X 3 1987 (11)

INFORMATYKA KOMPUTERY SYSTEMY

Dodatek „Żołnierza Wolności" ISSN 0860-2094 Cena 50

MIKROKOMPUTERY IBM PC

UDAWANIE RZECZYWISTOŚCI

W numerze:

- W szponach ATARI (2) Praca w trybie znakowym — str. 3
- Kreska CPC 464 (2) — str. 7
- Grafika (Amstrad) — str. 8
- Grafika Commodore 64 (2) — str. 9
- Mikrokomputery IBM PC — str. 11
- Biblioteka — str. 12
- „Home, Office, Personal Computer 1987" — Tradycja to nasz obowiązek — str. 13 — Na ekranie „Komputera'87" — str. 14 — Fotoreportaż — str. 16—17
- Udawanie rzeczywistości — str. 18
- Mikroserwis — str. 20
- Wyszukiwanie informacji w zbiorze — str. 21
- Sztuki i sztuczki (5) — str. 22
- Szkoleniowy system graficzny — Oprogramowanie znacznika świetlnego — str. 24
- Sami piszemy gry — str. 27
- W naszym komputerlandzie — str. 29
- OLIVETTI — str. 29
- Liga Myślących — str. 31
- Notowania komputerowe — str. 31

Podobnie jak przed rokiem parada komputerów wzbudziła duże zainteresowanie. Ponad siedemdziesiąt firm, w tym znaczące w świecie marki, ściągnęło do dusznych sal Pałacu Kultury i Nauki fachowców, i pasjonatów nowoczesnej techniki. A było na co popatrzeć.

Dla handlowców oczywiście niewielkie mają znaczenie okrzyki zachwytu, bo o nie w naszej sytuacji raczej nietrudno, ale konkretne kontrakty, których realność jest sprawą być, albo nie być na polskim rynku.

Oczywiście „być". I jeśli nawet teraz trudno się z nami handluje, szczególnie tak nowoczesną techniką, którą nie zawsze jesteśmy w stanie skonsumować, to przecież w przyszłości, dla światowych potentatów możemy być dobrym partnerem. Tymczasem komputerowe firmy muszą zadbać, aby ich przyszły klient miał za co kupić ich wyroby, czyli po prostu wcześniej trzeba mu dolary do kieszeni włożyć, żeby je później wyciągnąć. Z tą drugą czynnością nie ma kłopotów — gorzej z pierwszą.

Z kim handlować warto? Na to trzeba odrobiny doświadczenia i przede wszystkim wiedzy. Z dnia na dzień przybywa nam firm komputerowych, oferujących sprzęt i oprogramowanie. Handlowanie oprogramowaniem jest w wielu przypadkach czystym piractwem, w którym interes polega na oferowaniu cudzej pracy, za duże pieniądze, z jednoczesnym pozbawieniem wszelkich praw autorskich i majątkowych. Ot, po prostu kto pierwszy znajdzie dobry program przerzyna go natychmiast i puszcza w obieg.

Oczywiście dotyczy to nie tylko oprogramowania profesjonalnego, którego ceny sięgają często setek tysięcy, ale również i naszych programów drukowanych na łamach „IKS-a". Spotkać je można na komputerowych giełdach, starannie nagrane na kasety w niezmienionej wersji, albo przerobione na inny dialekt BASIC-a. W tej sytuacji autor jest nieważny — tu liczy się pieniądz.

Niestety dla wielu zwiedzających liczy się przede wszystkim efekt: błysk kamery, obficie rozdawane kolorowe foldery i drobne upominki.

Atrakcją komputerowych pokazów były oczywiście gry i kolorowe grafiki. Nikogo to nie dziwi. Tak jest na całym świecie. Taki jest właśnie pierwszy kontakt z komputerem. A że zainteresowanie tą techniką u nas nie spada — możemy sądzić, że za kilka lat komputer stanie się nie tylko kosztowną zabawką, ale powszechnie stosowanym narzędziem pracy. Czego i nasza redakcja życzyłaby sobie.

W. CETERA

Uwaga ośmioklasiści

Wojskowe licea ogólnokształcące czekają

Uczniowie klas ósmych muszą zdecydować o dalszej nauce. Jedną z ciekawszych możliwości są dla nich wojskowe licea ogólnokształcące. Aktualnie oczekuje na swoich przyszłych uczniów pięć wojskowych liceów w: Częstochowie, Lublinie, Olsztynie, Toruniu i Wrocławiu.

Ogólnokształcące licea wojskowe realizują pełny program średniej szkoły w klasie matematyczno-fizycznej. Dodatkową atrakcją są w procesie nauczania elementy politechnizacji, komputeryzacji, cybernetyki i informatyki. Licea dysponują bowiem nowoczesną bazą sprzętu komputerowego. Nauka w szkole trwa 4 lata. Kandydatów obowiązuje egzamin wstępny z matematyki i języka polskiego oraz sprawdzian sprawności fizycznej.

Podania o przyjęcie do wybranego wojskowego liceum ogólnokształcącego zainteresowani składają w terminie do 15 kwietnia br. w najbliższej wojskowej komendzie uzupełnień.

Uczniowie w czasie nauki w liceum wojskowym otrzymują bezpłatnie: zakwaterowanie w internacie, ubiór szkolny (wyjściowy i codzienny); wyżywienie, wojskową pomoc leczniczą, podręczniki i inne materiały szkolne; przejazdy (4 razy w roku) do miejsca zamieszkania najbliższej rodziny.

W okresie letnich wakacji organizuje się dla uczniów bezpłatne, trzytygodniowe obozy szkoleniowe, w ramach których można uzyskać między innymi kartę pływacką, patent żeglarza jachtowego, prawo jazdy, specjalność skoczka spadochronowego itp.

Podstawowym problemem każdego komputera, używającego odbiornika telewizyjnego dla celów zobrazowania informacji, jest to, że wyświetlanie jest procesem dynamicznym; z tego powodu telewizor nie pamięta obrazu. W rezultacie musi go pamiętać komputer i stale wysyłać sygnał do telewizora, mówiący co jest do wyświetlania. Wysyłanie informacji jest procesem ciągłym i wymaga stałej uwagi. Z tego powodu większość mikrokomputerów ma specjalne układy, które obsługują telewizor. W Atarii funkcje te spełnia ANTIC.

W szponach ATARI (2)

Praca w trybie znakowym

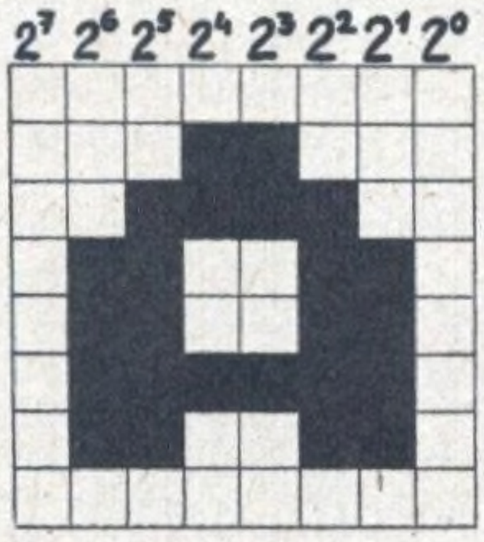
Tomasz MROWIEC
Ludwik PIELA

Mikroprocesor zapisuje informację w pamięci RAM. ANTIC stale kontaktuje się z tym obszarem pamięci, pobierając dane, które przekształca (z pomocą GTIA) na sygnały telewizyjne. Sygnały te idą do telewizora, który wyświetla informację. Pamięć ekranu odwzorowywana jest na ekranie w takiej samej kolejności, w jakiej występuje w RAM. Oznacza to, że pierwszy bajt pamięci ekranu odwzoruje lewy górny narożnik obrazu, drugi bajt odwzoruje jedną pozycję w prawo, następnie trzeci, czwarty, aż do ostatniego bajtu, który jest odwzorowany w prawym dolnym rogu ekranu.

ANTIC może pracować w trybach tekstowych lub graficznych. Najistotniejsza różnica między nimi polega na źródle danych, które tworzą obraz. W trybach graficznych dane z pamięci ekranu są bezpośrednio wyświetlane. W trybach tekstowych każdy bajt traktowany jest jako numer znaku. Na podstawie tego numeru ANTIC wybiera odpowiedni wzorzec i umieszcza go na ekranie. Dzięki temu każdemu znakowi wyświetlonemu na ekranie odpowiada jeden bajt w pamięci ekranu.

Od razu należy wyjaśnić, że numer znaku przechowywany w pamięci ekranu, nie jest kodem *ATASCII* tego znaku. Jest to po prostu numer porządkowy, który wskazuje wzorzec znaku przechowywany w pamięci ROM.

Aby zrozumieć, w jakiej postaci wzorce znaków przechowywane są w pamięci komputera, przyjrzyjmy się bliżej, w jaki sposób znak wyświetlany jest na ekranie. Wyobraźmy sobie siatkę 8x8 oczek, w której umieszczono 64 miniaturowe żarówki. Każda z nich reprezentuje elementarny punkt ekranu (pixel) i może być zapalona lub nie. Aby uzyskać kształt litery „A", muszą być zapalone żarówki jak na poniższym rysunku:



ATARI COMPUTERS

Każdy wiersz siatki może być opisany za pomocą bajtu. „Żarówka" zapalona może być oznaczona jako 1, a zgaszona jako O. Zatem poszczególne wiersze mogą być zapisane w postaci liczb dwójkowych, jakie umieszczone są z prawej strony siatki. Na prawo od nich umieszczone są odpowiedniki dziesiętne. Ci, którzy nie mają wprawy w posługiwaniu się liczbami binarnymi, mogą przekształcić każdy wiersz w następujący sposób:

1. Powyżej każdej kolumny umieszczone są liczby — są to potęgi liczby 2.
2. Jeżeli „żarówka" jest zapalona, bierzemy liczbę umieszczoną nad nią i dodajemy do zmiennej „suma". Sumujemy wszystkie zapalone „żarówki" w wierszu (np. czwarta linia „A" ma wartość 64 + 32 + 4 + 2 = 102).
3. Wykonujemy to dla ośmiu wierszy.

W ten sposób, za pomocą ośmiu bajtów możemy opisać kształt dowolnego znaku, utworzonego w powyższej siatce. W takiej właśnie postaci przechowywane są wzorce znaków w pamięci komputera. Zestaw znaków przechowywanych jest w ciągłym obszarze pamięci ROM w adresach 57344 do 58367 ($E000 do $E777 szesnastkowo): od $E000 zaczynają się znaki specjalne, interpunkcyjne i cyfry; od $E100 (57600) zaczynają się wielkie litery; od $E200 (57856) — znaki graficzne i od $E300 (58112) — małe litery. Łącznie zajęty jest obszar 1024 ($400) bajtów, 128 znaków po 8 bajtów każdy. Negatywy znaków nie są przechowywane, lecz uzyskiwane przez odpowiednie manipulowanie bajtami.

W komórkach pamięci o adresach 756 (CHBAS) i 54281 przechowywany jest adres początku obszaru, w którym przechowywane są wzorce znaków, a dokładniej, numer stronicy (256-bajtowego bloku). Dzięki temu istnieje możliwość zmodyfikowania kształtów istniejących symboli lub wręcz zdefiniowania własnego ich zestawu. Możliwe jest zatem posiadanie znaków polskich, greckich, cyrylicy lub innych specjalnych. Można, na przykład, utworzyć zestaw symboli elektronicznych, zawierający symbole tranzystorów, diod, oporników, kondensatorów i przewodów, dzięki którym możliwe byłoby kreślenie schematów elektronicznych.

Jeszcze ciekawsze efekty wynikają z możliwości zmiany zestawów znaków podczas działania programu. Można wyróżnić trzy tryby czasowe dla takiego multipleksowania zestawu znaków: wolny (rzadziej niż co 1 sek.), szybki (1/50 sek. do 1 sek.) i bardzo szybki (częściej niż co 1/50 sek.).

Wolne multipleksowanie znaków przydatne jest do „zmiany scenerii". Na przykład, program podróży w kosmosie może używać jednego zestawu znaków graficznych dla jednej planety, drugiego dla kosmosu i trzeciego dla innej planety. W miarę przemieszczania się podróżnika, program zmienia zestaw znaków dla otrzymania nowej, egzotycznej scenerii.

Szybkie przełączanie zestawów znaków może być przydatne do animacji. Efekty ruchu można uzyskać zmieniając znaki wewnątrz jednego zestawu, albo zmieniając całe zestawy znaków.

Bardzo szybka animacja cykliczna całego ekranu możliwa jest dzięki przygotowaniu kilku zestawów znaków, wykreśleniu obrazu, a następnie cyklicznemu wybieraniu zestawów. Jeśli każdy znak ma różne wcielenia w innych zestawach znaków, to przejdzie przez animowany ciąg w miarę zmian zestawów. W ten sposób ekran zapełniony obiektami może być cyklicznie przesuwany za pomocą bardzo prostej pętli.

Używanie zestawów znaków do grafiki i animacji ma wiele zalet i pewne ograniczenia. Największą zaletą jest zajmowanie małego obszaru pamięci RAM do utworzenia szczegółowego obrazu. Na przykład, w trybie zerowym możliwe jest umieszczenie na ekranie 960 znaków (24 linie po 40 znaków) i uzyskanie rozdzielczości takiej jak w trybie 8. Dla porównania, zajętość pamięci w trybie 0—960 bajtów, a w trybie 8—7680 bajtów, czyli osiem razy więcej. Obraz używający trybu 2 może dać tyle samo szczegółów i o jeden kolor więcej niż obraz w trybie 7. Dodatkowo będzie on zajmował 200 bajtów, podczas gdy w trybie 7 zajmuje 4000 bajtów. Ze względu na mniejszą objętość pamięci manipulowanie ekranem z grafiką znakową jest dużo szybsze.

Jednak grafika znakowa nie jest zbyt elastyczna. Nie można ustawić tego co chcemy w dowolnym miejscu ekranu. Ograniczenie to uniemożliwia używanie grafiki znakowej w pewnych zastosowaniach. Pozostaje jednak wiele zastosowań graficznych, w których potrzebujemy wyświetlić tylko ograniczoną liczbę wcześniej zdefiniowanych kształtów w ustalonych miejscach. W tych wypadkach grafika znakowa jest bardzo wygodna.

Interesującym przykładem wykorzystania grafiki znakowej jest program „Eastern front 1941", w którym uzyskano bardzo ciekawe efekty dzięki zdefiniowaniu odpowiednich znaków graficznych terenu.

Tworzenie własnych znaków

Komputery Atari, jak zresztą wiele innych, odznaczają się interesującą i pożyteczną własnością — możliwością tworzenia własnych znaków. Nie ma jednak do tego żadnych specjalnych instrukcji. Wszystko trzeba wykonać za pomocą **PEEK** i **POKE**, instrukcji umożliwiających pobieranie i modyfikowanie zawartości pamięci, w bardzo uciążliwy sposób.

Tworzenie własnych znaków rozpoczynamy od zaprojektowania ich kształtu. Można to zrobić za pomocą ołówka, kalkulatora i kilku arkuszy papieru milimetrowego. W celu utworzenia znaku w odpowiedni sposób wypełniamy siatkę 8 x 8 oczek, w której każde oczko reprezentuje elementarny punkt ekranu. Musimy uwzględnić to, że linie poziome wyświetlane są jaśniej niż pionowe, dlatego konieczne jest pogrubienie tych ostatnich, aby stały się widoczne.

Po zaprojektowaniu kształtów znaków musimy je przetłumaczyć na liczby, które są bardziej zrozumiałe dla komputera. Sposób zamiany kształtu znaku na liczby opisany został powyżej. Następnie zestawiamy liczby w instrukcje **DATA**. Gdy już to zrobimy, musimy podmienić liczby wybranego znaku naszymi. Nie jest to takie proste. Zestaw znaków przechowywany jest w pamięci ROM i nie może być zmieniany. Ale i na to jest rada. Wystarczy skopiować ten obszar do RAM, gdzie może być zmieniany, i poinformować komputer (a dokładnie ANTIC) o miejscu położenia nowego zestawu znaków. W tym celu musimy znaleźć odpowiednie miejsce w pamięci do ich przechowywania.

Jednym z rozwiązań jest umieszczenie wzorców znaków w górnych adresach pamięci RAM dostępnej dla użytkownika, a następnie przekonanie komputera, że ma mniej pamięci niż w rzeczywistości. Chroni to zestaw znaków przed przypadkowym uszkodzeniem. W tym celu używamy komórki o adresie 106 (RAMTOP), która przechowuje ilość dostępnych stronic pamięci (bloków po 256 bajtów). Atari 800 XL zawsze umieszcza program wyświetlania (display list) i dane do wyświetlania bezpośrednio poniżej RAMTOP. Jeśli zmienimy zawartość komórki 106, system operacyjny przesunie program wyświetlania i dane do niższych adresów RAM. Zmniejsza to obszar dostępny dla programu, lecz uwalnia część pamięci powyżej nowej wartości RAMTOP. Obszar ten nie będzie używany przez program ani system operacyjny i może być traktowany jako „zarezerwowany".

Poniższy program przedstawia, w jaki sposób można zmniejszyć RAMTOP o cztery strony (1024 bajty). Należy pamiętać o użyciu rozkazu GRAPHICS bezpośrednio po zmianie zawartości komórki 106, w celu przesunięcia programu wyświetlania i danych.

```
10 ? "WOLNE RAM = ";FRE(0)
20 RAMTOP=PEEK(106):? "RAMTOP =
 ";RAMTOP;" STRONIC":? "OSTATNI
 ADRES = ";RAMTOP*256
30 FOR W=1 TO 2000:NEXT W
40 MALERAM=RAMTOP-4
50 POKE 106,MALERAM
60 GRAPHICS 0
70 ARAMTOP=PEEK(106)
80 ? "AKTUALNIE WOLNE RAM = ";F
RE(0)
85 ? "STARE RAMTOP = ";RAMTOP;"
 STRONIC"
90 ? "NOWE RAMTOP = ";ARAMTOP;"
 STRONIC":? "OSTATNI ADRES = ";
ARAMTOP*256
100 ? "ZAREZERWOWANY OBSZAR ZAC
ZYNA SIE  OD   ";ARAMTOP*256+1
```

Używanie tak zarezerwowanego obszaru jest wygodne, ponieważ zawartość komórki 106, odtwarzana jest tylko w dwóch wypadkach: włączenia zasilania i naciśnięcia klawisza RESET. Należy go jednak używać ostrożnie, a w chwili rezerwowania tego obszaru uwzględnić:

1) każde użycie rozkazu GRAPHICS zeruje ekran oraz pierwsze 64 bajty powyżej RAMTOP,
2) każde użycie rozkazu CLEAR (lub PRINT CHR$(125)/ zeruje również pierwsze 64 bajty powyżej RAMTOP.

Dla bezpiecznej pracy najlepiej będzie pozostawić wolne pierwsze 800 bajtów powyżej RAMTOP. Wymaga to zarezerwowania większego obszaru pamięci, niż to jest niezbędne, a tym samym zmniejszenia obszaru dostępnego dla programu.

Innym rozwiązaniem jest umieszczenie znaków w dolnych adresach pamięci RAM, a następnie przekonanie systemu operacyjnego, że wolna pamięć zaczyna się w innym miejscu. W tym celu używamy wskaźnika początku obszaru wolnej pamięci — MEMLO. Przechowywany on jest w komórkach o adresach 743 i 744. Wskazuje adres pierwszej wolnej komórki RAM dostępnej dla programu. Bez DOS-u MEMLO wskazuje 1792, a z DOS-em 7420. Jego zawartość aktualizowana jest po włączeniu zasilania lub naciśnięciu klawisza RESET.

Możemy zmienić zawartość MEMLO w celu zarezerwowania obszaru pamięci poniżej programu. Musi to być jednak zrobione na samym początku, zanim BASIC przejmie sterowanie, ponieważ rozpoczyna on przechowywanie wprowadzanego programu od najniższych adresów. Wymaga to użycia prostego podprogramu w języku maszynowym, który działa następująco: przejmuje kontrolę nad komputerem i wstawia nową wartość do MEMLO, a następnie ponownie inicjuje BASIC — tak jak po naciśnięciu klawisza RESET. Kontrolę nad komputerem przejmuje BASIC nieświadomy, że ma teraz mniej wolnej pamięci. Poniżej zamieszczamy program, który umożliwia zmianę zawartości MEMLO.

```
10 DIM SUBR$(24)
20 FOR I=1 TO 24:READ PRG
30 SUBR$(I)=CHR$(PRG):NEXT I
40 ? "Ile bajtow chcesz zarezer
wowac ";:INPUT REZ
50 HI=INT(REZ/256):LO=REZ-256*H
I
60 SUBR$(6,6)=CHR$(LO):SUBR$(14
,14)=CHR$(HI)
70 Z=USR(ADR(SUBR$))
80 DATA 24,173,231,2,105,0,141
90 DATA 231,2,173,232,2,105,0,1
41
100 DATA 232,2,169,0,133,8,76,0
,160
```

W celu określenia adresu, od którego zaczyna się zarezerwowany obszar, musimy wprowadzić przed uruchomieniem tego programu następującą linię:

?PEEK(743) + 256 * PEEK(744)

Otrzymana liczba może być używana jako adres przeznaczenia dla dowolnych danych, które chcemy przechować w zarezerwowanym obszarze.

Najczęściej jednak obywamy się bez specjalnych sztuczek i umieszczamy własne znaki w obszarze pamięci przeznaczonym na program, zakładając optymistycznie, że nie zajmie pamięci całkowicie. Dlatego najchętniej umieszczamy znaki tuż przed pamięcią ekranu. Istnieje jednak pewna komplikacja. Wielkość pamięci ekranu zależy od używanego trybu graficznego, od 272 bajtów w GRAPHICS 3, do prawie 8000 bajtów w GRAPHICS 8. Możemy wykorzystać zawartość komórki 106 (RAMTOP) i cofnąć się wystarczająco daleko, aby zostawić miejsce na pamięć ekranu.

Zatem dla znalezienia miejsca do przechowania zestawu znaków w trybach 0, 1, 2, 3, 4 i 5 cofnijmy się o 4 stronice od RAMTOP — miejsce na pamięć ekranu, a następnie o 4 dodatkowe do przechowania znaków, razem 8. GRAPHICS 6 będzie wymagać 12, GRAPHICS 7 — 20, a GRAPHICS 8 — 36 stron pamięci.

Po wybraniu odpowiedniego obszaru pamięci możemy przesłać zestaw znaków z ROM do RAM. Mogą to wykonać następujące instrukcje:

```
10 CHBAS=57344
20 CHSET=(PEEK(106)-8)*256
30 FOR I=0 TO 1023
40 POKE CHSET+I,PEEK(CHBAS+I)
50 NEXT I
60 SUBR$(6,6)=CHR$(LO):SUBR$(14
,14)=CHR$(HI)
70 Z=USR(ADR(SUBR$))
80 DATA 24,173,231,2,105,0,141
90 DATA 231,2,173,232,2,105,0,1
41
100 DATA 232,2,169,0,133,8,76,0
,160
```

Przesłanie w Basic'u zajmuje dosyć dużo czasu, dla 1024 bajtów trwa około 15 sekund. Dlatego lepiej jest użyć znacznie szybciej działającej procedury w języku maszynowym. Na przykład podobnej do poniższej:

```
10 DIM BYTE$(80)
20 CHSET=(PEEK(106)-8)*256
30 FOR L=1 TO 32:READ B:BYTE$(L
,L)=CHR$(B):NEXT L
40 DATA 104,104,133,213,104,133
,212
50 DATA 104,133,215,104,133,214
,162
60 DATA 4,160,0,177,212,145,214
70 DATA 200,208,249,230,213,230
,215
80 DATA 202,208,240,96
90 Z=USR(ADR(BYTE$),224*256,CHS
ET)
100 POKE 756,CHSET/256
```

Kolejna linia w każdym z powyższych programów powinna informować ANTIC, gdzie umieściliśmy zestaw znaków:

60 POKE 756, CHSET/256

Normalnie w komórce o adresie 756 przechowywana jest wartość 224 (224 × 256 = 57344 — adres standardowego zestawu znaków w ROM).

Teraz, mając już zestaw znaków w RAM, możemy przystąpić do ich modyfikacji. W tym celu wybieramy znak, który chcemy zmienić, wyznaczamy jego adres i podmieniamy odpowiednie bajty. Po zmianie kształtów wszystkich interesujących nas znaków musimy przechować nowy zestaw znaków w pamięci zewnętrznej, dla wykorzystania w przyszłości do innych programów.

Jak widać z przedstawionego opisu, tworzenie własnych znaków jest czynnością niełatwą i bardzo żmudną. Na szczęście, istnieją specjalne programy, zwane edytorami kształtu znaków lub generatorami znaków, które pomagają nam zmieniać wygląd znaków, pokazują, jak będą wyglądać, wykonują niezbędne obliczenia i przechowują cały zestaw znaków na taśmie lub dyskietce. Do najbardziej znanych należą: The Next Step z Online, Instedit z APX, Font Edit z Code Works czy Ultrafont z Antic. Ze względu na kłopoty związane ze zdobyciem tych programów proponujemy własną wersję generatora znaków. Być może nie ma on wszystkich funkcji oferowanych przez wymienione, ale chcieliśmy, aby był napisany w języku Basic i nie był zbyt trudny do wyprowadzenia.

Generator znaków

Prezentowany program jest ilustracją materiału dotyczącego generowania własnych znaków. Umożliwia on zmodyfikowanie kształtu dowolnego, wybranego przez nas znaku lub zaprojektowanie go od nowa. Działanie rozpoczynamy od wybrania znaku, którego postać chcemy zmienić. W tym celu naciskamy klawisz SELECT i za pomocą manipulatora przesuwamy kursor na wybrane miejsce prezentowanego repertuaru znaków. Po naciśnięciu przycisku manipulatora znak, w dużym powiększeniu, pojawi się w dolnej części ekranu. W przedstawionej siatce możemy zmienić dowolne pole. Naciśnięcie przycisku zmienia jego stan na przeciwny. Jednocześnie, z prawej strony powiększonego znaku widoczna jest jego postać o wymiarach dwa razy większych niż normalnie (tak jak w GRAPHICS 2). Jeżeli uznamy, że uzyskany kształt znaku jest odpowiedni, naciskamy klawisz START. Spowoduje to zamianę znaku.

Po dokonaniu wszystkich modyfikacji musimy przechować zestaw znaków w pamięci zewnętrznej. W tym celu naciskamy klawisz OPTION. Na ekranie pojawi się pytanie o nazwę zbioru. Jeśli chcemy zapisać nowe znaki na taśmie magnetycznej, to podajemy **„C:"**, w wypadku zapisu na dyskietkę podajemy **„D: NAZWA ZB. LST"**. Zapis ma postać linii DATA o numerach od 32000 do 32042.

Jeśli chcemy zacząć pracę od nowa lub wygenerować inny zestaw znaków, to wystarczy nacisnąć klawisz HELP. Spowoduje to odtworzenie standardowego zestawu znaków i przejście do początku programu.

Zapisany w pamięci zewnętrznej zestaw znaków można dołączyć do własnego programu instrukcją ENTER. Włączenie tych linii do programu nie wystarcza. Należy wprowadzić te dane do tablicy znaków i przekazać komputerowi adres tej tablicy. Poniższy program, który może być włączony do własnego programu po ewentualnej zmianie numerów linii, rezerwuje obszar pamięci, niezbędny do przechowania zestawu własnych znaków oraz wpisuje do niego dane o znakach.

```
30000 MEM=PEEK(106)-8:POKE 106,
MEM-1
30010 CHACT=MEM*256
30020 RESTORE 32000
30030 FOR I=0 TO 1023
30040 READ D:POKE CHACT+I,D
30050 NEXT I
30060 POKE 756,MEM
30070 RETURN
```

Dla lepszego zrozumienia działania programu oraz ewentualnej jego rozbudowy podajemy w skrócie, co realizują poszczególne linie:

Linie 10—20 — rezerwowanie obszaru pamięci

dla zestawu znaków i grafiki P/M.

Linie 60—80 — przygotowanie podprogramów w języku wewnętrznym.
Linie 100—120 — ustawienie parametrów dla grafiki P/M.
Linie 140—190 — rozpoznawanie funkcji.
Linie 200—220 — przepisanie zestawu znaków z ROM do RAM.
Linie 300—690 — podprogram wyboru znaku do poprawiania.
Linie 700—795 — ustawienie kursora w środku ekranu.
Linie 800—860 — podprogram przemieszczania kursora.
Linie 900—940 — podprogram rozpoznawania położenia kursora na ekranie.
Linie 1000—1090 — podprogram wyświetlania powiększonego znaku.
Linie 1100—1195 — podprogramy w języku wewnętrznym,
Linie 1500—1550 — podprogram wyświetlania strony tytułowej programu.
Linie 1600—1700 — podprogram wyświetlania instrukcji programu.
Linie 2000—2090 — podprogram umieszczania napisu w środku ekranu.
Linie 2100 — podprogram wygaszania kursora.
Linie 3000—3170 — podprogram zapisu zestawu znaków w pamięci zewnętrznej.

Generator znaków

```
4 CLOSE #6:OPEN #6,4,0,"S:"
5 CLR :DIM A$(10),BUF$(32),UP$(
21),DOL$(21)
10 MEM=PEEK(106)-16
20 AD=PEEK(106)-8:WSK=0:POKE 10
6,MEM-1
30 CHACT=MEM*256
40 GOSUB 1500:REM SKOK DO STRON
Y TYTULOWEJ
50 RESTORE
60 FOR L=1 TO 32:READ PGM:POKE
708,L:BUF$(L,L)=CHR$(PGM):NEXT
L
70 FOR L=1 TO 21:READ PGM:POKE
708,L:UP$(L,L)=CHR$(PGM):NEXT L
80 FOR L=1 TO 21:READ PGM:POKE
708,L:DOL$(L,L)=CHR$(PGM):NEXT
L
85 GRAPHICS 0
90 GOSUB 1600
95 GOSUB 2100
100 POKE 54279,AD:POKE 53277,3:
POKE 559,46
110 START=AD*256+512+68:POKE 53
248,142
120 POKE 704,64:POKE 53256,0:PO
KE 752,1
130 GOSUB 200
140 KF=PEEK(53279):KH=PEEK(732)
150 IF KF=7 AND KH=0 THEN 140
160 IF KH=17 THEN POKE 732,0:GO
SUB 1600:GOSUB 200
170 IF KF=3 THEN GOSUB 3000
180 IF KF=5 THEN GOTO 300
190 GOTO 140
200 REM PRZEPISANIE TABLICY ZNA
KOW
210 Z=USR(ADR(BUF$),224*256,CHA
CT)
220 POKE 756,MEM:RETURN
300 REM WYBOR ZNAKU I POPRAWA
310 ? "}":POSITION 4,1:FOR L=0
TO 26
320 ? CHR$(L);:NEXT L
330 POSITION 4,2:FOR L=32 TO 64
:? CHR$(L);:NEXT L
340 POSITION 4,3:FOR L=65 TO 96
:? CHR$(L);:NEXT L
350 POSITION 4,4:FOR L=97 TO 12
4:? CHR$(L);:NEXT L
360 GOSUB 700
370 A=STICK(0):GOSUB 800
380 IF STRIG(0)<>0 THEN 370
390 GOSUB 900
400 B=M
410 IF B>127 THEN B=B-127
420 IF B<32 THEN B=B+64:GOTO 44
0
430 IF B<97 THEN B=B-32
440 POSITION 10,10:FOR I=0 TO 7
450 W=PEEK(CHACT+I+B*8)
460 POKE START+I,W
470 KK=256
480 FOR K=0 TO 7
490 KK=KK/2
500 IF W<KK THEN ? "+";:GOTO 52
0
510 ? " ";:W=W-KK
520 NEXT K
530 POSITION 10,11+I
540 NEXT I
550 GOSUB 700
560 A=STICK(0):GOSUB 800
565 IF PEEK(53279)=6 THEN 640
570 IF STRIG(0)<>0 THEN 560
580 GOSUB 900
590 IF M=160 THEN A$="+"
600 IF M=ASC("+") THEN A$=CHR$(
160)
610 POSITION XP,YP:? A$;
620 GOSUB 1000
630 IF PEEK(53279)<>6 THEN 560
640 FOR I=0 TO 7
650 W=PEEK(START+I)
660 POKE CHACT+I+B*8,W
670 NEXT I
680 GOSUB 700
690 GOTO 140
700 REM KURSOR W SRODKU
720 POKE 53249,142
730 X=142:Y=80:POKE 705,88
740 RESTORE 1200
750 AD1=AD*256+640
760 FOR I=AD1 TO AD1+127:POKE I
,0:NEXT I
770 FOR I=AD1+Y TO AD1+Y+7
780 READ A:POKE I,A
790 NEXT I
795 RETURN
800 REM RUCH KURSORA
810 IF (A=10 OR A=9 OR A=11) TH
EN X=X-1
820 IF (A=6 OR A=7 OR A=5) THEN
 X=X+1
830 POKE 53249,X
840 IF (A=10 OR A=14 OR A=6) TH
EN U=USR(ADR(UP$),AD1+Y):Y=Y-1
850 IF (A=9 OR A=13 OR A=5) THE
N U=USR(ADR(DOL$),AD1+Y):Y=Y+1
860 RETURN
900 REM LOKALIZACJA KURSORA
910 XP=(X-46)/4:YP=(Y-16)/4
920 A$=""
930 LOCATE XP,YP,M
935 POSITION XP,YP:? CHR$(M)
940 RETURN
1000 REM ZMIANA ZNAKU P/M
1010 FOR I=0 TO 7
1020 KK=256:W=0
1030 FOR J=0 TO 7
1040 LOCATE 10+J,10+I,M
1050 KK=KK/2:IF M=160 THEN W=W+
KK
1060 NEXT J
1070 POKE START+I,W
1080 NEXT I
1090 RETURN
1100 DATA 104,104,133,213,104,1
33,212
1110 DATA 104,133,215,104,133,2
14,162
1120 DATA 4,160,0,177,212,145,2
14
1130 DATA 200,208,249,230,213,2
30,215
1140 DATA 202,208,240,96
1150 DATA 104,104,133,204,104,1
33,203
1160 DATA 160,1,177,203,136,145
,203
1170 DATA 200,200,192,10,208,24
5,96
1180 DATA 104,104,133,204,104,1
33,203
1190 DATA 160,9,177,203,200,145
,203
1195 DATA 136,136,192,255,208,2
45,96
1200 DATA 0,0,24,60,24,0,0,0
1500 GRAPHICS 2+16
1510 A$="GENERATOR":P=3
1520 GOSUB 2000
1530 A$="ZNAKOW":P=6
1540 GOSUB 2000
1550 RETURN
1600 REM INSTRUKCJA
1610 ? "}":POSITION 10,5:? "
 M E N U          ":? :? :? :POKE
752,1
1620 ? "SELECT - wybor znaku"
1630 ? "START  - akceptacja zna
ku"
1640 ? "HELP   - powrot do pocz
atku"
1650 ? "OPTION - zapis znakow n
a kasete"
1660 ? "                lub dysk"
1690 GOSUB 2100
1700 RETURN
2000 Z=INT(0.5+(20-LEN(A$))/2)
2010 FOR I=1 TO LEN(A$)
2020 FOR J=19 TO Z STEP -1
2030 SOUND 0,J*2*I,10,10
2040 POSITION J,P:? #6;A$(I,I);
" ";
2050 FOR W=1 TO 10:NEXT W
2060 NEXT J
2070 Z=Z+1
2080 NEXT I
2090 SOUND 0,0,0,0:RETURN
2100 FOR I=AD*256+512 TO AD*256
+767:POKE I,0:NEXT I:RETURN
3000 REM ZAPIS ZNAKOW NA C: LUB
 D:
3010 CHR=PEEK(756)*256
3020 NUM=32000
3030 ? "} Podaj nazwe zbioru";:
INPUT A$
3040 FOR I=0 TO 1023 STEP 24
3050 ? "}";
3060 POSITION 2,4:? NUM;" ";"DA
TA ";
3070 ? PEEK(CHR+I);
3080 FOR J=1 TO 23
3090 ? ",";PEEK(CHR+I+J);
3100 IF I+J=1023 THEN 3120
3110 NEXT J
3120 ? :? "CONT"
3130 POSITION 2,0:POKE 842,13:S
TOP
3140 POKE 842,12:NUM=NUM+1
3150 NEXT I
3155 ? "}"
3160 LIST A$,32000,32042
3170 A$="":RETURN
```

Colossus chess

Jest to program szachowy, dlatego nie ma potrzeby omawiania zasad gry. Zainteresowani znają je z pewnością lepiej od nas. A ci, którzy nie znają, mogą je znaleźć w stosownej literaturze. Dlatego ograniczymy się do przedstawienia możliwości oferowanych przez program Colossus chess 3.0.

W czasie gry dysponujemy dwoma przełączanymi obrazami: jeden z nich przedstawia szachownicę, a drugi sześć ostatnich posunięć każdego z partnerów, nazwy graczy (Colossus i Opponent), zegary szachowe oraz sposób analizowania ruchów przez program. Przełączania dokonujemy klawiszem „SPACE BAR". Po zakończeniu analizowania pozycji program wyświetla kilka najlepszych posunięć.

dokończenie na str. 23

Zgodnie z zapowiedzią, w poprzednim odcinku obecnie przystąpimy do rozszerzenia naszego programu. Dołączymy do niego elementy, które realizują następujące funkcje:

— składowanie pamięci obrazu na kasecie magnetofonowej,
— wczytanie rysunku z pamięci kasetowej do pamięci obrazu,
— kreślenie odcinka, okręgu i czworokąta.

KRESKA

(Basic CPC 464) (2)

Pierwsza z tych funkcji realizowana jest w linii 198, przy użyciu komendy **SAVE**. Ogólny format instrukcji:
SAVE nazwa — kartoteki — zbioru (typ danych), adres początku zbioru w pamięci RAM, (długość zbioru).

W przypadku naszego programu, kartotekę nazwałem RYSUNEK. Typ danych — litera B oznacza, że tworzony jest na taśmie zbiór binarny. Taki sposób zapisu minimalizuje czas jego zapisywania i odczytywania. Adres C000 pokrywa się z adresem początkowym pamięci obrazu, która w przypadku CPC 464 wykorzystuje 16 Kbajtów RAM (od adresu C000 do FFFF). Długość zbioru określona jest liczbą szesnastkową 4000, co odpowiada 16 K:
4000 szesnastkowo = 16384 dziesiętnie $(4 \cdot 16^3+0 \cdot 16^2+0 \cdot 16^1+0 \cdot 16^0)$. W rezultacie na taśmie kasetowej tworzony jest zbiór składający się z 8 bloków po 2Kbajtów każdy. Szybkość przesyłania danych w trakcie zapisu określana jest przez instrukcję:

SPEED WRITE a

dla a=0 szybkość transmisji wynosi 1000 bodów (bitów na sekundę) dla a=1 przesyłanie odbywa się z szybkością 2000 bodów. Wczytywanie rysunku z pamięci kasetowej do pamięci obrazu — linia 194 — realizowana jest w oparciu o komendę LOAD.
Format:

LOAD nazwa — kartoteki.

Kreślenie odcinka prostej, okręgu oraz czworokąta dokonywane jest odpowiednio w liniach 4220—4400, 4590—4800, 4840—5090. Wspólną cechą przyjętych rozwiązań jest określanie punktów charakterystycznych poprzez naciskanie klawisza **COPY** bądź przycisku FIRE. I tak w wypadku rysowania odcinka pierwsze naciśnięcie **COPY/FIRE** spowoduje zaznaczenie jego punktu początkowego, ponowne zaś, wykreślenie linii do punktu aktualnie wskazywanego przez kursor graficzny. W przypadku kreślenia okręgu, pierwsze naciśnięcie **FIRE/COPY** wyznaczy jego środek. Dalsze przemieszczanie kursora graficznego po ekranie określa zmianę długości promienia. Ponowne wciśnięcie **FIRE/COPY** spowoduje wyliczenie tej długości wg wzoru:

$$R = (x_1-x_2)^2 + (y_1-y_2)^2$$

gdzie:
(x_1, y_1) — współrzędne środka okręgu
(x_2, y_2) — współrzędne punktu aktualnie wskazywanego przez kursor.

Samo rysowanie okręgu polega na zapalaniu na ekranie monitora wszystkich punktów (x,y) spełniających równania:

$$x = x_1 + R \cdot \cos(a)$$
$$y = y_1 + R \cdot \sin(a)$$

gdzie a zawiera się w przedziale od 0° do 360°. Kreślenie czworokąta (równoległoboku) sprowadza się do wyznaczenia trzech jego wierzchołków. Czwarty wyliczany jest wg następujących zależności:

$$x_4 = x_1-(x_3-x_2)$$
$$y_4 = y_3-(y_2-y_1)$$

Po obliczeniu współrzędnych punktu (x_4, y_4) następuje wykreślenie czworokąta za pomocą instrukcji DRAW. Połączenia wszystkich elementów programu KRESKA dokonujemy następująco:

Po wprowadzeniu II części programu do pamięci komputera, składujemy ją na taśmie kasetowej komendą **SAVE** np.:

SAVE „PLUS"

Następnie kasujemy zawartość pamięci komputera. Po wykonaniu tej czynności możemy przejść do utworzenia rozszerzonej wersji programu. W tym celu należy:
1. wczytać program **KRESKA** komendą **LOAD „KRESKA"**.
2. dołączyć część II instrukcją **MERGE „PLUS"**
3. usunąć zbędne linie rozkazem **DELETE**. Dotyczy to linii: 820, 1300 oraz 1480.
4. przenumerować nowo otrzymany program komendą **RENUM**.

Nową wersję programu **KRESKA** należy nagrać na taśmie magnetofonowej komendą:

SAVE „KRESKA V.2".

Na zakończenie uwaga praktyczna. Przy wprowadzeniu programu zaleca się pisanie komend małymi literami. Umożliwia to łatwe odnajdywanie błędów formalnych, gdyż wszystkie niepoprawnie wpisane instrukcje zostaną przez interpreter BASICa wylistowane na ekranie w niezmienionej postaci (interpreter poprawne komendy wypisuje wielkimi literami).

W kolejnych numerach zostaną przedstawione kolejne funkcje:
— wypełnianie kolorem figur zamkniętych
— aerograf
— kreślenie prostopadłościanu.

Janek

```
180 CLS #2:PRINT #2," (L)OAD (S)AVE (K)R
ESLENIE (C)LS";
190 LOCATE #2,2,2:PRINT #2,"(Z)AKONCZENI
E PRACY";
191 REM**********************************
192 REM      CZYTANIE RYSUNKU Z TASMY
193 REM**********************************
194 IF INKEY(36)=0 THEN CLS #2:PRINT #2,
" PROSZE WCISNAC <PLAY>":LOAD"!RYSUNEK":
GOTO 180
195 REM**********************************
196 REM     ZAPISANIE RYSUNKU NA TASMIE
197 REM**********************************
198 IF INKEY(60)=0 THEN CLS #2:PRINT #2,
" PROSZE WCISNAC <PLAY> I <REC> ":FOR i=
1 TO 2000:NEXT i:SPEED WRITE 1:SAVE"!RYS
UNEK",B,&C000,&4000:GOTO 180
320 GOTO 194
480 WINDOW #2,7,38,1,2:PRINT #2,"(M)ENU
GLOWNE (P)RZEMIESZCZENIE (K)OLOR (L)INIA
 (F)IGURY LUB("+CHR$(241)+")"
501 IF INKEY(36)=0 THEN 4220
502 IF INKEY(53)=0 THEN 4440
520 IF INKEY(37)=0 THEN GOTO 630
580 IF INKEY(52)=0 THEN GOTO 1340
4190 REM*********************************
4200 REM         KRESLENIE LINII
4210 REM*********************************
```

```
4220 CLS #2:PRINT #2," WYZNACZ PIERWSZY
PUNKT":LOCATE #2,2,2:PRINT #2,"UZYWAJ :"
+CHR$(240)+CHR$(242)+CHR$(243)+CHR$(241)
+" COPY LUB JOYSTICK"
4230 o%=0
4240 GOSUB 1600:GOSUB 930:GOSUB 1070
4250 GOSUB 870:GOSUB 930:GOSUB 980
4260 GOSUB 870
4270 IF o%=1 THEN PLOT a%,b%,1:MOVE x%,y
%
4280 IF INKEY(F%)=0 THEN GOTO 4310
4290 IF INKEY(27)=0 THEN 480
4300 GOTO 4240
4310 IF o%=1 THEN GOTO 4360
4320 a%=x%:b%=y%:o%=1
4330 LOCATE #2,1,1:PRINT #2," WYZNACZ DR
UGI PUNKT          "
4340 FOR i=1 TO 100:NEXT i
4350 GOTO 4240
4360 MOVE a%,b%
4370 DRAW x%,y%,kolor:o%=0
4380 LOCATE #2,1,1:PRINT #2," WYZNACZ PI
ERWSZY PUNKT  LUB P   "
4390 FOR i=1 TO 100:NEXT i
4400 GOTO 4240
4410 REM*********************************
4420 REM           FIGURY
4430 REM*********************************
4440 CLS #2:PRINT #2," (O)KRAG (C)ZWOROK
AT "
4450 LOCATE #2,2,2:PRINT #2,"(P)OWROT DO
 MENU "
4460 IF INKEY(34)=0 THEN 4590
4470 IF INKEY(27)=0 THEN 480
4475 IF INKEY(62)=0 THEN 4810
4480 GOTO 4460
4560 REM*********************************
4570 REM        KRESLENIE OKREGU
4580 REM*********************************
4590 CLS #2:PRINT #2," WYZNACZ SRODEK OK
REGU   LUB P"
4600 LOCATE #2,2,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241);"
 COPY LUB JOYSTICK"
4610 GOSUB 980:GOSUB 930:GOSUB 1070:GOSU
B 870:GOSUB 930
4620 IF INKEY(27)=0 THEN 480
4630 IF INKEY(F%)=0 THEN cx%=x%:cy%=y%:L
OCATE #2,2,1:PRINT #2,"WYZNACZ PROMIEN O
KREGU":FOR i=1 TO 150:NEXT i:GOTO 4660
4640 GOSUB 870
4650 GOTO 4610
4660 GOSUB 930:GOSUB 1070:GOSUB 980:GOSU
B 870:GOSUB 930
4670 PLOT cx%,cy%,1:MOVE x%,y%
4680 IF INKEY(F%)=0 THEN ox%=x%:oy%=y%:L
OCATE #2,2,1:PRINT #2,"WYZNACZ SRODEK OK
REGU   LUB P":GOTO 4710
4690 GOSUB 870
4700 GOTO 4660
4710 d1%=cx%-ox%:d2%=cy%-oy%
4720 d1%=ABS(d1%):d2%=ABS(d2%)
4730 r=SQR((d1%^2)+(d2%^2))
4740 FOR a%=1 TO 360
4750 DEG
4760 PLOT cx%+r*COS(a%),cy%+r*SIN(a%),ko
lor
4770 x=XPOS:y=YPOS:GOSUB 1530
4780 NEXT a%
4790 PLOT cx%,cy%,0:x%=cx%:y%=cy%
4800 GOTO 4590
4810 REM*********************************
4820 REM      KRESLENIE CZWOROKATA
4830 REM*********************************
4840 CLS #2:PRINT #2," WYZNACZ PIERWSZY
PUNKT   LUB P":LOCATE #2,2,2:PRINT #2,"U
ZYWAJ :"+CHR$(240)+CHR$(242)+CHR$(243)+C
HR$(241)+" COPY LUB JOYSTICK"
4850 q%=0
4860 GOSUB 1600:GOSUB 930:GOSUB 1070
4870 GOSUB 870:GOSUB 930:GOSUB 980
4880 GOSUB 870
4890 IF sk%=0 AND pr%=1 AND q%=0 THEN PL
OT xp%,yp%,kol:MOVE x%,y%:q%=1
4900 IF INKEY(F%)=0 THEN 4930
4910 IF INKEY(27)=0 THEN 480
4920 GOTO 4860
4930 PLOT x%,y%,1
4940 x1%=x%:y1%=y%
4950 LOCATE #2,1,1:PRINT #2," WYZNACZ DR
UGI PUNKT          "
4960 GOSUB 930:GOSUB 1070:GOSUB 870:GOSU
B 930:GOSUB 980:GOSUB 870
4970 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp
%,kol:MOVE x%,y%:sk%=1
4980 IF INKEY(F%)=0 THEN 5000
4990 GOTO 4960
5000 PLOT x%,y%,1:x2%=x%:y2%=y%:q%=0
5010 LOCATE #2,1,1:PRINT #2," WYZNACZ TR
ZECI PUNKT          "
5020 GOSUB 1600:GOSUB 930:GOSUB 1070:GOS
UB 870:GOSUB 930:GOSUB 980:GOSUB 870
5030 IF sk%=0 AND pr%=1 AND q%=0 THEN PL
OT xp%,yp%,kol:MOVE x%,y%:sk%=1:q%=1
5040 IF INKEY(F%)=0 THEN 5060
5050 GOTO 5020
5060 x3%=x%:y3%=y%:MOVE x1%,y1%
5070 x4%=x1%+(x3%-x2%):y4%=y3%-(y2%-y1%)
5080 DRAW x2%,y2%,kolor:DRAW x3%,y3%,kol
or:DRAW x4%,y4%,kolor:DRAW x1%,y1%,kolor
5090 x=XPOS:y=YPOS:GOSUB 1530:GOTO 4840
```

Grafika (AMSTRAD)

```
10 CLS
20 ORIGIN 0,0
30 FOR x=1 TO 15
40 DRAW 320,25*x
50 DRAW 10*x,5*x
60 NEXT x
70 ORIGIN 640,0
80 FOR y=1 TO 15
90 DRAW -320,25*y
100 DRAW -10*y,5*y
110 NEXT y
120 ORIGIN 0,400
130 FOR z=1 TO 15
140 DRAW 320,-25*z
150 DRAW 10*z,-5*z
160 NEXT z
170 ORIGIN 640,400
180 FOR k=1 TO 15
190 DRAW -320,-25*k
200 DRAW -10*k,-5*k
210 NEXT k
220 ORIGIN 0,200
230 FOR x=1 TO 15
240 DRAW 320,25*x
250 DRAW 10*x,5*x
260 NEXT x
270 ORIGIN 0,200
280 FOR y=1 TO 15
290 DRAW 320,-25*y
300 DRAW 10*y,-5*y
310 NEXT y
320 ORIGIN 640,200
330 FOR z=1 TO 15
340 DRAW -320,25*z
350 DRAW -10*z,5*z
360 NEXT z
370 ORIGIN 640,200
380 FOR x=1 TO 15
390 DRAW -320,-25*x
400 DRAW -10*x,-5*x
410 NEXT x
```

Możliwości Commodore 64 (2)

W tej części zajmiemy się znakowym trybem pracy mikrokomputera Commodore 64. W szczególności opisany zostanie GENERATOR ZNAKÓW oraz metoda programowania własnych znaków w trybie pracy dwukolorowej i wielokolorowej. Dla przybliżenia omawianych problemów posłużymy się dwoma programami:

1. **zobrazowanie bitowej reprezentacji znaków rezydujących w GENERATORZE ZNAKÓW;**
2. **dwukolorowa i wielokolorowa realizacja podstawowych elementów obwodu elektrycznego.**

Generator znaków

Wszystkie standardowe znaki graficzne widziane na ekranie otrzymywane są z GENERATORA ZNAKÓW (GZ) mieszczącego się w pamięci ROM. Bezpośredni dostęp do tej pamięci posiada VIC pracując w BANKU Ø lub w BANKU 2. W BANKU Ø standardowy GZ zlokalizowany jest od adresu 4Ø96 do 8191, a w BANKU 2 od adresu 36864 do 4Ø959. W obszarach występowania standardowego GZ pamięć RAM jest przesłaniana pamięcią ROM zawierającą GZ — z punktu widzenia VIC-a. System operacyjny i programy korzystają w tych miejscach z normalnej pamięci RAM o identycznej adresacji.

Dostęp do standardowego GZ system operacyjny i programy mają w obszarze od 53248 bajtu do 57343 bajtu. Napotykamy tu jednak pewną trudność: ten sam obszar pamięci jest wykorzystywany do dwóch różnych celów. Pamięć RAM zawiera w tym miejscu rejestry we/wy niezbędne do obsługi przerwań. O tym, która pamięć fizycznie występuje pod wspomnianymi adresami decyduje aktualne ustawienie przełącznika rodzaju pamięci zlokalizowanego w trzecim bicie pierwszego bajtu pamięci.

W celu uzyskania dostępu do standardowego GZ musimy poinformować system operacyjny, aby nie reagował na pojawiające się sygnały przerwań:

```
3Ø POKE 56334, PEEK (56334) AND 254
```

Posiadacze mikrokomputerów C-64 mogą jednocześnie wprowadzać do pamięci pierwszy z przykładowych programów, który posłuży nam do ilustracji omawianych zagadnień związanych z GENERATOREM ZNAKÓW.

Ustawmy teraz przełącznik rodzaju pamięci tak, aby w obszarze 53248 — 57343 mieć GZ z pamięci ROM:

```
4Ø POKE 1, PEEK (1) AND 251
```

Powrót do stanu poprzedniego tj. odłączenie pamięci ROM, przyłączenie pamięci RAM i umożliwienie obsługi przerwań uzyskamy sekwencją instrukcji:

```
6Ø POKE 1, PEEK (1) OR 4
7Ø POKE 56334, PEEK (56334) OR 1
```

GENERATOR ZNAKÓW podzielony jest na dwa zbiory znaków (po 256 znaków każdy) wielkości 2K. W procesie wyświetlania znaków na ekranie VIC pracuje zawsze tylko na jednym ze zbiorów, który możemy wybrać:

— przyciskając jednocześnie klawisze SHIFT i COMMODORE;
— ustawiając odpowiednią wartość 53272 bajtu według tabeli zamieszczonej w części pierwszej artykułu.

Każdy znak graficzny ma określony KOD EKRANOWY (KE), który jest liczbą porządkową tego znaku w zbiorze GZ. Ponieważ istnieją dwa zbiory znaków, ten sam KE może odnosić się do dwóch różnych znaków (np. kod 1 odpowiada znakom A i a). Zestawy znaków standardowych i ich KODY EKRANOWE zamieszczone są w każdej instrukcji do mikrokomputera C-64.

W naszym przykładowym programie znak będziemy lokalizować podając dane:

```
1Ø INPUT "KOD EKRANOWY ZNAKU Ø — 255"; KE
2Ø INPUT "ZBIÓR ZNAKÓW 1 — 2"; ZB
```

Każdy znak występujący w GENERATORZE ZNAKÓW definiowany jest na ośmiu bajtach. Rzeczywisty adres (RA) znaku w pamięci można wyznaczyć według reguły:

```
5Ø RA=51200+ZB*2048+KE*8
```

Po zlokalizowaniu znaku pozostaje tylko wyprowadzić zawartość pamięci definiującej znak. Każdy z ośmiu bajtów wyprowadzany ma być w postaci zerojedynkowej w kolejnych wierszach ekranu, zachowując uporządkowanie z pamięci. Przedstawiona poniżej grupa instrukcji wykonuje opisaną procedurę:

```
6Ø PRINT: PRINT "POSTAĆ BITOWA ZNAKU:":PRINT
7ØFOR I=Ø TO 7: LP=PEEK (RA+I): FOR J=7 TO Ø STEP —1
8Ø LK=INT(LP/2): T(J)=LP—LK*2: LP=LK: NEXT: Z$=""
9Ø FOR J=Ø TO 7: Z$=Z$+MID$(STR$(T(J)), 2,1): NEXT
1ØØ PRINT"       ";Z$: NEXT
```

Spróbujmy uruchomić nasz program dla następujących danych: KE=1, ZB=1 oraz KE=1, ZB=2. Uzyskamy na ekranie bitową reprezentację litery A i litery a. Litera A ma postać:

```
Ø Ø Ø 1 1 Ø Ø Ø
Ø Ø 1 1 1 1 Ø Ø
Ø 1 1 Ø Ø 1 1 Ø
Ø 1 1 1 1 1 1 Ø
Ø 1 1 Ø Ø 1 1 Ø
Ø 1 1 Ø Ø 1 1 Ø
Ø 1 1 Ø Ø 1 1 Ø
Ø Ø Ø Ø Ø Ø Ø Ø
```

Zatem, każdy znak zbudowany jest z 64 punktów (8 wierszy na 8 kolumn); a wartość bitu 1 odpowiada świecącemu punktowi na ekranie, zaś wartość Ø odpowiada kolorowi. Oczywiście jest to słuszne w trybie pracy dwukolorowej.

Dla ciekawskich pozostawiam dokładniejsze przebadanie GENERATORA ZNAKÓW za pomocą zaprezentowanego programu. Możemy wykryć dodatkowe cechy standardowego GZ (np. jakie znaki graficzne przedstawiają KE, które nie zostały pokazane w instrukcji do C-64?).

Programowanie własnych znaków

Ogólna metoda wykorzystania w grafice komputerowej niestandardowych znaków sprowadza się do budowy własnego GENERATORA ZNAKÓW i przekazania go VIC-owi. Przy realizacji tego typu programów należy wykonać następujące czynności:

— zaprojektować własne znaki (uwzględniając ewentualne ich wykorzystanie w trybie wielobarwnym);

— wybrać obszar pamięci RAM, w którym powstanie nowy GENERATOR ZNAKÓW (korzystaj z tabeli zamieszczonej w pierwszej części artykułu) i skopiować do tego obszaru niezbędne znaki ze standardowego GZ;

— wprowadzić zaprojektowane znaki w odpowiednie miejsca do nowego GZ;

— zlecić VIC-owi korzystanie z nowego GZ;

— przeprowadzić dodatkowe operacje związane z wykorzystywaniem własnego GZ (np. zabezpieczenie się przed niekontrolowanym zniszczeniem zawartości GZ przez program w Basicu).

Powyższe czynności zilustrowane zostały w kolejnym programie, którego zadaniem jest zobrazowanie na ekranie podstawowych elementów obwodu elektrycznego: rezystancji, pojemności, indukcyjności. W zestawie znaków standardowych C-64 nie ma symboli graficznych wymienionych elementów. Przyjęte zostały następujące założenia dotyczące projektowanych znaków:

— rezystancja zaprojektowana jest jako dwa znaki o KE 35 i 36 (# i $);

— pojemność reprezentuje jeden znak o KE 37 (%);

— indukcyjność przedstawiana jest za pomocą trzech znaków o KE 38, 39 i 40/&, ' i (/.

Poszczególne grupy instrukcji opatrzone zostały komentarzem ułatwiającym zrozumienie programu.

```
5 REM KOPIOWANIE STANDARDOWEGO GZ DO WYBRANEGO OBSZARU PAMIĘCI RAM
10 POKE56334, PEEK (56334) AND 254
20 POKE 1,PEEK(1) AND 251
30 FOR I=0 TO 4095: POKE 40960+I, PEEK (53248+I) : NEXT
40 POKE 1,PEEK(1) OR 4
50 POKE 56334, PEEK (56334) OR 1
55 REM WPROWADZENIE WŁASNYCH ZNAKÓW DO GZ W ODPOWIEDNIE MIEJSCA
60 FOR I=41240 TO 41287: READ L: POKEI,L: NEXT
70 DATA 0,170,128,128,128,128,170,0: REM REZYSTANCJA
80 DATA 0,170,2,2,2,2,170,0
90 DATA 130,130,130,130,130,130,130,130: REM POJEMNOŚĆ
100 DATA 40,130,130,130,130,0,0,0: REM INDUKCYJNOŚĆ
110 DATA 162,8,8,8,8,0,0,0
120 DATA 128,32,32,47,47,0,0,0
125 REM PRZEKAZANIE NOWEGO GZ VIC-owi
130 POKE 56576,149: POKE 648,140: POKE 53272,57
140 POKE 56,35840/256: CLR: REM OGRANICZENIE OD GÓRY PAMIĘCI RAM DLA BASICA
150 POKE 53280,1: POKE 53281,1: REM BIAŁY KOLOR OBRZEŻA I EKRANU
160 PRINTCHR$(147) : PRINT: PRINT„PODSTAWOWE ELEMENTY OBWODU ELEKTRYCZNEGO" : PRINT
170 POKE 646,11: REM ZNAKI KOLORU SZAREGO
180 PRINT" o——#$—%—&'(——o": STOP: REM WYŚWIETLENIE OBWODU ELEKTRYCZNEGO
215 REM POWROT DO SYTUACJI POCZĄTKOWEJ
220 POKE 56,40960/256: CLR
230 POKE 56576,151: POKE 53272,21: POKE 648,4
```

Występująca w wierszu 180 instrukcja STOP pozwala na pozostanie w BANKU 2. Kontynuacja programu spowoduje powrót VIC-a do BANKU 0.

W programie wykorzystana jest właściwość mikrokomputera C-64 polegająca na takiej samej adresacji dwóch różnych pamięci. Nowy GENERATOR ZNAKÓW zlokalizowany jest w obszarze 40960 — 45058 pamięci RAM. Pod tymi samymi adresami występuje interpreter Basica w pamięci ROM.

Kolorowanie znaków

W przedstawionych programach wykorzystywany był dotychczas dwukolorowy tryb pracy VIC-a. Pozwala on na wyświetlenie znaku mieszczącego się na 64 punktach ekranu (w ośmiu wierszach i kolumnach) odwzorowujących 64 bity pamięci (jeden bit — jeden punkt ekranu). W dowolnej chwili mieliśmy możliwość określenia koloru ekranu (jednego tła dla wszystkich znaków) i koloru każdego ze znaków występujących na ekranie. Kolor ekranu wybieramy wprowadzając do bajtu 53281 liczbę odpowiadającą wybranej barwie (jednej z szesnastu możliwych). Kolor znaku określany jest na podstawie zawartości najmłodszych trzech bitów odpowiedniego bajtu PAMIĘCI EKRANU (patrz cz.1).

W trybie wielokolorowym wyświetlany znak mieści się na 32 punktach ekranu (w ośmiu wierszach i czterech kolumnach) odwzorowujących również 64 bity pamięci, przy czym parze bitów odpowiada jeden punkt ekranu. Znak na ekranie w obu trybach zajmuje tę samą powierzchnię ekranu. Kolory punktów tworzących znaki są zależne od odpowiadających im par bitów. Para bitów wskazuje skąd należy pobrać kod barwy wyświetlanych punktów według poniższego zestawienia:

00 — bajt 53281;
01 — bajt 53282;
10 — bajt 53283;
11 — najmłodsze trzy bity w bajcie odpowiadają położeniu znaku w PAMIĘCI KOLORU.

Zapisując odpowiednie kody barw w wymienionych bajtach można realizować wielobarwną grafikę w trybie znakowej pracy VIC-a.

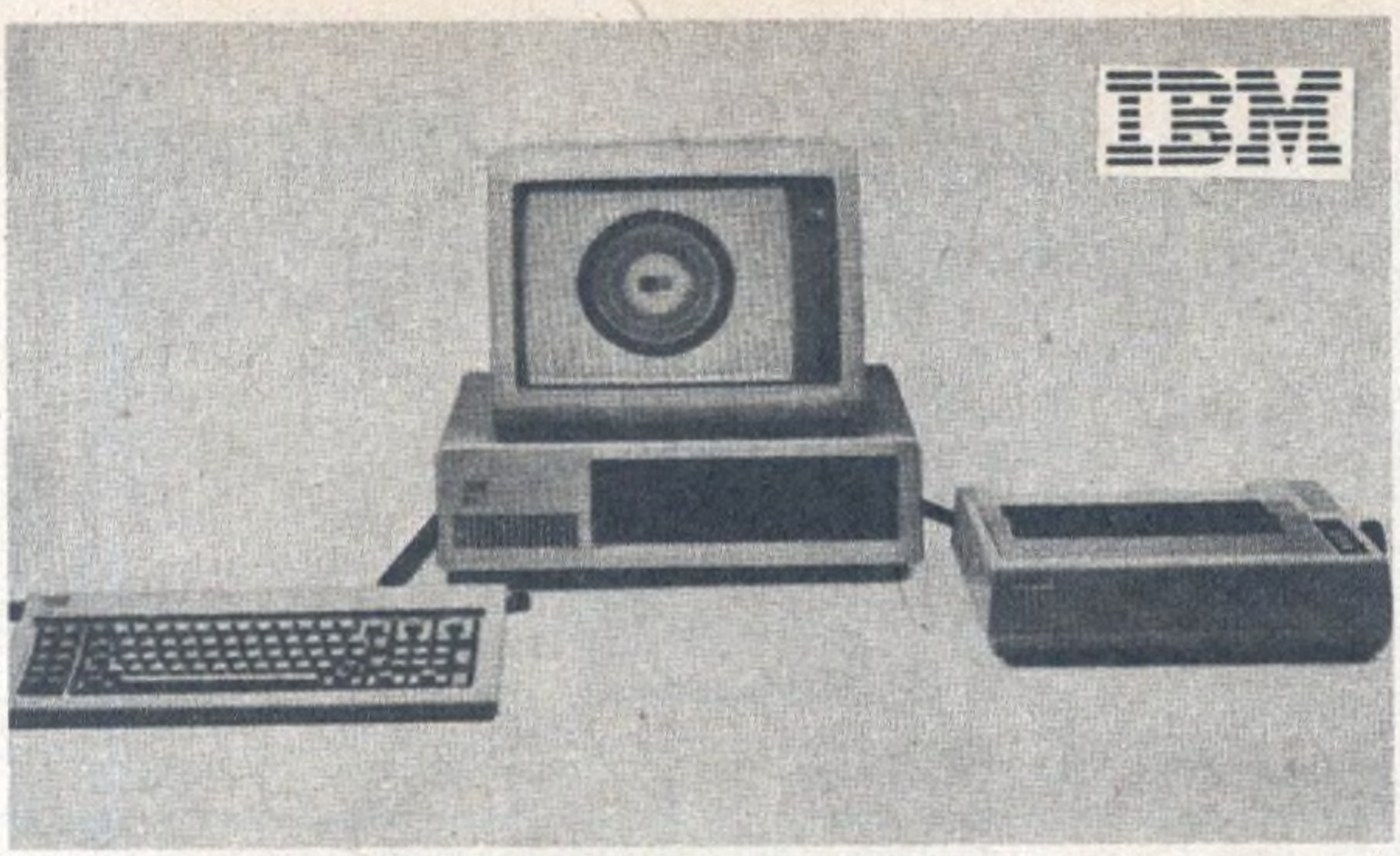
Wykorzystanie trybu wielokolorowego zilustrujemy wprowadzając do ostatniego programu dodatkowe wiersze:

```
190 POKE 53282,15: POKE 53283,0: REM WYBÓR KOLORÓW
200 POKE 53270, PEEK (53270) OR 16: STOP: REM PRZEŁĄCZENIE VICA NA TRYB WIELOKOLOROWY
210 POKE 53270, PEEK(53270) AND 239: STOP: REM POWRÓT DO TRYBU DWUKOLOROWEGO
```

O programowaniu grafiki o dużej rozdzielczości przeczytać będzie można w następnej części.

Sławomir WASIELEWSKI

Rys. Michał Przybyłowski



(1) Mikrokomputery IBM PC

Cykl artykułów o mikrokomputerach personalnych firmy IBM stanowi próbę zaprezentowania ich architektury, parametrów technicznych i oprogramowania. Próba ta poparta jest pewnym doświadczeniem autora, zdobytym w czasie pracy z tym ciekawym systemem.

Określenie komputer personalny (Personal Computer), lub komputer osobisty używane jest w odniesieniu do sprzętu, który może być wykorzystywany profesjonalnie. Jak dynamicznie kształtuje się światowy standard komputera personalnego możemy prześledzić właśnie na przykładzie firmy IBM.

Komputery **IBM PC** pojawiły się na rynku pod koniec 1981 roku. Jako jednostkę centralną wykorzystywano w nim 16-bitowy mikroprocesor INTEL 8088. Komputer posiadał pamięć operacyjną RAM 64 Kb, pamięć stałą ROM 40 Kb oraz pamięć zewnętrzną na dyskach elastycznych, zwanych dyskietkami (Diskette), o wymiarach 5 1/4 cala (133 mm) i czasie dostępu do informacji rzędu kilkudziesięciu milisekund.

Na początku 1983 roku pojawiła się wersja **IBM PC/XT**. Mikrokomputer wyposażono w większą pamięć operacyjną, o pojemności 128 Kb oraz umożliwiono rozbudowanie jej do pojemności 640 Kb. Obok pamięci masowej na dyskach elastycznych, o gęstości zapisu 184 320 bajtów (znaków informacji) lub 368 640 bajtów, zastosowano tzw. dysk stały (Hard Disk) o pojemności 10 Mb, później 20 Mb. W mikrokomputerze wykorzystano, obok jednostki centralnej 8088, koprocesor arytmetyczny INTEL 8087. Połączenie mikroprocesora 8088 z koprocesorem daje programiście jeden wspólny procesor tzw. interprocesor, o automatycznie skoordynowanych funkcjach. Koprocesor pracuje równolegle z procesorem głównym, wykonując te rozkazy, pobierane z pamięci programu, które są dla niego przeznaczone.

W połowie 1984 roku pojawiła się kolejna wersja **IBM PC/AT**, wyposażona w nową jednostkę centralną INTEL 80286 i koprocesor arytmetyczny INTEL 80287. Pamięć operacyjna ma pojemność 256 Kb i posiada możliwość rozszerzenia do 1 Mb. Mikrokomputer posiada dysk stały 20 Mb (lub większy) oraz mechanizmy dyskietkowe; jeden pozwalający na osiągnięcie na dyskietce 5 1/4 cala gęstości zapisu 1,2 Mb, a drugi o gęstości 368 Kb. IBM PC/AT posiada inne udoskonalenia.

W bieżącym roku zaprezentowano najnowszą wersję **IBM PC/RT** (lub **IBM 6150**). W mikrokomputerze zastosowano 32-bitowy procesor o architekturze RISC (Redunded Instruction Set Computer), dużo szybszy od jednostek centralnych stosowanych w poprzednich wersjach oraz koprocesor AT, umożliwiający wykorzystanie programów z IBM PC/AT. Pamięć operacyjna może mieć pojemność od 1 do 4 Mb. Pamięć masowa to dysk stały 210 Mb oraz dyski elastyczne, o gęstości zapisu 1,2 Mb.

Porównując wersję IBM PC/RT z pierwszą wersją IBM dochodzimy do wniosku, że jest to olbrzymi skok możliwości technicznych sprzętu, niewiarygodnie wysoko podnoszący standard komputera personalnego. Rozwiązania sprzętowe tworzą z IBM PC/RT komputer o bardzo dużych mocach obliczeniowych.

O wartości komputerów IBM PC świadczą nie tylko ich parametry techniczne. Można uznać, że sprzęt (Hardware) jest „drugoplanową" cechą tych komputerów. Cechą najważniejszą, decydującą o tym, że IBM PC opanowały już praktycznie wszystkie dziedziny życia, jest ich bardzo bogate oprogramowanie (software).

Mikrokomputery IBM PC/XT dysponując w typowej konfiguracji pamięcią operacyjną 640 Kb, pamięcią zewnętrzną na dysku stałym i dyskach elastycznych, stanowią uniwersalny system obliczeniowy, o dość dużej mocy obliczeniowej. Moc ta może być zwiększana poprzez utworzenie sieci, z reguły 16- lub 32- komputerowej. Na komputerach IBM PC/XT skupimy właśnie naszą uwagę w dalszych numerach „IKS-a".

W mikrokomputerach PC/XT zastosowano dyskowy system operacyjny (DOS — Disk Operating System), będący bogatym i uniwersalnym systemem, pozwalającym na budowę finezyjnych programów użytkowych. Wykorzystywane są dwie implementacje tych systemów: MS-DOS (wersje 2.0, 2.11, 3.10) oraz PC-DOS (wersja 3.0). Mikrokomputer może pracować również pod systemem operacyjnym CP/M.

Wersja PC/RT posiada system operacyjny IBM AIX, umożliwiający pracę wielodostępną i wieloprogramową, wzorowaną na wersji V systemu UNIX. Innym bardzo ciekawym systemem jest MS-WIN DOS, zorientowany graficznie pakiet oprogramowania, komunikujący się z użytkownikiem za pomocą okien z zestawem funkcji reprezentowanych przez piktogramy.

Oprogramowanie do PC/XT zawiera, oprócz interpretera BASIC-u, kompilatory BASIC-u, C, COBOL-u, FORTRAN-u, PASCAL-a. Z języków, posiadających implementację na PC/XT, kolejne to: MODULA-2 LOGO, FORTH. Do tworzenia oprogramowania w języku procesora służy MACRO ASSEMBLER. Istnieje wiele programów wspomagających tworzenie oprogramowania użytkowego, są to edytory czy programy, jak np. SIDEKICK, na stałe rezydujące po ich wpisaniu do pamięci operacyjnej, a służące do podręcznych kalkulacji i notatek.

Istnieje wiele zintegrowanych pakietów programowych, umożliwiających całą gamę zastosowań PC/XT — zaczynając od komputerowego wspoma-

➧

gania projektowania inżynierskiego (CAE — Computer Aided Engineering czy CAD.— Computer Adided Desing). wspomagania produkcji, a kończąc na programach do prowadzenia ewidencji i statystyk w gabinetach lekarskich, czy obsługujących zawody sportowe.

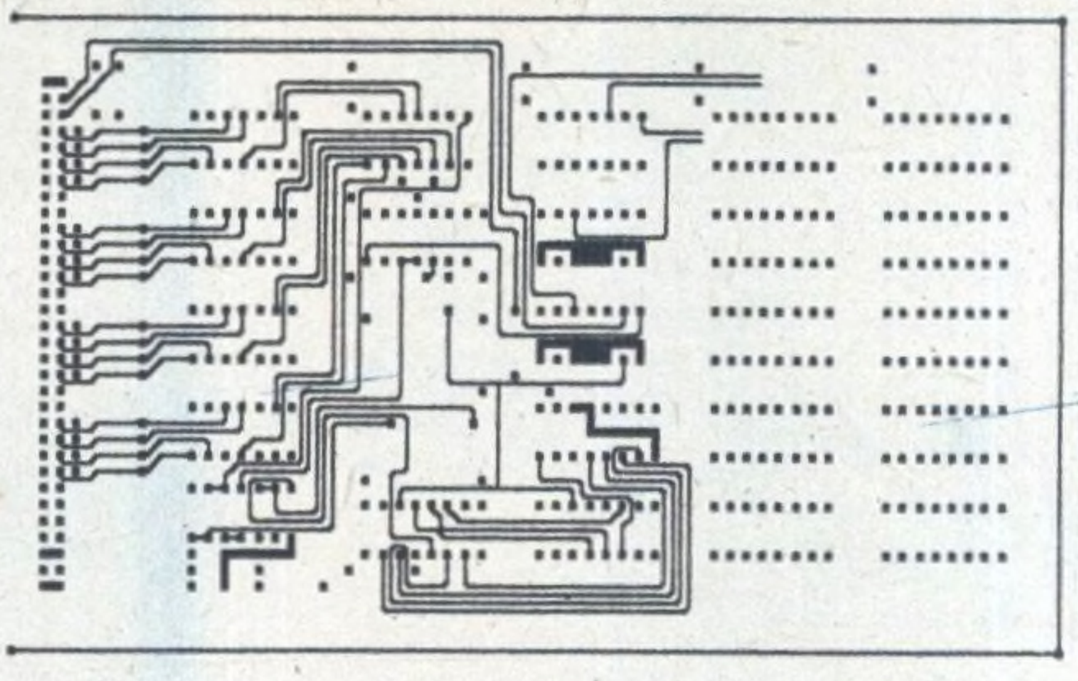
Wśród programów wspomagających projektowanie warto wymienić: sam — ARTWORK, program umożliwiający projektowanie dwustronnych płytek elektronicznych — przykładowe wydruki przedstawiono na rys. 1 i rys. 2; ANALOP i NODAL, programy do analizy obwodów mikrofalowych. Do obliczeń całek lub rozwiązywania równań różniczkowych może posłużyć mu — SIMP (mu — MATH).

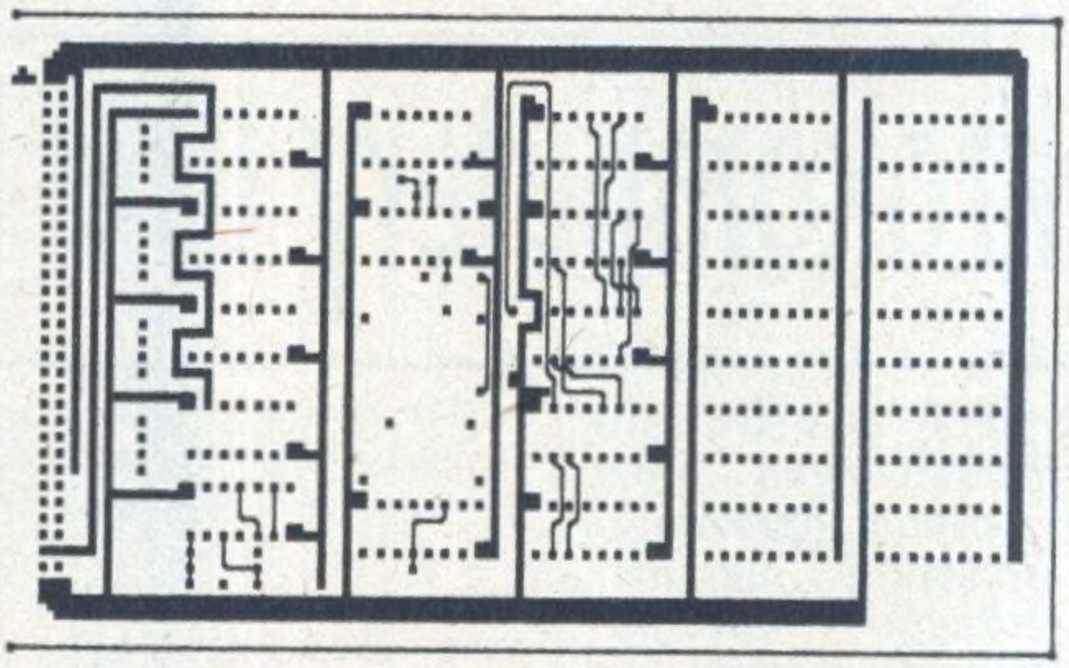
Inna grupa to programy typu baza danych, wśród nich: DBASE II, DBASE III, RBASE 4000 lub im podobne, takie jak: SYMPHONY czy LOTUS.

Następna grupa to programy służące do edycji tekstów np. WORDSTAR i WORDSTAR 2000 + (do pracy w sieci) lub programy wspomagające sporządzanie dokumentacji technicznej.

Kolejna grupa to programy graficzne, takie jak: ACAD, PAINT BRUSH, GEM, ENERGRAPHICS, 3 DESIGN.

Tak w skrócie wygląda najważniejsza, dostępna dla autora, część oprogramowania mikrokomputerów IBM PC/XT. Oczywiście zgodnie z zasadą „kompatybilności w górę" większość programów może pracować na wersji PC/AT i prawdopodobnie na PC/RT. Wersja IBM PC/RT, dopiero po opracowaniu nowego oprogramowania, zdecydowanie zwielokrotni swoje możliwości w stosunku do modeli XT i AT.





Rys.1 i 2 — Przykładowe wydruki płytek elektronicznych projektowane za pomocą programu SM-ARTWORK. (Skala 1:2)

Nie powinniśmy zapomnieć, że mikrokomputery IBM PC nie mogłyby znaleźć zastosowań profesjonalnych bez odpowiednich urządzeń graficznych, takich jak drukarki matrycowe czy plottery, służące do sporządzania rysunków, wykresów itp.

Do charakterystyki oprogramowania będziemy wracać w kolejnych numerach, natomiast w następnym numerze „IKS-a" poznamy architekturę mikrokomputera IBM PC/XT.

(sierpień 1986 r.)

Jacek WOJTALA

BIBLIOTEKA

Z listów do redakcji wynika, że „IKS-a" czytają: uczniowie, studenci i specjaliści — krótko: „IKS-a" czytają Ci, których łączy jedna wspólna cecha: ciekawość (Co komputer może? Jak działa? Jaki będzie?...).

Wychodząc naprzeciw tym, którym nie wystarcza „tylko palcowanie" rozpoczynamy nowy dział.

Mikrorecenzje będą opisami książek nadesłanych nam. Zaczynamy od Wydawnictw Komunikacji i Łączności.

1) **„24 proste układy elektroniczne do samodzielnego wykonania".** (cena 200 zł). Poziom hobbysty-majsterkowicza. Para autorska: Maria i Wojciech Nowakowscy szczegółowo opisują domowe urządzenia elektroniczne: alarm włamaniowy, interkom, minutnik... zasilacz do akumulatorów.

Autorzy bardziej niż zwięźle opisują zalety urządzeń, natomiast bardziej niż dokładnie konstrukcję, a w szczególności nowoczesną technologię (świadectwem ich dokładności jest powtórzenie na końcu rysunków obwodów drukowanych w skali 1:1)

* * *

2) **„PL/I dla inżynierów"**

„Elektronizacja — przewodnik zawodowy (zeszyt 22) autorstwa Jana Bieleckiego — przeznaczony jest jedynie dla profesjonalistów. PL/I zdobywa sobie coraz więcej użytkowników. W Polsce pojawił się wraz z komputerem IBM/369/379 i systemem RIAD, który jako Jednolity System M.C. zdobywa sobie rynek krajów socjalistycznych.

ciąg dalszy na str. 31

Rys. Michał Przybyłowski

TRADYCJA TO NASZ OBOWIĄZEK

Rozmawiamy z dyrektorem Oddziału INTERNATIONAL COMPUTERS LIMITED w Polsce — JANEM J. KLUKIEM

— Mówią o Was potentaci...

Jan J. Kluk: — Kto mówi?

— Fachowcy i Ci, dla których ICL oznacza po prostu dobry komputer.

J.J. Kluk: — Na tę opinię pracujemy już ponad osiemdziesiąt lat.

— I chyba nie tylko na nią.

J.J.K.: — Jesteśmy obecnie największą brytyjską firmą komputerową, posiadamy swoje oddziały w 80 krajach, na pięciu kontynentach. Zatrudniamy 23 500 osób, a obroty firmy przekraczają miliard funtów rocznie.

— W Polsce ICL obecny jest od 20 lat — jakie są tego efekty?

J.J.K.: — Dokładnie od 25 lat działa Biuro Handlowe ICL, a efektem jego pracy jest działanie ponad 400 instalacji systemów komputerowych ICL i ICL — pochodnych.

— Całe zastępy polskich fachowców wychowały się na ODRACH.

J.J.K.: — ODRA powstała na bazie ICL 1900. I oczywiście w tym przypadku liczył się nie tylko osprzęt (hardware), ale równie bogate oprogramowanie (software).

— Czy nie sądzi Pan, że właśnie ten pierwszy kontakt większości naszych informatyków z ICL, powoduje, że tak często decydują się w przyszłości na wasz sprzęt, który jednak nie poddał się światowemu standardowi IBMa?

J.J.K.: — W interesach nikt nie kieruje się sentymentami. Jeśli na nasze systemy zdecydowały się stocznie w Gdańsku, Gdyni, Szczecinie i Ustce, Huta im. Lenina, Zakłady Metalowe im. Cegielskiego, banki, centrale handlu zagranicznego — lista ta jest oczywiście niepełna — i ciągle realizowane są nowe kontrakty, to nie z sentymentu — tu liczy się jakość.

— I co jeszcze?

J.J.K.: — Nasze atuty to wiarygodność i elastyczność działania. Wymusza to stosowanie najnowszych technologii przy konstruowaniu sprzętu i tworzeniu oprogramowania. Musimy również budować urządzenia kompatybilne zarówno z IBM, jak i ze sprzętem krajowym, takim jak na przykład ODRA. Nowy komputer ICL, nie może wprowadzać organizacyjnego zamieszania i nie może tworzyć sytuacji, w której działające dotychczas systemy informatyczne będzie trzeba przeprogramowywać. Poza tym coraz częściej proponujemy rozwiązania kompleksowe. To znaczy sprzęt i specjalistyczne

➧

Rodzina mikrokomputerów ICL PC serii QUATTRO+ obejmuje cztery modele M19, M39, M49, M59 różniące się pojemnością pamięci zewnętrznej. Do jednostki centralnej ICL PC QUATTRO+ można podłączyć jednocześnie cztery stanowiska (monitor monochromatyczny lub kolorowy, klawiatura, myszka). Mikrokomputery te mogą współpracować z drukarkami mozaikowymi, laserowymi i termicznymi, plotterami, czytnikami taśm, czytnikami kodów paskowych, kasami sklepowymi, modemami i innymi urządzeniami dołączanymi poprzez interfejs RS232C/V24. QUATTRO+ pracuje pod systemem operacyjnym Multiuser Concurrent CP/M-86 3.1.8 lub Concurrent DOS 4.1. Systemy te umożliwiają emulację MS-DOS — a to pozwala na realizację wielu programów z IBM PC.

ICL CLAN jest 32-bitowym, szesnastostanowiskowym mikrokomputerem pracującym pod UNIXem. Trzy jego modele: 35, 45 i 55 różnią się pojemnością pamięci zewnętrznej, ilością stanowisk, które mogą obsłużyć i oczywiście ceną. Może współpracować podobnie jak PC QUATTRO z drukarkami mozaikowymi, termicznymi i laserowymi. System operacyjny zawiera ponad 150 programów standardowych realizujących między innymi: sortowanie, wyszukiwanie informacji. Oprogramowanie narzędziowe stanowią: język C, Fortran 77, SMC Basic, SVS, Pascal, Level II Cobol.

oprogramowanie, takie jak systemy magazynowe czy finansowe w oparciu o komputery osobiste do wielkich systemów komputerowych Serii 39.

— Czy nie jest to zbyt optymistyczne?

J.J.K.: — Jeśli coś zwyczajnie nawali, a to choć rzadko, to przecież się zdarza, mamy serwis gwarancyjny, pogwarancyjny i niezłą kadrę specjalistów. W Polsce mamy również własny skład konsygnacyjny co znacznie ułatwia pracę. Poza tym nasz klient wie, że w przyszłości kupi u nas coś nowoczesnego, utrzymanego w dotychczasowym standardzie. Nasze wyroby idą do przodu — nasi kontrahenci mają taką pewność.

— Ale nie mają pewności, że będzie ich na to stać.

J.J.K.: — ICL nie jest przeznaczony dla amatorów.

— Zawodowcy również nie zawsze mogą sobie pozwolić na takie inwestycje.

J.J.K.: — Zdajemy sobie z tego sprawę.

— I co?

J.J.K.: — I coraz częściej musimy dbać o to, aby nasz potencjalny klient miał za co nas kupić. To jest cena utrzymania się na tym rynku. Pracownicy naszego biura muszą poszukiwać towarów, które dadzą się sprzedać w drugim obszarze.

— Tymczasem coraz więcej firm decyduje się na popularny sprzęt tańszy od ICLa.

J.J.K.: — Ale nie są to rozwiązania kompleksowe. Obecnie np. AMSTRAD to rozwiązanie doraźne. Nie dorównuje nam ani sprzętowo, ani programowo. Jest jedynie tańszy.

— I to jest argument.

J.J.K.: — Przynajmniej na razie. Na Zachodzie firmy przeszły już przez ten etap. Tam dobre są jedynie rozwiązania takie jakie my proponujemy. Przez Polskę przechodzi fala masowej komputeryzacji, ale są to głównie zastosowania indywidualne. Ludzie kupują sprzęt dlatego, że mogą go kupić — a nie z konieczności wprowadzania informatyki. Komputer w domu to rozrywka. Ale kiedy trzeba będzie naprawdę stosować informatykę potrzebny będzie lepszy sprzęt.

— Zatem uważa Pan, że i tak do ICLa wrócą?

J.J.K.: — Nie chcę być zarozumiały ale uważam, że tak.

— Co Pan im zaproponuje?

J.J.K.: — To zależy oczywiście od zastosowania. Dziś proponujemy ICL — Clan — 16-stanowiskowy mikrokomputer pracujący pod systemem operacyjnym UNIX. PC Quattro — jedno- lub wielostanowiskowe i najnowocześniejszy nasz wyrób — DRS 300.

— Nie wszystko to jednak może trafić na nasz rynek...

J.J.K.: — Funkcjonują jeszcze zakazy — konsekwencje amerykańskiego embarga. Dotyczy ono między innymi komputerów o pojemności pamięci większej 2.4 Mb. ICL próbuje walczyć w Waszyngtonie o zniesienie tych ograniczeń. To leży w naszym interesie — przecież tradycyjnie gwarantujemy klientom najwyższy poziom — to nasz obowiązek.

— Dziękuję za rozmowę.

Rozmawiał: Wiesław CETERA

Po raz drugi AGPOL zorganizował międzynarodową wystawę „Home Office Personal Computer 1987". W dniach 10—13 lutego br. w Pałacu Kultury i Nauki w Warszawie 70 wystawców z 11 krajów zaprezentowało największą w krajach RWPG ekspozycję techniki komputerowej. Mimo że wystawa miała charakter handlowy i była przeznaczona dla specjalistów, cieszyła się ogromnym powodzeniem wśród wszystkich zwiedzających.

Tym, którzy nie mogli zapoznać się z wystawą „Komputer '87", dedykujemy nasze rozmowy i spostrzeżenia zobrazowane błyskiem flesza.

Na ekranie „Komputera'87"

„Prawda" łamana komputerowo...

MERA-BŁONIE
Krzysztof Woliński

— To przyszłość, miejmy nadzieję, że niedaleka. Obecnie w naszej produkcji dominują drukarki. Po odpowiednim przystosowaniu mogą współpracować ze Spectrum, z Commodore i Atari. Mamy kłopoty z tzw. wsadem dewizowym. Dlatego też nasza produkcja nie przekracza 30 tys. sztuk rocznie. Drukarki prezentują przyzwoity europejski poziom. Ponad 80 proc. jest wysyłane na eksport. W kraju łatwiej dostać je np. za lepik, (jeśli akurat jest potrzebny zakładowi) niż za pieniądze. Współpracujemy z zakładami elektronicznymi w Związku Radzieckim. MRAMOR powstał właśnie z tej kooperacji. Pod tym kryptonimem kryje się system komputerowego składu i łamania gazety. I będziemy to produkować. Projektem interesuje się redakcja dziennika „Prawda".

Fotografował: Jan Zelman

„Krówki"... i mikroprocesory...

ALMA — Przedsiębiorstwo Polonijno-Zagraniczne Zakład Produkcji i Oprogramowania Systemów Komputerowych
Wojciech Lipko

— Handlujemy słodyczami, ale nasza oferta obejmuje również komputerowe systemy użytkowe dla szkół i zakładów pracy. Dlaczego aż taka duża rozpiętość — od cukierka po mikroprocesor? Za dolary zdobyte na eksporcie łakoci możemy kupić potrzebny sprzęt i części w dalekiej Azji. Komputery montujemy w Polsce, tworząc na ich bazie sieci lokalne i wielodostępne. Oferujemy systemy przetwarzania informacji dotyczące gospodarki magazynowej oraz wspomagania prac biurowych. Naszym klientom proponujemy sprzęt i usługi w granicach od 4 do 8 milionów złotych. Wszystko zależy od wielkości zamówienia i zakresu wykonywanych prac. Doradzamy, szkolimy obsługi, wykonujemy projekty techniczne i oprogramowanie. Mamy własny serwis. Ceny umowne.

Chcemy wtargnąć do szkół...

ACORN
Komputerowa firma angielska
M. Neugarten

— Firma w Polsce pojawiła się... w grudniu 1986 roku. Zdążyła się już zaaklimatyzować na Śląsku. Na wystawie prezentuje komputery Master 128 i Compact 128. Jest to jedyna ekspozycja ogólnodostępna dla zwiedzających (dzieci tłuką w klawiaturę kilku komputerów od rana do wieczora). Ciekawostka, jaką po raz pierwszy prezentujemy, to płyta wizyjna z odczytem laserowym. Obraz Anglii przedstawiono w postaci 200 tys. fotografii, planów i schematów oraz 150 tys. stron tekstu. A wszystko to na jednej niewielkiej lśniącej płycie. Wyszukiwanie informacji trwa sekundy.

Nasze komputery można porównywać do wyrobów tak znanej firmy jak Amstrad. Nie przegrywają w tej rywalizacji, raczej są lepsze. W Anglii jesteśmy najlepsi w produkcji komputerów edukacyjnych. Zrobimy wszystko, aby jak najszybciej pojawiły się w polskich szkołach.

Nie musimy się reklamować...

GERPOL
Przedsiębiorstwo Polonijno-Zagraniczne
Marek Siwiński

— Nasza oferta jest obszerna. Obejmuje zarówno samostery do jachtów pełnomorskich, jak i elektronikę dla rolnictwa. Współpracujemy z ośrodkami naukowymi we Wrocławiu i Gdańsku. Największym sukcesem firmy jest opracowanie systemu METEOSAD, w którym dane meteorologiczne określa się za pomocą satelity. Zajmujemy się również komputeryzacją w przedsiębiorstwach. Dotyczy to takich działów jak gospodarka magazynowa, finanse, kadry itp. Ofert jest coraz więcej.

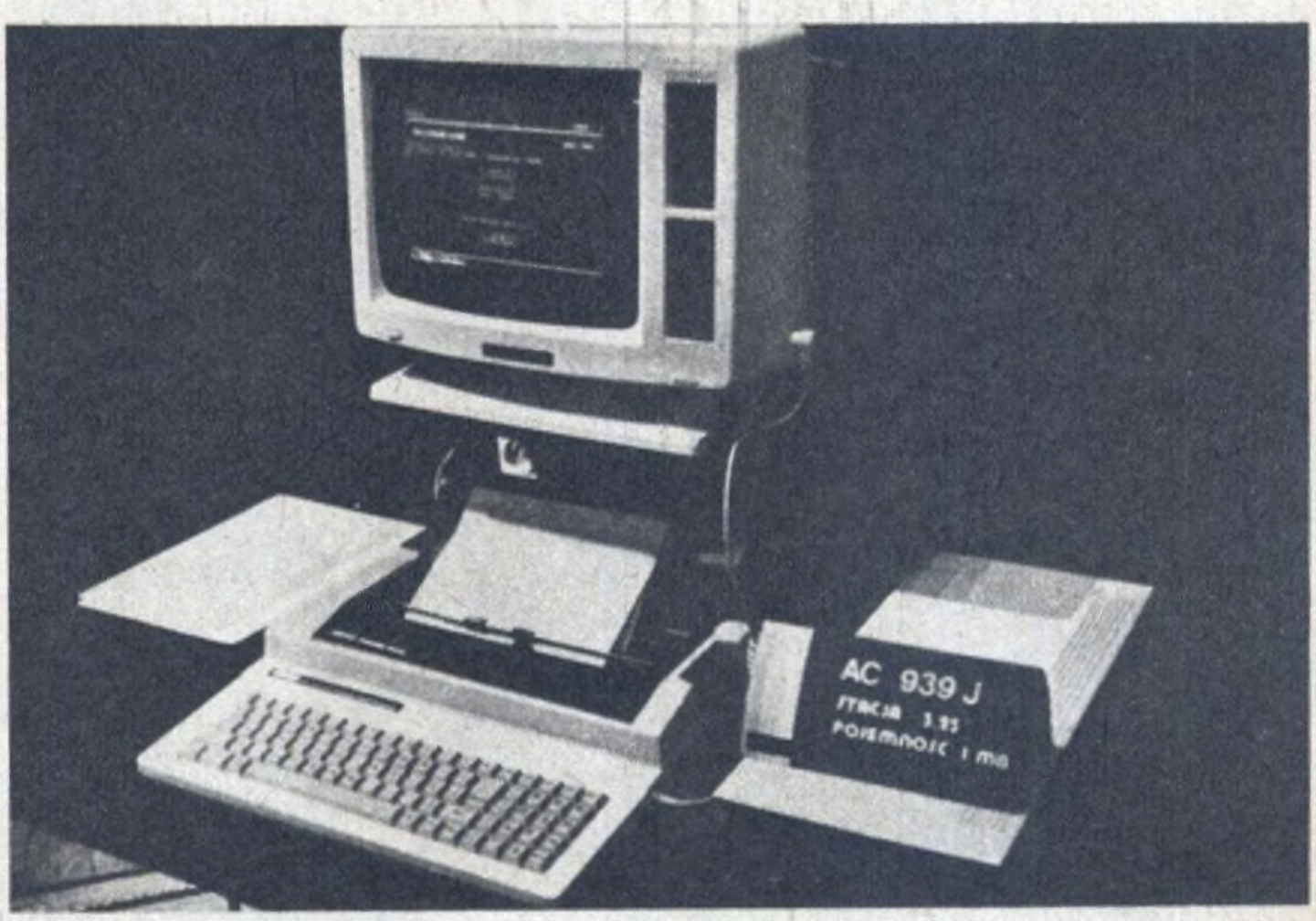
Komputery wyciskane z porzeczek...

INTEGRATED MICRO COMPUTERSETS
Szwajcarska Spółka Akcyjna
Elżbieta Duchowicz

— Handlujemy w zasadzie dużymi komputerami profesjonalnymi. Na polski rynek staramy się dostarczyć najnowsze konstrukcje, m.in. posiadamy licencje sprzedaży Fortune System (komputery 32- i 16-bitowe). Współpracujemy tylko z przedsiębiorstwami państwowymi, wymieniając towar za towar. Wiele firm patrzyło na nas z politowaniem, gdy komputery wymienialiśmy na soki porzeczkowe. Po pomyślnym zrealizowaniu przez nas tego typu umowy, inni też rozglądają się za podobnymi kontraktami. Przełamaliśmy pewien stereotyp myślenia. Uważam to za swój osobisty sukces. Chcemy rozszerzyć listę towarów, które można wymieniać na komputery. Jestem pełna optymizmu i wierzę, że pomogą skomputeryzować kraj, w którym się urodziłam.

Notował:

Ryszard ROGOŃ

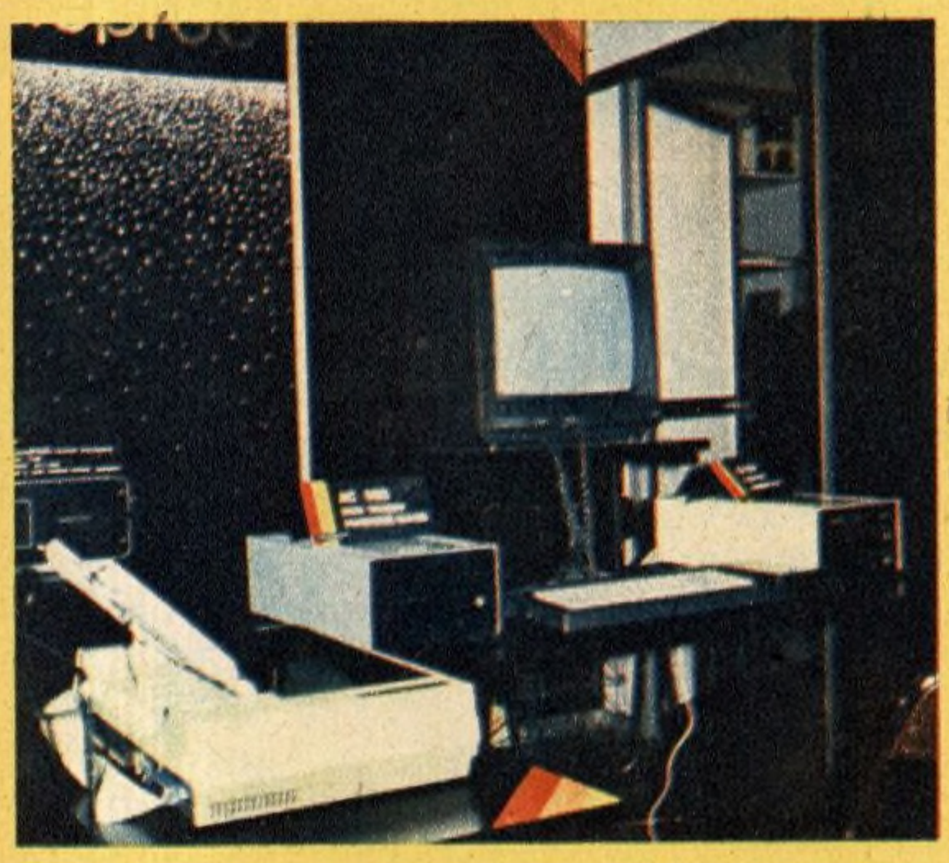


„Komputer '87"

Foto: Jan Zelman

Początek był nieciekawy

ale w miarę przybywania eksponatów
...było na co popatrzeć



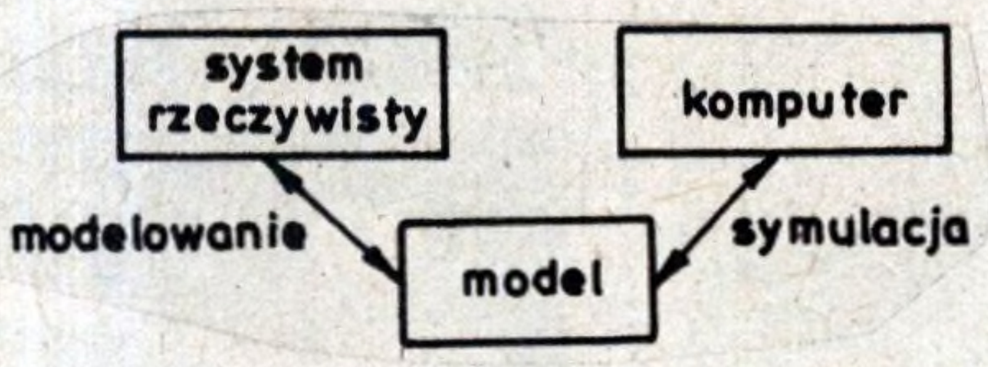
...wykorzystać do pracy, zabawy a przede wszystkim
...porozmawiać z przedstawicielami firm, uzgodnić warunki
i... kupić!

Tekst i foto:
Ryszard Rogoń

UDAWANIE RZECZYWISTOŚCI

SYMULACJA jest jedną z ważniejszych i ciekawszych możliwości wykorzystania mikrokomputera. Pozwala ona na przeanalizowanie zjawisk, które w rzeczywistym systemie przebiegają zbyt szybko lub wolno lub też wtedy, kiedy eksperymentowanie z rzeczywistym systemem byłoby zbyt kosztowne, niebezpieczne czy w ogóle niemożliwe (np. niemożliwe jest dowolne kształtowanie podaży i popytu w rzeczywistym systemie ekonomicznym, co łatwo można wykonać przy pracy z modelem; względy moralne uniemożliwiają manipulowanie mózgiem człowieka, natomiast eksperymentowanie z modelem nie stwarza takich ograniczeń). Ponadto komputerowe eksperymenty są całkowicie niedestrukcyjne (nie niszczą systemu rzeczywistego), zapewniają powtarzalność i umożliwiają łatwą zmianę parametrów modelu.

Aby przeprowadzić symulację należy skonstruować model systemu rzeczywistego. **Proces konstruowania modelu nazywamy modelowaniem**. Pojęcie „modelowanie i symulacja" oznacza zespół działań związanych z konstruowaniem modelu systemu rzeczywistego i symulowania go na komputerze. Z definicji tej wynika, że możemy wyróżnić trzy elementy: system rzeczywisty, model i komputer oraz dwie relacje pomiędzy tymi elementami: modelowanie i symulację. Związek ten przedstawiony jest na rys. 1.



Systemem rzeczywistym nazywamy interesującą nas część świata rzeczywistego.

Modelem jest zbiór informacji o systemie zebranych w celu jego zbadania. Model powinien nie tylko odtwarzać zaobserwowaną reakcję systemu rzeczywistego, ale również wiernie odzwierciedlać sposób powstawania tej reakcji w systemie rzeczywistym. Jest to postulat tzw. zasadności strukturalnej modelu.

Istnieje kilka klasyfikacji modeli. Najważniejsze z nich związane są:

— z podstawą czasu, w której występują zdarzenia (modele ciągłe w czasie, modele dyskretne w czasie);

— z zakresami wartości przyjmowanych przez zmienne opisowe (modele ciągłe w stanie, modele dyskretne w stanie);

— ze zmiennymi losowymi (modele determinityczne, modele stochastyczne).

— **Model jest ciągły** w czasie, jeżeli opis czasu dokonuje się w sposób ciągły, tzn. zegar czasu modelu zwiększa swoją wartość o stałe przyrosty.

— **Model jest dyskretny** w czasie, jeżeli czas płynie skokowo, tzn. zegar czasu modelu zwiększa się okresowo o różne wielokrotności umownej jednostki czasu.

— **Model jest ciągły w stanie**, jeżeli jego zmienne przyjmują wartości z pewnego przedziału liczb rzeczywistych (zmieniają się w sposób ciągły).

— **Model jest dyskretny w stanie**, jeżeli jego zmienne przyjmują wartości dyskretne.

— **Model nazywamy deterministycznym**, jeżeli w jego opisie nie występują zmienne; w przeciwnym przypadku model nazywamy stochastycznym lub probabilistycznym.

W niniejszym artykule przedstawiony zostanie prosty przykład symulacji rozwoju społeczeństwa. Jest to model deterministyczny, ciągły w czasie i dyskretny w stanie. Determinizm tego modelu polega na tym, że prawa rozwoju społeczeństwa opisane są ścisłymi matematycznymi formułami, bez zmiennych losowych.

Załóżmy, że w pewnym małym społeczeństwie struktura ludności jest następująca:

grupa	wiek	liczba osób
A	0-19	4000
B	20-39	3600
C	40-59	3100
D	60 i więcej	2450

Przyjmijmy również, że wskaźniki śmiertelności i reprodukcji w okresach dwudziestoletnich dla każdej z grup wynoszą:

grupa	wiek	WS (%)	WR (%)
A	0-19	5	32
B	20-39	12	67
C	40-59	18	1
D	60 i więcej	80	0

WS-wskaźnik śmiertelności

WR-wskaźnik reprodukcji

Na podstawie takiego systemu rzeczywistego można łatwo skonstruować model matematyczny. Model ten opisuje w postaci czterech równań liczbę ludności w poszczególnych grupach w kolejnych okresach czasowych, n=1, 2, 3, ... N. Przyjmujemy czas trwania jednego cyklu 20 lat.

Równanie (1)÷(4) tworzą matematyczny model systemu.

1) $A_{n+1} = 0.32\ A_n + 0.67\ B_n + 0.01\ C_n$
(liczba narodzin, czyli liczba osób, które w następnym cyklu zasilą grupę A)

2) $B_{n+1} = 0.95\ A_n$
(95% populacji z grupy A przeżywa i przechodzi do grupy B)

3) $C_{n+1} = 0.88\ B_n$
(88% populacji z grupy B przeżywa i przechodzi do grupy C)

4) $D_{n+1} = 0.82\ C_n + 0.2\ D_n$
(82% populacji z grupy C przeżywa i przechodzi do grupy D, ponadto 20% populacji grupy D przeżywa do następnego cyklu, tj. więcej niż 20 lat).

Warunki początkowe w chwili T =0 są następujące:
$A_0 = 4000$, $B_0 = 3600$, $C_0 = 3100$, $D_0 = 2450$

```
2995 REM program "Symul"
     ########################
     @ Janusz MORBITZER
3000 CLS : BRIGHT 1: PRINT AT 8,
7;"Symulacja dynamiki": PRINT AT
 11,0;"rozwoju  społeczeństwa w
zadanej": PRINT AT 14,0;"ilości
cykli  dwudziestoletnich.": BRIG
HT 0: BEEP .5,10: GO SUB 3500
3020 CLS : BRIGHT 1: PRINT AT 0,
2;"Załóżmy, że struktura  wieku"
: PRINT AT 2,2;"pewnego małego s
społeczeństwa": PRINT AT 4,2;"jes
t  następująca (model de-": PRIN
T AT 6,8;"terministyczny):": BRI
GHT 0
3025 PRINT AT 8,0;"**************
**********************": PRINT AT 9
,0;"*grupa*    wiek    * liczba o
sób*": PRINT AT 10,0;"**********
**********************"
3030 PRINT AT 11,0;"*  A  *     0
-19    *     4000      *": PRINT AT
12,0;"*-------------------------
------*"
3035 PRINT AT 13,0;"*  B  *    20
-39    *     3600      *": PRINT AT
14,0;"*-------------------------
------*"
3040 PRINT AT 15,0;"*  C  *    40
-59    *     3100      *": PRINT AT
16,0;"*-------------------------
------*"
3045 PRINT AT 17,0;"*  D  *60 i
więcej*     2450      *": PRINT AT
18,0;"**************************
******"
3050 GO SUB 3500
3055 CLS : PRINT AT 1,0;"  Załóż
my, że zdolność do repro-": PRIN
T AT 2,0;"dukcji i śmiertelność
w  okresie": PRINT AT 3,0;"20-le
tnim dla  każdej z grup wy-"
3060 PRINT AT 4,0;"nosi (w %% w
odniesieniu  do li-": PRINT AT 5
,0;" czby ludności z  danej grup
y): "
3065 PRINT AT 8,0;"**************
**********************": PRINT AT 9,
0;"*grupa*    wiek    *WS(%) *WR(%
)*": PRINT AT 10,0;"************
**********************"
3070 PRINT AT 11,0;"*  A  *     0
-19    *  5  * 32  *": PRINT AT 1
2,0;"*  B  *    20-39    * 12  * 6
7  *": PRINT AT 13,0;"*  C  *
40-59    * 18  *  1  *"
3075 PRINT AT 14,0;"*  D  *60 i
więcej* 80  *  0  *": PRINT AT 1
5,0;"**************************
****"
3080 PRINT BRIGHT 1;AT 17,4;"WS-
wskaźnik umieralności": PRINT BR
IGHT 1;AT 19,4;"WR-wskaźnik  rep
rodukcji"
3085 GO SUB 3500
3120 REM struktura ludności
3130 READ a,b,c,d
3140 DATA 4000,3600,3100,2450
3150 REM wskaźnik śmiertelności
3160 READ d1,d2,d3,d4
3170 DATA .05,.12,.18,.8
3180 REM wskaźnik urodzeń
3190 READ b1,b2,b3,b4
3200 DATA .32,.67,.01,0
3210 BEEP .5,10: CLS : PRINT AT
9,2;"Symulacja może być przeprow
a-": PRINT AT 11,2;"dzona dla n
cykli dwudziesto-": PRINT AT 13,
12;"letnich.": PRINT AT 17,3; BR
IGHT 1;"Podaj żądaną liczbę cykl
i:"
3212 INPUT "                  n=";LI
NE n$: LET l=LEN n$: IF l=0 THEN
 GO TO 3212
3214 FOR i=1 TO l: IF n$(i)>="0"
 AND n$(i)<="9" THEN NEXT i: LET
 n=VAL n$: GO TO 3220
3216 GO TO 3212
3220 IF n<=0 THEN GO TO 3212
3230 LET t=0
3250 BEEP .5,10: CLS : PRINT AT
0,0;"----------": PRINT AT 1,0;"|cykl| 0-1
9 |20-39 |40-59| 60+ |": PRINT A
T 2,0;"----------"
3270 PRINT "|";TAB (4-LEN (STR$
t));t;TAB 5;"|";TAB 7;a;TAB 12;"
|";TAB 14;b;TAB 19;"|";TAB 21;c;
TAB 25;"|";TAB 27;d;TAB 31;"|"
3275 IF a=0 AND b=0 AND c=0 AND
d=0 THEN PRINT "----------": BEEP .8,0: PA
USE 50: GO SUB 3500: CLS : PRINT
 BRIGHT 1;AT 12,5;"Społeczeństwo
 wymarło": GO SUB 3500: GO TO 34
50
3280 IF t>=n THEN GO TO 3360
3290 LET a1=b1*a+b2*b+b3*c+b4*d
3300 LET d=INT ((1-d3)*c)
3310 LET c=INT ((1-d2)*b)
3320 LET b=INT ((1-d1)*a)
3330 LET a=INT a1
3340 LET t=t+1
3350 GO TO 3270
3360 PRINT "----------": PRINT : BEEP .8,1
0: GO SUB 3500
3370 GO TO 3450
3450 RESTORE : CLS : PRINT AT 13
,8; BRIGHT 1;"Koniec symulacji":
 BEEP .8,10: GO SUB 3500: GO TO
3000
3500 REM podprogram
3505 PRINT #1;AT 0,4; BRIGHT 1;
FLASH 1;"Naciśnij dowolny klawis
z": PAUSE 0: RETURN
```

Przedstawiony program napisany w języku BASIC na ZX Spectrum pozwala na przeprowadzenie symulacji dla dowolnie długiego okresu. Instrukcje 3000 ÷ 3085 służą do wydrukowania informacji ogólnych o problemie i mogą zostać pominięte. Użytkownik ma możliwość zmiany parametrów modelu społeczeństwa:
(a) początkowej struktury ludności (instrukcja DATA w linii 3140);
(b) wskaźników śmiertelności (instrukcja DATA w linii 3170);
(c) wskaźników urodzeń (instrukcja DATA w linii 3200).

W ten sposób można przeprowadzić prognozy demograficzne dla różnych wartości poszczególnych parametrów modelu. Dla parametrów podanych w artykule otrzymujemy następujący przebieg symulacji:

cykl	0-19	20-39	40-59	60+
0	4000	3600	3100	2450
1	3723	3800	3168	2541
2	3769	3536	3344	2597
3	3606	3580	3111	2742
4	3584	3427	3150	2551
5	3474	3404	3015	2582
6	3422	3300	2995	2472
7	3335	3250	2904	2455
8	3273	3168	2860	2381
9	3198	3109	2787	2345
10	3134	3038	2735	2285
11	3065	2977	2673	2242
12	3002	2911	2619	2191
13	2937	2851	2561	2147
14	2875	2790	2508	2100
15	2814	2731	2455	2056
16	2754	2673	2403	2013
17	2696	2616	2352	1970

Dokładniejszy model można uzyskać, przyjmując krótszy czas trwania jednego cyklu, np. 5 lat.

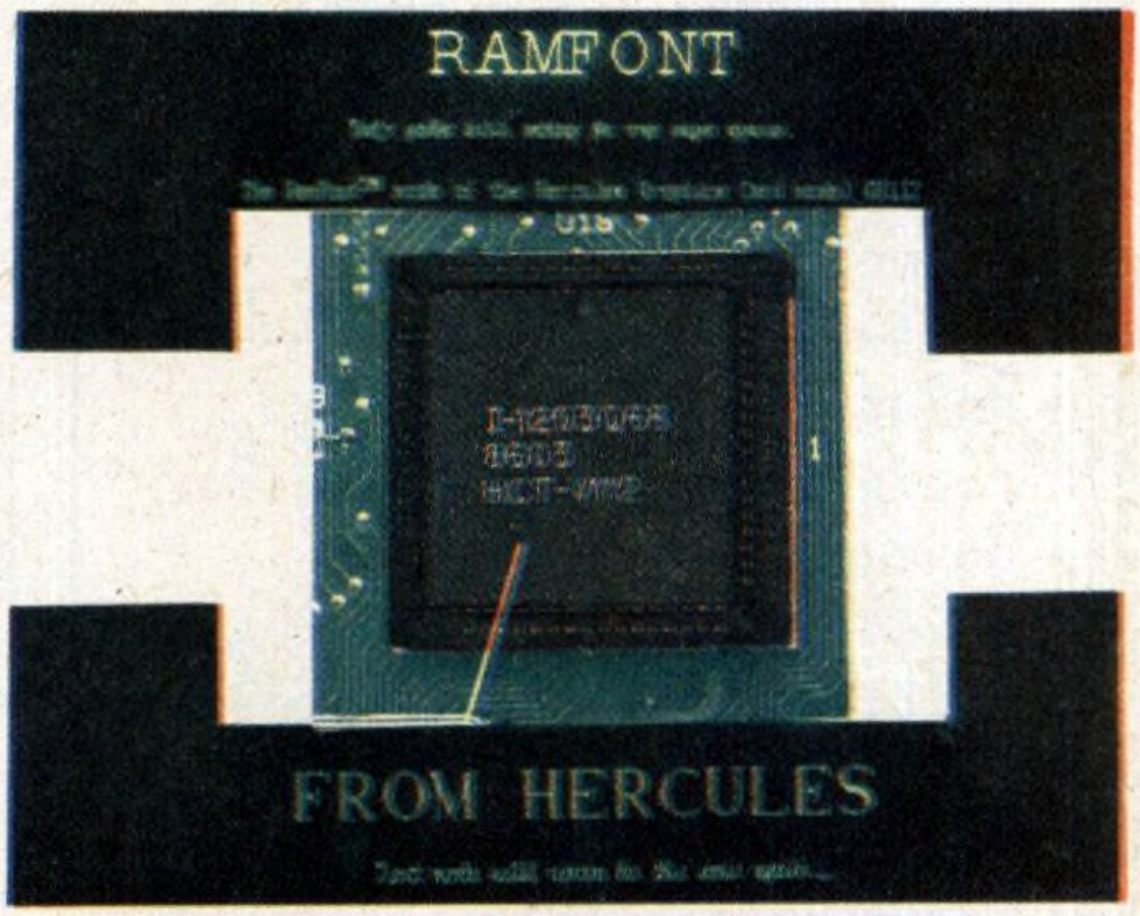
Janusz MORBITZER

H E R C U L E S P L U S

Cztery lata temu pojawiła się na rynku karta graficzna firmy Hercules do mikrokomputera IBM PC , która w krótkim czasie stała się światowym standardem . Karta *HERCULES* posiada dwa tryby pracy: tekstowy i graficzny . Tryb tekstowy charakteryzuje się dużą szybkością , lecz ograniczony jest do 256 znaków . Tryb graficzny pozwala na dowolną liczbę znaków , ale do przetwarzania tekstów jest zbyt wolny . Nowy produkt firmy ,karta *HERCULES PLUS*, posiada dodatkowy , trzeci tryb pracy **RAMFont** . Łączy on w sobie szybkość trybu tekstowego z elastycznością trybu graficznego i umożliwia definiowanie 3072 znaków o różnej szerokości i wysokości . Prace nad nową kartą trwały dwa lata . Głównym ich celem było zaprojektowanie procesora **V112** . Cena karty *HERCULES PLUS* 299 $.

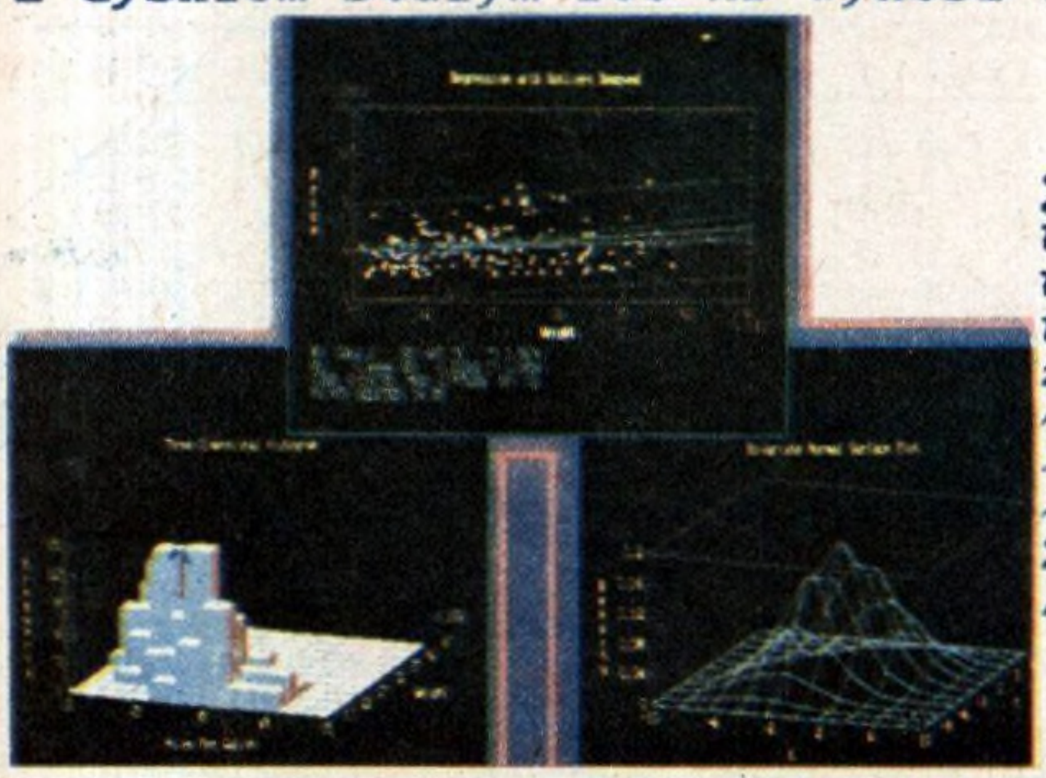
T A N D Y 1 0 2

Firma TANDY wyprodukowała nowy mikrokomputer przenośny , oznaczony symbolem 102 . Posiada on 8-bitowy mikroprocesor 80C85 , pracujący z częstotliwością 2,4 MHz , pamięć zapisywalną RAM 24KB i pamięć stałą 32KB . TANDY 102 posiada wskaźnik na ciekłych kryształach LCD , pozwalający na pracę : w trybie alfanumerycznym w 8 liniach po 80 znaków , w graficznym 240x64 znaki adresowane bezpośrednio . Mikrokomputer posiada klawiaturę QWERTY , złożoną z 56 klawiszy , rozbudowaną o 8 klawiszy funkcyjnych . TANDY wyposażony jest w port szeregowy RS-232 i równoległy oraz interfejs do magnetofonu i modemu . Istnieje możliwość podłączenia przenośnej stacji dysków i rozszerzenia RAM-u o 8KB. Cena TANDY 102 wynosi 299 $.



COMPAQ DESKPRO 386

DESKPRO 386 jest komputerem osobistym kompatybilnym z IBM PC . Zastosowano w nim 32-bitowy mikroprocesor *INTEL 80386*,pracujący z częstotliwością 16 MHz .DESKPRO 386 może posiadać pamięć zapisywalną RAM o pojemności do 14 MB . W mikrokomputerze zastosowano kartę graficzną EGA . DESKPRO 386 posiada stację dysków elastycznych 5 1/4 cala o gęstości zapisu 1,2 MB . Mikrokomputer może mieć dysk stały 40 MB lub 130 MB . DESKPRO wyposażono w dyskowy system operacyjny *DOS* firmy *MikroSoft* . Cena mikrokomputera DESKPRO 386 z dyskiem stałym 130 MB wynosi 9 000 £ .



STATGRAPHICS

Jest to nowy pakiet do IBM PC pozwalający na wizualizację analizowanych danych w formie atrakcyjnych wykresów i diagramów . Zawiera 255 procedur z zakresu analizy regresji, analizy danych eksperymentalnych oraz analizy szeregów czasowych. Współpraca z DBASEIII, LOTUS SYMPHONY, można wykorzystywać zbiory ASCII . Cena 449 $.

opracowali: Jacek Wojtala i Mariusz Jarzębowski

Wyszukiwanie informacji w zbiorze

Wykorzystanie komputera do gromadzenia dużej ilości danych pociąga za sobą konieczność opracowania metod wyszukiwania informacji w zbiorze.

Wyszukiwanie odbywa się zazwyczaj wg pewnej wybranej cechy (lub cech) rekordu, np. nazwiska, tytułu książeczki czy numeru katalogowego. Cechę, wg której realizowane jest wyszukiwanie nazywamy **deskryptorem** lub **kluczem wyszukiwania.**

Zbiór, w którym wyszukujemy informacje może być uporządkowany wg kluczy (np. alfabetycznie) lub nieuporządkowany.

W zbiorze nieuporządkowanym jedyną rozsądną strategią wyszukiwania informacji jest wyszukiwanie proste. Polega ono na sekwencyjnym przeglądaniu zbioru (pobieraniu rekordów) i sprawdzaniu czy natrafiliśmy na żądany rekord (klucz wyszukiwania zgodny z kluczem danego rekordu).

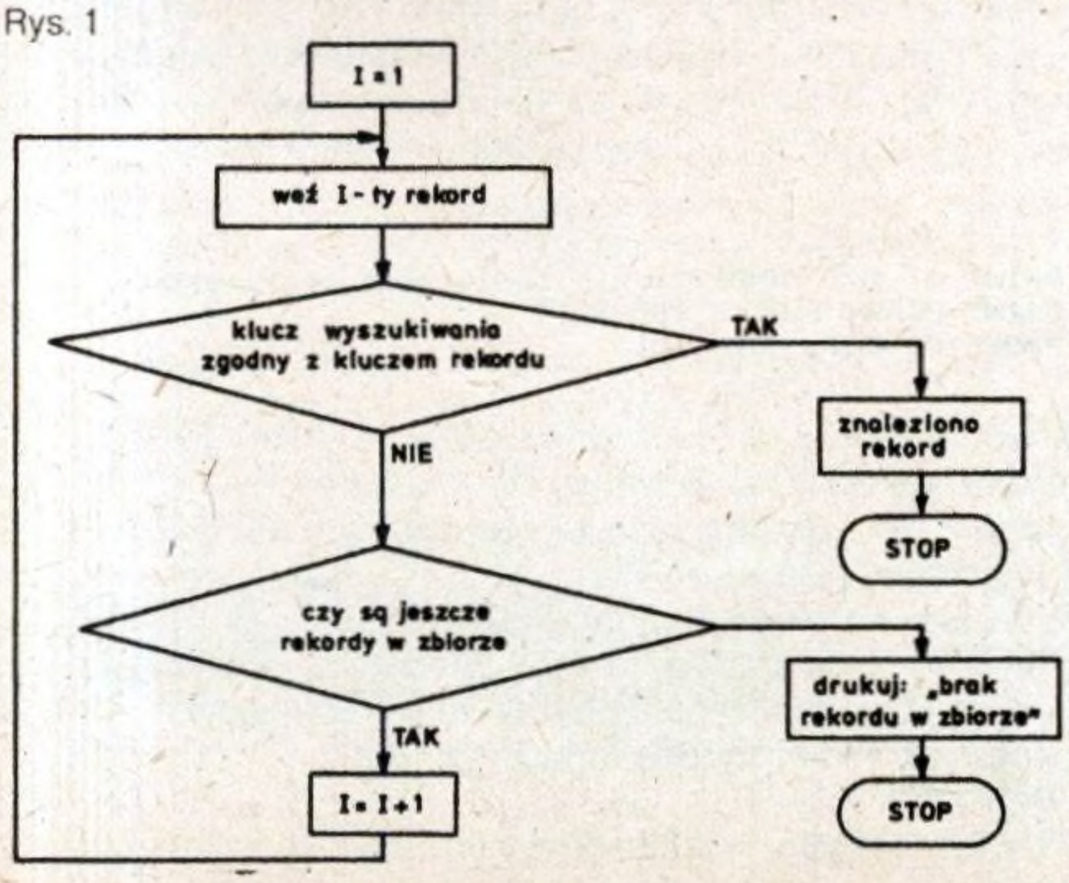
Schemat blokowy takiego wyszukiwania przedstawia rys. 1:

Powyższy schemat przedstawia sytuację, gdy wystarcza nam znalezienie jednego rekordu spełniającego określony warunek. Jeżeli takie rozwiązanie nas nie zadowala, należy sekwencyjnie przeglądnąć wszystkie rekordy w zbiorze.

Pobranie jednego rekordu ze zbioru w celu porównania jego klucza z kluczem wyszukiwania nazywamy **cyklem.** Oszacujmy średnią liczbę cykli niezbędną do odnalezienia informacji w zbiorze metodą wyszukiwania sekwencyjnego. Niech N będzie liczbą rekordów w zbiorze. W najlepszym razie poszukiwany rekord może być odnaleziony w pierwszym cyklu, w najgorszym zaś — w ostatnim, czyli N-tym. Średnia liczba cykli wynosi więc

$$C = (1 + N)/2$$

Rys. 1



Wadą wyszukiwania sekwencyjnego jest jego powolność, zaletą — oprócz prostoty — łatwość aktualizacji zbioru. Zbiór można uzupełniać nowymi rekordami, dopisując je na końcu lub w wolne miejsca; można też usuwać rekordy bez scalania zapisów zakładając, że przy wyszukiwaniu rekordy proste są pomijane.

Jeżeli znane jest uporządkowanie rekordów w zbiorze, możliwe jest zastosowanie szybszych metod wyszukiwania informacji np. metody podziałów połówkowych, zwanej również metodą binarną.

Wyszukiwanie binarne jest przykładem wyszukiwania **dychotomicznego,** tj. procesu, w którym każdy cykl wyszukiwania związany jest z podziałem zbioru na dwie części, z których jedna może być odrzucona. W metodzie tej porównywany jest z kluczem wyszukiwania najpierw klucz rekordu środkowego. Jeżeli wartości kluczy są równe rekord został odnaleziony. Jeżeli wartość klucza rekordu jest większa od wartości klucza szukania, odrzucana jest dolna część zbioru, jeżeli jest ona mniejsza, odrzucana jest górna część zbioru. Odrzuconą część zbioru nazywany „martwą", pozostałą zaś — „żywą". Opisane postępowanie prowadzone jest z kolejnymi „żywymi" częściami zbioru, aż do znalezienia rekordu lub stwierdzenia, że poszukiwanego rekordu nie ma w zbiorze (część „żywa" jest pusta). W każdym cyklu obszar poszukiwań zmniejszany jest o połowę względem poprzedniej jego objętości — stąd też wywodzi się nazwa metody.

Należy zwrócić uwagę, że dokładniej rzecz ujmując w kolejnych cyklach zbiór zmniejszany jest nie o połowę, lecz o połowę —1, gdyż klucz rekordu środkowego był już sprawdzony i w dalszych działaniach może zostać pominięty. Liczba cykli potrzebna do znalezienia rekordu wynosi (dla zbioru o N rekordach) $\log_2 N \leq c \leq 1 + \log_2 N$

Stosowanie metody binarnej jest opłacalne przy większych zbiorach; dla małych zbiorów szybszą może się okazać metoda wyszukiwania sekwencyjnego (należy wziąć również pod uwagę koszt czasowy określenia kolejnego rekordu do sprawdzenia, który jest dużo większy w metodzie binarnej oraz koszt utrzymania rekordów w stanie uporządkowania). Przykładowo, znalezienie rekordu w zbiorze liczącym 1000 rekordów metodą wyszukiwania sekwencyjnego wymaga średnio 500 cykli, podczas gdy zastosowanie metody wyszukiwania binarnego — tylko 9 do 10 cykli. Poniżej przedstawiony został napisany w języku BASIC dla ZX Spectrum algorytm realizujący wyszukiwanie binarne.

Zaletą tego algorytmu jest to, że może on być używany nie tylko do wyszukiwania rekordów, ale również do sortowania zbioru, przy czym sortowanie odbywa się bezpośrednio po wprowadzeniu nowego rekordu tzn. rekordy zapisywane są od razu we właściwym miejscu. Sortowanie takie odbywa się szybko. Wykorzystana jest tu naturalna inercja człowieka przy wprowadzaniu kolejnych rekordów (wprowadzanie rekordów przeplata się z ich sortowaniem). Metoda taka pozwala na uniknięcie dłuższego oczekiwania na posortowanie zbioru, jakie występuje przy tradycyjnym układzie: wprowadzenie wszystkich rekordów — sortowanie całości.

Janusz MORBITZER

Program: Wyszukiwanie binarne

```
20 REM  Wyszukiwanie  binarne
30 REM p,k - początek i koniec
   żywej części zbioru
40 REM m-nr rekordu środkowego
50 REM a$(n,q)- tablica z upo-
   rządkowanymi rekordami
60 REM w$ - klucz wyszukiwania
70 REM d=LEN w$ (0<d<=q)
80 REM gdzie=0- użycie podpro-
   gramu do wyszukiwania (poz-
   nr znalezionego rekordu)
90 REM gdzie=1 -użycie podpro-
   programu do sortowania (poz
   -pozycja, na której ma zna-
   leźć się rek. o kluczu  w$)
100 REM ************************
110 LET p=1: LET k=n
120 IF (k-p)>=0 THEN GO TO 150
130 IF gdzie=1 THEN LET poz=x:
   RETURN
140 PRINT "Brak rekordu!":
   PAUSE 100: RETURN
150 LET m=1+INT ((p+k-1)/2)
160 IF a$(m, TO d)=w$ THEN LET
   poz=m: PRINT "Znaleziono
   rekord": RETURN
170 IF a$(m, TO d)<w$ THEN LET
   p=m+1: GO TO 120
180 LET k=m-1: GO TO 120
190 REM  koniec  podprogramu
```

Zmienne systemowe ROM Spectrum

ATRYBUTY

BORDCR	23624
ATTR—P	23693
ATTR—T	23695
MASK—P	23694
MASK—T	23696
P—FLAG	23697
K—DATA	23565
TVDATA	23566

Atrybuty ekranu to przede wszystkim kolory, ale niektórzy rozszerzają to pojęcie i wyróżniają:

INK — kolor atramentu, czyli liter i punktów;
PAPER — kolor tła;
BRIGHT — tryb rozjaśniający kolory;
FLASH — tryb cyklicznej wymiany kolorów atramentu i tła powodujący efekt migania;
INVERSE — tryb zamiany kolorów atramentu i tła;
OVER — tryb nakładania punktu na punkt powodujący jego zniknięcie (1 XOR 1 = 0)

Kolory podstawowe (INK, PAPER) skojarzone są z polem tekstowym, co oznacza, że znak na ekranie może posiadać tylko jedną wartość atramentu i tła. Rodzi to wiele kłopotów głównie w trybie graficznym.

Obszar atrybutów ekranu zajmuje tylko 768 bajtów (adresy 22528... 23295). Każdy bajt składa się z ośmiu bitów:

bit	7	6	5	4	3	2	1	0
stan	0	0	1	1	1	0	0	0
wartość	128	64	32	16	8	4	2	1

Bity 0, 1, 2 określają INK, bity 3, 4, 5, PAPER, bit 6 odpowiada za BRIGHT, a bit 7 za FLASH w danym polu tekstowym. W przykładzie powyższym zawartość bajtu wynosi 56, co oznacza PAPER 7, INK 0.

W Spectrum występują dwa pojęcia atrybutu — główny i tymczasowy. Jeśli podamy instrukcję np. PAPER 4: INK 7: CLS ustalamy atrybut główny, jeśli zaś: PLOT PAPER 1; INK 6; 10, 10 ustawiamy tymczasowy dla jednej instrukcji.

Atrybut główny trzyma zmienna ATTR P, tymczasowy — ATTR—T. Pozostałe tryby (INVERSE, OVER) przechowywane są w P—FLAG w następującej konfiguracji:

```
bit     ¦  znaczenie
--------------------------------
0       ¦  OVER tymczasowy
1       ¦  OVER główny
2       ¦  INVERSE tymczasowy
3       ¦  INVERSE główny
4       ¦  INK 9 tymczasowy
5       ¦  INK 9 główny
6       ¦  PAPER 9 tymczasowy
7       ¦  PAPER 9 główny
--------------------------------
```

Tryb 9 atramentu lub tła określa wydruk w kontrastowym kolorze niezależnie od stanu atrybutu głównego.
Stan atrybutów podaje następujący program:

```
9000 LET a=PEEK 23693
9002 LET b=PEEK 23697
9004 PRINT "FLASH ";a>127
9006 LET a=a-128*(a>127)
9008 PRINT "BRIGHT ";a>63
9010 LET a=a-64*(a>63)
9012 PRINT "PAPER ";INT (a/8),CHR$ (32+25*(b>127))
9014 LET b=b-128*(b>127)
9016 PRINT "INK ";a-8*INT (a/8),CHR$ (32+25*(b>31))
9018 LET b=b-32*(b>31)
9020 PRINT "INVERSE ";b>7
9022 LET b=b-8*(b>7)
9024 PRINT "OVER ";b>1
```

Jeśli **PAPER** lub **INK** zostały wymuszone przez kontrast pojawia się przy nich cyfra 9.

Tak się akurat składa, że BASIC zawiera wszystkie potrzebne rozkazy dotyczące atrybutów, przez co wyszczególnione na początku zmienne stosowane są głównie w języku wewnętrznym. Mało wykorzystywaną właściwością Spectrum są tzw. atrybuty maskujące (lub przezroczyste). Ustawia się je komendami **PAPER** 8, **INK** 8, **BRIGHT** 8 i **FLASH** 8. Powodują zamaskowanie bitów odpowiedzialnych za wyróżniony atrybut w pamięci ekranu, więc po wydruku pozostaje on bez zmiany. „Ósemki" przechowują zmienne **MASK—P** i **MASK—T** w podobny sposób, jak w **ATTR—P**. Sprawdźmy te atrybuty.

```
 10 LET a$="Każda litera jest widoczna !"
 20 PAPER 7: INK 7: CLS: REM mgła
 30 FOR a=22528 TO 22559
 40 POKE a,INT (RND*128)
 50 NEXT a: PAUSE 0
 80 FOR a=1 TO LEN a$
 90 PRINT a$(a);
110 PAUSE 10
120 NEXT a
```

Wydruk nastąpił, ale nic nie widać, gdyż atrament jest taki sam, jak tło. Zamaskujmy więc przypadkowe atrybuty, które wprowadziliśmy bezpośrednio do pamięci ekranu.

60 **POKE** 23694,127: **REM PAPER** 8, **INK** 8, **BRIGHT** 8

Teraz wydruk jest widoczny; choć gdzieniegdzie słabo. Zależy to od stanu atrybutu w danym polu. Dodajmy kontrastowy **INK**.

70**POKE** 23697, 32: **REM INK** 9

Obecnie w każdym polu atrament musi kontrastować z tłem, przez co każda litera staje się dobrze widoczna. Jeszcze tylko kropka nad z...

100 **IF** a= 3 **THEN PRINT CHR$** 8; "";

70**POKE** 23697, 34: **REM INK** 9, **OVER** 1

Zmienna TVDATA przechowuje pod 23566 ostatnio używany kod kontrolny atrybutu ekranu (16... 21), kod AT (22) lub TAB (23), natomiast pod 23567 pierwszy parametr funkcji AT (czyli numer linii) lub TAB (numer kolumny). Możemy więc zlokalizować linię, w której nastąpił ostatni wydruk:

```
10 PRINT AT INT (RND*22),0;" linia jest przypadkowa ";
20 PRINT FLASH 1;PEEK 23567
30 PAUSE 0: CLS: GOTO 10
```

Atrybuty można również wprowadzać w trybie EXTENDED bezpośrednio z klawiatury. Bajt ostatnio wykorzystywanego w ten sposób koloru lub stanu innych atrybutów (0 lub 1) przechowuje K—DATA.

Poza tym na ekranie widać jeszcze ramkę (BORDER), której kolor umieszczony jest w BORDCR. Wartość tej zmiennej jest jednocześnie atrybutem linii edycyjnych. Oto sposób na zgranie kolorów ramki i edytora:

10 **BORDER 4**
20 **INPUT**:: **REM** czyszczenie dwu linii edycji oraz tła ekra-

nu i edytora:
10 **PAPER** 4: **CLS**
20 **POKE** 23624, 32: **REM PAPER** 4, **INK** 0
30**INPUT**;

Ciekawym efektem jest podzielenie ramki na szereg pasków, a nawet próba jej zgrania z kolorami głównego ekranu.

```
10 BORDER 4: INPUT ;
20 FOR a=1 TO 44
30 PRINT PAPER 1+3*(a>22),: NEXT a
40 PAUSE 1: BORDER 1: BORDER 1: BORDER 1: BORDER 1:
   BORDER 1: BORDER 4: BORDER 4: BORDER 4: BORDER 4:
   GOTO 40
```

Maksymalna liczba pasków osiągana w BASIC'u wynosi 10 (w języku wewn. 625). Nieznaczne ruchy ramki przy naciskaniu klawiszy są spowodowane długością czasu odszukiwania klawisza (kolor ramki i stan klawiatury są identyfikowane w tym samym porcie).

W kolejnym odcinku trochę szczegółów o interpretacji i błędach.

Krzysztof MAMCARZ

„W szponach ATARI" dokończenie ze str. 6

Podaje również informację o posiadanych figurach (Mtr1) oraz ocenę aktualnej pozycji (Psnl). Dodatnie wartości świadczą o przewadze komputera.

Informacje o naszym ruchu możemy przekazać dwojako:

1) podając współrzędne pola początkowego i końcowego bierki szachowej; po podaniu współrzędnych każdego pola naciskamy RETURN,

2) za pomocą strzałek sterujących przesuwamy kursor na pole, z którego chcemy wykonać ruch, naciskamy RETURN, przesuwamy kursor na pole, na które chcemy wykonać posunięcie i naciskamy RETURN.

Po wykonaniu nieprawidłowego ruchu komputer wyświetla komunikat „Illegal move". Współrzędne wprowadzone pomyłkowo możemy kasować za pomocą klawiszy ESC i DELETE.

Komputer sygnalizuje zamiar wykonania ruchu komunikatem „Let me think". Naciśnięcie klawisza ESC, w czasie gdy komputer „myśli" nad ruchem, spowoduje przerwanie analizy możliwych posunięć, wyświetlenie komunikatu „Escape" i wykonanie najlepszego ruchu, jaki został znaleziony do momentu przerwania analizy. Jeśli komputer przewiduje szybką wygraną, uprzedza nas o tym komunikatem „Mate in N" — gdzie N jest liczbą ruchów dzielących nas od mata.

Po zakończeniu gry komputer informuje o wyniku: Dawn — remis, checkmate — mat, Stalemat — pat, Time up — przekroczenie limitu czasu. Po tych komunikatach pyta: „What now" (co teraz?) i czeka na wprowadzenie nowych instrukcji.

Naciśnięcie CTRL R pozwala na odtworzenie rozegranej partii. Program zatrzymuje się po każdym ruchu na czas od 0 do 20 sekund (podawany przez nas). Odtwarzanie może być w każdej chwili przerwane klawiszem ESC.

Po naciśnięciu CTRL N — nowa gra. Możliwe jest określenie stopnia trudności (CTRL Q) oraz wybranie typu gry (CTRL T). Po CTRL Q komputer pyta, czy ma korzystać z posiadanej książki debiutów (Book?) — odpowiadamy 1 lub 0 (tak/nie). Możemy również określić, czy komputer ma dążyć do zwycięstwa, czy do remisu. Podajemy wartości dla mtrl z przedziału —9,+9 oraz dla psnl z przedziału —60,60; im niższe wartości, tym silniejsze dążenie komputera do zwycięstwa.

Program zawiera 6 typów partii rozgrywanych w różnym tempie i stylu. Po naciśnięciu CTRL T jesteśmy proszeni o wybranie jednego z sześciu wariantów:

1) partia turniejowa — należy podać liczbę ruchów do pierwszej i drugiej kontroli czasu oraz terminy kontroli,

2) partia, dla której określamy średni czas ruchu,

3) partia o zadanym czasie trwania — przekroczenie czasu kończy się porażką,

4) partia równoważna — komputer „myśli" tak długo jak przeciwnik,

5) partia o nieograniczonym czasie — stosowana przy poszukiwaniu najlepszego wariantu ruchu (czas poszukiwania może trwać nawet dobę);

6) rozwiązywanie zadań — komputer pyta o liczbę ruchów do mata.

Możliwe są także dodatkowe utrudnienia gry, których celem jest ćwiczenie pamięci. Po naciśnięciu **CTRL V** możemy wybrać jedną z następujących wersji: 1 — niewidoczne białe bierki; 2— niewidoczne czarne bierki; 3 — niewidoczne białe i czarne. Dodatkowo za pomocą **CTRL Z**, możemy wyłączyć dźwięk.

Program jest bardzo elastyczny pod względem wyboru partnerów. Dostępne są następujące warianty gry: komputer — komputer (**CTRL P**), człowiek — człowiek (**CTRL S**), bez podawania czegokolwiek człowiek (**Opponent**) gra z komputerem (**Colossus**). Podczas gry z komputerem możemy zmusić go do wykonania ruchu za nas (**CTRL G**), następuje wtedy odwrócenie szachownicy i kontynuujemy grę bierkami komputera. Zmianę koloru bierek realizujemy po **CTRL O**, a zmianę koloru ekranu — **CTRL C**.

W czasie gry możliwe jest cofnięcie ruchu (**CTRL B**), zarówno naszego, jak i komputera, nawet kilkakrotnie. Możliwe jest wtedy wykonanie jednego lub więcej ruchów — **CTRL F**.

Można również „poprosić" komputer o wykonanie nieco gorszego ruchu (następny najlepszy) — **CTRL U**.

W wypadku rozwiązywania różnorodnych zadań szachowych pożądana jest możliwość swobodnego ustawiania figur na szachownicy. Po naciśnięciu **CTRL A** pojawia się komunikat: *„Alter position: White. Command?"*, który sygnalizuje gotowość komputera do ustawiania białych bierek. W tym celu należy ustawić kursor w wybranym polu i nacisnąć jedną z następujących liter: C (czyszczenie pola), P (pionek), N (skoczek), B (goniec), R (wieża), Q (hetman) lub K (król). Naciśnięcie klawisza W usuwa wszystkie bierki. Aby ustawić figury drugiej strony, należy nacisnąć S. Wyjście z trybu ustawiania poprzez naciśnięcie E.

Jeśli partia szachów przeciąga się i nie mamy czasu na jej dokończenie, możemy przechować na taśmie magnetycznej aktualną sytuację i przełożyć partię na później. Po naciśnięciu CTRL D pojawia się komunikat „Load or Save". W zależności od potrzeby naciskamy L lub S (naciśnięcie innego klawisza powoduje wyjście z tego trybu pracy). Komputer poleca przygotować magnetofon i nacisnąć dowolny klawisz.

Colossus chess 3.0 jest najbardziej interesującym programem szachowym, w wersji taśmowej, jaki udało nam się znaleźć dla komputerów Atari 800 XL i 130 XE.

Tomasz MROWIEC
Ludwik PIELA

Oprogramowanie znacznika świetlnego

Procedury graficzne w systemach komputerowych stosowane są do wprowadzania, przetwarzania i zobrazowania danych w postaci graficznej. Wprowadzanie danych w postaci graficznej jest ułatwione dzięki użyciu urządzeń operatorskich takich jak manipulator kulowy lub pióro świetlne [1]. Użytkownik wskazuje określone punkty na ekranie monitora za pomocą znacznika świetlnego, sterowanego odpowiednio oprogramowanym urządzeniem operatorskim.

Znacznik świetlny może być również sterowany uniwersalną klawiaturą alfanumeryczną. Rozwiązanie takie przedstawione zostanie poniżej.

Wybrany podzbiór znaków klawiatury alfanumerycznej został oprogramowany dla potrzeb sterowania znacznikiem. Oprogramowanie to jest częścią szkoleniowego systemu graficznego (SSG), który został opracowany w języku BASIC i uruchomiony w komputerze MERITUM I.

[1] *Chmurzyński J.: Grafika komputerowa. „IKS", nr 2/1986*

Struktura SSG

Struktura SSG przedstawiona została na rys. 1. W skład SSG wchodzą tablice współrzędnych — rys. 2 oraz procedury, zestawione w tabeli 1. Początkowy fragment programu SSG przedstawiono poniżej:

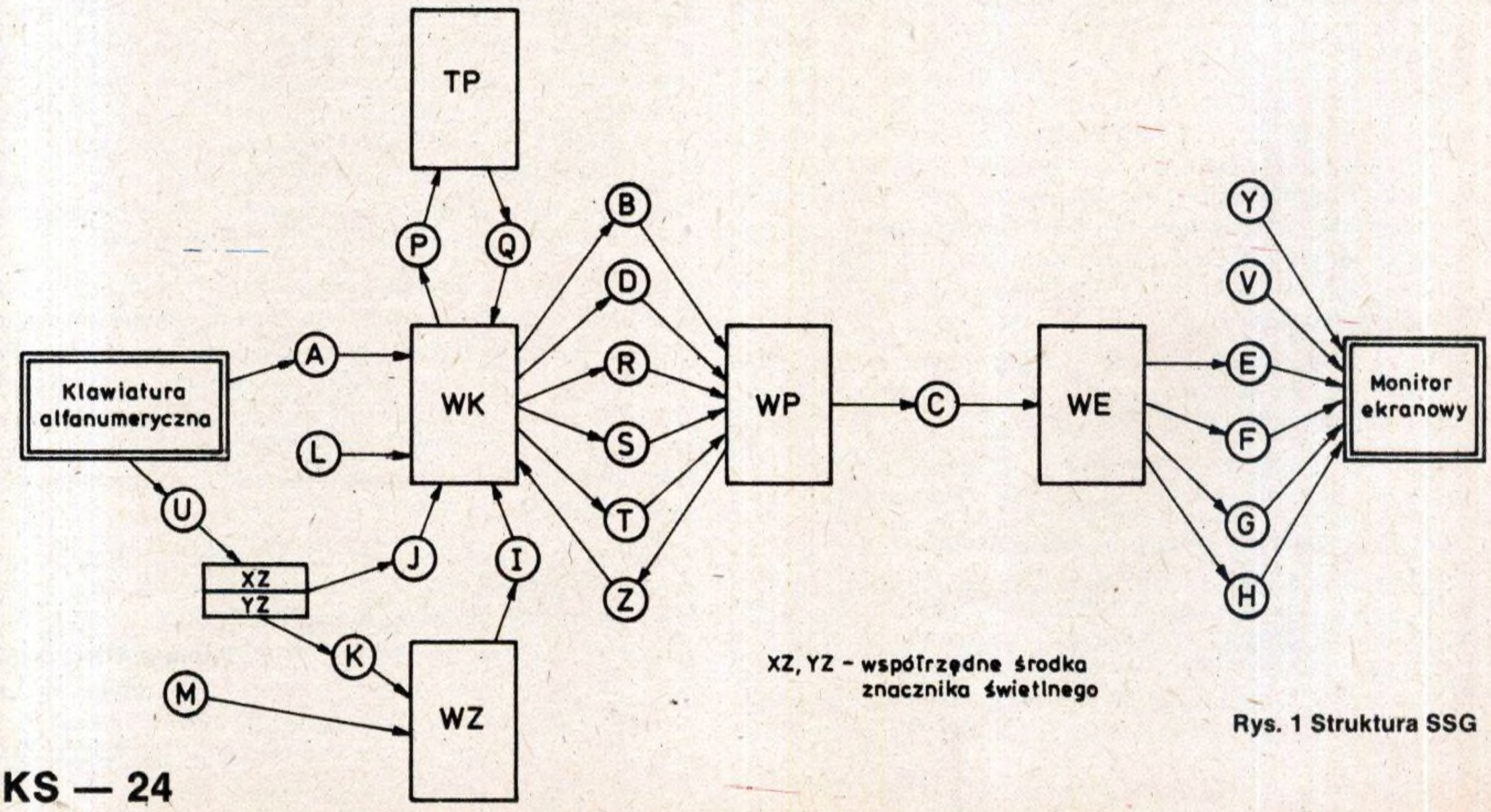
```
  1 REM * * * * * SZKOLENIOWY SYSTEM GRAFICZNY
  2 REM * * * * * OPRACOWAŁ J. CHMURZYŃSKI 1985
  3 REM * * * * * MERITUM 1 W. 1. 0
 10 DIM WK (21,2)
 11 DIM WZ (21,2)
 12 DIM WP (21,2)
 13 DIM WE (21,2)
 14 DIM TP (21,2)
 20 DIM P1 (2)
 21 DIM P2 (2)
 22 DIM P3 (2)
 23 DIM T1 (2,2)
 24 DIM T2 (2,2)
 25 DIM T3 (2,2)
 99 CLS
100 INPUT „NR PROCEDURY"; NP
101 IF NP =  1 GOTO 250
102 IF NP =  2 GOTO 150
103 IF NP =  3 GOTO 153
104 IF NP =  4 GOTO 155
105 IF NP =  5 GOTO 157
106 IF NP =  6 GOTO 170
107 IF NP =  7 GOTO 172
108 IF NP =  8 GOTO  99
109 IF NP =  9 GOTO 290
110 IF NP = 10 GOTO 174
111 IF NP = 11 GOTO 164
112 IF NP = 12 GOTO 168
113 IF NP = 13 GOTO 160
114 IF NP = 14 GOTO 162
115 IF NP = 15 GOTO 400
116 IF NP = 16 GOTO 555
117 IF NP = 17 GOTO 560
118 IF NP = 18 GOTO 166
148 PRINT „ZLY NR PROCEDURY"
149 GOTO 100
```

Poszczególne tablice przechowują następujące dane:

- tablice WK i TP — współrzędne kartezjańskie punktów charakterystycznych obrazu,
- tablica WP — przekształcone współrzędne kartezjańskie punktów charakterystycznych obrazu,
- tablica WE — współrzędne ekranowe punktów charakterystycznych obrazu,
- tablica WZ — współrzędne kartezjańskie punktów charakterystycznych obrazu wskazanych znacznikiem świetlnym.

Obieg danych w SSG

Dane w SSG przyjmować mogą postać pojedynczych współrzędnych X, Y punktu lub ciągów współrzędnych X, Y punktów. Punkty te mogą być elementami składowymi obrazu lub też punktami początkowymi i końcowymi odcinków linii prostej.



Rys. 1 Struktura SSG

Tabela 1

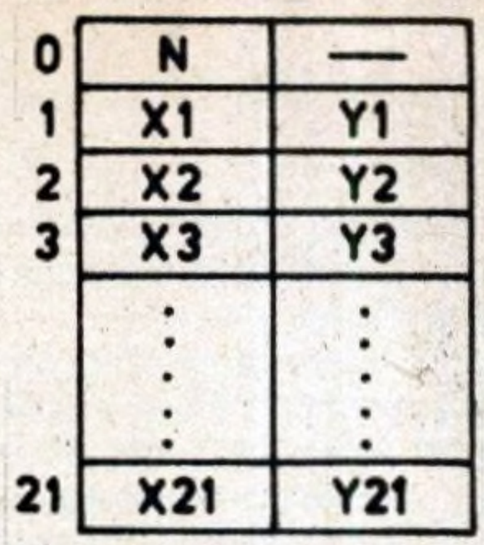
Procedury SSG

Nazwa	FUNKCJA
A	Wprowadzenie ciągu N par współrzędnych kartezjańskich z klawiatury alfanumerycznej do tablicy WK
B	Przesyłanie zawartości tablicy WK do tablicy WP bez przekształceń współrzędnych
C	Zamiana współrzędnych kartezjańskich punktów z tablicy WP na współrzędne ekranowe i przesyłanie ich do tablicy WE
D	Zmiana skali współrzędnych punktów z tablicy WK i przesyłanie ich do tablicy WP
E	Zobrazowanie na monitorze ekranowym linii według punktów charakterystycznych z tablicy WE
F	Wygaszenie linii według punktów charakterystycznych z tablicy WE
G	Zobrazowanie na monitorze ekranowym punktów charakterystycznych z tablicy WE
H	Wygaszanie punktów charakterystycznych z tablicy WE
I	Przesyłanie zawartości tablicy WZ do tablicy WK
J	Przesyłanie współrzędnych środka znacznika świetlnego na pierwsze miejsce tablicy WK
K	Przesyłanie współrzędnych środka znacznika świetlnego na kolejne miejsce tablicy WZ
L	Zerowanie tablicy WK
M	Zerowanie tablicy WZ
P	Przesłanie zawartości tablicy WK do tablicy TP
Q	Przesłanie zawartości tablicy TP do tablicy WK
R	Przekształcenie obrazu opisanego punktami charakterystycznymi w tablicy WK: przesunięcie o dany wektor
S	Przekształcenie obrazu opisanego punktami charakterystycznymi w tablicy WK: obrót wokół środka układu współrzędnych
T	Przekształcenie obrazu opisanego punktami charakterystycznymi w tablicy WK: obrót wokół dowolnego punktu
U	Sterowanie znacznikiem świetlnym (patrz rys. 4; tabela 3)
V	Wygaszenie wszystkich punktów na ekranie monitora
Y	Zobrazowanie na ekranie monitora osi X — Y układu współrzędnych
Z	Przesłanie zawartości tablicy WP do tablicy WK

Współrzędne punktów wprowadzane są z klawiatury alfanumerycznej do tablicy WK (procedura A). Drugim sposobem wprowadzania współrzędnych punktów jest użycie znacznika świetlnego, sterowanego klawiaturą alfanumeryczną (procedura U). Współrzędne środka znacznika świetlnego XZ, YZ kierowane są do tablicy WZ (procedura K) oraz zobrazowane są na ekranie monitora (procedury J, B, C, G). Po skompletowaniu ciągu współrzędnych punktów, wskazanych za pomocą znacznika świetlnego, zawartość tablicy WZ może być przeniesiona do tablicy WK (procedura I).

Obraz opisany punktami charakterystycznymi, których współrzędne znajdują się w tablicy WK, może być przekształcony i przesłany do tablicy WP (procedury D, R, S, T). Współrzędne kartezjańskie punktów charakterystycznych z tablicy WP przekształcane są do postaci współrzędnych ekranowych (procedura C). Postać ekranowa współrzędnych służy do wyświetlania punktów lub odcinków linii (procedury G, E) albo też do wygaszania punktów lub odcinków (procedury H, F).

Tablica TP służy do chwilowego przechowywania obrazu.



N — ilość par współrzędnych X, Y znajdujących się w tablicy

Rys. 2 Struktura tablicy współrzędnych

Podczas złożonych transformacji obrazu stosowane jest przesyłanie zawartości tablicy WP do tablicy WK (procedura Z).

Dyrektywy operatorskie SSG

Kolejnością wykonywania procedur SSG sterują dyrektywy operatorskie. Dyrektywy złożone są z identyfikatorów i parametrów. Dyrektywy operatorskie wprowadzane są do SSG przez klawiaturę alfanumeryczną. Identyfikatory dyrektyw przyjmują wartość liczbową z przedziału od 1 do 18. W tabeli 2 zestawiono dyrektywy operatorskie SSG wraz z operacjami wykonywanymi po wprowadzeniu dyrektyw do SSG.

Sterowanie znacznikiem świetlnym

Znacznik świetlny w SSG złożon jest z 4 punktów ułożonych w kształci krzyża (rys. 3). Współrzędne środk

Rys. 3 Znacznik świetlny

znacznika przechowywane są w zmiennych XZ, YZ. Procedura U ustawia początkowo znacznik na środku ekranu. W celu łatwego odróżnienia znacznika od innych elementów obrazu, znacznik migocze. Migotanie znacznika zapewnia program cyklicznego wyświetlania i wygaszania znacznika na ekranie monitora.

Do sterowania znacznikiem świetlnym wydzielono dodatkowe dyrektywy, przedstawione w tabeli 3. Znacznik może być przesuwany ze zmienną szybkością, zależną od skoku znaczni-

ka SZ. Wielkość skoku ustala się dyrektywami sterowania znacznikiem świetlnym 1, 3, 5 i 7.

Algorytm sterowania znacznikiem świetlnym przedstawia rys. 4. Poszczególne bloki algorytmu oznaczono dodatkowo numerami początkowych wierszy procedury sterowania znacznikiem świetlnym (procedura U).

Procedúra sterowania znacznikiem świetlnym

Poniżej przedstawiono procedurę sterowania znacznikiem świetlnym w SSG. Procedura ta inicjowana jest dyrektywą operatorską nr 15 i zapewnia sterowanie znacznikiem zgodnie z algorytmem przedstawionym na rys. 4. Procedura ta umożliwia:

— wprowadzenie do tablicy WZ opisu obrazu w postaci ciągu par współrzędnych charakterystycznych,
— wyświetlanie punktów na ekranie monitora w miejscach położenia punktów charakterystycznych (dyrektywa Q),
— wyświetlanie linii łamanej, łączącej punkty charakterystyczne (dyrektywa L).

Po wprowadzeniu wymaganej liczby punktów charakterystycznych dane te mogą być przesłane z tablicy WZ do tablicy WK (procedura 1) i poddane przekształceniom za pomocą procedury SSG.

```
400 REM * * * * * * PROCEDURA U
401 WW = 0
402 XZ = 64
403 YZ = 24
405 SZ = 1
405 XP = 64
406 YP = 24
410 IF WW = 0 GOTO 412
411 IF WW = 1 GOTO 422
412 REM * * * * * * * WYSWIETL ZNACZNIK
413 XZ = XP
414 YZ = YP
415 WW = 1
417 SET (XZ + 1, YZ)
418 SET (XZ — 1, YZ)
419 SET (XZ, YZ + 1)
420 SET XZ, YZ — 1)
421 GOTO 430
422 REM * * * * * * WYGAS ZNACZNIK
423 WW = 0
425 RESET (XZ + 1, YZ)
426 RESET (XZ — 1, YZ)
427 RESET (XZ, YZ + 1)
428 RESET (XZ, YZ — 1)
429 GOTO 430
430 REM * * * * * * * SLEDZENIE ZNACZNIKA
431 Z$ = INKEY$
432 IF Z$ = "J" GOTO 460
433 IF Z$ = "K" GOTO 470
434 IF Z$ = "I" GOTO 480
435 IF Z$ = "M" GOTO 490
436 IF Z$ = "Q" GOTO 500
437 IF Z$ = "L" GOTO 520
438 IF Z$ = "S" GOTO 100
449 GOTO 540
450 GOTO 410
460 REM * * * * * * KLAWISZ J
461 XP = XZ — 1 * SZ
462 IF XP — 1 < GOTO 465
464 GOTO 410
465 XP = XZ
466 GOTO 410
470 REM * * * * * * KLAWISZ K
471 XP = XZ + 1 * SZ
472 IF XP + 1 > 127 GOTO 475
474 GOTO 410
475 XP = XZ
476 GOTO 410
480 REM * * * * * * KLAWISZ I
481 YP = YZ — 1 * SZ
482 IF YP — 1 < 0 GOTO 485
484 GOTO 410
485 YP = YZ
486 GOTO 410
490 REM * * * * * * KLAWISZ M
491 YP = YZ + 1 * SZ
492 IF YP + 1 > 47 GOTO 495
494 GOTO 410
495 YP = YZ
```

Dyrektywy operatorskie SSG

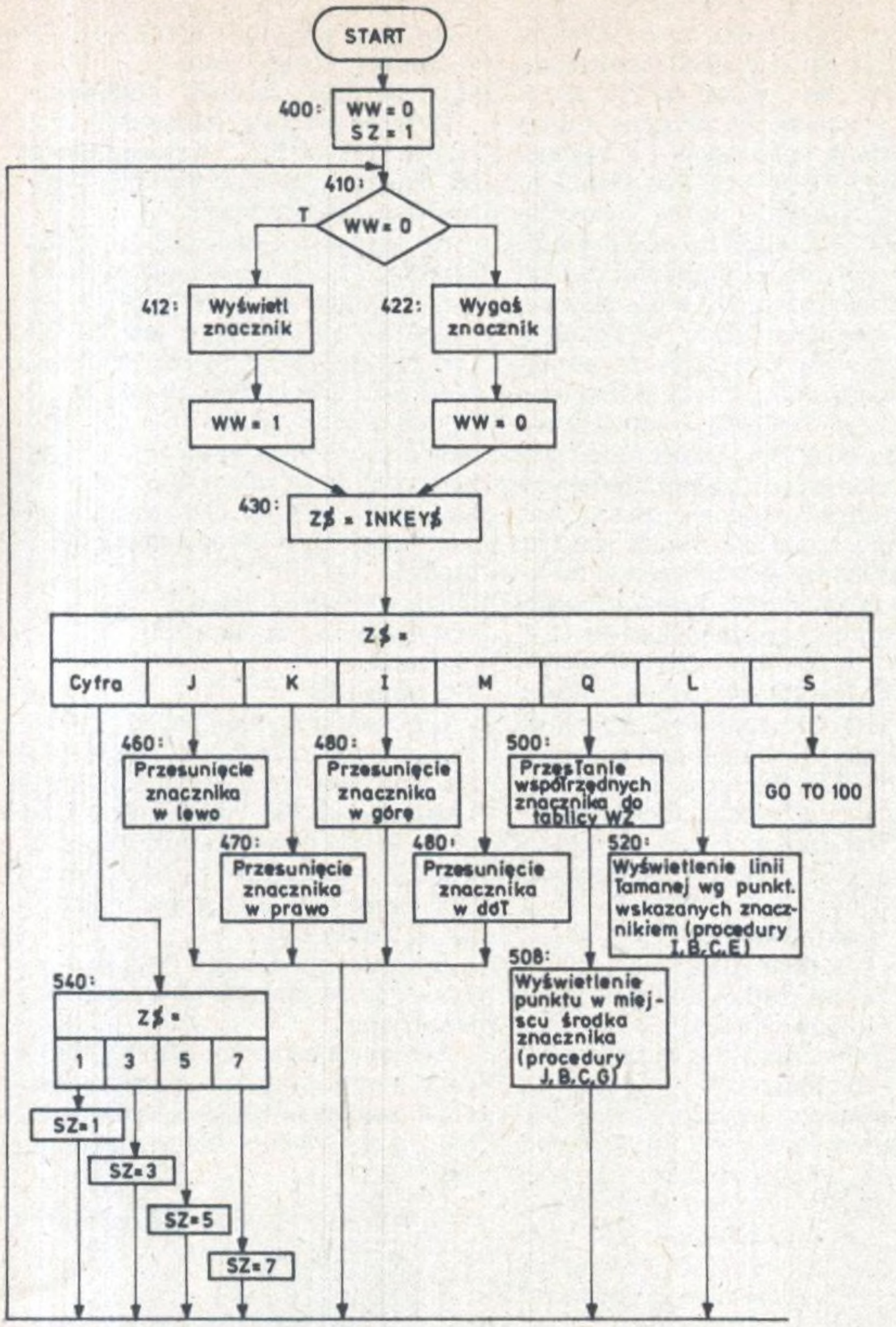
NUMER DYREKTYWY (IDENTYFIKATOR)	OPERACJE DYREKTYWY	PARAMETRY DYREKTYWY
1	A	— LICZBA PUNKTÓW — WSPÓŁRZĘDNE X, Y PUNKTÓW
2	B + C	—
3	E	—
4	F	—
5	D + C	—
6	P	—
7	Q	—
8	V	—
9	Y	—
10	R + C	— DX, DY WEKTOR PRZESUNIĘCIA OBRAZU
11	S + C	— FI — KĄT OBROTU
12	T + C	— DX. DY WSPÓŁRZĘDNE PUNKTU OBROTU — FI — KĄT OBROTU
13	G	—
14	H	—
15	U	patrz tabela 3, rys. 4
16	M	—
17	L	—
18	Z	—

Tabela 2

Dyrektywy sterowania znacznikiem świetlnym

Tabela 3

IDENTYFIKATOR DYREKTYWY	OPERACJE DYREKTYWY
J	Przesunięcie znacznika w lewo
K	Przesunięcie znacznika w prawo
I	Przesunięcie znacznika w górę
M	Przesunięcie znacznika w dół
Q	Wykonanie operacji K oraz ciągu operacji J, B, C, G
L	Wykonanie ciągu operacji I, B, C, E
1	Ustalenie skoku znacznika SZ = 1 j. r.
3	Ustalenie skoku znacznika SZ = 3 j. r.
5	Ustalenie skoku znacznika SZ = 5 j. r.
7	Ustalenie skoku znacznika SZ = 7 j. r.
S	Koniec sterowania znacznikiem świetlnym GOTO 100



Rys. 4 Algorytm sterowania znacznikiem świetlnym

```
496 GOTO 410
500 REM * * * * * * * WYŚWIETL PUNKT
WG ZNACZNIKA
501 WZ (0, 0) = WZ (0, 0) + 1
502 N = WZ (0, 0)
503 WZ (N, 0) = XZ — 64
504 WZ (N, 1) = (24 — YZ) * 1.845
505 WK (0, 0) = 1
506 WK (1, 0) = XZ — 64
507 WK (1, 1) = (24 — YZ) * 1.845
508 GOSUB 260
509 GOSUB 280
510 GOSUB 380
511 GOTO 410
520 REM * * * * * * * KRESLENIE LINII
ZA ZNACZNIKIEM
521 FOR I = 1 TO WZ (0, 0)
522 WK (I, 0) = WZ (i, 0)
523 WK (I, 1) = WZ (I, 1)
524 NEXT I
525 WK (0, 0) = WZ (0, 0)
526 GOSUB 260
527 GOSUB 280
528 GOSUB 300
529 GOTO 410
540  REM * * * * * * * ZMIANA SZE-
ROKOŚCI PRZESUWU ZNACZNIKA
541 IF Z$ = "1" GOTO 546
542 IF Z$ = "3" GOTO 548
543 IF Z$ = "5" GOTO 550
544 IF Z$ = "7" GOTO 552
545 GOTO 410
546 SZ = 1
547 GOTO 410
548 SZ = 3
549 GOTO 410
550 SZ = 5
551 GOTO 410
552 SZ = 7
553 GOTO 410
```

Programy pozostałych procedur SSG przedstawione zostaną w drugiej części artykułu.

Jerzy CHMURZYŃSKI

SAMI PISZEMY GRY

Cezary SOBCZAK

Znając podstawowe instrukcje języka BASIC, spróbujmy sami napisać grę. Zaczynamy od wymyślenia scenariusza. A oto przykład: *podróżując rakietą po wszechświecie w poszukiwaniu przygód, w pewnym momencie nasz czujnik pokładowy zasygnalizował, że kończy się paliwo. Na nasze szczęście w pobliżu znalazła się planeta, o której nasz komputer dał nam informację, iż znajdują się na niej składniki niezbędne do wytworzenia potrzebnego nam paliwa. Lecz cóż to? Nasze czujniki wykryły obecność na planecie zbiorników ze skumulowaną antymaterią i spotkanie z takim zbiornikiem grozi wybuchem.* Wprawdzie komputer pokładowy wyświetla nam położenie niebezpiecznych zbiorników, lecz pojawiające się zakłócenia powodują to, że ich rozmieszczenie po pewnym czasie staje się niewidoczne. Wybierając się w drogę musimy zatem zapamiętać ich rozmieszczenie i tak się poruszać, aby się z nimi nie zderzyć.

Przystępujemy teraz do budowy gry. Komputer osobisty AMSTRAD CPC 6128 może wyświetlać znaki w trzech trybach (wielkościach): tryb 0, tryb 1 i tryb 2.

W trybie 0 (MODE 0) może zapisać 20 znaków w linii,
w trybie 1 (MODE 1) może zapisać 40 znaków w linii,
w trybie 2 (MODE 2) może zapisać 80 znaków w linii,

w każdym trybie można zapisać 25 linii.

Aby zrozumieć różnice pomiędzy poszczególnymi trybami, po włączeniu komputera (automatycznie ustawiany jest tryb 1) wyświetl kilka dowolnych znaków. Następnie zmień tryb pisząc np. MODE 0 i wyświetl znów kilka znaków. Zobaczysz, że w trybie 2 znaki są najmniejsze, w trybie 0 — największe. Do naszej gry wybieramy tryb 0 (MODE 0), a więc mamy do dyspozycji pole ekranu o wymiarach 20 na 25 znaków. Przystępujemy do precyzowania założeń gry, ustalania zmiennych i tablic oraz rysujemy schemat gry.

Załóżmy, że:

— nasza załoga składa się z trzech osób, które pojedynczo wychodzą na

poszukiwanie składników paliwa,

— rozlokowano 10 niebezpiecznych zbiorników,

— na planecie znajduje się 5 składników naszego paliwa, z czego do jego wytworzenia potrzeba nam przynajmniej trzech.

Lista zmiennych i tablice:

zal — określa aktualną liczbę członków załogi,

n — ilość rozlokowanych zbiorników,

m — ogólna ilość rozmieszczonych na planecie składników paliwa,

k — niezbędna ilość składników potrzebnych do wykonania paliwa ($k \leq m$), a więc umożliwiających start rakiety i opuszczenie planety,

l — ilość aktualnie zebranych składników paliwa, $l \geq k$,

minx (n) — tablica współrzędnych x zbiorników

miny (n) — tablica współrzędnych y zbiorników

palx (m) — tablica współrzędnych x składników paliwa

paly (m) — tablica współrzędnych y składników paliwa

Rozpoczynając grę, kolejny członek załogi znajduje się w lewym górnym rogu ekranu (współrzędne $x = 1$, $y = 2$; wiersz 1 rezerwujemy na wyświetlanie informacji np. o pozostałych przy życiu członkach załogi itp.), natomiast rakieta w prawym dolnym rogu ekranu ($x = 20$, $y = 25$). Rys. 1.

Przystępujemy do kodowania naszego programu. Ustalmy jeszcze, że podprogramy będą rozpoczynały się od linii 1000.

```
10 rem nadanie wartości początkowych
20 mode O: zal=3:n=10:m=5:k=3:1=0
30 dim minx (n) : dim miny (n) : dim palx (m) : dim paly (m)
```

Generujemy losowe współrzędne rozmieszczenia niebezpiecznych zbiorników oraz składników paliwa. Wykorzystamy instrukcję **RND** obliczającą kolejną liczbę losową z przedziału (0—1). Ponieważ współrzędne muszą być liczbami całkowitymi i z zakresu odpowiednio x od 1 do 20 i y od 2 do 25, wygenerowane liczby mnożymy przez 100, zaokrąglamy do liczb całkowitych i badamy przynależność do określonego przedziału. Może się zdarzyć, że wygenerowane współrzędne zbiornika będą identyczne ze współrzędnymi składnika paliwa. Można tego uniknąć wprowadzając odpowiednie badanie i zmieniając współrzędne jednego z elementów, ale pozbawiłoby to dodatkowego elementu „dreszczyka" naszej gry. Może się również zdarzyć, że wygenerowane współrzędne zbiornika będą identyczne z początkowymi współrzędnymi naszego ludzika (1,2) oraz położenia rakiety (20,25). W tym wypadku należałoby jednak zmienić położenie niebezpiecznego zbiornika, gdyż nie dałoby to nam szansy rozpoczęcia bądź zakończenia gry.

Napiszemy teraz podprogram generujący współrzędne,

```
1000 rem podprogram wyznaczania współrzędnych,
1010 x = cint (rnd* 100) : if x < — 1 or x > — 20 then 1010
1020 y = cint (rnd* 100) : if y < — 2 or y > — 25 then 1020
1025 if (x = 1 and y = 2) or (x = 20 and y = 25) then 1010
1030 return
```

który wykorzystamy do zobrazowania ustawienia zbiorników. Jako znak niebezpiecznego zbiornika przyjmujemy jeden ze znaków naszego komputera oznaczony numerem 231: 40 **rem** generowanie i wyświetlanie zbiorników

```
50 for i = 1 to n
60 gosub 1000
70 minx (i) = x : miny (i) = y
80 locate x,y : print chr$ (231); : next
```

Dopisz teraz wiersz

```
85 for a = 1 to 1000 : next:cls:goto 40
```

i uruchom program. Zobaczysz, jak wyglądają kolejne warianty rozmieszczenia niebezpiecznych zbiorników.

Przerwij program naciskając dwukrotnie klawisz [ESC], **wykasuj wiersz 85**, napisz mode 1 lub mode 2 i piszemy nasz program dalej.

Do zakrycia („utajnienia") rozmieszczenia zbiorników wykorzystamy rozkazy grafiki komputera. W tym wypadku ekran podzielony jest na 640 punktów (tzw. pixeli) w poziomie i 400 punktów w pionie. Początkowe współrzędne kursora grafiki ($x = 0$, $y = 0$) umieszczone są w lewym dolnym rogu ekranu. Pole ze zbiornikami będziemy zasłaniać „kurtyną" przemieszczającą się z lewej strony ekranu na prawą. Piszemy:

```
90 for i = 0 to 640 step 3
100 move i, 1 : drawr 1,380
120 next
125 goto 125
```

Uruchom program i zobacz, jak zasłaniane są niebezpieczne zbiorniki. Naciśnij dwa razy klawisz [ESC], **wykasuj wiersz 125**, napisz mode 1 lub mode 2. Aby nieco utrudnić, zakrywajmy pole jednocześnie z dwóch stron:

```
110 move 640—i, 1:drawr 1,380 : if i > 320 then 130
```

Spróbuj sam dopisać instrukcje powodujące zasłanianie pola z czterech stron naraz.

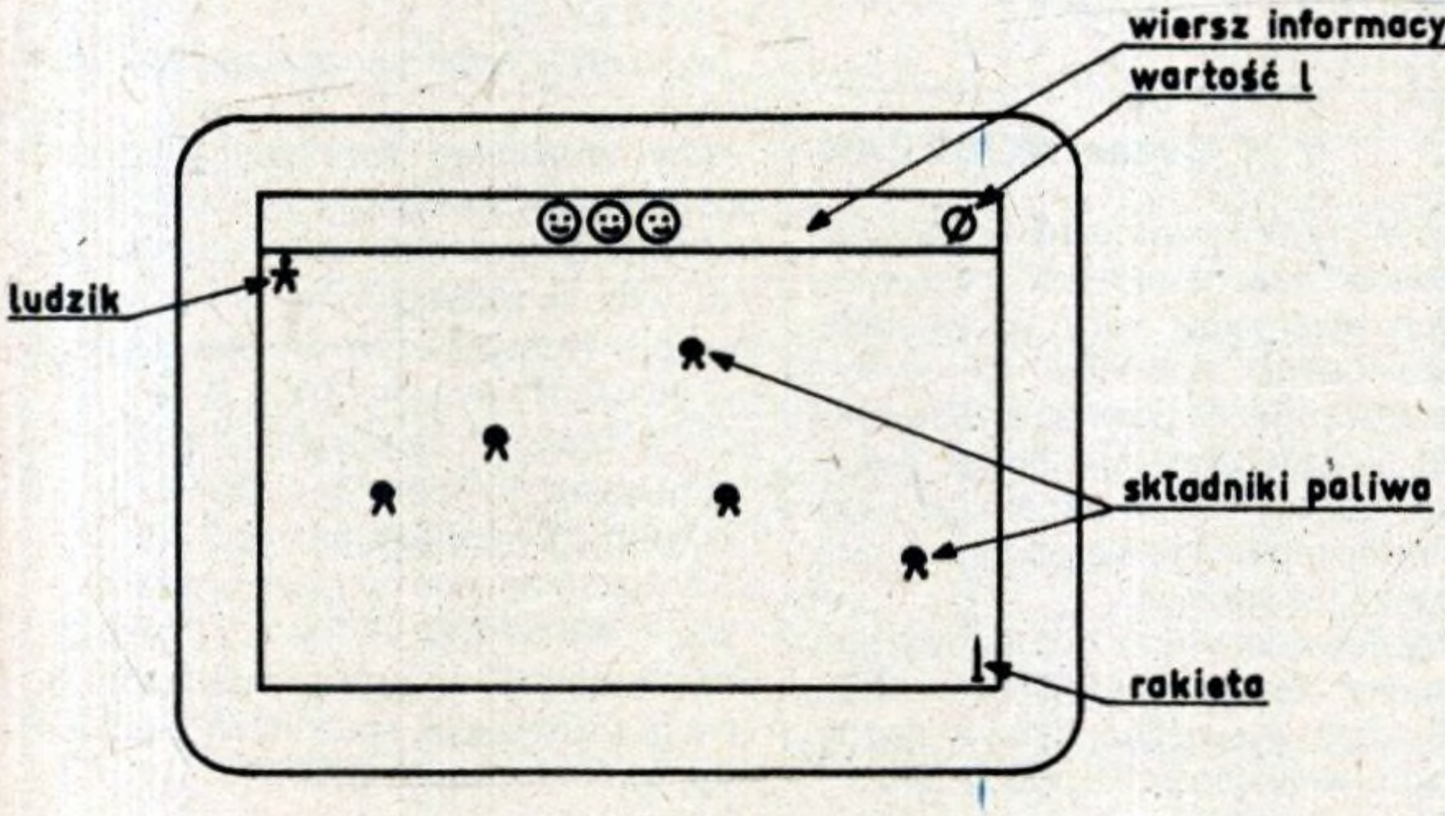
Skoro niebezpieczne zbiorniki są już niewidoczne, umieśćmy na naszej planecie komponenty paliwa, które musimy zebrać. Wykorzystamy znak o numerze 229.

```
130 rem generowanie i wyświetlanie składn. paliwa
140 for i = 1 to m
150 gosub 1000
160 palx (i) = x : paly (i) = y
170 locate x,y : print chr$ (229); : next
175 goto 175
```

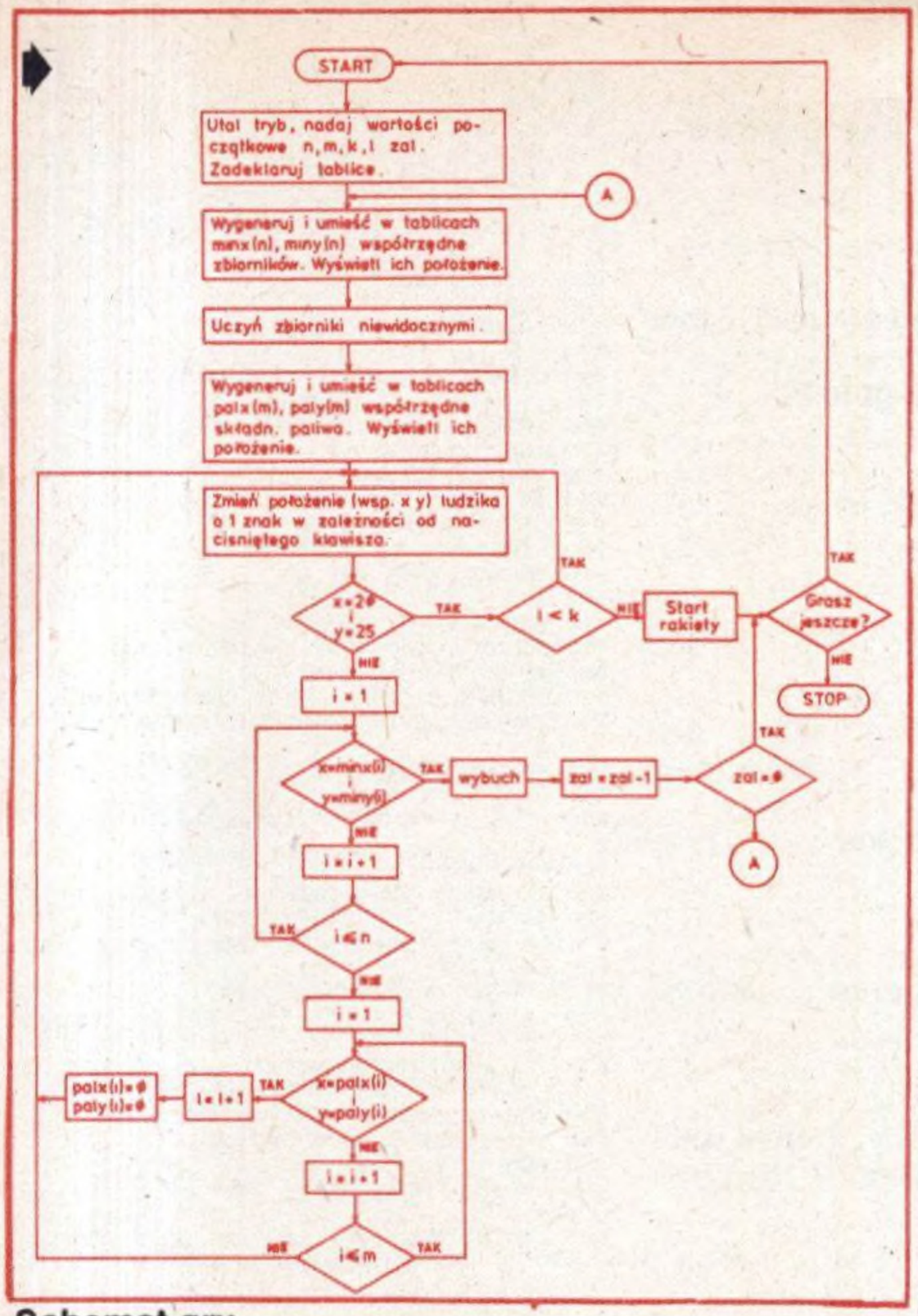
Uruchom program i zobacz jeden z możliwych wariantów rozmieszczenia składników paliwa. Naciśnij dwa razy klawisz [ESC], **skasuj wiersz 175**, napisz mode 1 lub mode 2. Umieścimy teraz na odpowiednich miejscach naszego ludzika (znak numer 249) i rakietę (znak numer 239) oraz napiszemy podprogram wyświetlający informację o pozostałych przy życiu członkach załogi:

```
180  x = 1 : y = 2 locate  x,y : print chr$ (249); : locate 20, 25 : print chr$ (239);
190 gosub 1130
1130 rem podaje ilość pozostałej przy życiu załogi
1140 locate 18,1 : print 1;
1150 for i = 1 to zal
1160 locate 8 + i, 1 : print chr$ (224); : next : return
```

Przystępujemy do sterowania ruchem ludzika w zależności od naciśniętego klawisza. Wszystkim klawiszom komputera



Rys. 1 Ekran początkowy



Schemat gry

dokończenie na str. 30

W naszym komputerlandzie

W naszym komputerlandzie trwał właśnie niczym nie zmącony świąteczny nastrój. Z różnych stron napływały na ręce pracowitego Spektrusia urodzinowe życzenia, głównie by kolejne lata były wesołe, zdrowe i pogodne, wypełnione nieskończonym pasmem sukcesów.

Atmosferę tę zakłócił jednak telegram oznaczony symbolem „PILNY". Jego treść początkowo obraźliwa, następnie życzliwa i radosna, w miarę dalszego czytania stawała się wulgarna, a na końcu nawet tragiczna — ogólnie jednak mało zrozumiała. Brzmiała ona:

stop OCH NIEROBY stop SERDECZNE ŻYCIE NA stop WIELE CZĘŚCI DO MASZYNKI stop MI DUPA SIEKIERA DOŚĆ NAKOPAŁ stop PIOTRUŚ NIE ŻYJE stop PRZYJEDŹCIE.

Telegram wywołał w zespole robotów ogromny niepokój i poruszenie. Podczas gdy gorączkowo zaczęto się zastanawiać co począć, jeden z robotów nie przejmując się poddawał pod wątpliwość prawdziwość treści, w związku z czym przestawił się na kontestację. Rozmyślając nad tekstem wykonał on we współpracy z komputerem operację zwaną tłumaczeniem telegramu. Okazało się, że właściwa treść telegramu jest następująca:

stop KOCHANE ROBOTY stop SERDECZNE ŻYCZENIA stop WIELE SZCZĘŚCIA I W DOMU SZYNKI stop MIODU PASIEKĘ RADOŚCI NA KOPY stop SPEKTRUŚ NIECH ŻYJE stop PRZYJACIEL.

No! Roboty odetchnęły z ulgą...

Podglądał: Eugeniusz MLECZAK

OLIVETTI M19

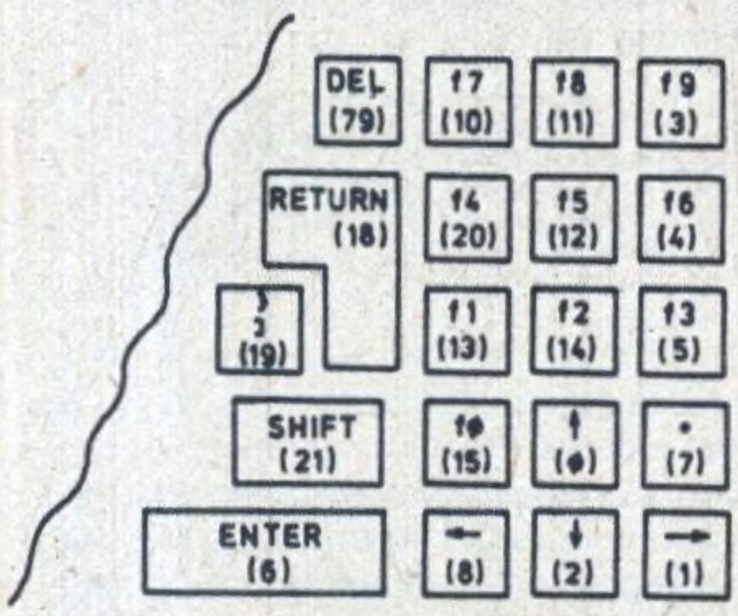
Komputer klasy XT przeznaczony do użytku jako komputer edukacyjny. Pracuje pod systemem MS-DOS 2.11 lub 3.1. M19, ma RAM 256 Kb natomiast pamięć zewnętrzną stanowić mogą floppy dyski po 360 Kb lub floppy dysk 360 Kb plus dysk twardy 20 Mb. Jeśli cena tego sprzętu będzie przyzwoita to ma szansę na szerokie zastosowanie nie tylko w ośrodkach naukowych, ale również w szkołach. Bogate oprogramowanie i możliwość współpracy z wieloma urządzeniami zewnętrznymi są najważniejszym atutem M19.

OLIVETTI M28

Nowy model opracowany w firmie Olivetti, kompatybilny z IBM. Może pracować pod MS-DOS i XENIX V.286 SCo — po raz pierwszy oferowanym w Polsce zgodnym ze standardem światowym. Bogate oprogramowanie narzędziowe jest atutem tego urządzenia. Pozostałe parametry: RAM 512 Kb MOS ROM z możliwością rozbudowy do 1 Mb; dwa tryby znakowe na monitorze monochromatycznym lub kolorowym — 80×25 lub 40×25 znaków; rozdzielczość obrazu 640×400, 640×200 lub 320×200 punktów.

dokończenie ze str. 29

przyporządkowane są określone numery (rys. 2). Instrukcja **INKEY** sprawdza, które klawisze zostały naciśnięte i generuje odpowiednią wartość odpowiedzi w zależności od naciśniętego klawisza. W naszej grze wykorzystamy klawisze sterujące ruchem kursora znakowego (strzałki w lewo, w prawo, w górę, w dół) mające numery odpowiednio: 8, 1, 0, 2.



Zasada sterowania ruchem ludzika jest następująca: w zależności od naciśniętego klawisza wyznacza się nowe współrzędne położenia ludzika (zwiększone lub zmniejszone o jeden znak), drukowany jest jego symbol graficzny, natomiast w miejscu określonym poprzednimi (starymi) współrzędnymi drukowana jest spacja: Aby ludzik „nie uciekł" nam z ekranu wprowadzimy odpowiednie ograniczenia na jego współrzędne.

Ujmiemy to w następujące podprogramy:

```
1040 rem w lewo
1050 if inkey (8) = 0 and x > 1 then gosub 1100 : x = x — 1
1055 rem w prawo
1060 if inkey (1) = 0 and x < 20 then gosub 1100 : x = x +1
1065 rem do góry
1070 if inkey (0) = 0 and y > 2 then gosub 1100 : y = y — 1
1075 rem w dół
1080 if inkey (2) = 0 and y < 25 then gosub 1100 : y = y + 1
1085 locate x,y : print chr$ (249);
1090 return
1100 locate x,y : print "";
1110 sound 3,500,1,,,13
1120 return
```

Napisz jeszcze:

```
200 gosub 1040
205 goto 200
```

Uruchom program i naciskając klawisze ze strzałkami zobaczysz jak ludzik porusza się po ekranie.

Naciśnij dwa razy klawisz [ESC], **skasuj wiersz 205,** powróć do poprzedniego trybu.

Piszemy nasz program dalej zgodnie ze schematem gry:

```
210 rem dojście do rakiety
220 if x = 20 and y = 25 then 420
230 rem wejście na niebezpieczny zbiornik
240 for i = 1 to n
250 if x = minx (i) and y = miny (i) then 340
260 next
270 rem trafienie na składnik paliwa
280 for i = 1 to m
290 if x = palx (i) and y = paly (i) then palx (i) = 0 : paly (i) = 0 : l = 1 + 1 : locate 18,1 : print l ; goto 200
300 next
310 goto 200
420 rem po dojściu do rakiety
430 if l < k then 580
440 rem start rakiety
450 forw = 1 to 150
460 sound 1,500—w, 2, 15—cint (0,1 * w)
470 if w > 19 then 510
480 locate 10,25 — w : print chr$ (239); : cls
490 ink 0, 9, 24 : locate 5, 12 : print „KONIEC"
500 ink 0, 1 : ink 1, 26
510 next
520 mode 1 : print „GRASZ JESZCZE? (T/N)"
530 d$ = inkey$
540 if d$ = "" then 530
550 if d$ = "T" or d$ = "t" then run
560 if d$ = "N" or d$ = "n" then 640
570 gotto 530
340 rem wybuch
350 sound 3,500,120,,,,10
360 for i = 1 to 7
370 locate x,y : print chr$ (144); : cls : locate x,y : print chr$ (239); : ink 0,9,24 : next : ink 0,1 : cls
380 zal = zal — 1
390 if zal = 0 then 520
400 gosub 1130 : goto 40
580 rem za mało składników
590 locate 2,1 : print "ZA MALO SKLADNIKOW";
600 for a = 1 to 1000 : next
610 locate 2,1 : print space$ (15);
620 locate 20,25 : print chr$ (239);
630 x = x — 1 : gosub 1130 : goto 200
640 mode 2
```

Uporządkujmy numerację linii naszego programu pisząc **RENUM** i ostatecznie program nasz wygląda następująco:

```
10 REM
20 MODE 0:zal=3:n=10:m=5:k=3:l=0
30 DIM minx(n):DIM miny(n)
40 DIM palx(m):DIM paly(m)
50 REM
60 FOR i=1 TO n
70 GOSUB 670
80 minx(i)=x:miny(i)=y
90 LOCATE x,y:PRINT CHR$(231);:NEXT
100 FOR i=0 TO 640 STEP 3
110 MOVE i,1:DRAWR 1,380
120 MOVE 640-i,1:DRAWR 1,380:IF i>320 THEN 140
130 NEXT
140 REM
150 FOR i=1 TO m
160 GOSUB 670
170 palx(i)=x:paly(i)=y
180 LOCATE x,y:PRINT CHR$(229);:NEXT
190 x=1:y=2:LOCATE x,y:PRINT CHR$(249);
200 LOCATE 20,25:PRINT CHR$(239);
210 GOSUB 840
220 GOSUB 710
230 REM
240 IF x=20 AND y=25 THEN 450
250 REM
260 FOR i=1 TO n
270 IF minx(i)=x AND miny(i)=y THEN 360
280 NEXT
290 REM
300 FOR i=1 TO m
310 IF palx(i)=x AND paly(i)=y THEN 340
320 NEXT
330 GOTO 220
340 palx(i)=0:paly(i)=0:l=l+1:LOCATE 18,1
350 PRINT l:GOTO 220
360 REM
370 SOUND 3,500,120,,,,10
380 FOR i=1 TO 7
390 LOCATE x,y:PRINT CHR$(144);:CLS:LOCATE x,y
400 PRINT CHR$(238);:INK 0,9,24:NEXT
410 INK 0,1:CLS
420 zal=zal-1:l=0
430 IF zal=0 THEN 550
440 GOSUB 840:GOTO 50
450 REM
460 IF l<k THEN 610
470 REM
480 FOR w=1 TO 150
490 SOUND 1,500-w,2,15-CINT(0.1*w)
500 IF w>19 THEN 540
510 LOCATE 10,25-w:PRINT CHR$(239);:CLS
520 INK 0,9,24:LOCATE 5,12:PRINT "K O N I E C";
530 INK 0,1:INK 1,26
540 NEXT
550 MODE 1:PRINT "GRASZ JESZCZE ? (T/N)" ;
560 d$=INKEY$
570 IF d$="" THEN 560
580 IF d$="t" OR d$="T" THEN RUN
590 IF d$="n" OR d$="N" THEN 880
600 GOTO 560
610 REM za malo skladnikow
620 LOCATE 2,1:PRINT "ZA MALO SKLADNIKOW";
630 FOR A=1 TO 1000:NEXT
640 LOCATE 2,1:PRINT STRING$(18," ");
650 LOCATE 20,25:PRINT CHR$(239);
660 x=x-1:GOSUB 840:GOTO 220
670 REM
680 x=CINT(RND*100):IF x<=1 OR x>=20 THEN 680
690 y=CINT(RND*100):IF y<=2 OR y>=25 THEN 690
700 RETURN
710 REM w lewo
720 IF INKEY(8)=0 AND x>1 THEN GOSUB 810 :x=x-1
730 REM w prawo
740 IF INKEY(1)=0 AND x<20 THEN GOSUB 810:x=x+1
750 REM do gory
760 IF INKEY(0)=0 AND y>2 THEN GOSUB 810 :y=y-1
770 REM na dol
780 IF INKEY(2)=0 AND y<25 THEN GOSUB 810:y=y+1
790 LOCATE x,y:PRINT CHR$(249);
800 RETURN
810 LOCATE x,y:PRINT " ";
820 SOUND 3,500,1,,,13
830 RETURN
840 REM
850 LOCATE 18,1:PRINT l;
860 FOR i=1 TO zal
870 LOCATE 8+i,1:PRINT CHR$(224);:NEXT:RETURN
880 MODE 2
```

Program możemy sobie uatrakcyjnić zmieniając np. ilości zbiorników (wartość n), ilości składników paliwa (wartość m), i ilość niezbędnych składników do wytworzenia paliwa (k) — zmiany w wierszu 20. Trzeba jednak zaznaczyć, że zbyt duże wartości n i m spowodują, że ludzik będzie się poruszał wolniej (przeglądanie coraz większych tablic). Może ktoś z Was zaproponuje inny, bardziej efektywny sposób poruszania się ludzika, bądź szybszego przeglądania tablic.

Wprowadzenie ograniczenia czasowego (np. w ciągu 30 sek.) na dojście do rakiety i wystartowanie na pewno zwiększy atrakcyjność gry. Życzę dobrej zabawy.

Cezary SOBCZAK

Liga Myślących

A oto wylosowane nazwiska Czytelników, którzy za prawidłowe rozwiązanie wszystkich zadań LIGI MYŚLĄCYCH z 5 nr. „IKS-a" otrzymują nagrody książkowe: Andrzej BARCEWICZ z Łomży, Sławomir RYDEL z Kolna, Wojciech WÓJCIK z Ostrowca Świętokrzyskiego, Paweł URBAN z Rzeszowa, Janina SZMYDT z Lubonia.

Oto wylosowane nazwiska Czytelników, którzy za prawidłowe rozwiązanie wszystkich zadań z 6 nr. „IKS-a" otrzymują nagrody książkowe: Marek ZDEB z Pszczyny, Marek KRUŻEL z Tarnowa, Barbara GOŁKOWSKA z Grudziądza, Ryszard BANIEWICZ z Bydgoszczy, Tadeusz FRYSKA z Poznania.

Za poprawne rozwiązanie wszystkich zadań z 7 nr. „IKS-a" książki otrzymują: Andrzej ERD z Radomia, Janusz GWÓŹDŹ z Zawiercia, Marian RABCZAK z Gdańska, Krzysztof TOPOLSKI ze Zbąszynia, Witold WRÓBEL z Katowic.

W siódmym numerze „IKS-a" skończyliśmy pierwszą edycję LIGI MYŚLĄCYCH. W siedmiu numerach „IKS-a" przedstawiono 27 zadań o różnym stopniu trudności. Wszystkie zadania rozwiązał prawidłowo i otrzymał maksymalną liczbę 37 punktów HUBERT NOWACZYK z Grudziądza. Został on zwycięzcą pierwszej edycji LIGI MYŚLĄCYCH i otrzymuje magnetofon MK 232 p.

Kolejne miejsca, z 36 punktami, zajęli:
Andrzej ERD z Radomia, Antoni GORYL z Krakowa, Janusz MORBITZER z Krakowa, Andrzej WRZEŚNIOWSKI z Krakowa i otrzymują radioodbiorniki ANETA.

Za miesiąc rozpoczynamy kolejną edycję LIGI MYŚLĄCYCH. Czekamy na propozycję zadań — najciekawsze trafią na łamy LIGI. Myślący do pracy!

Notowania komputerowe

GIEŁDA

ZX Spectrum Plus	125 tys. zł
ZX Spectrum 128 KB	195 tys. zł
Commodore 16 + magnet. 1531	100 tys. zł
Commodore C-64	170 tys. zł
Atari 800 XL	125 tys. zł
Atari 1050 + joystick 5 + 10 dyskietek	185 tys. zł
Magnetofon 1050	50 tys. zł
Stacja dyskietek 1050	170 tys. zł
Atari — programy: gry	300 — 500 zł
programy użytkowe	200 — 400 zł
programy kopiujące	500 — 700 zł
Amstrad 464 z monit. kolor.	320 tys. zł
Drukarka Seikosha GP 50S	100 tys. zł

Studio SYSTEM — Warszawa-Wola

Commodore C-16	92 tys. zł
Sharp M-701 z plotterem	150 tys. zł
Schneider (Amstrad) CPC 6128	885 tys. zł
Schneider (Amstrad) PCW 8256	1 200 tys. zł
Schneider (Amstrad) PCW 8512	1 540 tys. zł
Interface'y KEMPSON	10 200 zł
Interface'y SINCLAIR	10 620 zł
Dyskietki 3"	6,2 — 8,4 tys. zł
Dyskietki 5,25" 2D	1 071 zł
Dyskietki 5,25" 2D firmowe	3 500 zł
Zestaw programów SPECTRUM (min. 8)	1 600 zł
Bogata biblioteka ATARI	400 zł/szt.

dokończenie ze str. 12

3) **„Elementy i przyrządy półprzewodnikowe powszechnego zastosowania"** (cena 790 zł). Poziom conajmniej studencki. Michał Polowczyk zaczynając od fizyki półprzewodników, poprzez rezystory, halotrony i diody doprowadza nas do tranzystorów oraz elementów fotooptycznych.

Jest to bazą do trzech głównych części: obwody scalone analogowe, obwody scalone cyfrowe TTL oraz obwody scalone cyfrowe MOS. Wytrwały czytelnik pozna tajemnice „cegiełek" komputera.

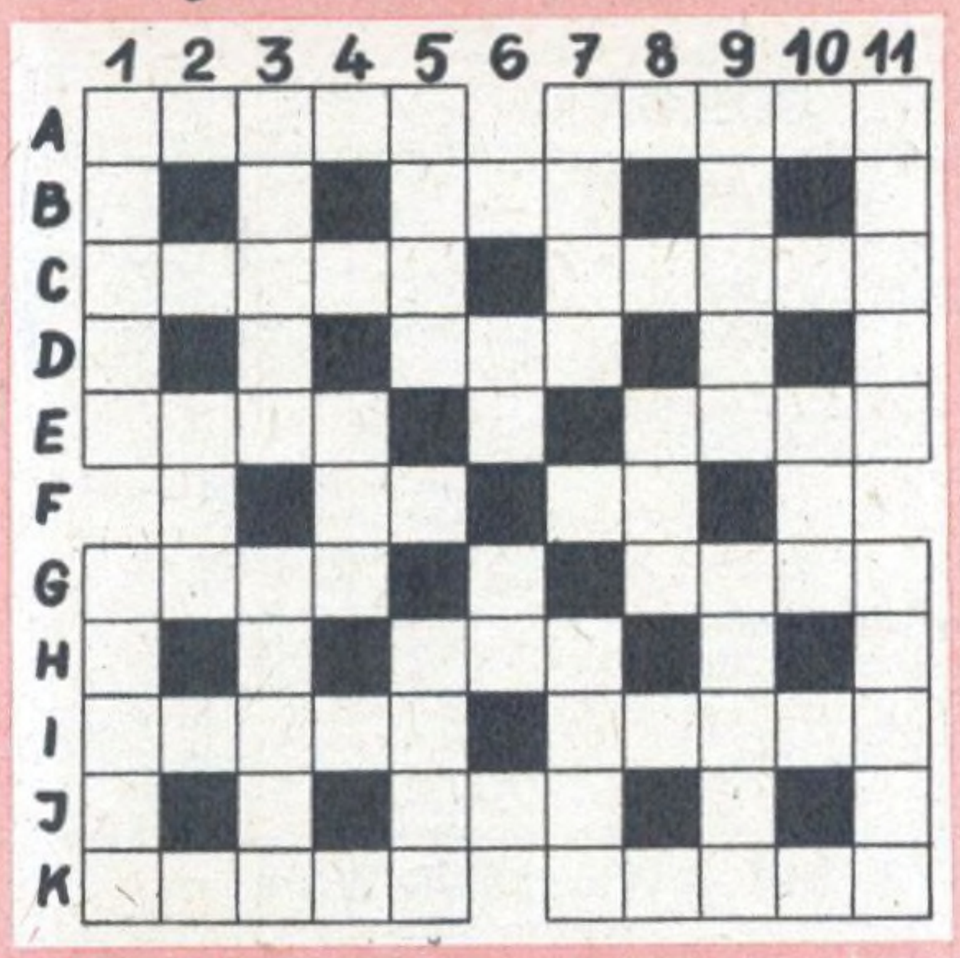
* * *

4) Użytkownicy ZX Spectrum mają wreszcie dobry podręcznik: **„Przewodnik po ZX Spectrum"** (cena 480 zł) K. Kuryłowicz, D. Madej i K. Marasek. Starali się wnikliwie omówić wszystkie zakamarki i możliwości tego najpopularniejszego w Polsce mikrokomputera. Może to być z pewnością książka dla początkujących, zaś zaawansowani znajdą w niej wskazówki, gdzie poszukiwać dalszych informacji. Uzupełnieniem wiadomości o zmiennych systemowych ROM jest też „Iksowski" cykl „Sztuki i sztuczki". A rozszerzeniem informacji o Basicu nasz kurs tego języka.

„Przewodnik" przyda się każdemu fanowi mikrokomputerów, często bowiem wykracza poza ramy Spectrum — ta wiedza z pewnością będzie pożyteczna. Szkoda jedynie, że na pierwsze tego typu wydawnictwo na tak dobrym poziomie — tak długo trzeba było oczekiwać. Uwaga nakład 150 000 tys.

C.M.

Krzyżówka nr 3



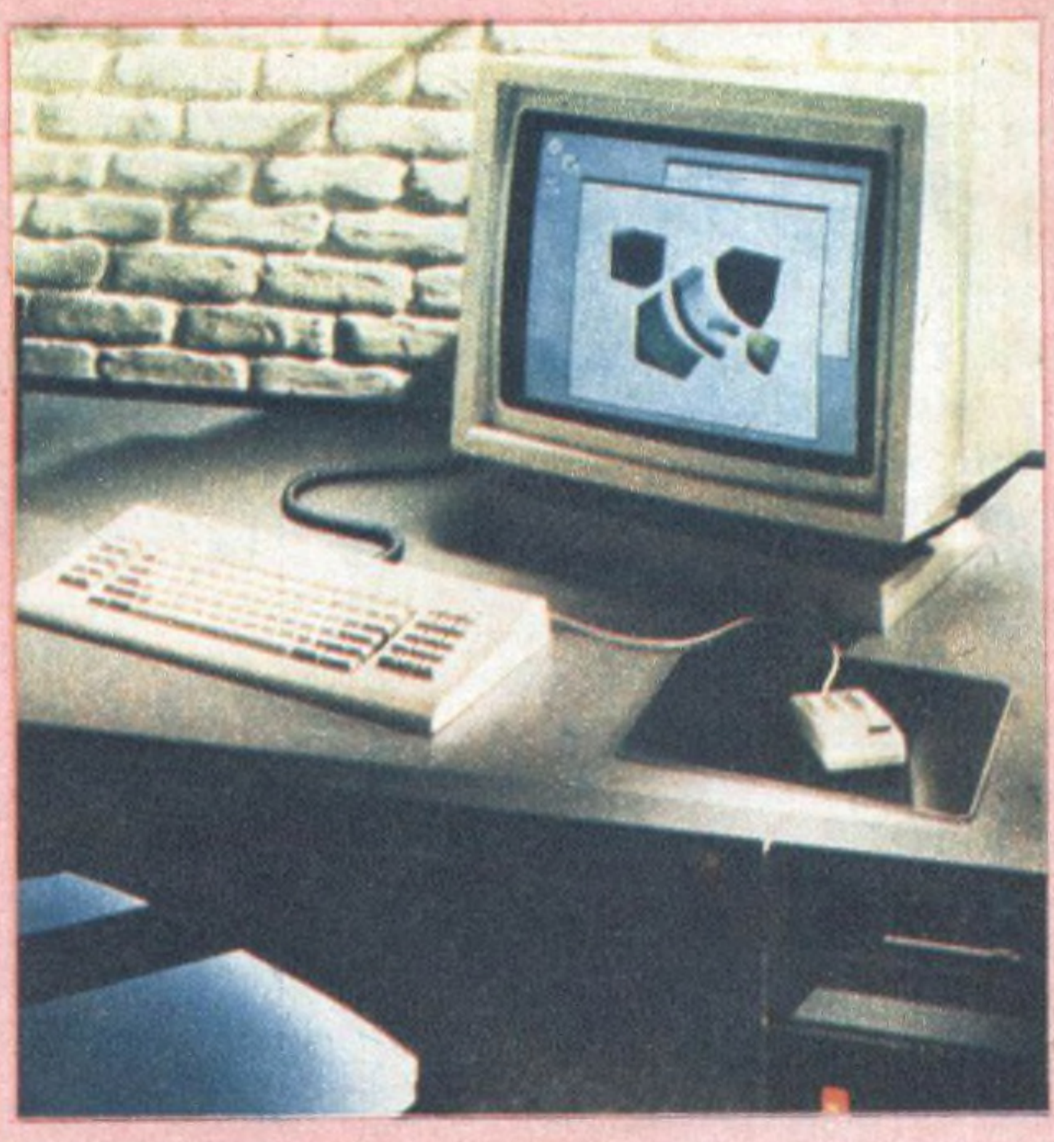
POZIOMO: A1) Może być na przykład aperturowa, dualna, indeksowa itp., A7) Sekwencja rozkazów, która została celowo określona przez programistę i jest wykonywana przez procesor w sposób powtarzalny, aż do momentu spełnienia założonego warunku, B5) Jednostka gęstości zapisu na magnetycznym nośniku danych, C1) Określona porcja np. promieniowania, C7) Uniwersalny język programowy stosowany w zagadnieniach ekonomicznych i zarządzania, D5) Jedno z wcieleń Janusza Gajosa, E1) Robocza nazwa komputerów „Jednolitego systemu", E8) Grecki Mars, F4) Jednostka oporności elektrycznej, F7) Stała część nazwy systemów operacyjnych dla komputerów rodziny RIAD lub firmy IBM, G1) Skrótowa nazwa angielskiego specjalistycznego systemu przetwarzania danych przeznaczonych do informowania kierownictwa, G8) Część ciała ludzkiego, H5) Potrawa na ratunek, I1) Osoba żyjąca przy klasztorze nie składająca ślubów, I7) Odwrotność różniczki, J5) Imię żeńskie, K1) Metropolita Jordanii, K7) W starożytnym Rzymie pieśń żałobna.

PIONOWO: A1) Urządzenie zmieniające symbol pewnego alfabetu na symbole innego alfabetu, G1) Obraz religijny w sztuce wsch.-chrześcijańskiej, E2) Skrótowa nazwa firmy będącej największym producentem sprzętu komputerowego na świecie, A3) Pokaz, parada, G3) Mahometanizm, E4) Ogólna nazwa systemów operacyjnych dla komputerów rodziny RIAD, lub firmy IBM zapewniających mniejsze możliwości aniżeli system OS, A5) Starożytne liczydła, H5) Pistolet maszynowy skonstruowany przez Shepparda i Turpina, D6) Przyjęta z języka angielskiego nazwa spójnika — lub, G6) Klasyczny japoński balet dramatyczny, A7) Nazwa systemu informatycznego przeznaczonego do kierowania produkcją, H7) Nazwa pakietu programów przeznaczonych dla komputerów serii Odra-1300 oraz ICL-1900 i stosowanych do kontroli zapasów magazynowych, E8) Stosowana w języku rosyjskim skrótowa nazwa zautomatyzowanego systemu zarządzania, A9) Środki transportowe, G9) Może być operacyjny (mnożenia, sumujący itp.), E10) Skrót oznaczający komputer hybrydowy, A11) Najpopularniejszy instrument muzyczny w starożytnej Grecji, atrybut Euterpe, G11) Lęk, strach.

Hasło: (D3, H1, H3) (B6, K10, F8, F5, B1) (E4, I3, E8) (I7, B7, B3, I2, E2, E10)

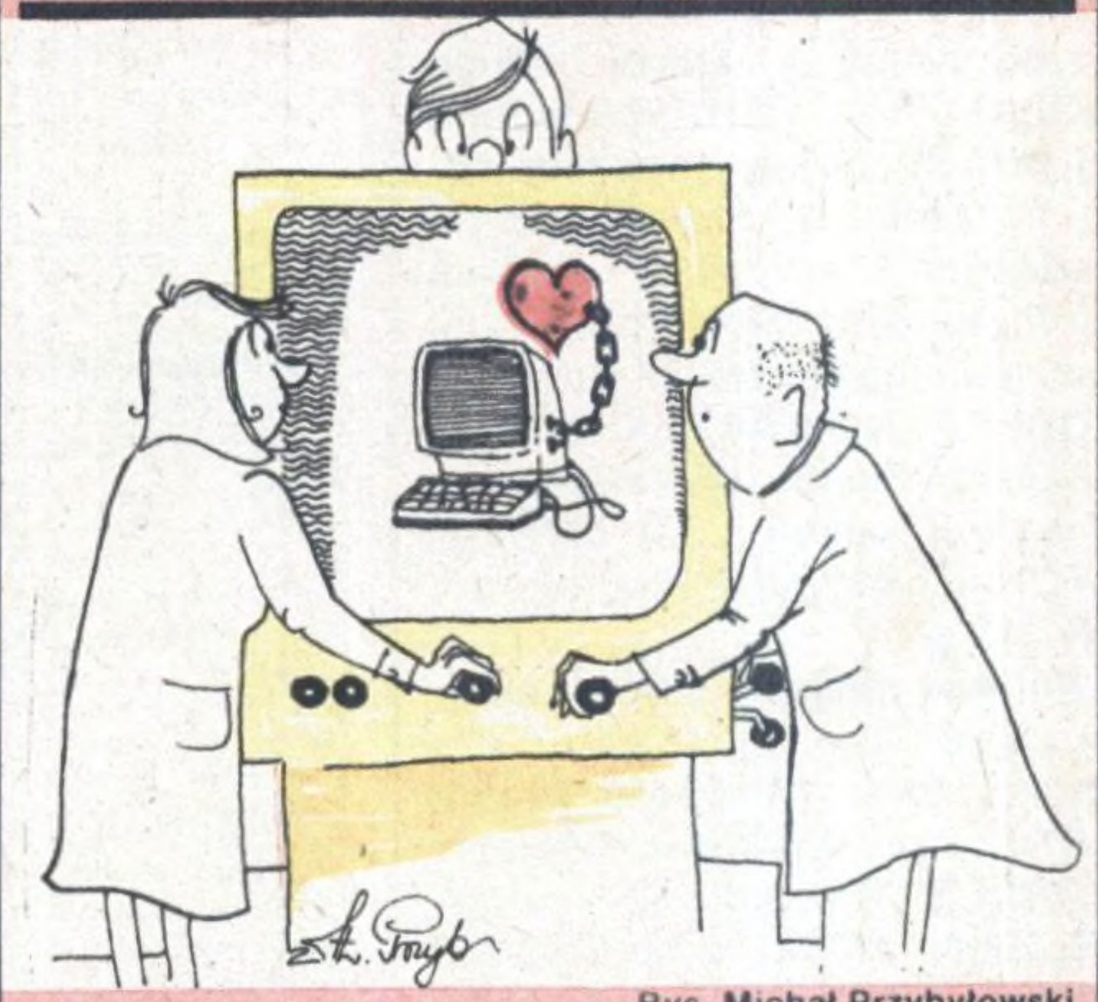
Rozwiązanie (tylko hasło) należy przesłać pod adresem redakcji do końca kwietnia br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

MIKROCIEKAWOSTKI



SUN — GRAFIK — SYSTEM

Szczególną właściwością tego urządzenia jest rozdzielczość jego ekranu, 1152 × 900 punktów. Komputer dysponuje 256 barwami, z których różne jasności tworzą paletę zawierającą 16 700 000 kolorów!



Rys. Michał Przybyłowski

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogon. 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. 8587. Nr. ind. 361682. K-71