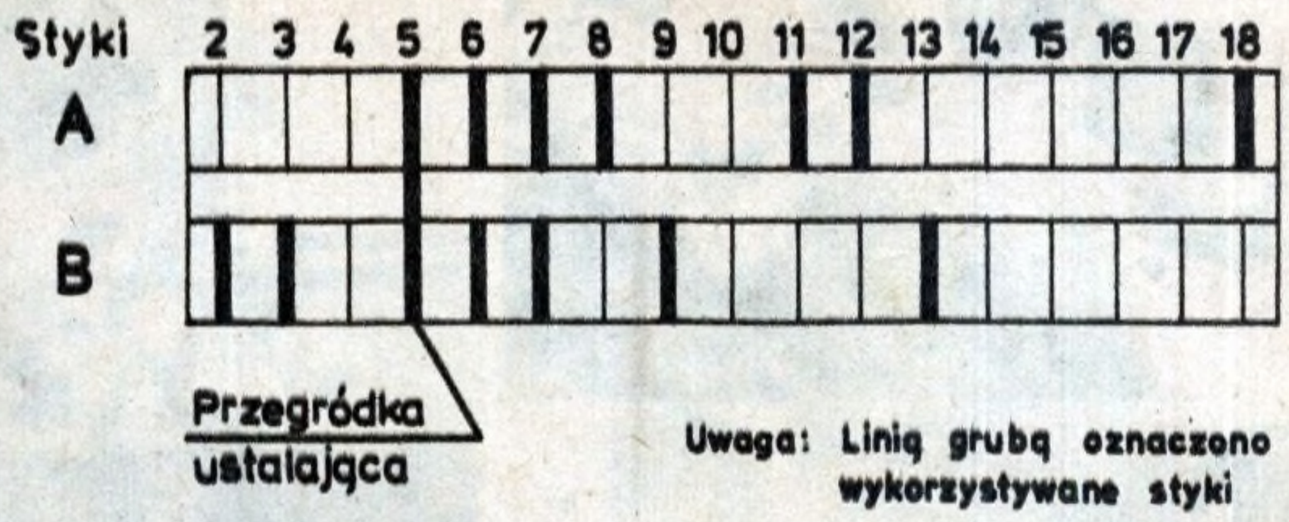

Rys. 2 Rozmieszczenie elementów



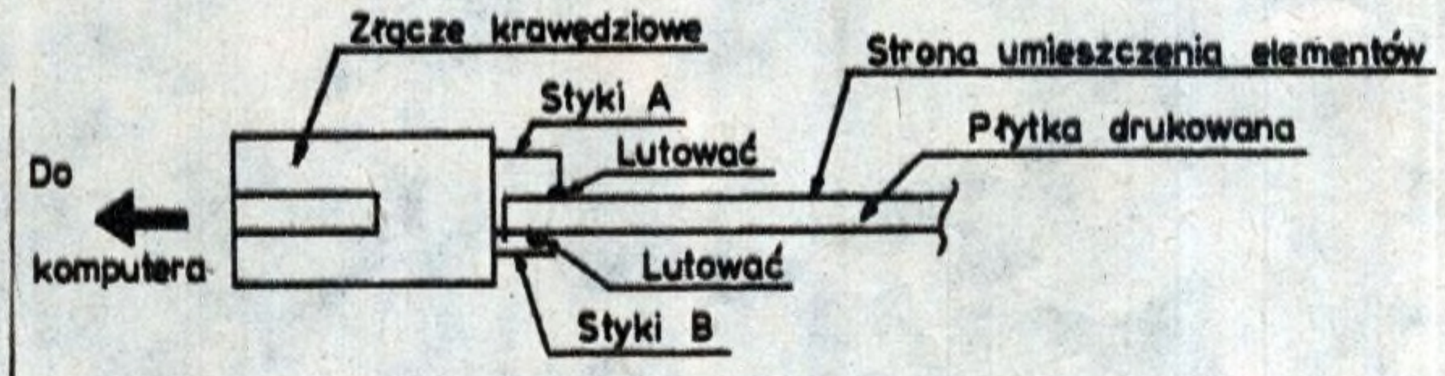

Rys. 3 Płytka drukowana — widok od strony części




**Rys. 4 Płytka drukowana widok od strony lutowania**




**Rys. 6. Wykorzystanie styków złącza krawędziowego**




**Rys. 5 Szkic montażu złącza krawędziowego do płytki drukowanej**

lone. Tak wykonany interfejs nie wymaga żadnych regulacji. Po wizualnej kontroli montażu możemy interfejs dołączyć do ZX Spectrum.

UWAGA!

Dołączenie interfejsu do komputera należy przeprowadzić przy wyłączonym zasilaniu komputera. Joystick możemy dołączyć do interfejsu zarówno przy wyłączonym, jak i włączonym zasilaniu komputera.

Po dołączeniu wg powyższych uwag interfejsu i joysticka włączamy zasilanie komputera. Uaktywnienie dowolnej funkcji manipulatora powinno powodować wyświetlanie na ekranie telewizora cyfr. Poszczególnym funkcjom joysticka powinny odpowiadać następujące cyfry:

| Funkcja joysticka | Cyfra |
|---|---|
| w lewo | 6 |
| w prawo | 7 |
| w dół | 8 |
| w górę | 9 |
| strzał | 0 |

Przy zastosowaniu sprawnych elementów oraz prawidłowym montażu układ powinien działać od razu po podłączeniu.

**(B. L.)**

**Wykaz elementów:**

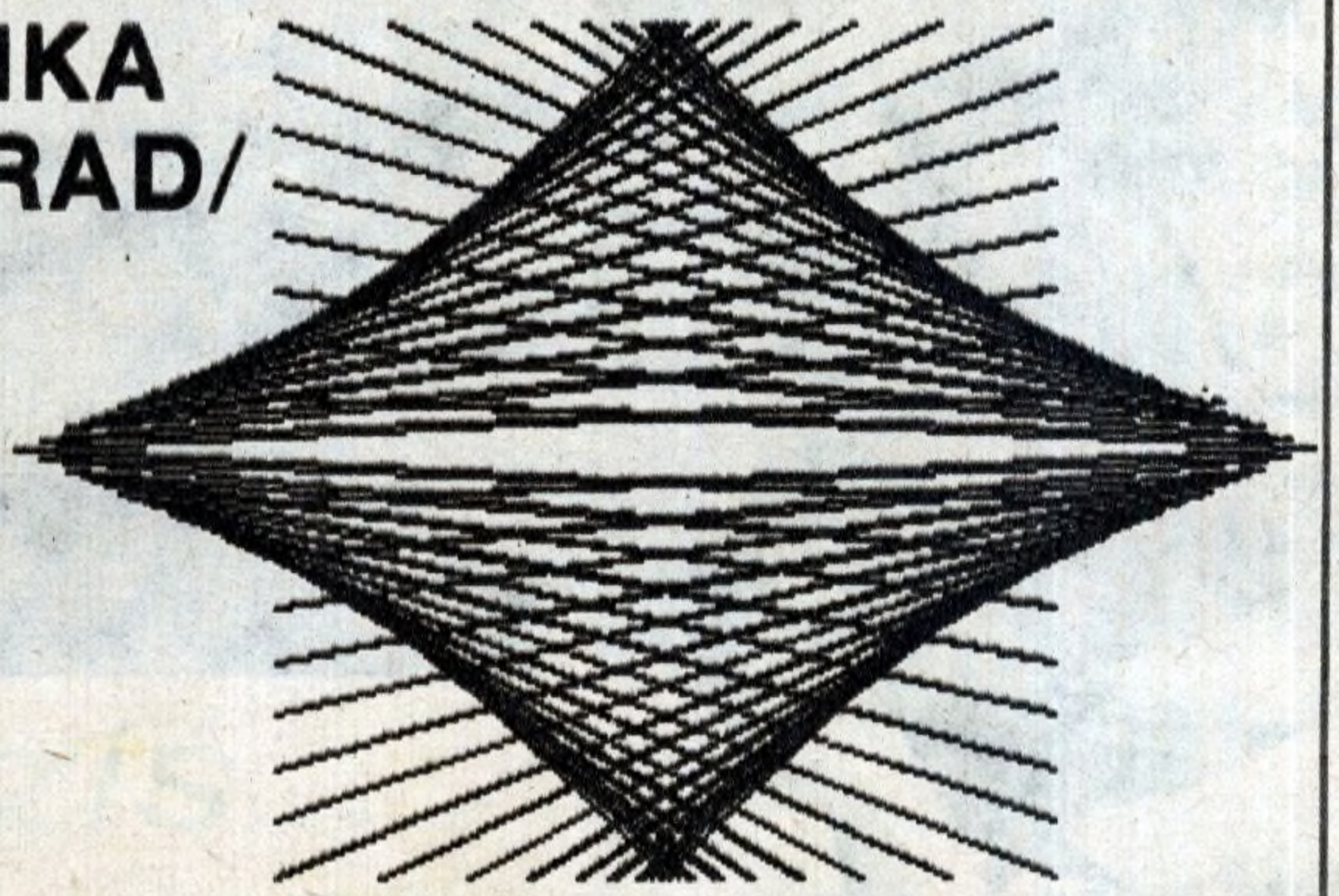
Układy scalone:
US1 — UCY 74LS05,
USZ,U53 — UCY 74LS02;
Dioda:
DD1 — BAVP 19 (lub dowolna impulsowa);
Rezystory (wszystkie O,125 W):
R1 — 4,7 kΩ,
R2 ÷ R6 — 6,8 k ÷ 10 kΩ;
Kondensatory:
C1 — D,1 μF/163 V typu KFPm (lub MKSE),
C2 — 22 μF/6,3 V elektrolityczny;
Inne:
Gniazdo krawędziowe typu Eltra 801 (2X24 styki),
Gniazdo joysticka — wtyk szufladowy typu Eltra 871 — 9 styków.

**Literatura:**

[1] Bednarski P.: Dodatkowa klawiatura do mikrokomputera ZX Spectrum, Radioelektronik 7185
[2] Dickens A.: Spectrum Hardware Manual, Melbourne House, 1983.

# GRAFIKA AMSTRAD/

```
2 CLS
3 ORIGIN 70,200
5  FOR x=1 TO 15
11 DRAW 400,25*x
20 DRAW 10*x,5*x
25 NEXT x
30 ORIGIN 570,200
35 FOR y=1 TO 15
40 DRAW -400,25*y
50 DRAW -10*y,5*y
60 NEXT y
70 ORIGIN 70,200
80 FOR z=1 TO 15
85 DRAW 400,-25*z
90 DRAW 10*z,-5*z
100 NEXT z
110 ORIGIN 570,200
120 FOR k=1 TO 15
130 DRAW -400,-25*k
140 DRAW -10*k,-5*k
150 NEXT k
```



# WOJSKOWY

Foto: Stanisław Iwan

*czoraj i dziś...*

# INSTYTUT INFORMATYKI

# AMSTRAD — BASIC

Pomimo wielu prób standaryzacji komend BASIC-a ciągle istnieje tendencja do pojawiania się kolejnych wersji tego języka. Każdy nowy typ mikrokomputera zawiera zbiór unikalnych słów kluczowych, a z oryginału pozostaje nie więcej jak kilkanaście. Podobnie jest z maszyną **AMSTRAD**.

Poniżej opisane zostały te jej słowa kluczowe, które nie występują w innych dialektach BASIC-a.

**Komendy:**

MEMORY — ustala wielkość pamięci dostępnej dla BASIC-a przez podanie najwyższego bajtu (analogia do CLEAR w Spectrum);
CALL — wywołuje podprogram w języku wewnętrznym;
FRAME — powoduje czasowe zgranie wydruków na ekranie upłynniając ruch;
AFTER...GOSUB — obsługa przerwań (!); po pewnym, określonym przedziale czasowym wywołuje podprogram w BASIC-u;
EVERY...GOSUB — obsługa przerwań (!); regularnie, co pewien okres wywołuje podprogram w BASIC-u;
ERROR — wywołuje błąd o podanym kodzie;
RESUME — wznawia normalną egzekucję programu po powrocie z podprogramu obsługi błędów wywołanego przez ON ERROR GOTO lub ON BREAK GOTO;
CHAIN — ładuje (LOAD) i uruchamia (RUN) program od podanego numeru linii;
OPENIN — otwiera plik ASCII do odczytu z dyskietki;
OPENOUT — otwiera plik ASCII do zapisu na dyskietkę;
CLOSEIN — zamyka odczyt pliku ASCII;
CLOSEOUT — zamyka odczyt pliku ASCII;
ENT — definiuje parametry wysokości dźwięku;
ENV — definiuje parametry głośności dźwięku;
RELEASE — uaktywnia kanały dźwiękowe;
WAIT — czeka na określoną wartość w porcie;
WIDTH — ustala długość linii dla drukarki;
SPEED KEY — określa szybkość samopowtarzania klawisza;
SPEED INK — określa częstotliwość zmian kolorów w trakcie migania (Spectrum FLASH);
SPEDD WRITE — określa szybkość zapisu do pamięci zewnętrznej;
CLG — czyści ekran graficzny;
ORIGIN — ustala miejsce graficznego punktu (0,0) na ekranie graficznym;
MASK — podaje parametr dla instrukcji DRAW powodujący możliwość modyfikowania linii (np. przerywana, punktowa);
FILL — wypełnia kolorem figurę zamkniętą;
LOCATE — pozycja wydruku znaku (Spectrum AT);
MODE — ustala tryb ekranu tekstowego na 20,40 lub 80 znaków w linii;
MOVE — przesuwa kursor graficzny;
SYMBOL — definiuje grafikę użytkownika;
TAG — włącza tryb wydruku znaków na ekranie graficznym;
TAGOFF — wyłącza powyższy tryb;
ZONE — określa liczbę znaków, o jaką przesunie się kursor tekstowy napotykając przecinek;
DEG — włącza tryb obliczania kątów w stopniach;
RAD — włącza tryb obliczania kątów w radianach;
DEFINT — deklaracja zmiennych typu integer;
DEFREAL — deklaracja zmiennych rzeczywistych;
DEFSTR — deklaracja zmiennych tekstowych;
ERASE — wymazanie zmiennej z pamięci;
KEY — definicja klawisza funkcyjnego;
KEYDEF — zmienia znaczenie klawisza;
CLEAR INPUT — powoduje wyczyszczenie bufora klawiatury;
CURSOR — odpowiada za wydruk kursora podczas symulacji instrukcji INPUT;
DI — wyłącza system przerwań;
EI — włącza system przerwań;

**Funkcje:**

CINT — zmienia liczbę rzeczywistą na całkowitą;
CREAL — zmienia liczbę całkowitą na rzeczywistą;
FIX — wyodrębnia część całkowitą liczby;
MAX — podaje najwyższą wartość z wyszczególnionych argumentów numerycznych;
MIN — podaje najniższą wartość;
ROUND — zaokrągla wartość do podanej liczby miejsc po przecinku;
UNT — podaje dziesiętnie adres wyrażony binarnie lub heksadecymalnie;
ASC — podaje kod znaku w systemie ASCII;
BIN$ — podaje wartość binarną liczby całkowitej;
HEX$ — przedstawia liczbę w postaci heksadecymalnej;
DEC$ — przedstawia liczbę w postaci dziesiętnej;
LOWER$ — zmienia w łańcuchu tekstowym duże litery na małe;
UPPER$ — zmienia w łańcuchu tekstowym małe litery na duże;
STRING$ — tworzy łańcuch tekstowy złożony z podanej ilości wybranego znaku;
SPACE$ — tworzy łańcuch tekstowy złożony z podanej ilości spacji;
COPYCHR$ — odczytuje znak z ekranu (Spectrum SCREEN$);
INSTR — szuka łańcucha tekstowego w innym i podaje pozycję, jeśli znajdzie;
INKEY — sprawdza, w jakiej konfiguracji naciśnięto podany klawisz (shift, control);
JOY — odczytuje stan dołączonego joystick'a;
POS — określa poziome położenie kursora tekstowego, głowicy drukarki lub ilość przesłanych znaków na dyskietkę;
REMAIN — wyłącza zegar obsługujący przerwania podając jego dotychczasowy stan;
SQ — podaje stan jednego z trzech kanałów dźwiękowych;
VPOS — określa pionowe położenie kursora tekstowego;
EOF — wskaźnik końca zbioru podczas pracy z urządzeniami WE/WY;
ERR — podaje numer ostatniego błędu;
DERR — podaje numer linii ostatniego błędu AMSDOD;
ERL — podaje numer linii ostatniego błędu;
FRE — wskazuje liczbę wolnych bajtów pamięci dla BASIC-a;
HIMEM — podaje najwyższy adres dostępny dla BASIC-a;
TEST — podaje kolor wyszczególnionego punktu na ekranie graficznym;
TIME — zwraca czas rzeczywisty od momentu wyzerowania komputera.

**K. MAMCARZ**

# SŁOWNIK SHARP — PC 1500

Japońska firma SHARP produkuje mikrokomputery posługujące się językiem BASIC. Obowiązuje tu typowy dla BASIC słownik i gramatyka BASIC.

Na tej podstawie poszczególne firmy, a w tym i SHARP opracowały własne WYRAŻENIA.

Ponieważ Czytelnicy posługują się RÓŻNYMI komputerami podajemy skrócony „słownik", właściwy dla PC 1500 SHARP. To ułatwi „przetłumaczenie" z SHARP na SPECTRUM".

Wyrażenia (software) wynikają z architektury (hardware). Zatem krótko o PC 1500: ma on wbudowany WSKAŹNIK 26-pozycyjny z ciekłych kryształów LCD. Na nim wydawane są informacje literowo-cyfrowe. Przez konsolę komputer jest połączony z minipisakiem czterokolorowym. Może on drukować (wypisywać) teksty literowo-cyfrowe, lub kreślić minirysunki na taśmie 59 mm zwykłego papieru.

## PAMIĘCI SHARP

**Rejestracja LICZB:** Jest tu cała mnogość sposobów znakowania pamięci: Oto odmiany zapisu:
— jednoliterowa: A,B...,Z
— dwuliterowa: AA, AB...,AZ, BA, BB....,BZ, ZA, ZB...,ZZ.
— literowo-cyfrowa: A0, A1,....,A9,....,Z0 Z1,...,Z9.

**Rejestracja WYRAZÓW:** Operuje TAKIMI SAMYMI ZBIORAMI SYMBOLI z tą uwagą, iż dla WYRAZÓW dodajemy sufix $.

Np. CZ$ = „czwartek", A$ = „Jan", W1$ = „12 lipca".

**Pamięci ZBIOROWE:** Wyrażenie DIM A(3) oznacza CZTERY pamięci liczbowe A(0), A(1), A(2), A(3). Wyrażenie DIM T$ (7) oznacza OSIEM pamięci wyrazów.

Zapis typu DIM A(1,2) oznacza pamięć liczbową MACIERZOWA, mającą DWA wiersze i TRZY kolumny, zaś zapis typu DIM IN$ (5,10).

Przykład ŁĄCZENIA pamięci słów:
W$ = „WAC":W$ = W$ + „ — " + STR$1234
Daje W$ = „WAC—1234"

## ROZKAZY SHARP

RUN 397, RUN „A" — startuj od wiersza 397, etykiety „A"
NEW — kasowanie całego programu
LIST 870, LIST „Z" — podaj wiersz 870, etykietę „Z" programu

## DEKLARACJE SHARP

INPUT — wejście (wprowadzenie) danych
PRINT — wyjście: wyświetlenie na wskaźniku.
CURSOR 13 — zacznij od 13 pozycji wskaźnika
PAUSE — krótkotrwale wyświetlenie na wskaźniku
USING — deklaracja formy zapisu danych
WAIT 50 — występuje łącznie z PRINT. Jest to czas wyświetlenia na wskaźniku
CLS — wygaszenie wskaźnika
BEEP 7,123,300 — generacja dźwięku o 7 powtórzeniach, częstotliwości (umownej) 123, impulsu trwającego 300 jednostek czasu
CLEAR — zerowanie wszystkich pamięci
RANDOM — generacja liczb losowych
DEGREE — kąt w stopniach
RADIAN — kąw w radianach

## FUNKCJE SHARP

SIN, COS, TAN — sinus, cosinus, tangens
ASN, ACS, ATN — arcusy: sinusa, cosinusa, tangensa
LN, LOG — logarytmy naturalny i dziesiętny
EXP — exponenta
DEG — stopnie i ich ułamki dziesiętne
DMS — zapis typu STOPNIE, MINUTY, SEKUNDY
RND 7 — generowanie liczb losowych od 1 do 7
SQR — pierwiastek kwadratowy (lub $\sqrt{\ }$)
ABS — wartość bezwzględna
INT — część całkowita

## OPERACJE SHARP

+, —, *, / — cztery działania
(,) — nawiasy
<, <=, >, >= — nierówności
=, <> — równość, nierówność
^ — „do potęgi"
AND — funkcja logiczna ORAZ
OR — funkcja logiczna LUB
NOT — logiczne ZAPRZECZENIE
INKEY$ — startowanie dowolnym klawiszem
TIME — wywołanie zegara-kalendarza

## STEROWANIE PISAKIEM SHAPR

LLIST 230, 1500 — wydruk programu od wiersza 230 do 1500 włącznie
TEST — próba pisaka. Rysuje cztery kolorowe kwadraciki
LPRINT — wydruk tekstu literowo-cyfrowego
COLOR 3 — pisak czerwony (0 — czarny, 1 — niebieski, 2 — zielony)
GLCURSOR 138 — przesunięcie pisaka o 138 „punktów"
SORGN — ustawienie początku w zadanym punkcie
LINE — (0,0) — (20,—53) — wyrysowanie linii od współrzędnych x,y = 0,0 do 20,—53
RLINE — (10,—12) — wyrysowanie odcinka o rzutach 10,—12
TEXT — stan typowy — gotów do druku tekstów literowo-cyfrowych
GRAPH — przestawienie na rysowanie linii (LINE, RLINE...)

**Wybrał: Janusz MILLER**


Rys. Michał Przybyłowski

# MIKROKOMPUTERY

## STUDIO MIKROKOMPUTEROWE „BIT"

— U C Z Ą

— B A W I Ą

— P R A C U J Ą

*Klub mikrokomputerowy „Bit"*

**Foto: Jan Zelman**

*Elementy systemów mikrokomputerowych (4)*

# ARYTMETYKA I ADRESOWANIE

**Odcinek ten rozpoczynamy od dokończenia, omawianej w poprzednim numerze, logiki i arytmetyki systemów cyfrowych.**

Zapis w systemie dwójkowym, inaczej naturalny zapis binarny (dwójkowy), ma pewne wady, które są nie bez znaczenia dla mikroprocesora. Wymaga on przeznaczenia najstarszego (najbardziej znaczącego) bitu liczby na zapis jej znaku oraz nie bardzo wiadomo co robić z faktem, że istnieją, jakby dwa zera (+0 i —0) zapisywane jako 0 0000 i 1 0000. W 1 bajcie za pomocą naturalnego zapisu binarnego, w skrócie NKB (natural binary code), można zapisywać liczby całkowite z przedziału (+0,...,255).

Z uwagi na niedogodność zapisu w NKB w kodzie znak-moduł, powodującym skomplikowanie operacji arytmetycznych, arytmetyka mikroprocesorów oparta jest na zapisie w kodzie uzupełnień do 2, oznaczanym U2. Kod ten umożliwia zapisanie w 1 bajcie liczby całkowitej ze znakiem z przedziału (—128, +127). Liczba w kodzie U2 zapisywana jest następująco:

$$(22)\quad L_{U2} = -a_n \times 2^n + \sum_{i=0}^{n-1} a_i \times 2^i$$

Najważniejsze cechy charakterystyczne zapisu w kodzie uzupełnień do 2 to: zmiana znaku liczby poprzez zanegowanie bajtu i dodanię jedynki; wykonywanie dodawania i odejmowania jak na liczbach w NKB; traktowanie całkowitych liczb dodatnich jak zapisanych w NKB.

Mikroprocesory 8-bitowe nie są w stanie realizować wprost operacji mnożenia i dzielenia. Wyjątek stanowią mnożenie i dzielenie przez 2. Dzielenie przez 2 wykonywane jest poprzez przesunięcie liczby w prawo i powtórzenie na najbardziej znaczącej pozycji bitu, jaki się tam znajdował przed wykonaniem przesunięcia. Mnożenie przez 2 wykonywane jest poprzez przesunięcie liczby w lewo i dopisanie na najmniej znaczącej pozycji zera.

Z uwagi na używanie przez mikrokomputery zapisu dwójkowego, a przez człowieka zapisu dziesiętnego dobrym rozwiązaniem problemu komunikacji między człowiekiem i mikrokomputerem mógłby być zapis dwójkowo-dziesiętny, zwany BCD *(binary code decimal notation)*. W zapisie BCD każda cyfra dziesiętna kodowana jest oddzielnie za pomocą 4 bitów.

Przykładowo zapiszmy liczbę dziesiętną w kodzie BCD 8421:

1957 = 0001'1001'0101'0111
10

Z uwagi na architekturę mikroprocesora nie opłaca się stosować tego kodu w mikrokomputerach, nabiera on znaczenia w dużych i średnich systemach komputerowych.

Niejako rozwiązaniem pośrednim problemu komunikacji jest zastosowanie zapisu szesnastkowego, znacznie ograniczającego długość zapisu liczb, łatwego do konwersji na zapis dwójkowy.

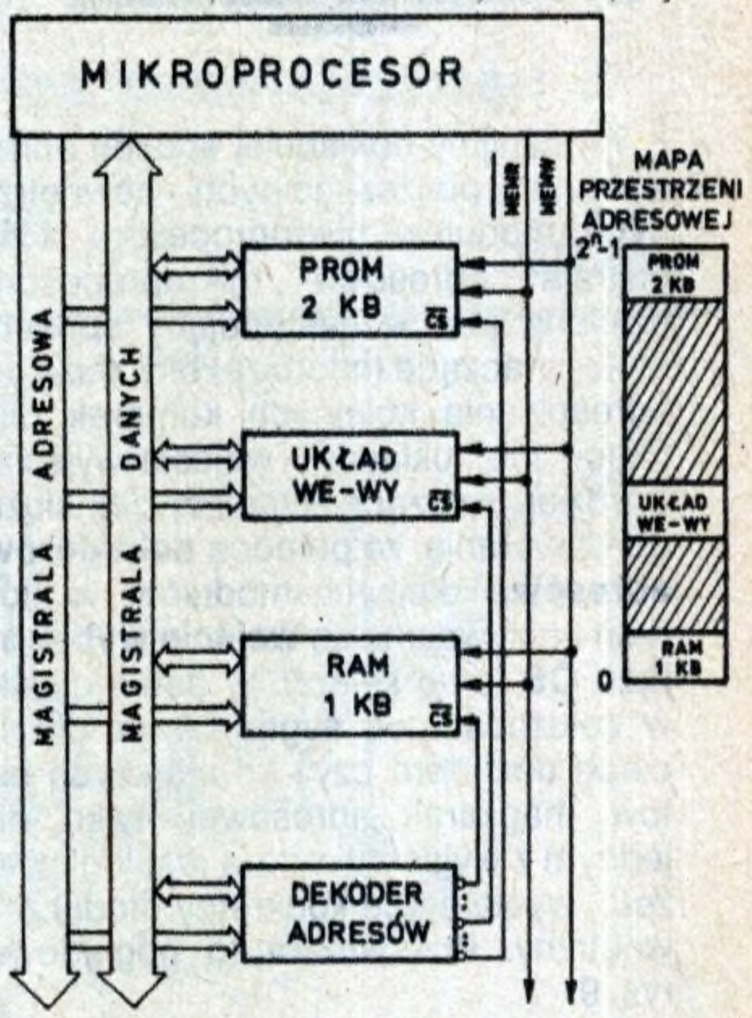
Znane są już nam pojęcia: cykl rozkazowy mikroprocesora *(instruction set)* i cykl maszynowy *(machine cycle)*, zwany cyklem procesora. Z uwagi na sygnały sterujące, wytwarzane przez system mikroprocesorowy, można wyróżnić kilka typów cykli procesora:
— cykl odczytu z pamięci;
— cykl zapisu do pamięci;
— cykl odczytu z układów wejścia-wyjścia;
— cykl zapisu do układów wejścia-wyjścia;
— cykl przyjęcia przerwania.

W czasie tych cykli mikroprocesor generuje sygnały sterujące, w przypadku mikroprocesora INTEL 8080 są nimi:
— **MEMR** *(memory read)*, sygnał odczytu z pamięci;
— **MEMW** *(memory write)*, sygnał zapisu do pamięci;
— **IOR** *(input/output read)*, sygnał odczytu z układów wejścia-wyjścia;
— **IOW** *(input/output write)*, sygnał zapisu do układów wejścia-wyjścia;
— **INTA** *(interrupt acknowledge)*, sygnał potwierdzenia przyjęcia przerwania.

Za pomocą takich sygnałów mikroprocesor 8080 komunikuje się z pamięcią i układami wejścia-wyjścia, stanowiącymi jego moduły zewnętrzne. Wybieranie modułów, z którymi mikroprocesor wymienia dane, następuje poprzez podanie magistralą adresową kombinacji zer i jedynek logicznych, stanowiącej adres modułu.

W zależności do realizacji układowej (sprzętowej) można wyróżnić dwie metody adresowania zewnętrznych modułów mikroprocesora:
— **jednolite adresowanie** pamięci i układów wejścia-wyjścia *(memory maped i/o)*;
— **odrębne adresowanie** pamięci i układów wejścia-wyjścia *(isolated i/o)*.

W pierwszym przypadku istnieje jednolita przestrzeń adresowa, o $2^n$ adresach (gdzie N — liczba linii adresowych). W przestrzeni tej pamięci jak i układy wejścia-wyjścia mogą być umieszczone w dowolny sposób. Rejestry układów wejścia-wyjścia traktowane są tu jak komórki pamięci. Metodę jednolitego adresowania przedstawiono na rys. 7. Do odczytu i zapisy-




**Rys. 7 Jednolite adresowanie modułów zewnętrznych mikroprocesora**

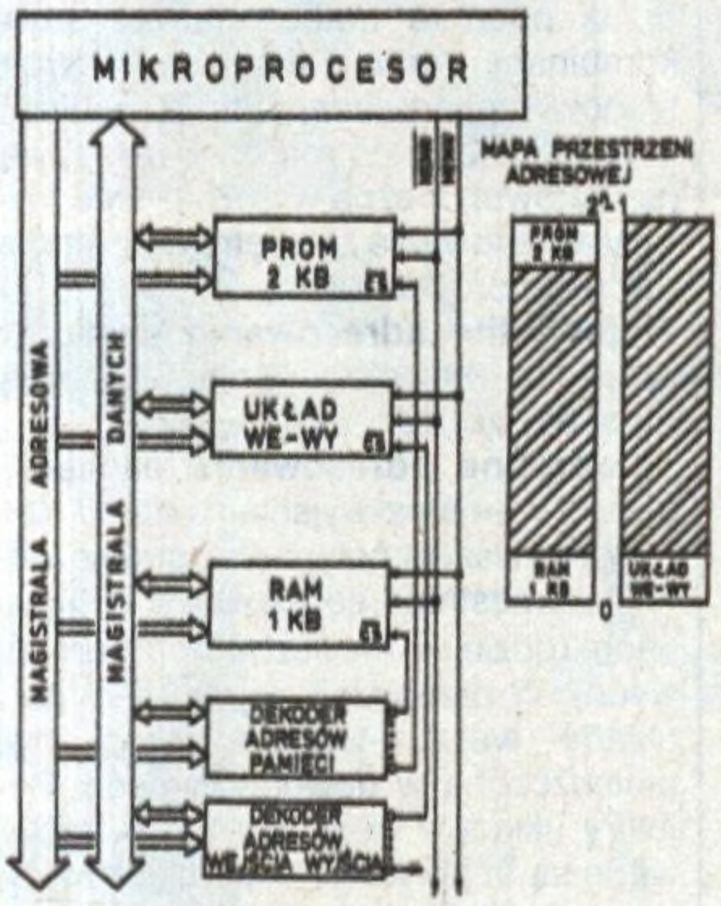
wania danych między mikroprocesorem a modułami zewnętrznymi wykorzystywane są dwa sygnały sterujące: MEMR i MEMW. Ten sposób adresowania posiada bardzo ważną zale-

**dokończenie na str. 22**

**dokończenie ze str. 21**

tę, do komunikacji mikroprocesora z układami wejścia-wyjścia można wykorzystać bogatszy zbiór rozkazów i trybów adresowania, odnoszących się w przypadku odrębnego adresowania tylko do pamięci.

W drugim przypadku, przy odrębnym adresowaniu, istnieją dwie odrębne przestrzenie adresowe. Do komunikacji z modułami zewnętrznymi mikroprocesor wykorzystuje cztery sygnały: MEMR, MEMW, IOR, IOW. Do przesyłania danych z/do układów wejścia-wyjścia służą dwa rozkazy IN *(input)* i OUT *(output)*. Metodę odrębnego adresowania przedstawiono na rys. 8.

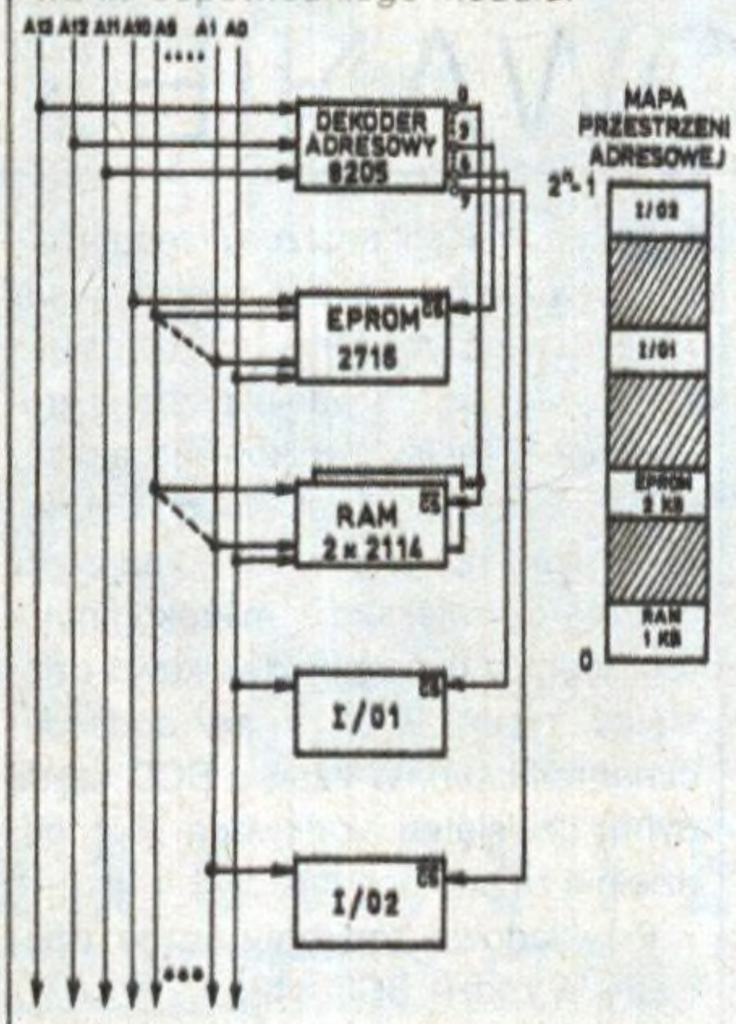



Rys. 8 Odrębne adresowanie modułów zewnętrznych mikroprocesora

Wyjaśnijmy dokładniej sposób adresowania poszczególnych zewnętrznych modułów mikroprocesora. Magistrala adresowa mikroprocesora dzielona jest w następujący sposób. Mniej znaczące (młodsze) bity służą do adresowania kolejnych komórek pamięci lub układów wejścia-wyjścia. Bardziej znaczące (starsze) bity służą do wybierania, za pomocą **dekoderów adresów**, kolejnych modułów, w których uaktywniane są **wejścia wybierające CS** *(chip select)*. W danej chwili, w zależności od sygnałów na wejściach dekodera czyli od starszych bitów magistrali adresowej, tylko na jednym z wyjść dekodera jest logiczne zero, wybierające konkretny moduł zewnętrzny. Przedstawia to poglądowo rys. 9.

Jako dekodery mogą być wykorzystywane również pamięci PROM lub EPROM. Na ich wyjściach danych można zaprogramować odpowiednie sekwencje zer i jedynek logicznych, przełączające — w zależności od bitów na wejściach adresowych pamięci — kolejne moduły mikroprocesora.

W systemach mikroprocesorowych stosuje się zwykle **dwustopniowe dekodowanie adresu**. Pierwszy dekoder służy do zdekodowania grupy modułów drugi natomiast umożliwia zaadresowanie odpowiedniego modułu.




Rys. 9 Wybieranie modułów przy jednolitym adresowaniu

Za pomocą dekoderów możemy podzielić dysponowaną w systemie przestrzeń adresową, utworzyć **mapę przestrzeni adresowej**. W zależności od potrzeb użytkowych przestrzeń ta może być **ciągła, z dekodowaniem pełnym** lub **nieciągła, z dekodowaniem częściowym**, jak na rys. 9.

Zanim przejdziemy do trybów adresowania *(addressing mode)* czyli do sposobów określenia miejsca umieszczenia argumentu danego typu operacji mikroprocesora, przypomnijmy określenie — słowo mikroprocesora. Jest to elementarna jednostka informacji, jaka jest przesyłana w systemie mikroprocesorowym. Słowo może mieć długość 4 bity, 8 bitów (tzw. bajt), 16 bitów, 32 bity. W słowie może być zawarty kod rozkazu (operacji), kod rozkazu wraz z pośrednim adresem argumentu, liczba binarna lub w kodzie 8CD, kod znaku alfanumerycznego. Rozkazy mogą mieć postać jednosłowową, lub wielosłowów, gdy zawierają argument będący liczbą lub numer komórki pamięci (adres), w której zapisana jest liczba.

W **liczniku rozkazów** mikroprocesora zawsze zawarty jest adres komórki pamięci, w której zapisano kod wykonywanego rozkazu. W zależności od zastosowanego w rozkazie trybu adresowania argumenty można określać w różny sposób. Wśród sposobów adresowania, wykorzystywanych w mikroprocesorach, wyróżnia się następujące podstawowe tryby:

— adresowanie natychmiastowe *(immediate addresing)*;

— adresowanie bezpośrednie *(direct addressing)*;

— adresowanie pośrednie *(indirect addressing)*;

— adresowanie względne *(relative addressing)*;

— adresowanie indeksowe *(indexing addressing)*;

**Adresowanie natychmiastowe** dotyczy rozkazów wielosłowowych, 2- lub 3-słowowych. W pierwszym słowie rozkazu umieszczony jest kod operacji, w słowach następnych znajduje się argument (operand). Tryb ten stosowany jest do umieszczania w pamięci danych stałych.

**Adresowanie bezpośrednie** odnosi się do rozkazów co najmniej 3-słowowych. W rozkazie, bezpośrednio za kodem operacji, umieszcza się adres komórki pamięci, z której należy pobrać argument, lub do której należy ten argument przesłać. Tryb ten wykorzystywany jest również przy wykonywaniu operacji na rejestrach wewnętrznych mikroprocesora. Są to zazwyczaj rozkazy 1-słowowe. Programiści stosują ten tryb wówczas, gdy znany jest adres argumentu, dotyczy to np. rozkazów skoków.

**Adresowanie pośrednie** dotyczy rozkazów, w których poza kodem operacji, umieszczony jest adres pamięci, pod który zapisany jest adres argumentu operacji. Do adresowania pośredniego można również zaliczyć adresowanie rejestrowe pośrednie *(pointer addressing)*, w którym adres argumentu umieszczony jest w rejestrze (rejestrach) roboczym mikroprocesora. Rejestr taki nazywany jest wówczas licznikiem danych *(data counter)* lub wskaźnikiem danych *(data pointer)*. Tryb adresowania pośredniego umożliwia korzystanie z argumentów znajdujących się w całej przestrzeni adresowej mikroprocesora.

**Adresowanie względne** odnosi się do rozkazów, w których adres argumentu jest wynikiem dodania liczby umieszczonej w rozkazie (za kodem operacji) i zawartości licznika rozkazów. Tryb ten stosuje się w rozkazach skoków warunkowych, adresowanie to jest stosowane w programach, które mogą pracować w dowolnym miejscu przestrzeni adresowej.

**Adresowanie indeksowe** jest to odmiana adresowania względnego. Adres argumentu jest sumą adresu zawartego wewnątrz rozkazu, bezpośrednio za kodem operacji, a zwanego przesunięciem oraz adresu umieszczonego w rejestrze indeksowym mikroprocesora. Adresowanie to jest oczywiście stosowane w mikroprocesorach posiadających taki rejestr. Tryb wykorzystywany jest głównie, gdy konieczny jest dostęp do danych umieszczonych kolejno w pamięci jako tablica.

Omówione tryby adresowania są podstawowymi, spotykanymi w większości mikroprocesorów. Jak wiele jest sposobów adresowania niech świadczy to, że 16-bitowy mikrokomputer INTEL 8086 posiada ich aż 24.

Źródłem danych dla mikroprocesora jest świat zewnętrzny, z którym to komunikuje się on za pomocą układów wejścia-wyjścia, a są nimi urządzenia zewnętrzne takie jak: klawiatura, monitor lub wyświetlacz cyfrowy, drukarka, pamięć masowa (np. na dyskach elastycznych) etc.

Wymiana danych z urządzeniami zewnętrznymi może być realizowana m.in. poprzez:
— system przerwań (interrupt system);
— bezpośredni dostęp do pamięci (direct memory access DMA).

**System przerwań** umożliwia przerwanie wykonywania przez mikroprocesor **programu głównego** i przejście do **podprogramu** obsługującego urządzenie zgłaszające przerwanie. Sygnały żądania przerwania (każdy z nich posiada przyporządkowany odpowiednio priorytet) można potraktować jako sygnały sterujące pracą mikroprocesora. System przerwań umożliwia więc reakcje mikroprocesora na zdarzenia w jego otoczeniu zewnętrznym. Aby było to możliwe mikroprocesor, pod koniec każdego cyklu rozkazowego, sprawdza czy nie pojawiły się sygnały żądania przerwania. Jeśli w systemie zastosowano **jednopoziomową strukturę przerwań** to mikroprocesor obsłuży zgłoszenie przerwania, które pojawiło się jako pierwsze. Ingorowane są inne zgłoszenia — nawet o wyższym priorytecie, aż do końca obsługi pierwszego. Jeśli pojawiło się kilka sygnałów zgłoszeń przerwania to obsłużone zostanie to z nich, które posiada najwyższy priorytet. W ten sposób mikroprocesor obsłuży wszystkie zgłoszenia i dopiero powróci do wykonywania programu głównego. Jeśli w systemie zastosowano wielopoziomową strukturę przerwań, to podczas wykonywania obsługi przerwania, gdy pojawi się zgłoszenie przerwania o wyższym priorytecie, następuje zawieszenie obsługi danego przerwania, aż do momentu zakończenia obsługi zgłoszenia ważniejszego. Dopiero wtedy następuje powrót do obsługi mniej ważnego zgłoszenia (posiadającego niższy priorytet).

**Bezpośredni dostęp do pamięci DMA** umożliwia przeprowadzanie szybkich przesłań danych (tzw. transmisji blokowych) między pamięcią systemu mikroprocesorowego a urządzeniami zewnętrznymi. Transmisje te wykonywane są bez udziału mikroprocesora, poprzez specjalizowany układ wejścia-wyjścia tzw. sterownik DMA. Gdy ze sterownikiem współpracuje wiele urządzeń zewnętrznych, to podobnie jak w systemie przerwań określany jest priorytet urządzenia.

**Terminologia i pojęcia wprowadzone w pierwszych czterech odcinkach cyklu staną się lepiej zrozumiałe już w następnym numerze „IKS-a", od którego rozpoczynamy omawianie budowy i programowania konkretnych systemów mikroprocesorowych.**

**Jacek WOJTALA**

---

**Jak to zrobić — dokończenie ze str. 3**

```
315 W(Y-2)=WY
320 POKE ZN+Y-3,W(Y-2):POSITION 20,Y:P
RINT W(Y-2);"      "
325 GOTO 160
345 REM Zapamietanie danych dla
      definicji znaku
350 IF PEEK(53279)<3 THEN GOTO 500
355 W(Y-2)=WY:POKE ZN+Y-3,W(Y-2):POSIT
ION 20,Y:PRINT W(Y-2);"      "
360 GRAPHICS 0:POSITION 2,2
365 PRINT "Podaj kod ASCII definiowane
g          znaku  ";
370 INPUT CH:PRINT " Prawidlowa postac
 instrukcji  DATA"
375 POSITION 2,7
380 PRINT LAB;" DATA ";CH;:LAB=LAB+5
385 FOR I=1 TO 8
390 PRINT ",";W(I);
395 NEXT I
400 REM Wprowadzenie poprawki w
      tresci programu
405 POSITION 2,9:PRINT "CONT":POSITION
 2,4
410 POKE 842,13:STOP
415 POKE 842,12
420 GRAPHICS 0
425 REM Odtworzenie srodowiska
430 MEM=PEEK(106)-8:POKE 106,MEM:POKE
756,MEM
435 CHADR=256*MEM:POKE 752,2
440 ZN=ASC("!"):ZN=CHADR+(ZN-32)*8
445 GOSUB 1055:GOTO 75
500 IF PEEK(53279)<2 THEN GOTO 305
505 REM OPTION/START -Koniec programu
510 GRAPHICS 0
515 POSITION 2,4:PRINT "1056"
520 PRINT "CONT":POSITION 2,0
525 POKE 842,13:STOP
530 POKE 842,12
540 END
1000 DIM PROC$(32)
1010 REM Definiowanie procedury
      kopiujacej znaki
1015 FOR I=1 TO 32
1020 READ B:PROC$(I)=CHR$(B)
1025 NEXT I
1030 DATA 104,104,133,213,104,133,212,
104,133,215,104,133,214,162,4,160,0,17
7,212,145,214
1035 REM
1040 DATA 200,208,249,230,213,230,215,
202,208,240,96
1045 REM Kopiowanie zbioru znakow
1050 REM
1055 A=USR(ADR(PROC$),224*256,CHADR)
1060 READ J:IF J=0 THEN RETURN
1065 RETURN
1070 READ B:POKE CHADR+(J-32)*8+I,B
1075 NEXT I
1995 DATA 0
2000 GOTO 1060
```

**GRZEŚ**


Rys. Michał Przybyłowski

# PASCAL (3)

**Dotychczas poznane instrukcje wyczerpują zbiór najczęściej wykorzystywanych instrukcji języka PASCAL. Jednak o użyteczności języka decydują nie tylko dostępne instrukcje, ale także, w znacznej mierze, dostępne dla programisty typy danych. Występujące w przedstawionych wcześniej przykładach typy INTEGER i REAL należą do grupy typów prostych. Oprócz typów prostych w języku PASCAL istnieją typy strukturalne, umożliwiające opis obiektów, traktowanych jako zbiory wartości typów prostych i strukturalnych. Do typowych obiektów tej klasy należą znane z matematyki wektory i macierze. W języku PASCAL odpowiednikiem wektorów i macierzy są tzw. tablice, którym poświęcamy niniejszą część.**

### 1. Deklaracje i wykorzystanie tablic

Jednowymiarowa tablica to ciąg ponumerowanych elementów, których wartości są tego samego typu. Dwuwymiarowa tablica to ciąg wierszy lub kolumn, z których każda liczy tyle samo elementów tego samego typu. Rozmiary dwuwymiarowej tablicy to liczba wierszy i liczba elementów w wierszu (czyli liczba kolumn). Za pomocą jednowymiarowych tablic opisujemy zwykle takie obiekty jak wektory, natomiast za pomocą tablic dwuwymiarowych — macierze. (PASCAL pozwala na definiowanie tablic o większej ilości wymiarów. Ponieważ nie przewidujemy przedstawiania problemów wymagających takich tablic poprzestaniemy na omówieniu tablic co najwyżej dwuwymiarowych).
Naturalnym sposobem opisu tablic jest więc notacja:

a) **ARRAY [**1...*n*, 1...*n*] **OF** *integer*,

dla opisu tablicy jednowymiarowej o „*n*" elementach typu *integer*,

b) **ARRAY [**1...*m*, 1...*n*] **OF** *real*,

dla tablicy dwuwymiarowej o rozmiarach $m \times n$ i elementach typu *real*.

Sposoby deklarowania i wykorzystania tablic w programach przedstawiamy na przykładach.

Rozważmy następujący problem: należy uporządkować rosnąco 6 zadanych liczb całkowitych. Jeden ze sposobów rozwiązania tego problemu opiera się na zasadzie porównywania i zamiany par sąsiadujących ze sobą liczb dopóty, dopóki wszystkie liczby nie zostaną ustawione w kolejności od najmniejszej do największej. Z przedstawionego sposobu wynika, że warunkiem jego zrealizowania jest jednoczesne udostępnienie wszystkich sześciu liczb. Jeżeli potraktujemy zadane liczby jako elementy „pionowej" tablicy jednowymiarowej to, przy pewnej dozie wyobraźni, możemy je uważać za znajdujące się w naczyniu z wodą bąbelki w „wagach" proporcjonalnych do ich wartości. Zatem każde przejście przez tablicę da w wyniku wypchnięcie bąbelka na poziom odpowiadający jego wadze. Metoda ta jest znana pod nazwą sortowania bąbelkowego. Przykładowy przebieg sortowania bąbelkowego ilustruje poniższa tabelka.

**Tabela nr 1**

| Ustawienie początkowe | Pierwsze przejście | Drugie przejście | Trzecie przejście | Czwarte przejście | Piąte przejście |
|---|---|---|---|---|---|
| 574 | 8 | 8 | 8 | 8 | 8 |
| 303 | 574 | 23 | 23 | 23 | 23 |
| 34 | 303 | 574 | 34 | 34 | 34 |
| 125 | 34 | 303 | 574 | 125 | 125 |
| 8 | 125 | 34 | 303 | 574 | 303 |
| 23 | 23 | 125 | 125 | 303 | 574 |

Program realizujący porządkowanie dowolnych sześciu liczb całkowitych metodą sortowania bąbelkowego zamieszczono w przykładzie 8.

**Przykład 8**

```
PROGRAM prog8;
  {porzadkowanie zadanych 6 liczb rosnaco}
 VAR liczby:ARRAY[1..6] OF integer;
     x,i,j:integer;
 BEGIN
  FOR i:=1 TO 6 DO  {wczytanie liczb}
    READ(liczby[i]);
  WRITELN;
  WRITELN('wprowadzone liczby:');
  FOR i:=1 TO 6 DO
   WRITE(liczby[i]:5,' ':3);
  FOR i:=2 TO 6 DO {porzadkowanie liczb}
   FOR j:=6 DOWNTO i DO
     IF liczby[j-1]>liczby[j] THEN
       BEGIN
        x:=liczby[j-1];
        liczby[j-1]:=liczby[j];
        liczby[j]:=x
       END;
  WRITELN;
  WRITELN('liczby uporzadkowane:');
  FOR i:=1 TO 6 DO
   WRITE(liczby[i]:5,' ':3)
END.
```

Zadane liczby wczytywane są do jednowymiarowej sześcioelementowej tablicy „liczby". Działania na elementach tablicy przeprowadza się tak, jak na zmiennych typu zgodnego z typem elementów tablicy. Każdy element tablicy jest identyfikowany jednoznacznie przez nazwę tablicy oraz tzw. indeks, określający położenie elementów w tablicy, np.:

liczby [i] — i-ty wskazywany przez wartość zmiennej „i" element jednowymiarowej tablicy „liczby",

liczby [j — 1] — [j — 1]-szy (wskazany przez wartość wyrażenia „j—1" element jednowymiarowej tablicy „liczby".

W przypadku, gdy indeks jest wyrażeniem, wartość wyrażenia jest obliczona przed udostępnieniem elementu tablicy. Należy zauważyć, że bieżąca wartość indeksu musi mieścić się w przedziale wartości dopuszczalnych dla tego indeksu (w naszym programie <1,6>). Zmiana elementów tablicy miejscami odbywa się za pośrednictwem pomocniczej „x" (dlaczego?).

Program wykonano dla danych z przedstawionej tabelki, uzyskując następujące wyniki:

```
wprowadzone liczby:
  574    303     34    125      8     23
liczby uporzadkowane:
    8     23     34    125    303    574
```

Wprowadzając dane do programu „prog 8" należy pamiętać, aby po każdej wprowadzanej liczbie nacisnąć klawisz ENTER (co spowoduje wykonanie instrukcji READ wczytującej w tym przypadku jedną liczbę). Wyjaśnienie, dlaczego jest to konieczne, przedstawimy w odpowiednim czasie.

Nowo wprowadzonym typom — w tym przypadku tablicom — można przypisywać nazwę, która będzie traktowana jako synonim danego typu. Nazwa ta może być następnie wykorzystana między innymi w deklaracji zmiennych. Taki sposób deklarowania tablic zastosowano w programie dodawania dwóch macierzy o ustalonych wymiarach (przykład 9). Przypomnijmy, że sumą dwóch macierzy **A** i **B** tego samego wymiaru jest trzecia macierz **C**, której ele-

menty wyznaczane są według zależności:

$c_{ij} = a_{ij} + b_{ij}$

gdzie:

*i — indeks wierszy,*

*j — indeks kolumny*

**Przykład 9**

```
PROGRAM prog9;
  {dodawanie macierzy o ustalonych wymiarach}
 TYPE
  macierz=ARRAY[1..3,1..5] OF real;
 VAR
  a,b,          {macierze wejsciowe}
  c:macierz;  {macierz wyjsciowa}
  i,j:integer;{zmienne pomocnicze}
BEGIN
 {wczytanie elementow macierzy a}
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     READ(a[i,j]);
    READLN
   END;
 {wczytanie elementow macierzy b}
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     READ(b[i,j]);
    READLN
   END;
 {dodawanie macierzy a i b}
  FOR i:=1 TO 3 DO
   FOR j:=1 TO 5 DO
     c[i,j]:=a[i,j]+b[i,j];
 WRITELN;
 WRITELN('dane wejsciowe:');
 {wydruk macierzy a}
 WRITELN(' ':10,'macierz a');
 FOR i:=1 TO 3 DO
  BEGIN
   FOR j:=1 TO 5 DO
    WRITE(' ':3,a[i,j]:10);
   WRITELN
  END;
 {wydruk macierzy b}
 WRITELN(' ':10,'macierz b');
 FOR i:=1 TO 3 DO
  BEGIN
   FOR j:=1 TO 5 DO
    WRITE(' ':3,b[i,j]:10);
   WRITELN
  END;
 WRITELN;
 WRITELN('wynik:');
 {wydruk macierzy c}
 WRITELN(' ':10,'macierz c=a+b');
 FOR i:=1 TO 3 DO
  BEGIN
   FOR j:=1 TO 5 DO
    WRITE(' ':3,c[i,j]:10);
   WRITELN
  END
```

W programie, w deklaracji zmiennych typu tablica (a,b,c) wykorzystano wcześniej zdefiniowany typ „macierz" opisujący macierze o wymiarach 3 × 5 i elementach typu REAL. Typ ten został zdefiniowany zgodnie z następującą, przyjętą w języku PASCAL, ogólną postacią definicji typu:

**TYPE** t = T;

gdzie:

**TYPE** — *słowo kluczowe oznaczające definicję typu,*

t — *nazwa definiowanego typu („macierz"),*

T — *opis definiowanego typu*

(**ARRAY** [1...3,1...5] **OF** *real*)

Program wykonano dla macierzy:

$$A = \begin{bmatrix} 10 & -6 & 15 & 1 & 33 \\ 25 & -12 & 31 & 6 & 89 \\ 16 & 4 & 8 & 19 & -7 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

uzyskując następujące wyniki:

```
dane wejsciowe:
        macierz a
   1.0000E+01   -6.000E+00   1.5000E+01   1.0000E+00   3.3000E+01
   2.5000E+01   1.2000E+01   -3.100E+01   6.0000E+00   8.9000E+01
   1.6000E+01   4.0000E+00   8.0000E+00   1.9000E+01   -7.000E+00
        macierz b
   1.0000E+00   2.0000E+00   3.0000E+00   4.0000E+00   5.0000E+00
   1.0000E+00   2.0000E+00   3.0000E+00   4.0000E+00   5.0000E+00
   1.0000E+00   2.0000E+00   3.0000E+00   4.0000E+00   5.0000E+00

wynik:
        macierz c=a+b
   1.1000E+01   -4.000E+00   1.8000E+01   5.0000E+00   3.8000E+01
   2.6000E+01   1.4000E+01   -2.800E+01   1.0000E+01   9.4000E+01
   1.7000E+01   6.0000E+00   1.1000E+01   2.3000E+01   -2.000E+00
```

Elementy obu macierzy zostały wprowadzone wierszami przy czym, w odniesieniu do każdego wiersza, obowiązywały te same zasady wprowadzania danych, jak w programie poprzednim. Przejście do nowego wiersza następowało po dodatkowym wciśnięciu klawisza ENTER, po wprowadzeniu elementów danego wiersza (odpowiada to wykonaniu instrukcji READLN).

Zastanówmy się obecnie, jakie zmiany byłyby konieczne w przedstawionych programach w przypadku zmiany wymiarów tablic. Nietrudno zauważyć, że zmiana wymiarów „rozprzestrzeni" się na cały program. Należy bowiem zmodyfikować również wszystkie instrukcje programu, w których występują wartości graniczne wymiarów tablic. I tak, zmiana wymiarów tablicy „liczby" z przykładu 8 wymaga wprowadzenia zmian w sześciu wierszach programu a zmiana wymiarów tablicy „macierz", z przykładu 9 — aż w trzynastu wierszach!

Wad tych jest pozbawiony program mnożenia dwóch macierzy, z przykładu 10. Przypomnijmy, że iloczyn **A** * **B** dwóch macierzy **A** i **B** jest taką macierzą **C**, w której element i-tego wiersza i j-tej kolumny jest sumą iloczynów elementów i-tego wiersza macierzy **A** przez odpowiednie elementy j-tej kolumny macierzy **B**. Liczba kolumn macierzy **A** musi być przy tym równa liczbie wierszy macierzy **B**. Jeżeli więc macierz **A** jest stopnia p * q i macierz **B** — stopnia q * r, to macierz **C** jest stopnia p * r:

$C_{p*r} = A_{p*q} \times B_{q*r}$

gdzie:

$$c_{ij} = \sum_{k=1}^{q} a_{ik} * b_{kj}$$

**Przykład 10**

```
PROGRAM prog10;
 {mnozenie macierzy}
 CONST
  dw=1; {dolna wartosc indeksu wierszy}
  gw=7; {gorna wartosc indeksu wierszy}
  dk=1; {dolna wartosc indeksu kolumn}
  gk=7; {gorna wartosc indeksu kolumn}
 TYPE
  tablica=ARRAY[dw..gw,dk..gk] OF real;
 VAR
  p,          {liczba wierszy macierzy a}
  q1,         {liczba kolumn macierzy a }
  q2,         {liczba wierszy macierzy b}
```

**dokończenie na str. 26**

dokończenie ze str. 25

```
 r:integer;  {liczba kolumn macierzy b }
 a,b,        {macierze wejsciowe}
 c:tablica;  {macierz wynikowa  }
 i,j,k:integer; {zmienne robocze}
 s:real;
BEGIN
 READLN(p,q1);{wczytanie wymiarow macierzy a}
 IF (p>gw) OR (q1>gk) THEN
  BEGIN
   WRITELN;
   WRITELN('niepoprawne wymiary macierzy a')
  END
 ELSE
  BEGIN
   FOR i:=dw TO p DO  {wczytanie elementow}
    BEGIN             {macierzy a        }
     FOR j:=dk TO q1 DO
      READ(a[i,j]);
     READLN
    END;
   READLN(q2,r);{wczytanie wymiarow macierzy b}
   IF (q2>gw) OR (r>gk) THEN
    BEGIN
     WRITELN;
     WRITELN('niepoprawne wymiary macierzy b')
    END
   ELSE
    BEGIN
     FOR i:=dw TO q2 DO  {wczytanie elementow}
      BEGIN              {macierzy b        }
       FOR j:=dk TO r DO
        READ(b[i,j]);
       READLN
      END;
     WRITELN;{wypisanie wczytanych macierzy }
     WRITELN(' ':5,'macierz a');
     FOR i:=dw TO p DO
      BEGIN
       FOR j:=dk TO q1 DO
        WRITE(a[i,j]:10,' ');
       WRITELN
      END;


     WRITELN(' ':5,'macierz b');
     FOR i:=dw TO q2 DO
      BEGIN
       FOR j:=dk TO r DO
        WRITE(b[i,j]:10,' ');
       WRITELN
      END;
     IF q1=q2 THEN {sprawdzenie zgodnosci  }
      BEGIN        {wymiarow macierzy a i b}
       FOR i:=dw TO p DO {mnozenie macierzy}
        FOR j:=dk TO r DO
         BEGIN
          s:=0;
          FOR k:=dk TO q1 DO
           s:=s+a[i,k]*b[k,j];
          c[i,j]:=s
         END;
       WRITELN;
       WRITELN(' ':5,'macierz c=a*b');
       FOR i:=dw TO p DO
        BEGIN
         FOR j:=dk TO r DO
          WRITE(c[i,j]:10,' ');
         WRITELN
        END
      END
     ELSE
      WRITELN('niezgodne wymiary macierzy a i  b')
    END
  END
END.
```

Wymiary tablic zostały określone poprzez nazwy stałych (dw, gw, dk, gk), zdefiniowanych w części definiującej programu po słowie kluczowym CONST. Ponieważ w części operacyjnej programu również używamy zdefiniowanych stałych, nie jest ona „wrażliwa" na zmiany wymiarów tablic. Ewentualna zmiana wymiarów tablic wymaga jedynie przedefiniowania odpowiednich stałych.

Analizując program „prog 10" można zauważyć, że aktualnie wymagane wymiary tablic są wczytywane w czasie wykonywania programu. Pozwala to na zmianę wymiarów tablic bez konieczności modyfikacji tekstu programu. Oczywiście zakres zmian jest ograniczony zdefiniowanymi wymiarami tablic. Zatem po wczytaniu bieżących wartości wymiarów tablic należy sprawdzić, czy mieszczą się one w przedziale wartości dopuszczalnych. Program wykonano dla macierzy:

$$A = \begin{bmatrix} 1 & 34 & 0 \\ 22 & -14 & -5 \end{bmatrix}$$

$$B = \begin{bmatrix} -7 & 2 & 1 & 2 & 0 \\ 12 & 2 & 5 & 1 & -1 \\ 3 & 2 & -9 & 1 & 0 \end{bmatrix}$$

uzyskując następujące wyniki:

```
     macierz a
1.0000E+00 3.4000E+01 0.0000E+00
2.2000E+01 -1.400E+01 -5.000E+00
     macierz b
-7.000E+00 1.0000E+00 2.0000E+00 0.0000E+00
1.2000E+01 5.0000E+00 1.0000E+00 -1.000E+00
3.0000E+00 -9.000E+00 1.0000E+00 0.0000E+00

     macierz c=a*b
4.0100E+02 1.7100E+02 3.6000E+01 -3.400E+01
-3.370E+02 -3.000E+00 2.5000E+01 1.4000E+01
```

### 2. Składowanie zbiorów roboczych

Czytelnicy wykorzystujący praktycznie system **TURBO PASCAL** zauważyli zapewne, że żądanie utworzenia nowego zbioru roboczego, gdy w pamięci istniał już zbiór (np. poddawany edycji), powodowało wyprowadzenie pytania dotyczącego składowania istniejącego zbioru. Odpowiedź twierdząca (Y) powoduje w tym przypadku składowanie zbioru w pamięci dyskowej, przecząca zaś — jego zniszczenie poprzez otworzenie nowego zbioru roboczego. Widzimy zatem, że składowanie zbioru zapobiega jego zniszczeniu. Składowanie programu można uzyskać również przez wykonanie komendy S. Jeżeli istnieje poprzednia wersja zbioru o tej samej nazwie, co składowany zbiór roboczy, to typ zbioru w tej wersji zostanie przemianowany na **BAK**. Należy podkreślić, że wprowadzając nowy program o dużej ilości wierszy źródłowych warto dokonywać jego składowania po wprowadzeniu kolejnej porcji wierszy.


Rys. Michał Przybyłowski

Zapobiegnie to utracie całego wprowadzonego programu w przypadku jakiejkolwiek awarii. Wczytanie składowanego programu umożliwia znana już komenda **W**.

### 3. Zadania

**Zadanie 7**
Napisać program obliczający iloczyn skalarny dwóch wektorów o n współrzędnych całkowitych.

**Zadanie 8**
Napisać program porządkowania n-elementowego ciągu liczb rzeczywistych metodą „przez proste wybieranie". W metodzie tej przyjmuje się następujące zasady:
- wybieramy liczbę najmniejszą,
- zamieniamy ją z pierwszą liczbą ciągu,
- operacje powyższe powtarzamy dla pozostałych n-1 liczb, następnie dla n-2 liczb itd. aż pozostanie tylko liczba największa.

**Zadanie 9**
Napisać program odejmowania dwóch macierzy o zadanych wymiarach i elementach rzeczywistych.

**Zadanie 10**
Napisać program, który dla zadanej macierzy kwadratowej oblicza sumy elementów leżących:
a) nad przekątną główną,
b) na przekątnej głównej,
c) pod przekątną główną.

**Stefan ROZMUS**

## Przepraszamy...

zamieszczając poprawne, dobrze napisane programy z pierwszej części PASCALA.

```
PROGRAM prog2;
  VAR x,y,suma,roznica,iloczyn:integer;
      iloraz:real;  {deklaracje zmiennych}
  BEGIN
    READ(x,y);
    WRITELN;
    WRITELN('pierwsza liczba = ',x:2,
            ' druga liczba = ',y:2);
    suma:=x+y;
    roznica:=x-y;
    iloczyn:=x*y;
    iloraz:=x/y;
    WRITELN('suma = ',suma:3,' roznica = ',
          roznica:2,' iloczyn = ',iloczyn:4,
          ' iloraz = ',iloraz:2:5)
  END.

PROGRAM prog3;
  VAR a,b,c,  {wspolczynniki trojmianu}
      x1,x2,  {pierwiastki trojmianu}
      delta:real; {zmienna pomocnicza}
  BEGIN
    READLN(a);
    IF a <= 0 THEN
      WRITELN('  niedopuszczalna wartocsc ',
              'wspolczynnika a')
    ELSE
      BEGIN
        READ(b,c);
        WRITELN;
        WRITELN('dla wspolczynnikow : a = ',a);
        WRITELN('                     b = ',b);
        WRITELN('                     c = ',c);
        delta:=b*b-4*a*c; {obliczenie wyroznika}
                          { trojmianu          }
        IF delta < 0 THEN
          WRITELN('brak pierwiastkow rzeczywi',
                  'stych')
  ELSE
    IF delta = 0 THEN
      BEGIN
        x1:=-b/(2*a);
        WRITELN('trojmian ma jeden pierwi',
                'astek podwojny x = ',x1)
      END
    ELSE
      BEGIN
        x1:=(-b-SQRT(delta))/(2*a);
        x2:=(-b+SQRT(delta))/(2*a);
        WRITELN;
        WRITELN('trojmian ma dwa pierwiastki',
                ' :  x1 = ',x1);
        WRITELN(' ':27,' :  x2= ',x2)
      END
  END
END.


PROGRAM prog4;
  VAR x,y:integer;
  BEGIN
    READ(x,y);
    WRITELN;
    WRITE('najwiekszy wspolny podzielnik',
          ' liczb ',x,' i ',y,' wynosi ');
    WHILE x<>y DO
      IF x>y THEN
        x:=x-y
      ELSE
        y:=y-x;
    WRITELN(x)
  END.
```


Rys. Michał Przybyłowski

osiągniemy następujące efekty:

— VIC korzysta z BANKU drugiego,

— PAMIĘĆ EKRANU rozpoczyna się od 35840,

— GENERATOR ZNAKÓW jest standardowy (z ROM) od adresu 36864,

— system operacyjny poinformowany jest o aktualnym położeniu PAMIĘCI EKRANU (35840:256=140).

b) Wykonując sekwencję instrukcji:

POKE 56576,150: POKE 648,120: POKE 53272,25

POKE 56,28672/256: CLR lub POKE 56,112: CLR

osiągamy następujące efekty:

— VIC korzysta z BANKU pierwszego,

— PAMIĘĆ EKRANU rozpoczyna się od 30720,

— GENERATOR ZNAKÓW rozpoczyna się od 28672,

— system operacyjny poinformowany jest o aktualnym położeniu PAMIĘCI EKRANU (30720:256=120),

— zabezpieczenie GENERATORA ZNAKÓW i PAMIĘCI EKRANU przed ingerencją Basica w tym obszarze. Wskaźnik (bajt 56) określający największy adres używany przez Basic ustala go na 28671.

Po włączeniu mikrokomputera VIC pracuje w BANKU 0. PAMIĘĆ EKRANU zlokalizowana jest począwszy od 1024 bajtu, a GENERATOR ZNAKÓW (standardowych) od 4096 bajtu. Jeżeli zmienimy położenie PAMIĘCI EKRANU, to powrót do standardowego obszaru uzyskamy przyciskając jednocześnie klawisze (energicznie): RUN/STOP i RESTORE. Brak kursora na ekranie będzie świadczył o niezgodności rzeczywistego położenia PAMIĘCI EKRANU ze wskazaniami bajtu 648. Standardowa wartość wynosi 4.

Jak programować własne znaki i jak korzystać z graficznego trybu pracy VIC w następnej części.

Program demonstracyjny:

```
0 REM C-64 ZADEMONSTRUJE UZYWANIE DWOCH EKRANOW
10 POKE 56576,140: REM WYBIERAMY BANK 3
20 S$="WERT": REM DOWOLNE 4 ZNAKI
99 REM WYSWIETLAMY JEDEN EKRAN JEDNOCZESNIE ZAPISUJAC NASTEPNY
100 GOSUB 1000: POKE 53272,17: POKE 648,200: REM EKRAN 1
110 GOSUB 1000: POKE 53272,33: POKE 648,196: REM EKRAN 2
120 GOTO 100
999 REM BUDUJEMY LOSOWY LANCUCH ZNAKOW DO WYSWIETLENIA
1000 X$="": FOR J=1 TO 9: L=RND(1)*4+1
1010 X$=X$+MID$(S$,L,1): NEXT
1020 PRINTCHR$(147);: FOR J=1 TO 111: PRINTX$;: NEXT
1030 RETURN
```

BANK 0: wybieramy za pomocą instrukcji

POKE 56576, (PEEK(56576) AND 252) OR 3

(zwykle wartość 151)

| Adresy początkowe PAMIĘĆ EKRANU | POKE 648, | GENERATOR ZNAKÓW | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | ROM | | | | | | |
| | | 2048 | 4096 | 6144 | 8192 | 10240 | 12288 | 14336 |
| | | POKE 53272, | | | | | | |
| 1024 | 4 | 19 | 21 | 23 | 25 | 27 | 29 | 31 |
| 2048 | 8 | 35 | 37 | 39 | 41 | 43 | 45 | 47 |
| 3072 | 12 | 51 | 53 | 55 | 57 | 59 | 61 | 63 |
| PAMIĘĆ EKRANU nie może być zlokalizowana między 4096--8191, ponieważ VIC widzi tu GENERATOR ZNAKÓW standardowych z ROM-u | | | | | | | | |
| 8192 | 32 | 131 | 133 | 135 | 137 | 139 | 141 | 143 |
| 9216 | 36 | 147 | 149 | 151 | 153 | 155 | 157 | 159 |
| 10240 | 40 | 163 | 165 | 167 | 169 | 171 | 173 | 175 |
| 11264 | 44 | 179 | 181 | 183 | 185 | 187 | 189 | 191 |
| 12288 | 48 | 195 | 197 | 199 | 201 | 203 | 205 | 207 |
| 13312 | 52 | 211 | 213 | 215 | 217 | 219 | 221 | 223 |
| 14336 | 56 | 227 | 229 | 231 | 233 | 235 | 237 | 239 |
| 15360 | 60 | 243 | 245 | 247 | 249 | 251 | 253 | 255 |

BANK 1: wybieramy za pomocą instrukcji

POKE 56576, (PEEK(56576) AND 252) OR 2

(zwykle wartość 150)

| Adresy początkowe PAMIĘĆ EKRANU | POKE 648, | GENERATOR ZNAKÓW | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 16384 | 18432 | 20480 | 22528 | 24576 | 26624 | 28672 | 30720 |
| | | POKE 53272, | | | | | | | |
| 16384 | 64 | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 |
| 17408 | 68 | 17 | 19 | 21 | 23 | 25 | 27 | 29 | 31 |
| 18432 | 72 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 |
| 19456 | 76 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 |
| 20480 | 80 | 65 | 67 | 69 | 71 | 73 | 75 | 77 | 79 |
| 21504 | 84 | 81 | 83 | 85 | 87 | 89 | 91 | 93 | 95 |
| 22528 | 88 | 97 | 99 | 101 | 103 | 105 | 107 | 109 | 111 |
| 23552 | 92 | 113 | 115 | 117 | 119 | 121 | 123 | 125 | 127 |
| 24576 | 96 | 129 | 131 | 133 | 135 | 137 | 139 | 141 | 143 |
| 25600 | 100 | 145 | 147 | 149 | 151 | 153 | 155 | 157 | 159 |
| 26624 | 104 | 161 | 163 | 165 | 167 | 169 | 171 | 173 | 175 |
| 27648 | 108 | 177 | 179 | 181 | 183 | 185 | 187 | 189 | 191 |
| 28672 | 112 | 193 | 195 | 197 | 199 | 201 | 203 | 205 | 207 |
| 29696 | 116 | 209 | 211 | 213 | 215 | 217 | 219 | 221 | 223 |
| 30720 | 120 | 225 | 227 | 229 | 231 | 233 | 235 | 237 | 239 |
| 31744 | 124 | 241 | 243 | 245 | 247 | 249 | 251 | 253 | 255 |

BANK 3: wybieramy za pomocą instrukcji

POKE 56576, (PEEK(56576) AND 252)

(zwykle wartość 148)

UWAGA: pamięć o adresach 49152-53247 może być tylko używana jako PAMIĘĆ EKRANU (patrz komentarz w BANKU 2).

| Adresy początkowe PAMIĘĆ EKRANU | POKE 648, | GENERATOR ZNAKÓW | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 49152 | 51200 | 53248 | 55296 | 57344 | 59392 | 61440 | 63488 |
| | | POKE 53272, | | | | | | | |
| 49152 | 192 | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 |
| 50176 | 196 | 17 | 19 | 21 | 23 | 25 | 27 | 29 | 31 |
| 51200 | 200 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 |
| 52224 | 204 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 |

BANK 2: wybieramy za pomocą instrukcji

POKE 56576, (PEEK(56576) AND 252) OR 1

(zwykle wartość 149)

| Adresy początkowe | GENERATOR ZNAKÓW | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ROM | | | | | | | |
| PAMIĘĆ EKRANU POKE 648, | 32768 | 34816 | 36864 | 38912 | 40960 | 43008 | 45056 | 47104 |
| | POKE 53272, | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 32768 | 128 | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 |
| 33792 | 132 | 17 | 19 | 21 | 23 | 25 | 27 | 29 | 31 |
| 34816 | 136 | 33 | 35 | 37 | 39 | 41 | 43 | 45 | 47 |
| 35840 | 140 | 49 | 51 | 53 | 55 | 57 | 59 | 61 | 63 |

PAMIĘĆ EKRANU nie może być zlokalizowana między 36864-40959, ponieważ VIC widzi tu GENERATOR ZNAKÓW standardowych z ROM.

Obszaru pamięci o adresach 40960-49151 nie należy używać jako PAMIĘCI EKRANU. VIC ma dostęp w tym obszarze do pamięci RAM, natomiast mikroprocesor widzi pamięć ROM. Pamięć RAM i ROM jest w tym rejonie zrównoleglona i stanowi dwie odrębne części. Chcąc użyć tych obszarów jako PAMIĘCI EKRANU należy wyłączyć pamięć ROM. VIC i mikroprocesor widzą wtedy ten sam rzeczywisty obszar pamięci.

# W naszym komputerlandzie

Wśród robotów z kierowanego przez siebie zespołu Spektruś uchodził zawsze za szefa dobrego, wyrozumiałego lecz pryncypialnego, wymagającego i dominującego nad wszystkimi. W jego domu od dawna jednak panuje równouprawnienie. Może właśnie dlatego trzepanie dywanu, pastowanie podłogi i zmywanie naczyń znużyło go już trochę. Jego żona, Elwra, parokrotnie zwracała na to uwagę. Najpierw delikatnie, potem nieco ostrzej. Spektruś próbował się mobilizować, ale bezskutecznie. Stosunki partnerskie polegają jednak na tym, że gdy jedna strona przeżywa kryzys — druga wyciąga do niej pomocną dłoń.

Obliczywszy, że trzepanie dużego dywanu może kosztować 2 tys. zł, a pastowanie i mycie podłóg jakieś 5 tys., żona postanowiła zastosować elementy reformy.

— Daj mi najlepiej te pieniądze — rzekła kiedyś. Przecież gdybyś był obcym robotem, musiałabym ci właśnie tyle zapłacić. A przecież jakoś ostatnio praca w domu wcale cię nie interesuje.

Trudno było Spektrusiowi zaprzeczyć. Chwycił za portfel i nieśmiało zapytał:

— A jak będzie z porządkami?

— Nie martw się — odrzekła. Za taką sumę można wypucować mieszkanie do połysku.

Po dwóch dniach żona pojawiła się w nowej sukni. Do tego zmieniła uczesanie i klipsy. Z zadowoleniem przeglądała się w lustrze.

— No, jeśli chcesz, żebym była elegancka, zabieraj się do roboty. Chyba nie chcesz, żebym pobrudziła tę elegancką sukienkę.

Spektruś bez namysłu złapał za ścierkę. Tak więc dzięki drobnym reformatorskim posunięciom wróciło współpartnerstwo i harmonia...

**Podglądał: Eugeniusz MLECZAK**

# PROGRAM 6

# SHARP

## BITWA MORSKA

Proponowana bitwa morska odbywa się na ograniczonym AKWENIE: ma on 25 PÓL uszeregowanych w 5 wierszy i 5 kolumn. (Rys. 1)

Moja pierwsza salwa została oddana w pole o współrzędnych „wiersz" 1, „kolumna" 5, czyli skrótowo (1,5).

Komputer zameldował pudło.

Moja druga salwa ma współrzędne (4,4). Komputer: W CELU ZATOPIONY.

Mam satysfakcję, ponieważ zniszczyłem pierwszy cel. Prócz tego dostałem cenną informację: ponieważ cel został „zatopiony", zatem wokół niego są obszary, w które nie warto celować.

Moja trzecia salwa wywołała odpowiedź „w celu nie tonie" (2,2).

Mam drugie trafienie i bardziej niż istotną informację: ponieważ „nie tonie", zatem jest to „duża jednostka" i należy ją ostrzelać z zachodu, ze wschodu, z północy lub z południa.

Ja w moich rozgrywkach posługuję się kratkowanym „akwenem", rejestruję każdą salwę i starannie planuję współrzędne kolejnej salwy.

Zdaję sobie sprawę, że taki mały akwen może być zbyt skromnym „teatrem wojny". Łatwo dokonać modyfikacji dla DUŻEGO AKWENU (100 pól, 10 wierszy i 10 kolumn. 40 celów i 50 salw).

Przed każdą bitwą generator liczb losowych rozmieszcza cele. Są one „nieznane" w czasie bitwy. Ich lokalizację, oraz raport walki poda meldunek „po bitwie".

Program wywołuje RUN „B".

Życzymy rozgromienia armady.

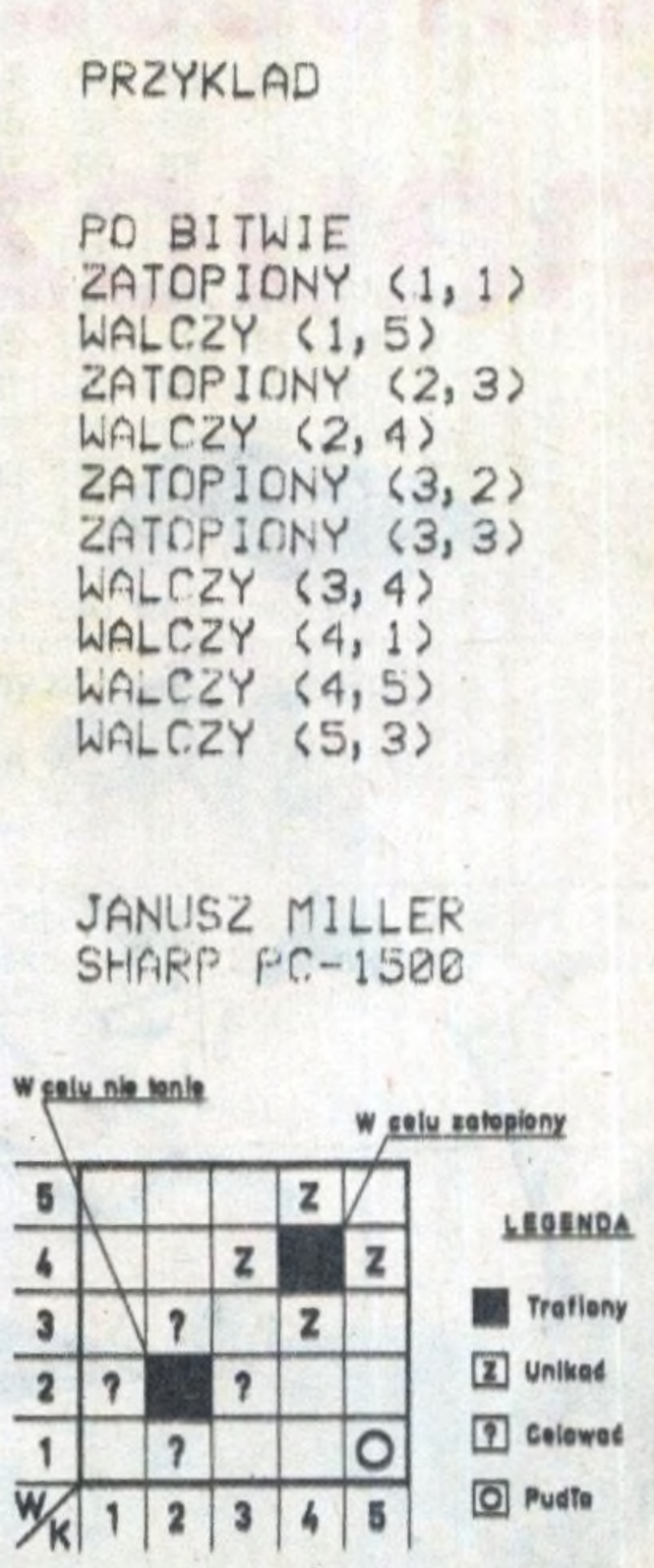
```
1050:"B":CLEAR :
     WAIT :USING
     :DIM T(6,6)
1060:FOR L=1TO 10
1070:W=RND 5:K=
     RND 5
1080:IF T(W,K)=1
     GOTO 1070
1090:T(W,K)=1
1100:NEXT L:PRINT
     "BITWA MORSK
     A"
1110:FOR L=1TO 10
1120:INPUT "CELUJ
     (1,2,3,4,5)
     W=";W:B=0
1130:W=INT W:W=
     ABS W:IF W<1
     OR W>5LET B=
     1
1140:INPUT "CELUJ
     (1,2,3,4,5)
     K=";K:WAIT
     0
1150:K=INT K:K=
     ABS K:IF K<1
     OR K>5OR B=1
     THEN PRINT "
     1,2,3,4,5":
     BEEP 1,100,5
     00:GOTO 1120
1160:IF T(W,K)=0
     THEN PRINT "
     P U D L O":
     BEEP 3,100,2
     00:GOTO 1230
1170:PRINT "W CEL
     U":CURSOR 7:
     Q=Q+1:C=0:D=
     0:T(W,K)=-1
1180:IF T(W-1,K)=
     1OR T(W+1,K)
     =1LET C=1
1190:IF T(W,K+1)=
     1OR T(W,K-1)
     =1LET D=1
1200:IF (C+D)>0
     THEN PRINT "
     NIE TONIE":
     GOTO 1220
1210:PRINT "ZATOP
     IONY"
1220:BEEP 10,50,1
     00
1230:WAIT :PRINT
     "SALWA";L;"
     TRAFIEN";Q
1240:CLS :NEXT L:
     WAIT :PRINT
     "PO BITWIE"
1250:FOR W=1TO 5
1260:FOR K=1TO 5
1270:IF T(W,K)=0
     GOTO 1310
1280:Z$="("+STR$
     W+","+STR$ K
     +")"
1290:IF T(W,K)=-1
     THEN PRINT "
     ZATOPIONY ";
     Z$:GOTO 1310
1300:PRINT "WALCZ
     Y ";Z$
1310:NEXT K
1320:NEXT W:END
```

Wydruk „Przykładu" dotyczy innego przypadku niż przedstawiony na rysunku.

```
PRZYKLAD


PO BITWIE
ZATOPIONY (1,1)
WALCZY (1,5)
ZATOPIONY (2,3)
WALCZY (2,4)
ZATOPIONY (3,2)
ZATOPIONY (3,3)
WALCZY (3,4)
WALCZY (4,1)
WALCZY (4,5)
WALCZY (5,3)


JANUSZ MILLER
SHARP PC-1500
```




Rys. 1

# Pocztowa giełda

**Rozwiązanie krzyżówki z „IKS-a" nr 8**

Hasło brzmi: „DO SIEGO ROKU". Bony pieniężne (1000 zł) wylosowali: Garlak Piotr — Legnica, Jolanta Mycek — Ruda Śl., Beata Konieczna — Leszno, Mariusz Kordys — Rzeszów, Dariusz Tokarew — Złotów. Nagrody książkowe otrzymują: Dorota Masłowska — Aleksandrów Kuj., Magdalena Grześk — Bydgoszcz, Mirosław Cwalina — Łomża, Tomasz Malus — Ostrołęka, Aneta Czartoryska — Zambrów, Jan Cholewa — Ustroń, Tomasz Radziewicz — Braniewo, Dorota Tyszka — Tychy, Andrzej Kwiatkowski — Szczecin, Sławomir Czerwiecki — Bytom.

## Stawką w tej grze jest komputer

Aby złamać szyfr, trzeba odnaleźć klucz. Oto zapis, który należy odczytać:

```
@Y$ 2*)"zN^)x$5$
dU!K1^!#zN"#o!7N
dP^J$$Yu4;"/ P$N
d@pU+$zT*"!-yqcN
```

Przypominamy: obowiązują znaki ASCII zastosowane na Sharpie. Klucz jest datą wydarzenia historii współczesnej zapisaną w międzynarodowym systemie:

DATA: RRRR. MM. DD.

Jakie to wydarzenie? Szukajcie go w naszym stuleciu, dodajmy, że suma cyfr roku, w którym do niego doszło, wynosi 19.

O tym, jak łamać szyfry pisaliśmy w poprzednim numerze „IKS-a" (nr 1/87). A teraz do dzieła, rozwiązanie konkursu już w maju.

## Poszukuję książki:

1. Układy scalone serii UCA 64/VC 74. Parametry i zastosowania — Włodzimierz Sasal WKŁ Warszawa 1985.
2. Mikrokomputer — elementy, budowa, działanie Andrzej Rydzewski, Krzysztof Socha. Sigma Warszawa 1985.
3. Mikrokomputery — programowanie w języku Basic. Bogdan Frelek, Andrzej Lewandowski. Sigma Warszawa 1986.

JANUSZ JANASIK, ul. Zawadzkiego 22/68 63-100 Śrem.

| Giełda | |
|---|---|
| Commodore 128 + magnetofon | 340 tys. zł |
| Commodore 128 + joystick + stacja dyskietek | 500 tys. zł |
| Commodore 16 | 65 tys. zł |
| Commodore 16 + magnetofon 1531 | 100 tys. zł |
| Commodore Plus 4 + magnetofon + 4 programy | 150 tys. zł |
| ZX Spectrum 128 | 175 tys. zł |
| ZX Spectrum 48 KB + magnetofon + joystick | 195 tys. zł |
| Sharp MZ-700 | 160 tys. zł |
| Sharp MZ-721 | 155 tys. zł |
| Drukarka Serial 8056 | 115 tys. zł |
| Drukarka Seikosha GP 505 | 105 tys. zł |
| Drukarka Seikosha GP 500 | 180 tys. zł |
| Drukarka MPS 802 | 220 tys. zł |
| Atari — kaseta | 800 — 1500 zł |
| Atari — kaseta zawierająca 5 gier i 3 programy kopiujące | 1700 zł |
| Atari — gry | 100 — 300 zł/szt. |
| Spectrum — kaseta | 1200 zł |
| Spectrum — gry | 100 — 300 zł/szt. |
| Spectrum — programy użytkowe | od 150 zł/szt. |
| Commodore — gry, programy użytkowe | 150 — 300 zł/szt. |
| **Studio SYSTEM — W-wa Wola** | |
| Schneider Amstrad CPC 6128 | 885 tys. zł |
| Schneider Amstrad PCW 8256 | 1 200 tys. zł |
| Schneider Amstrad PCW 8512 | 1 540 tys. zł |
| Komputer kompatybilny XT:<br>— 640 KB RAM<br>— 2 × FD 360 KB<br>— 20 MB HD<br>— Hercules<br>— RS 232 | 5 400 tys. zł |
| Komputer kompatybilny AT:<br>— 640 KB RAM (512)<br>— 20 MB HD<br>— 1,2 MB FD (5 360 KB)<br>— (Coprocessor 80287)<br>— Hercules (EGA)<br>— RS 232 | 11 — 13 mln zł |
| Drukarki:<br>— SG 15<br>— 15 Xl<br>— SG 10<br>— 10 Xl<br>— NL 10 | <br>988 tys. zł<br>920 tys. zł<br>700 tys. zł<br>630 tys. zł<br>840 tys. zł |
| Manipulatory kulowe (myszki) do IBM | od 280 tys. zł |
| Interface'y KEMPSTON | 10 200 zł |
| Interface'y SINCLAIR | 10 620 zł |
| Dyskietki 5.25" 2D | 1 071 zł |
| Dyskietki 5.25" 2D firmowe | 3 500 zł |
| Dyskietki 5.25" 2HD | 6,5 — 9,5 tys. zł |
| Dyskietki 3" | 6,2 — 8,4 tys. zł |
| Zestaw programów SPECTRUM (min 8) | 1 600 zł |
| Bogata biblioteka programów ATARI | 400 zł/szt. |

# Krzyżówka nr 2

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | K | A | R | T | A | | P | Ę | T | L | A |
| B | O | ■ | E | ■ | | | | ■ | A | ■ | |
| C | D | A | W | K | A | ■ | C | O | B | O | L |
| D | E | ■ | I | ■ | K | O | S | ■ | O | ■ | |
| E | R | | A | | ■ | R | ■ | A | R | E | S |
| F | | | ■ | O | M | ■ | O | | ■ | | |
| G | I | | I | | ■ | | ■ | U | | | |
| H | K | ■ | S | ■ | S | | | ■ | | ■ | |
| I | O | B | L | | T | ■ | C | A | Ł | K | A |
| J | N | ■ | A | ■ | E | W | A | ■ | O | ■ | |
| K | A | M | M | A | N | | | E | | | |

**POZIOMO: A-1) Może być na przykład aperturowa, dualna, indeksowa itp. A-7) Sekwencja rozkazów, która została celowo określona przez programistę i jest wykonywana przez procesor w sposób powtarzalny, aż do momentu spełnienia założonego warunku B-5) Jednostka gęstości zapisu na magnetycznym nośniku danych C-1) Określona porcja np. promieniowania C-7) Uniwersalny język programowy stosowany w zagadnieniach ekonomicznych i zarządzania D-5) Jedno z wcieleń Janusza Gajosa E-1) Robocza nazwa komputerów „jednolitego systemu" E-8) Grecki Mars F-4) Jednostka oporności elektrycznej F-7) Stała część nazwy systemów operacyjnych dla komputerów rodziny Riad lub firmy IBM G-1) Skrótowa nazwa angielskiego specjalistycznego systemu przetwarzania danych przeznaczonych do informowania kierownictwa G-8) Część ciała ludzkiego H-5) Potrawa na ratunek I-1) Osoba żyjąca przy klasztorze nie składająca ślubów I-7) Odwrotność różniczki J-5) Imię żeńskie K-1) Metropolia Jordanii K-7) W starożytnym Rzymie pieśń żałobna**

**PIONOWO: A-1) Urządzenie zmieniające symbole pewnego alfabetu na symbole innego alfabetu G-1) Obraz religijny w sztuce Wschodnio-Chrześcijańskiej E-2) Skrótowa nazwa firmy, będącej największym producentem sprzętu komputerowego na świecie A-3) Pokaz, parada G-3) Mahometanizm E-4) Ogólna nazwa systemów operacyjnych dla komputerów rodziny RIAD lub firmy IBM A-5) Starożytne liczydła H-5) Pistolet maszynowy skonstruowany przez Shepparda i Turpina D-6) Przyjęta z języka angielskiego nazwa spójnika LUB G-6) Klasyczny japoński balet dramatyczny A-7) Nazwa systemu informatycznego przeznaczonego do kierowania produkcją H-7) Nazwa pakietu programów przeznaczonych dla komputerów serii ODRA 1300 oraz ICL-1900 i stosowanych do kontroli zapasów magazynowych E-8) Stosowana w języku rosyjskim skrótowa nazwa zautomatyzowanego systemu zarządzania A-9) Środki transportowe G-9) Może być operacyjny (mnożenia, sumujący itp.) E-10) Skrót oznaczający komputer Hybrydowy A-11) Najpopularniejszy instrument muzyczny w starożytnej Grecji, Atrybut Euterpe G-11) Lęk, strach przed czymś**

**HASŁO: (3-B, 3-I, 3-C, 6-E, 7-F)(4-A, 8-C)(10-I, 9-J, 2-K, 7-A, 8-G, 9-A, 8-K, 3-A)(1-C, 11-C, 2-A)(7-C, 3-D, 5-J, 2-I, 1-G, 10-E)**

**Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji do końca marca 1987 r. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.**

# MIKROCIEKAWOSTKI

**SAM 3001 AT produkowany jest przez koreańską firmę Samsung Semiconductor Telecommunications Co. Ltd. Jest kompatybilny z komputerem IBM PC AT. Pracuje w oparciu o mikroprocesor 80286 z możliwością dołączenia dodatkowego procesora arytmetycznego 80287. Pamięć operacyjna ma 640 Kbajtów z możliwością powiększenia jej do 16 Mbajtów. Pojemność pamięci na dyskach elastycznych 1,2 Mbajta i na dyskach sztywnych 20 Mbajtów z możliwością powiększenia do 80 Mbajtów. Cena podstawowa 4395 dolarów.**


Rys. Michał Przybyłowski

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Michał Przybyłowski, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. 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. 8501. Nr. ind. 361682. K-62