INFORMATYKA KOMPUTERY SYSTEMY

CENA — 80 zł

DODATEK „ŻOŁNIERZA WOLNOŚCI" NR 9/1987 ISSN 0860-2794

Uwaga! Konkurs: „45 lat LWP" Atari dla zwycięzcy

— str. 2

Uczniowie rozpoczęli kolejny rok trudnej pracy. Dla wielu jest to również kolejny rok nauki z komputerem. Niestety, w niewielu przypadkach mikrokomputer jest w szkole rzeczywistą pomocą — najczęściej jedynie w pracowni informatycznej można spotkać to urządzenie. W najlepszej sytuacji są ci, którzy mają go w domu, ale czy takich jest większość?

Niestety, z informatyki uczyniono jedynie przedmiot nauczania i to jak się okazuje, dla wielu bardzo trudny, jakby zapominając zupełnie, że nie chodzi o wychowanie pokolenia informatyków, a o wykształcenie ludzi potrafiących ten nowoczesny sprzęt wykorzystać. Dlaczego zatem mikrokomputer tak rzadko obecny jest na lekcjach fizyki, chemii czy matematyki? Dlaczego najczęściej ważniejsza jest zasada działania mikroprocesora niż umiejętność posługiwania się podstawowymi programami użytkowymi?

Ideałem byłaby sytuacja, kiedy każda specjalistyczna pracownia miałaby swój sprzęt informatyczny i odpowiednie oprogramowanie. Tymczasem brakuje wszystkiego. Nie mamy nadal komputera szkolnego, brak jest odpowiedniego oprogramowania, nie ma również zbyt wielu chętnych nauczycieli do poznawania nowych przecież możliwości.

Dla tych jednak, którzy nie rezygnują, prezentować będziemy na łamach „IKS-a" nasze propozycje programów. Są to programy, których w wielu przypadkach autorami są nauczyciele wykorzystujący je na co dzień w swojej pracy pedagogicznej. Każda nasza propozycja może być wzbogacona dodatkowymi rozwiązaniami — liczą się jednak pomysły i tych będziemy dostarczać.

Tymczasem w nowym roku szkolnym życzymy samych piątek i sukcesów.

W NUMERZE:

Uwaga! Konkurs: „45 lat LWP"

Wojsko w komputerze

Czy historia, wydarzenia z przeszłości i współczesności dają się zapisać komputerowym kodem? Wiemy, że tak i dlatego proponujemy coś, czego jeszcze nie było!

45 lat temu, w trudnych wojennych czasach powstało ludowe Wojsko Polskie. Powstało by walczyć o wyzwolenie kraju, powstało by w pokojowych warunkach jak najlepiej służyć pomyślności ojczyzny. Dziesiątki wojennych wydarzeń, tysiące dobrych i złych ludzkich losów, wiele dni zapisanych pracą i służbą — przebogata mozaika tworzy 45-letnią historię wojska.

Książka, artykuł, film, opowieść — to źródła naszej wiedzy o niej. Chcemy je jednak wzbogacić. Niech także komputer uczy, zaciekawia i rozbudza historyczną pasję.

Dlatego ogłaszamy nasz nowy konkurs dla miłośników historii i... komputerów. Niech ten — tyleż zaskakujący co interesujący — mariaż zaowocuje dorobkiem, z którego skorzystamy potem wszyscy — w szkole, w domu i w klubie komputerowym.

Proponujemy:

1. Opracowanie scenariusza programu komputerowego, którego treścią będą tradycje i współczesność WP. Wyobraźnia plus wiedza niech wskażą co wybrać: czy epizod z bojowego szlaku, czy wybrany okres wojennych zmagań, czy też współczesny wątek związany z życiem lub działalnością wojska.

Dla autorów najlepszych scenariuszy przeznaczone są nagrody:

I miejsce — ATARI 130XE
II miejsce — 25 tys.
III miejsce — 20 tys.

oraz trzy wyróżnienia po 10 tys.

2. Opracowanie programu komputerowego na podstawie jednego z nagrodzonych scenariuszy lub na własnym, oryginalnym pomyśle.

W tej konkurencji programów komputerowych pierwszą nagrodą jest również mikrokomputer ATARI 130XE, zaś autorzy wyróżniających się programów zostaną uhonorowani sprzętem elektronicznym.

Regulamin konkursu

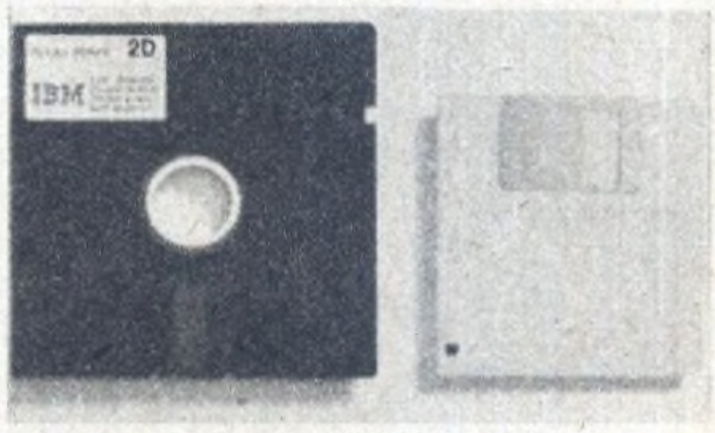
1) Zgłoszone do konkursu prace muszą być oryginalne
2) Termin nadsyłania scenariuszy programów (w trzech egzemplarzach ok. 5 stron maszynopisu) upływa 31.12.1987 roku. Ogłoszenie wyników tego etapu nastąpi w lutym na łamach „Żołnierza Wolności" i „IKS-a".
3) Autorzy programów, którzy podejmą się udziału w konkursie będą mogli otrzymać nagrodzone scenariusze w redakcji od 15 lutego 1988 roku.
4) Oceniane i klasyfikowane będą programy na Atari, Spectrum, Amstrad 464 i 6128 oraz Commodore 64.
5) Programy na kasetach magnetofonowych lub dyskietkach oraz scenariusze programów należy przesyłać pod adresem redakcji 00—950 Warszawa ul. Grzybowska 77.

M * I * K * R * O * S * E * R * W * I * S

Dyskietki 3 1/2 cala dla mikrokomputera

IBM PC/XT i IBM PC/AT

Uzytkownicy komputerów rodziny IBM PC mogą obecnie wykorzystywać programy zapisane na dyskietkach 3 1/2 cala. Umożliwia to większą elastyczność pracy z tymi komputerami .



Napędy dyskowe 3 1/2 cala mogą być dołączone do mikrokomputera IBM PC / XT lub IBM PC/AT , pracującego pod kontrolą systemu operacyjnego DOS 3.20. Transmisja zbiorów i programów użytkowych między stacjami dysków 5 1/4 '' i dołączoną , zewnętrzną stacją dysków 3 1/2 '' jest tak łatwa, jak wykonanie kopii zastępczej dyskietki (BACKUP) .Zastosowanie napędów 3 1/2 cala daje mozliwość łatwego przenoszenia oprogramowania, pomiędzy komputerami używającymi dyskietek 5 1/4 i 3 1/5 cala . Dyskietki 3 1/2 ''posiadają dwukrotnie większą pojemność, 720 KB pozwala na przechowywanie ponad 350 stron maszynopisu .

B u l l M i c r a l 60

Mikrokomupter firmy **B U L L**, *jest kompatybilny z IBM PC/AT. Zastosowano w nim : 16-bitowy mikroprocesor 80286, koprocesor 80287, pracujące z f zegarowa 6 / 8 MHz. Na płycie systemowej znajduje się RAM o pojemności 512 KB , która można rozszerzyć maksymalnie do 3 640 KB. Pamięć masową stanowią : stacja dysków elastycznych o pojemności 1.2 MB i dysk stały 40 MB . Firma proponuje cztery typy klawiatury, w tym QWERTY . Karty graficzne umożliwiają : ● tryb alfanumeryczny 25 linii po 80 znaków ; ● tryb graficzny : z kartą monochromatyczną 720 punktów w 348 liniach ; średnią i wysoką rozdzielczość z kartą kolorową czyli 320 punktów w 200 liniach z paletą 16-tu kolorów lub dwoma paletami po 4 kolory. Micral60 współpracuje z dukarkami PRT1910, 1911, 1912. Systemy operacyjne komputera to DOS 3.10 lub CP/M . Języki progra - mowania : BASIC (kompilator i interpreter), C, COBOL, FORTRAN, PASCAL.*

□□ ■ □□ **M I K R O D E. P E S Z E** □□ ■ □□

Firma MOTOROLA zamierza w pierwszym kwartale tego roku uruchomić produkcję mikroprocesora typu RISC , który będzie wykonywał 20 milionów instrukcji na sekundę . Nowy procesor oznaczony zostanie symbolem **78000** . ■ □□ ■ □□ ■ *OEM Marketing wprowadziła na rynek Gorąca Kartę, która kontroluje temperaturę wewnątrz komputera. Alarm termiczny włączany jest, gdy temperatura przekracza 50 stopni C. Jaki będzie następny produkt ? Zimna karta ?* ■ □□ ■ □□ ■ *Firma Maxell pracuje nad dyskami 5 1/4 cala o pojemności* **100 MB** . *Dyski o rekordowej pojemności mają pojawić się na rynku już w tym roku .*

opracowali: Jacek Wojtala i Mariusz Jarzębowski

Komputer w medycynie

Tomografia a układ krwionośny

W latach siedemdziesiątych naszego wieku obraz diagnostyki medycznej uległ, za pomocą elektronicznej techniki obliczeniowej, jakościowej zmianie. W oparciu o istniejące już metody bezinwazyjnego badania organizmu ludzkiego np. rentgenografię, jak i o nowe rewelacyjne odkrycia fizyki nuklearnej np. zjawisko magnetycznego rezonansu jądrowego, zbudowano szereg urządzeń obrazujących na monitorze telewizyjnym wnętrze ciała ludzkiego. W poprzednich artykułach poznaliśmy już rentgenowski tomograf komputerowy oraz tomograf magnetycznego rezonansu jądrowego. Pierwszy z nich w oparciu o przetworzone przez komputer dane, uzyskane podczas serii prześwietleń pacjenta promieniami X, umożliwia uzyskanie obrazu wideo dowolnego narządu wewnętrznego pacjenta. Gdy obraz zewnętrzny badanego narządu, uzyskany techniką rentgenowskiej tomografii komputerowej nie odbiega od normy, można zastosować drugi z wymienionych tomografów dla upewnienia się czy komórki budujące dany narząd nie podlegają wczesnym zmianom chorobowym. Badania tomograficzne dają więc dużą gwarancję prawidłowej diagnozy lekarskiej, a także ułatwiają śledzenie postępu choroby w każdym, nawet bardzo wczesnym jej stadium rozwojowym. W przypadku chorób nowotworowych, gdy od szybkości rozpoznania zmian chorobowych zależy pozytywny wynik leczenia, jedynie urządzenia tomograficzne dają szansę stłumienia choroby w zarodku. Zarówno rentgenowski tomograf komputerowy, jak i tomograf magnetycznego rezonansu jądrowego są urządzeniami niesłychanie kosztownymi, a przez to wciąż unikalnymi w normalnej praktyce diagnostycznej.

Oto dwie powszechne dziś metody diagnostyki medycznej, których wspólną cechą jest to, **że są prostymi realizacjami prostych pomysłów**. Celem ich twórców nie było uzyskanie dokładnego, trójwymiarowego obrazu badanego narządu ani zbadanie właściwości struktur atomowych narząd ten budujących. Chcieli oni uzyskać maksimum informacji o systemie zasilającym każdą komórkę ciała ludzkiego, czyli układzie krwionośnym, odgrywającym kluczową rolę w prawidłowym funkcjonowaniu organizmu. Dla uzyskania czytelnego obrazu telewizyjnego naczyń krwionośnych w obu przypadkach użyto znaczników kontrastujących, podawanych dożylnie pacjentowi. Zadaniem ich jest — jak sama nazwa wskazuje, nadanie krwi własności kontrastujących, a więc innych od własności otoczenia. Pierwszy pomysł po ubraniu go w materialne kształty, uzyskał nazwę **tomografu emisji pozytronowej**. W tej metodzie jako znacznika kontrastującego, użyto substancji radioaktywnej, której rozchodzenie się w naczyniach krwionośnych może być śledzone czujnikami promieniowania gamma. Czujniki przekazują wyniki swojej pracy do pamięci komputera, który zamyka cykl badania, porządkując uzyskane dane i dając w efekcie kolorowy obraz wideo naczyń krwionośnych badanego narządu.

Dla dokładniejszej ilustracji tej techniki diagnostycznej opiszemy badanie serca przy użyciu tomografu emisji pozytronowej. Pacjentowi podaje się wówczas dożylnie znacznik radioaktywny w czasie połowicznego rozpadu 10 minut. Znaczy to, że przez pierwsze 10 minut znacznik straci połowę swej aktywności, przez następne 10 minut połowę pozostałej radioaktywności itd. Gwarantuje to pacjentowi bezpieczeństwo, a od lekarza natomiast wymaga szybkości i precyzji w działaniu. Najpierw serce badane jest w stanie normalnej pracy, po czym pobudzane jest do szybkich skurczów specjalnym, łagodnym narkotykiem dozowanym pacjentowi dożylnie. Komputer zapamiętujący informacje o stanie ukrwienia serca w czasie całego badania, jako rezultat obliczeń podaje dwa obrazy naczyń krwionośnych serca: w czasie normalnej pracy i w stanie jego pobudzenia. Porównanie tych obrazów pozwala na wykrycie zaburzeń pracy serca lub zatorów w naczyniach krwionośnych, co jest symptomem zbliżającego się zawału. Dobre efekty daje użycie tomografu emisji pozytronowej do badania mózgu, umożliwiając wykrycie i w konsekwencji leczenie chorób psychicznych.

Drugi pomysł wizualizacji systemu krwionośnego człowieka zaowocował budową **tomografu odejmującego**. Tomograf ten, mówiąc w dużym uproszczeniu, to aparat rentgenowski sprzężony z komputerem. Badanie polega na wykonaniu serii prześwietleń rutynowych pacjenta, których wyniki rejestrowane są w pamięci komputera wraz z wynikami drugiej serii prześwietleń, lecz tym razem z podaną pacjentowi dożylnie substancją kontrastującą, pochłaniającą promieniowanie X. Komputer porównuje, uzyskane w ten sposób dwa zbiory danych, z których oczywiście drugi jest bogatszy o wielkości charakterystyczne dla układu krwionośnego, uwydatnionego przez zawarty we krwi kontrast. Właśnie dane różniące pierwszy zbiór od drugiego dotyczą układu krwionośnego i są materiałem obrabianym przez komputer, który daje na wyjściu graficzne przedstawienie naczyń krwionośnych badanego narządu. Tomograf odejmujący utorował drogę nowoczesnej technice usuwania zatorów żylnych oraz tamowania krwotoków wewnętrznych. Zabiegi takie, wykonywane dotychczas operacyjnie przez chirurgów, przeprowadzane są obecnie w sposób bezbolesny i nie wymagają długiej rekonwalescencji — przeprowadza się je za pomocą sondy wpuszczonej do tętnicy lub żyły wprawną ręką lekarza. Na ekranie tomografu śledzony jest komputerowy obraz sondy przesuwającej się wzdłuż naczyń krwionośnych do miejsca zatoru lub krwotoku. Następnie — przez sondę podawane są środki rozpuszczające zator lub hamujące przepływ krwi. Zabiegi takie trwają około 90 minut i zwykle kończą się pełnym sukcesem. Fotografia zamieszczona powyżej przedstawia jeden z bardziej wyspecjalizowanych tomografów odejmujących, zapewniający obraz badanego narządu pod dwoma kątami widzenia. Na 4 monitorach umieszczonych przy konsoli tomografu prezentowane są równocześnie: naturalny obraz serca, dwa obrazy opracowane przez komputer oraz dokładna historia choroby. Wyniki te uzyskujemy za naciśnięciem jednego guzika, a lekarzom pozostaje już tylko... leczenie.

AT

Co on ci takiego powiedział?!

Niepowodzenia sztucznej inteligencji

W. GOGOŁEK

Od dawna, a ostatnio coraz częściej, informatyczna literatura profesjonalna zwraca uwagę na zagadnienia związane ze sztuczną inteligencją (AI — Artificial Inteligence). Rangę tego problemu podkreślają oczekiwania olbrzymiego postępu wdrożeń tego typu systemów, zarówno przez profesjonalistów, jak i innych użytkowników mikro. Jednak rzeczywistość w dużej części nie dorównuje planom. Przykładem tego jest „interfejs naturalnego języka" do Lotusa 1—2—3. Idzie o to, by móc sterować pracą programu, przekazywać do niego polecenia i pytania stosując codzienny język „mówiony". Omawiany interfejs umożliwia wykorzystanie wszystkich zalet Lotusa — poprzez konwersację. Producent — GNP Development Hal — dopiero po roku od ogłoszenia gotowości dystrybucji „interfejsu" rzeczywiście rozpoczął jego sprzedaż. Opóźnienie to wynikało głównie ze względów na różnice pomiędzy proponowanymi możliwościami programu, a oczekiwaniami użytkowników w zakładach pracy, uczelniach i innych instytucjach. Okazało się bowiem, że pierwotna uniwersalność programu musiała być zastąpiona oprogramowaniem o ściśle określonej specjalizacji. Praktyka jeszcze raz okazała się bezwzględnym sędzią założeń teoretycznych, dowiodła, że zwrot „sztuczna inteligencja" sam siebie „oskarża" — obiecuje bowiem coś magicznego, co faktycznie nie jest możliwe do osiągnięcia. Wydaje się to typowym „problemem armat" — inteligencja nie jest dostatecznie poznana, rozumiana, by ją tworzyć.

Użytkownicy komputerów w zasadzie nie oczekują tego, co rzeczywiście określone jest jako sztuczna inteligencja. Tak jak tłumaczą ją w takich dyscyplinach nauki jak: psychologia, lingwistyka, a także informatyka. Praktycznie AI można opisać jako nieosiągalną zdolność urządzeń elektronicznych.

Zazwyczaj dostęp do sprzętu o możliwościach, które wcześniej oceniane były jako sztuczna inteligencja, zmienia naszą ocenę zdolności tego sprzętu — staje się on nowym „normalnym" produktem. Analizując to zagadnienie przytaczany jest przykład samochodu, który „rozmawia" z kierowcą. Zgodnie z tradycyjnymi wyobrażeniami było to uznawane jako „sztuczna inteligencja". Okazało się jednak, że w tym, jak i wielu innych przypadkach, liczy się nazwa, określenie, a nic ponadto — samochód pozostał samochodem, tyle że z jeszcze jednym urządzeniem stanowiącym o komforcie jazdy.

Zazwyczaj sztuczna inteligencja stanowi uzupełnienie (otoczenie przestrzeni, „muszlę") systemów typu ekspert — **„ekspert-systemów"**. Mamy z nią do czynienia pod postacią interfejsu komunikacji z użytkownikiem w naturalnym języku. Użytkownik może sterować programami używając komend wyrażanych codziennym językiem angielskim; pisząc polecenia na klawiaturze, także wypowiadając je do mikrofonu sprzężonego z komputerem. Coraz częściej pojawiają się także systemy wzbogacone o „komputerowe widzenie" — „okiem" jest odpowiednio sprzężona z maszyną kamera.

Większość prac nad produktami typu AI dla personalnych komputerów dotyczy ekspert-systemów oraz naturalnego języka.

W przyszłym roku spodziewane są na światowym rynku systemy komputerowe, które z powodzeniem będą zastępowały sekretarki — na razie w kwestii pisania pism dyktowanych przez szefa... Natomiast wspomniane „widzenie maszynowe" obecnie praktycznie wykorzystywane jest masowo tylko w przemyśle m.in. samochodowym.

Ekspert-system jest to program komputerowy, który prezentuje wiedzę eksperta np.: lekarza, inżyniera, tłumacza. Okazuje się, że inwestowanie w tego typu oprogramowanie gwarantuje aktualnie najszybszy zwrot kosztów ponoszonych na badania nad sztuczną inteligencją.

Gromadzenie wiedzy w bazach danych, dostępnych ekspert-systemom, manipulowanie nią, diagnozowanie problemów wraz z sugerowaniem czynności prowadzących do ich rozwiązania, szacowanie ryzyka określonego przedsięwzięcia jest o wiele łatwiejsze niż stworzenie programu rozumiejącego naturalny, mówiony język.

Przykładem wyspecjalizowanego ekspert-systemu jest „Analizator zbożowego rynku", łączy on w sobie naturalny język oraz „wiedzę" niezbędną do wiarygodnej pomocy (rady) farmerom poszukującym najlepszej rynkowej strategii dla ich zboża (produkcji i handlu) — bazując na spodziewanych cenach i innych parametrach rynku.

Podobne systemy budowane są dla potrzeb przetwarzania tekstów, baz danych, a także... produkcji i handlu winem.

Postęp w tworzeniu programów rozpoznających naturalny język jest znacznie trudniejszy niż postęp budowy ekspert-systemów. Produkty bowiem „naturalnego języka" nie funkcjonują samodzielnie. Mają na celu ułatwić używanie ściśle określonego programu i w rzeczywistości są tylko jego uzupełnieniem (otoczeniem, muszlą). Od czasu, gdy każdy pakiet programowy musi mieć swego rodzaju interfejs dla użytkownika — język naturalny jest oczywistym wyborem. Należy przy tym pamiętać, że technika „naturalnego języka" ma o wiele dłuższą historię niż ekspert-systemy.

Pierwszym produktem wykorzystującym naturalny język była wprowadzona w połowie 1983 roku baza danych R:base 460, następnie także baza danych, tym razem firmy Q&A. Wydawanie komend sterujących pracą tych programów odbywa się za pośrednictwem zdań wypowiadanych w języku naturalnym (angielskim). Użytkownicy zmieniają, wprowadzają i wyszukują dane „rozmawiając" z maszyną. Program ten często nazywany jest „inteligentnym asystentem" — szybko „rozumie" wielokrotne warunki. Na przykład: „podaj mi nazwiska osób, których pensja łącznie z innymi dochodami jest większa od 10 000$".

Niektóre programy języków naturalnych, zamiast dokonywać analizy gramatycznej rozpoznawanego zdania, wykorzystują analizę częstotliwościową występowania liter w słowach, funkcjonując w efekcie, jak gdyby „rozumiały" angielski. Pierwszym przykładem tego typu programu był MindReader. Następnie pod koniec ubiegłego roku pojawił się Write Now, który poza typowymi zdolnościami przetwarzania tekstów, pomaga użytkownikowi uniknąć robienia błędów ortograficznych. Podczas pisania programu sprawdza każdy znak słowa (porównując z pamiętanym słownikiem ortograficznym). Gdy napiszesz błędną literę — program wyświetla ją w negatywie, „buczy" i sugeruje poprawkę.

Prawdopodobnie najbardziej wyróżniającym się przykładem „wbudowanej" techniki języka naturalnego jest „Hal". Jest to narzędzie wspomagające użytkowanie znanego od dawna programu Lotus 1-2-3. „Język naturalny" znacznie rozszerzył możliwości Lotusa. Użytkownik może np.: wydać polecenie w postaci: „wyświetl nazwy i liczby produktów typu X", wówczas Hal „samodzielnie" odnajdzie poszukiwane informacje i wyświetli lub wydrukuje na drukarce. Można wydawać także bardziej

➡

złożone polecenia — np.: „wymień nazwy artykułów droższych od 100 $". Producent Lotusa zdając sobie sprawę z odpowiedzialności stosowania zwrotu „sztuczna inteligencja" — nie stosuje go w żadnym kontekście przy nazwie Hal.

Przykładem ekspert-systemu jest Paradox. Posiada on tę samą moc obliczeniową jak programowana baza danych typu dBase lub R:base. Zasadniczą techniką w Paradox'ie jest sposób pytania o wiedzę w nim zgromadzoną — poprzez przykłady. Także jego „ekspertowa wiedza" gromadzona jest w podobny sposób — użytkownik podaje przykłady określonych sytuacji, które są odpowiednio analizowane i zapisywane do bazy.

Jedną z istotnych przyczyn stosunkowo wolnego „zdobywania" przez ekspert-systemy rynku jest kosztowne przygotowanie ich użytkowników (idzie tu o nauczenie zmian, dostosowania struktury systemu do specyficznych zastosowań). Na przykład systemy M.1, S.1 firmy Teknowledge kosztują po 5000 $, a nauka ich wykorzystania aż 2500 $!

Ekspert-systemy realizują polecenia wydawane zgodnie z określonymi zasadami (od 40 do 500 zasad). Wyrażają one w ogólności zdania typu: czy (wówczas in/then). Warto w tym miejscu wspomnieć plany sprzed 10 lat, kiedy to zakładano, iż zasad tych będą tysiące...

W praktyce stosuje się obecnie względnie małe systemy, które szybko „się spłacają". Niewielkie ekspert-systemy są bardziej ekonomiczne dla obu stron — producenta i użytkownika. Dowodzą tego doświadczenia ze sprzedaży i eksploatacji tańszych, ale bardziej sprawdzonych („dopasowanych" do konkretnego zastosowania) narzędzi. Przykładami są wyprodukowane w ostatnich 10 latach systemy: 1 st Class, Insight 2 Plus, Personal Consultant Easy, the Arity Expert System Development Package i Wisdom XS. Systemy te mają moc podobną do narzędzi sprzed 10 lat, ale kosztują $^1/_{10}$ ówczesnej ceny i generalnie są łatwiejsze w wykorzystaniu.

Przykładem „życzliwego", o dużej mocy, ekspert-systemu jest Personal Consultant Series (produkt Texas Instruments), a także o dwa lata młodsze jego klony". Dołączone do nich poprawki to między innymi „okna" (fragmenty ekranu informujące o stanie i realizacji programu) oraz znacznie dokładniejsza dokumentacja.

Nowy Personal Consultant Series reprezentuje wiedzę uporządkowaną w struktury. Termin ten został wprowadzony przez Marvin'a Minsky'ego — jednego z inicjatorów prac nad AI w latach pięćdziesiątych. Struktury dzielą wiedzę na części, które są następnie zapamiętane tak, że opisują tę wiedzę w hierarchicznych relacjach. Dla przykładu, silnik samochodu powinien być widziany jako struktura, która jest rozłożona na elementy takie jak: system elektryczny, system chłodzenia (każdy subsystem jest także strukturą) itp.

Uporządkowana w ten sposób wiedza stwarza większe możliwości programowego manipulowania na niej — przeglądanie, wybieranie, łączenie itp.

Podsumowując należy podkreślić, iż nie osiągnięto zamierzonego postępu w produktach sztucznej inteligencji. Skoncentrowano się natomiast na systemach specjalistycznych i uzyskano na tym polu pewne sukcesy. Podobnie systemy „naturalnego języka" stają się coraz mniej uniwersalne — rozwijane są specjalistyczne oprogramowania przeznaczone dla określonych grup użytkowników.

Wydaje się, że zjawisko to — zwolnienia — stanowić może symptom wyhamowania gwałtownego postępu oprogramowania profesjonalnego sprzętu mikrokomputerowego.

W. GOGOŁEK

Opracowane na podstawie: H. Fersko Weiss, 1986 hits and misses AI UPDATE Personal Computing/December 1986.

W szponach Atari (7)

Igraszki z duchami

Ludwik PIELA,
Tomasz MROWIEC

Kłopoty z animacją

Dowolny domowy system komputerowy byłby niewiele wart bez możliwości animacji. Czyż można wyobrazić sobie efektowną grę bez ruchu obiektów na ekranie?

Konwencjonalnym sposobem uzyskania animacji jest przesuwanie obrazu w obszarze pamięci ekranu. Odbywa się to w dwóch etapach. Po pierwsze, program musi wykasować stary obraz poprzez zapisanie wartości tła do RAM zawierającej aktualny obraz. Następnie trzeba zapisać dane obrazu w obszarze pamięci odpowiadającym nowej pozycji obrazu. Poprzez ciągłe powtarzanie tego procesu uzyskujemy przemieszczanie się obrazu po ekranie.

Technika ta stwarza dwa problemy. Pierwszy, jeśli animacja wykonywana jest w trybie graficznym z dużymi punktami elementarnymi, ruch nie będzie płynny; obraz będzie przeskakiwał przez ekran. Dla innych komputerów jedynym rozwiązaniem jest użycie trybu graficznego o wysokiej rozdzielczości. Drugi problem jest o wiele gorszy. Obraz na ekranie jest dwuwymiarowy, a obszar pamięci ekranu jest jednowymiarowy. Oznacza to, że obraz, który jest ciągły na ekranie, nie będzie ciągły w RAM.

Znaczenie tego stanie się oczywiste jeżeli spróbujemy napisać program do przesuwania takiego obrazu. Przyjrzyjmy się, jak rozproszone są w pamięci bajty tworzące obraz. W celu ich wykasowania program musi obliczyć ich adresy, co nie zawsze jest łatwe do zrobienia. Poza tym nawet w kodzie maszynowym obliczenia zajmą trochę czasu. Oznacza to, że ten typ pamięci (zwany „animacją pola gry") jest zbyt wolny dla wielu celów.

Rozwiązaniem tego problemu jest technika wykorzystująca „sprite'y" czyli „duchy". W nomenklaturze Atari nosi ona nazwę grafiki gracz-pocisk (ang. player-missile graphics). Gracze (players) i pociski (missiles) są specjalnymi obiektami graficznymi przeznaczonymi do szybkiego przemieszczania się po ekranie.

Udało się utworzyć obiekt graficzny, który jest jednowymiarowy zarówno na ekranie jak i w RAM. Taki obiekt (zwany graczem) występuje w pamięci jako tablica (ciąg bajtów) o długości 128 lub 256 bajtów. Pojawia się jako pionowy pasek rozciągający się od góry do dołu ekranu. Każdy bajt tablicy odwzorowywany jest w jednej lub dwóch poziomych liniach ekranu (wybór należy do nas). Obraz jest bezpośrednim odwzorowaniem bitów danych w tablicy. Jeśli bit jest równy 1; wtedy zapalony jest odpowiadający mu punkt elementarny; jeśli bit jest równy 0, punkt jest zgaszony.

Grafika gracz-pocisk jest niezależna od innych trybów graficznych Atari, które nazywane są grafikami pola gry (ang. playfield). Instrukcje PRINT, PLOT, DRAWTO realizują grafikę pola gry. Technikę gracz-pocisk należy traktować jako drugi obraz na ekranie. Granice tego obrazu przekraczają wielkość pola gry i nie są związane z aktualnym trybem graficznym. Ponadto obrazy te mogą występować jakby były przed lub za grafiką pola gry na ekranie.

Dodatkowo grafika gracz-pocisk zwiększa ilość kolorów na ekranie. Każdy obiekt typu gracz ma swój własny rejestr koloru, rejestry te są niezależne od używanych dla grafiki pola gry. Bez względu na używany tryb zawsze będą dostępne cztery dodatkowe rejestry koloru dla obiektów graczy. Dzięki temu w trybie 0 (BASIC'u) można uzyskać na ekranie pięć kolorów.

Początkowo nie jest łatwo zrozumieć grafikę gracz-pocisk (G-P), ponieważ Atari BASIC nie ma odpowiednich instruk-

cji. Jej organizacja i wykorzystanie są przez to bardziej absorbujące, ponieważ muszą być realizowane na poziomie języka maszynowego. Czyni to tworzenie programów znacznie trudniejszym.

Definiowanie gracza

Każdy obiekt typu gracz ma szerokość ośmiu bitów. W celu zdefiniowania obiektu musimy utworzyć jego odwzorowanie w pamięci. Najpierw kreślimy żądany rysunek na papierze milimetrowym. Następnie tłumaczymy go na kod binarny, wstawiając jedynki dla punktów zapalonych i zera dla zgaszonych. Odwzorowaniem gracza jest zatem ciąg liczb jednobajtowych, które wstawimy do tablicy grafiki G-P. Przed wpisaniem musimy je przetłumaczyć na liczby dziesiętne lub szesnastkowe.

Lepszym rozwiązaniem jest wykorzystanie programu nr 1, który uwalnia nas od żmudnych obliczeń i umożliwia kreślenie kształtu obiektu bezpośrednio na ekranie za pomocą manipulatora drążkowego. Zauważmy, że ramki ograniczają nas do definiowania ośmiu komórek obrazu w poziomie. Po zakończeniu projektowania wyglądu gracza naciśnijmy klawisz (RETURN) i obraz gracza pojawi się w swojej naturalnej wielkości. Jeśli chcemy dokonać dodatkowych zmian w kształcie obiektu, musimy nacisnąć klawisz spacji. Po ponownym pojawieniu się migającego kursora, znowu możemy użyć manipulatora do zmiany obrazu gracza. Poprzez naciśnięcie (RETURN), po obejrzeniu kształtu gracza naturalnej wielkości, zobaczymy na ekranie odwzorowanie binarne obiektu wraz z innym spojrzeniem na utworzony obraz.

Obiekt typu gracz może być zdefiniowany w 128 lub 256 bajtach. Gdy zdefiniowany jest w 128 bajtach, każdy jego bajt zajmuje dwie linie ekranu. Gracze zdefiniowani w 256 bajtach będą używać tylko jednej linii ekranu dla każdego bajtu obiektu.

Zauważmy, że obiekty graczy różnią się swoimi wielkościami na obrazie. Gracze zdefiniowani w 256 bajtach mają rozdzielczość pionową dwa razy większą niż obiekty 128-bajtowe i są na ekranie mniej „zblokowane". Powinniśmy zdecydować, czy potrzebujemy tej dodatkowej rozdzielczości. Ponieważ wszyscy gracze muszą być zdefiniowani z taką samą długością, decyzja ta może nam zaoszczędzić trochę pamięci. Gracze zdefiniowani w 128 bajtach nazywani są graczami o rozdzielczości dwuwierszowej, a gracze zdefiniowani w 256 bajtach są zwani graczami o rozdzielczości jednowierszowej.

Tablica grafiki gracz-pocisk

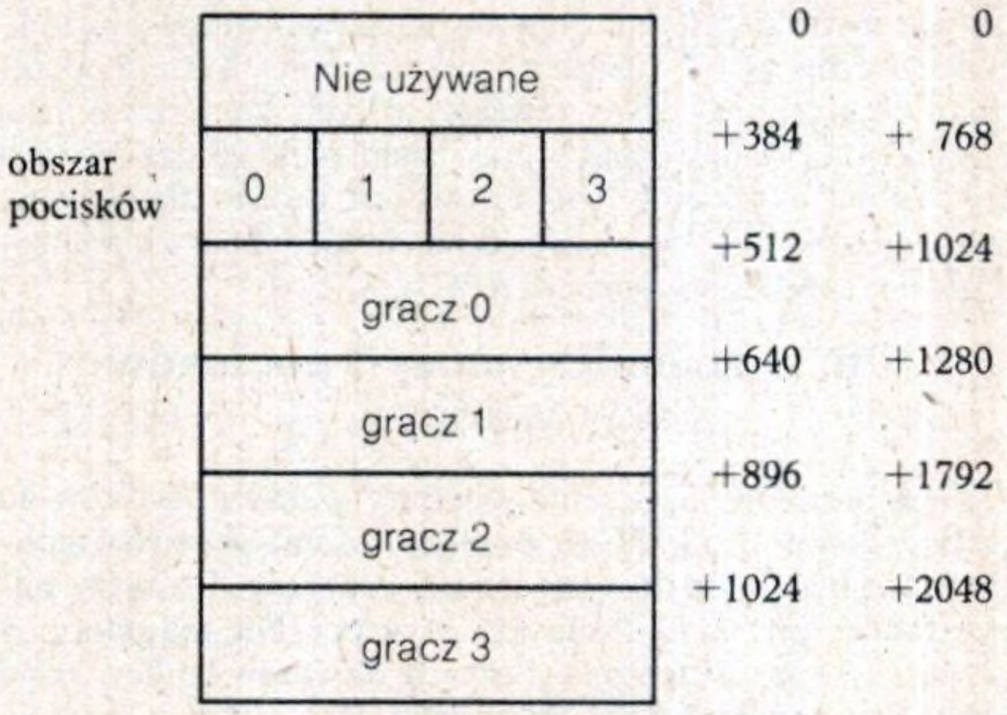
Tablica grafiki G-P musi rozpoczynać się od adresu podzielnego przez 1024, dla graczy o rozdzielczości dwuwierszowej, lub 2048 dla graczy o rozdzielczości jednowierszowej. Instrukcja PRINT PEEK (106)*256 wyświetli ostatni używany adres pamięci naszego komputera. W celu właściwego umieszczenia tablicy w pamięci, musi być znaleziony najbliższy adres na granicy 1024 lub 2048 bajtów.

Zawartość tablicy

Tablica grafiki G-P, bez względu na ilość zdefiniowanych graczy, zawsze będzie mieć wielkość 1024 lub 2048 bajtów dla umieszczenia czterech graczy i pocisków. Pierwsza część tablicy jest pusta; ten obszar dostępny jest dla innych celów, takich jak przechowywanie alternatywnych definicji graczy lub programów wyświetlania. Po tym pustym obszarze jest pięć innych, w których zdefiniowani są gracze i pociski.

Obszar definiowania pocisków będzie przechowywał cztery obiekty pocisków, każdy o szerokości dwóch bitów. Podobnie jak gracze, pociski także mogą być definiowane z rozdzielczością jedno- lub dwuwierszową.

Następne cztery obszary są jednakowej wielkości, każdy z nich przechowuje jeden obiekt gracza. Poniższy rysunek pokazuje przesunięcie od początku tablicy grafiki G-P dla każdego gracza i pocisku.



Następnym krokiem jest ponowna zmiana zawartości komórki 106, poprzez wpisanie adresu tablicy grafiki G-P. Ten krok jest konieczny, ponieważ system operacyjny będzie używał całej dostępnej pamięci, najwyższe dostępne adresy obszaru pamięci zawsze używane są do ustawienia grafiki pola gry. Ten konflikt użycia pamięci może zmienić grafikę pola gry, grafikę G-P, lub obie. W najgorszym przypadku może dojść do zablokowania komputera.

Obliczanie adresu początkowego

Komputer nie rozwiązuje automatycznie konfliktu pamięci, musimy to zrobić za niego. Chociaż wiemy już, że tablica grafiki G-P musi rezydować na granicy 1K lub 2K, obraz pola gry także ma ograniczenia. Umieszczenie tablicy grafiki G-P w najwyższej części pamięci spowoduje problemy adresowania dla grafiki pola gry. Na przykład, pewne obszary obrazu mogą być nie do użycia, lub instrukcje PLOT nie umieszczą punktów graficznych w oczekiwanych wierszach i kolumnach.

Jeśli pamięć ekranu grafiki pola gry przydzielona jest normalnie, możemy umieścić tablicę grafiki G-P tuż poniżej niej bez jakichkolwiek konfliktów pamięci. Jeśli jednak program zmienia tryby graficzne, wtedy możliwe jest powstanie innego konfliktu, który mógłby zniszczyć całkowicie tablicę grafiki G-P. Problem ten może wystąpić, jeśli program przełączy GRAPHICS 0 na GRAPHICS 7. W tym przypadku tablica może być całkowicie wykasowana. Należy zatem przyjrzeć się instrukcjom GRAPHICS w naszym programie; znaleźć tę, która przydziela najwięcej RAM i odpowiednio zaplanować umieszczenie tablicy. Dla obliczenia adresu początkowego grafiki G-P, wykonajmy następujące kroki:

1. Użyjmy funkcji PEEK dla określenia zawartości komórki 560.
2. Użyjmy funkcji PEEK dla określenia zawartości komórki 561 i pomnóżmy wynik przez 256.
3. Dodajmy wyniki kroków 1 i 2.
4. Podzielmy wynik kroku 3 przez 1024, jeśli używamy grafiki o rozdzielczości dwuwierszowej lub przez 2048, jeśli używamy grafiki o rozdzielczości jednowierszowej.
5. Odrzućmy resztę, odejmijmy 1 i pomnóżmy to przez 1024 (rozdzielczość dwuwierszowa) lub 2048 (rozdzielczość jednowierszowa).

Wynik kroku 5 jest adresem początkowym tablicy grafiki G-P. Programowo można to wykonać następująco:

```
1000 REM JEDN=1 oznacza rozdzielczość jednowierszową
1010 PMBASE=PEEK (560)+PEEK(561)*256
1020 IF JEDN THEN DIV=2048
1030 IF (NOT JEDN) THEN DIV=1024
1040 PMBASE=INT (PMBASE/DIV—1)*DIV
```

Ochrona tablicy grafiki G-P

Po określeniu adresu końcowego tablicy grafiki G-P, użyjmy instrukcji POKE do wstawienia dwubajtowej wartości do adresów 14 i 15 (najpierw młodszy bajt). System operacyjny interpretuje adres zawarty w tych komórkach jako bezwzględną dolną granicę przydziału pamięci grafiki pola gry. Zatem ustawienie tego adresu jest istotne dla ochrony tablicy grafiki G-P przed zniszczeniem kiedykolwiek realizowana jest nowa instrukcja GRAPHICS.

Umieszczenie graczy i pocisków w tablicy

Teraz możemy już wprowadzić bity graczy i pocisków do tablicy. Pierwszym krokiem jest oczyszczenie obszarów tablicy, które będą aktualnie przechowywać dane. Pierwszy obszar tablicy grafiki G-P nie jest używany. Nie ma potrzeby czyszczenia go, nie trzeba też czyścić obszarów tablicy, które nie będą zawierać aktywnych bitów.

Sterowanie wyświetlaniem

Istnieje kilka rejestrów sterujących, które jak wynika z ich nazwy, sterują bieżącym wyświetlaniem grafiki G-P:

— rejestr bazowy,
— rejestr sterujący grafiki i DMA,
— rejestry szerokości,
— rejestry kolorów,
— rejestry pozycji poziomych,
— rejestr sterowania priorytetami.

Pewne z tych rejestrów ustawiane są tylko raz, podczas inicjowania grafiki G-P, inne wymagają stałego ustawiania w zależności od tego jak nasz program będzie manipulował graczami i pociskami. Podręczniki Atari używają skróconych nazw tych rejestrów, które są umieszczone w nagłówkach.

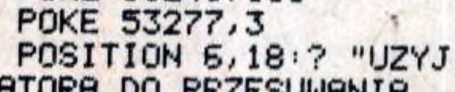
Rejestr bazowy gracz-pocisk (PMBASE)

Komórki pamięci o adresach 54279 i 54280 (D407 i D408 szesnastkowo) będą zawierać adres początkowy tablicy grafiki gracz-pocisk. Ponieważ adres ma być na granicy 1K lub 2K, komórka 54279 zawsze musi zawierać 0. Znaczący jest tylko numer stronicy (starszy bajt adresu).

Rejestr sterujący grafiki (GRACTL)

Rejestr sterujący grafiki wraz z rejestrem DMACTL zezwala na bezpośredni dostęp do pamięci (DMA) dla grafiki

```
DL 4 REM ******************
MC 5 REM *                 *
DY 6 REM *  Program nr 1   *
ME 7 REM *                 *
DP 8 REM ******************
NO 9 REM
QJ 10 DIM KURSOR(2),GRACZ(23,7)
ND 20 GRAPHICS 3
RY 30 SETCOLOR 2,0,0:REM CZARNE O
   KNO TEKSTOWE
LR 40 ? "WLACZ MANIPULATOR DO GNI
   AZDA 1":? "NACISNIJ PRZYCISK"
BO 50 GOSUB 610:?
RH 60 PRINT "UZYJ PRZYCISKU DO KR
   ESLENIA LUB      KASOWANIA GR
   ACZA"
DA 70 PRINT "NACISNIJ <RETURN> PO
    ZAKONCZENIU      KRESLENIA.";
TH 80 GOSUB 670
JK 90 IF PEEK(764)=12 THEN GOTO 2
   90
WG 100 GOSUB 770
DR 110 IF (GORA AND (KURSOR(2)<=0
   )) OR (DOL AND (KURSOR(2)>=23)
   ) THEN GOTO 90
GJ 120 IF (LEWO AND (KURSOR(1)<=0
   )) OR (PRAWO AND (KURSOR(1)>=7
   )) THEN GOTO 90
XP 130 COLOR 1
ZU 140 IF GRACZ(KURSOR(2),KURSOR(
   1))=0 THEN COLOR 0
IT 150 PLOT KURSOR(1)+16,KURSOR(2
   )
SS 160 KURSOR(2)=KURSOR(2)-GORA
PV 170 KURSOR(2)=KURSOR(2)+DOL
GQ 180 KURSOR(1)=KURSOR(1)-LEWO
RP 190 KURSOR(1)=KURSOR(1)+PRAWO
XV 200 COLOR 2
IM 210 PLOT KURSOR(1)+16,KURSOR(2
   )
UF 220 X1=STRIG(0)
GE 230 IF X1=1 THEN 90
XA 240 GRACZ(KURSOR(2),KURSOR(1))
   =1-SGN(GRACZ(KURSOR(2),KURSOR(
   1)))
YQ 250 COLOR 3
ZZ 260 IF GRACZ(KURSOR(2),KURSOR(
   1))=0 THEN COLOR 0
IY 270 PLOT KURSOR(1)+16,KURSOR(2
   )
SV 280 GOTO 90
WT 290 GRAPHICS 7
KC 300 POKE 764,0
WD 310 X2=2:X3=0
EG 320 X1=76:Y1=20
UQ 330 GOSUB 910
QW 350 PRINT "<RETURN> - KONIEC":
   ? "<SPACE> - POWROT DO GRACZA"
BL 360 IF PEEK(764)=12 THEN GOTO
   440
LD 370 IF PEEK(764)<>33 THEN 360
GV 380 GRAPHICS 3+16
VQ 390 GOSUB 740
ZN 400 X1=16:Y1=0
VW 410 X2=1:X3=0
UP 420 GOSUB 910
SN 430 GOTO 90
SR 440 GRAPHICS 0
YQ 450 FOR Y=0 TO 23
FQ 460 POSITION 0,Y
AX 470 PRINT "BAJT ";Y;
YF 480 X1=0
UI 490 FOR X=0 TO 7
LR 500 X2=GRACZ(Y,7-X)
QK 510 IF X2=0 THEN 530
VC 520 X1=X1+INT((X2*2)^X+0.01)
LV 530 NEXT X
BK 540 POSITION 12,Y:PRINT X1;
MJ 550 NEXT Y
ZQ 560 X1=24:Y1=0
FR 570 X2=ASC("□"):X3=ASC(" ")
JZ 580 POKE 752,1
VE 590 GOSUB 910
QU 600 GOTO 590
YG 610 REM OCZEKIWANIE NA PRZYCIS
   K/GENERACJA DZWIEKU
YZ 620 IF STRIG(0)=1 THEN 620
RT 630 SOUND 0,50,10,4
MW 640 FOR OP=1 TO 10:NEXT OP
WA 650 SOUND 0,0,0,0
ZO 660 RETURN
XQ 663 REM USTAWIENIE ZMIENNYCH I
    EKRANU
RX 670 KURSOR(1)=0
SL 680 KURSOR(2)=0
WP 690 FOR R=0 TO 23
MM 700 FOR C=0 TO 7
WX 710 GRACZ(R,C)=0
DT 720 NEXT C
JP 730 NEXT R
YT 740 COLOR 3
NE 750 PLOT 15,0:DRAWTO 15,23:PLO
   T 24,0:DRAWTO 24,23
FL 760 GRAPHICS 3+48:RETURN
WX 770 REM ODCZYT MANIPULATORA
IL 780 S=STICK(0)
AN 790 GORA=0
ZM 800 DOL=0
EG 810 LEWO=0
KE 820 PRAWO=0
YK 830 IF S=15 THEN RETURN
OQ 840 IF S=14 THEN GORA=1
TT 850 IF S=7 THEN PRAWO=1
CY 860 IF S=13 THEN DOL=1
YX 870 IF S=11 THEN LEWO=1
UW 880 GOSUB 630
ZW 890 RETURN
QL 900 REM WYSWIETLANIE GRACZA
YN 910 FOR Y=0 TO 23
TZ 920 FOR X=0 TO 7
LM 930 COLOR X2
ES 940 IF PLAYER(Y,X)=0 THEN COLO
   R X3
CQ 950 PLOT X+X1,Y+Y1
MF 960 NEXT X
MR 970 NEXT Y
ZV 980 RETURN
```



```
DM 5 REM ******************
MD 6 REM *                 *
EU 7 REM *  Program nr 2   *
MF 8 REM *                 *
DQ 9 REM ******************
SG 50 GRAPHICS 0:SETCOLOR 2,0,0:P
   OKE 755,0
NA 60 PMB=PEEK(106)-16
SN 70 POKE 54279,PMB
AB 80 PMBASE=PMB*256
LU 90 FOR J=0 TO 255:POKE PMBASE+
   1024+J,0:NEXT J
YE 100 FOR J=0 TO 12:READ A:POKE
   PMBASE+1024+140+J,A:NEXT J
RH 110 POKE 559,62
QK 120 POKE 704,204
MV 130 POKE 712,34
HU 140 POKE 53248,130
VM 150 POKE 53277,3
TA 160 POSITION 6,18:? "UZYJ MANI
   PULATORA DO PRZESUWANIA
            GRACZA":?
XB 170 H=130:V=140
ZM 180 S=STICK(0):RESTORE
BR 190 IF S=11 THEN H=H-1
HL 200 IF S=7 THEN H=H+1
HE 210 POKE 53248,H
XH 220 V1=0
AR 230 IF S=14 THEN V=V-1:V1=1
YS 240 IF S=13 THEN V=V+1:V1=1
KW 250 IF V1=1 THEN FOR J=0 TO 12
   :READ A:POKE PMBASE+1024+V+J,A
   :NEXT J
VT 260 POSITION 6,21:? "POZYCJA P
   OZIOMA =   ";CHR$(254);CHR$(25
   4);CHR$(254);H
JL 270 POSITION 6,22:? "POZYCJA P
   IONOWA =   ";CHR$(254);CHR$(25
   4);CHR$(254);V
PH 280 GOTO 180
JS 290 DATA 0,40,146,214,254,238,
   254,238,186,170,56,16,0
```

gracz-pocisk. GRACTL umieszczony jest pod adresem 53277 (D01D szesnastkowo) i możemy wybrać zezwolenie tylko dla gracza (POKE 53277,2), tylko dla pocisku (POKE 53277,1) lub łącznie dla gracza i pocisku (POKE 53277,3).

Rejestr sterujący DMA (DMACTL)

Ustawienie rejestru sterującego DMA włącza lub wyłącza grafikę G-P. Jeśli rejestr GRACTL nie jest ustawiony na zezwolenie DMA, będziemy widzieć tylko grafikę pola gry. DMACTL i GRACTL muszą być ustawione jednocześnie w celu wyświetlania graczy i pocisków. DMA działa jak pasożyt na mikroprocesorze 6502, któremu kradnie cykle maszynowe. Jeśli chcemy wstrzymać wyświetlanie obiektów grafiki G-P, musimy wyzerować rejestr DMACTL. Przyspieszy to trochę działanie mikroprocesora.

Użyjmy instrukcji POKE do wstawienia jednej z poniższych wartości do komórki o adresie 559 (22F szesnastkowo) w celu ustawienia rejestru DMACTL.

Wartość	Wynik działania
4	włącza DMA tylko dla pocisku
8	włącza DMA tylko dla gracza
12	włącza DMA dla gracza i pocisku
+16	rozdzielczość jednowierszowa (domyślnie dwuwierszowa)

Rejestry szerokości graczy (SIZEP0—SIZEP3)

Cztery rejestry ośmiobitowe, o adresach 53256 do 53259 (D008 do D00B szesnastkowo), sterują poziomą wielkością czterech graczy. Poprzez zmianę zawartości tych komórek możemy zwiększyć dwukrotnie lub czterokrotnie szerokości obiektów (lecz nie ich wysokości). Jeśli wielkość gracza nie będzie zmieniana w programie, pozostaje normalnej wielkości.

Adres 53256 steruje szerokością pierwszego gracza, adres 53257 — drugiego, i tak dalej. Podczas pisania programów do przesuwania poziomego obiektów ustawienie rejestru szerokości może zmienić zawartość rejestru pozycji poziomej gracza. W celu przełączenia gracza na podwójną szerokość musimy ustawić jego rejestr szerokości na 1; dla poczwórnej szerokości — na 3. Wartości 0 i 2 określają normalną szerokość. Przykładem może być instrukcja ustawienia trzeciego gracza na podwójną szerokość:

```
POKE 53258,1.
```

Rejestr szerokości pocisku (SIZEM)

Jeden rejestr, o adresie 53260 (D00C szesnastkowo), steruje wielkością wszystkich pocisków. Takie same wartości, jak pokazane powyżej dla szerokości gracza, stosują się do rejestru szerokości pocisku: 0 lub 2 dla normalnej szerokości, 1 dla podwójnej i 3 dla poczwórnej.

Rejestry kolorów gracz-pocisk (COLPM0—COLPM3)

Cztery rejestry kolorów gracz-pocisk, o długości 1 bajta każdy, rozpoczynają się od adresu 704 (2C0 szesnastkowo) dla pierwszego gracza i kończą w 707 (2C3 szesnastkowo) dla czwartego. Zarówno gracz, jak i skojarzony z nim pocisk, mają ten sam kolor.

Rejestry pozycji poziomej graczy (HPOSP0—HPOSP3)

Rejestry pozycji poziomej graczy używane są do przemieszczania obiektów wzdłuż osi poziomej. Poprzez prostą zmianę zawartości rejestru instrukcją POKE, możemy przesunąć gracza do nowej pozycji poziomej. W zależności od ustawienia rejestru szerokości możemy ustawić gracza przy lewym brzegu ekranu, następnie ustawić nową wartość pozycji poziomej, która spowoduje, że obiekt ponownie pojawi się gdzieś na ekranie. Minimalną wartością każdego rejestru pozycji jest 0, maksymalną — 227. W zależności od wielkości gracza, określonej w DMACTL, ustawienia tych rejestrów będą zmieniać się od 40, jako widocznej pozycji najbardziej z lewej, do 190, jako pozycji najbardziej z prawej.

Rejestry te mogą być używane tylko w trybie zapisu, to znaczy nie będziemy mogli użyć instrukcji PEEK dla określenia pozycji gracza. Zatem nasz program musi mieć zmienne, które przechowują aktualne pozycje poziome graczy i pocisków na ekranie. Rejestr pozycji poziomej gracza 0 mieści się pod adresem 53248 (D000 szesnastkowo); gracza 1 — pod 53249; gracza 2 — pod 53250 i gracza 3 — pod 53251.

Rejestry pozycji poziomej pocisków (HPOSM0—HPOSM3)

Cztery rejestry, od adresu 53252 (D004 szesnastkowo), przechowują wartości używane do zmiany pozycji pocisków na osi poziomej.

Manipulowanie duchami

Po tym, nieco przydługim wprowadzeniu, niezbędnym z powodu trudności ze zdobyciem odpowiedniej literatury (a także jej giełdowej ceny), przejdziemy do omówienia praktycznego wykorzystania opisanej techniki.

Jak wspomnieliśmy na wstępie, podstawowym przeznaczeniem grafiki gracz-pocisk jest uzyskanie efektu animacji poprzez przemieszczanie obiektów na ekranie. Ruch obiektów rozłożymy na poziomy i pionowy, nie tyle ze względów formalnych, ile z powodu różnych sposobów jego realizacji.

Poziome przemieszczanie gracza jest bardzo proste. Po prostu zwiększamy lub zmniejszamy wartość przechowywaną w rejestrze pozycji poziomej odpowiadającym graczowi, którego chcemy przemieszczać. Jedyny problem z rejestrami pozycji poziomej polega na tym, że nie możemy pobrać ich zawartości dla określenia aktualnej pozycji poziomej gracza lub pocisku. Jeśli wprowadzimy:

```
POKE 53248,140:PRINT PEEK(53248)
```

moglibyśmy oczekiwać, że komputer wyprowadzi 140, wartość, którą zapamiętaliśmy w komórce 53248. Zamiast tego otrzymujemy liczbę 0. Co zatem stało się ze 140? Układ ANTIC pobrał 140 do swoich wewnętrznych rejestrów, odpowiednio zmienił pozycję poziomą gracza i wyzerował zawartość komórki 53248, zanim została wykonana instrukcja PRINT PEEK (53248). Jest to dosyć niewygodne, jeśli chcemy pamiętać kolejne współrzędne poziomego położenia gracza.

Program nr 2 demonstruje podstawową metodę przemieszczania graczy. Pierwsza sekcja, linie 50 do 150, zawiera standardowe procedury inicjowania. POKE 755,0 w linii 50 wyłącza kursor, zatem jeśli zatrzymamy program i chcemy z powrotem włączyć kursor, musimy nacisnąć (RESET) lub wprowadzić POKE 755,2.

Dla większości procedur animacji musimy przydzielić zmienną do przechowywania aktualnej pozycji paska gracza. Linia 170 używa zmiennej H do zapamiętania aktualnej wartości położenia poziomego gracza 0, który początkowo umieszczony jest w 130.

Linia 180 przydziela inną zmienną, S, do przechowywania wartości STICK(0) — stanu manipulatora. Możliwymi wartościami są: 11—lewo, 7—prawo, 14—w górę, 13—w dół i 15—środek.

Linie 190 i 200 sprawdzają czy S jest równe 11, czy 7 — manipulator wskazuje lewo lub prawo i wartość H zmienia się o jeden, zgodnie z kierunkiem ruchu gracza. Linia 210 wstawia wartość przechowywaną w H do rejestru pozycji poziomej gracza 0, w celu przesunięcia go o jedną pozycję w prawo lub w lewo. Każde przemieszczenie gracza równe jest szerokości punktu elementarnego.

U dołu ekranu wyświetlana jest aktualna wartość pozycji poziomej. Od wartości równej 48 gracz rozpoczyna opu-

szczanie niebieskiego obszaru i przechodzi przez tło, aż wyjdzie poza ekran, jeśli kontynuujemy ruch manipulatora w lewo. Dalsze zmniejszenie wartości poziomej nie daje żadnego efektu na ekranie.

Jeśli wartość H stanie się mniejsza od 0 lub większa od 255, otrzymamy komunikat błędu, gdy program próbuje wstawić wartość H do rejestru pozycji poziomej 53248 (linia 210). Wynika to z faktu, że nie możemy wstawić, za pomocą POKE, liczby nie mieszczącej się na pozycji jednego bajtu, do pojedynczej komórki pamięci. Dla uniknięcia tego problemu powszechnie stosuje się w takich programach procedury śledzenia błędów. Program nr 2 nie ma takiej procedury, dlatego możemy sprawdzić co zdarzy się, kiedy spróbujemy przesunąć gracza zbyt daleko.

Jak widać, uzyskanie ruchu poziomego jest bardzo łatwe. Kiedy przychodzi do ruchu pionowego, sprawa się komplikuje. Nie jest możliwe przesuwanie paska gracza w pionie. Zawsze będzie on rozciągał się od góry do dołu ekranu, bez względu na to, ile punktów elementarnych wewnątrz niego jest wyświetlanych.

Pionowe położenie obiektu na ekranie określa odległość, od początku obszaru gracza do miejsca, w które ładujemy dane. Dlatego w linii 100 dane o kształcie gracza wprowadzane są o 140 bajtów od początku obszaru danych gracza 0 dla upewnienia się, że obiekt wystąpi na ekranie. Jeśli zmienimy wartość 140 na 141, gracz pojawi się o jedną linię ekranu niżej. Wartość 139 spowoduje wyświetlenie gracza o jedną linię wyżej.

Teraz powinno być oczywiste jak uzyskać iluzję ruchu w pionie. Potrzebna jest procedura do przesuwania wszystkich danych kształtu o jeden bajt w górę lub w dół pamięci. Funkcje te wykonuje linia 250.

Początkowo, w linii 170, zmienna V przydzielona jest do przechowania aktualnej wartości pozycji pionowej gracza, w taki sam sposób jak zmienna H.

Linie 230 i 250 sprawdzają górną i dolną pozycję manipulatora i zwiększają lub zmniejszają odpowiednio wartość V. Jeśli manipulator został przesunięty w górę lub w dół, wtedy zmienna V1 ustawiana jest na 1. V1 działa jak flaga. Kiedy V1 jest jedynką, wtedy procedura przemieszczania pionowego w linii 250 jest aktywna. Kiedy V1 równe jest zero, żaden ruch pionowy nie jest potrzebny, procedura w linii 250 jest ignorowana.

Procedura ta podobna jest do procedury ładowania danych kształtu w linii 100 i wprowadza informacje o kształcie do obszaru danych gracza 0. Zamiast dodawania stałej wartości, podobnej do 140 w linii 100, do adresu bazowego obszaru danych gracza 0 (PMBASE+1024) dodawana jest wartość zmiennej V. Obiekt będzie przesuwany o odległość równą jednemu punktowi elementarnemu w górę lub w dół.

Instrukcja DATA zawierająca informacje o kształcie gracza 0, zawarta jest w linii 290. Zauważmy, że na początku i końcu tych danych występują zera, w celu wykasowania wcześniej wykreślonego punktu od góry lub dołu ostatniego kształtu gracza.

Aby zobaczyć co się zdarzy, kiedy pominiemy te zera, zmieńmy linie 290 poprzez wykasowanie obu zer i zmieńmy licznik pętli w liniach 100 i 250 z 12 na 10, aby czytana była poprawna ilość danych. Teraz, po uruchomieniu programu, zdarzą się straszne rzeczy! W miarę przesuwania manipulatora w górę lub w dół gracz pozostawia ślad.

Z pewnością nie będziemy zbyt zadowoleni z otrzymanego ruchu pionowego obiektu. Procedura ponownego wyświetlania w linii 250 zajmuje sporo czasu i zawsze będzie zauważalne opóźnienie, gdy program dokonuje aktualizacji informacji przechowywanej w obszarze danych gracza. Jeśli chcemy poprawić jakość przemieszczania, musimy wprowadzić procedurę w kodzie maszynowym.

Program nr 3 jest ulepszoną wersją poprzedniego. Zawiera dwie procedury w kodzie maszynowym, jedną dla ruchu w górę, a drugą dla ruchu w dół. Nie martwmy się, jeśli nie mamy pojęcia o kodzie maszynowym. Obie procedury mogą być skopiowane i włączone do używania we własnych programach bez dokładnego rozumienia jak one działają.

Po uruchomieniu tego programu zobaczymy, że działa on tak samo jak poprzedni. Dopiero podczas przesuwania w górę lub w dół zobaczymy, że animacja jest dużo płynniejsza.

W linii 170 czytane są dane zawarte w liniach 300 do 370 — procedury w kodzie maszynowym — i przechowywane w nieużywanej części obszaru danych gracz-pocisk. Ten nie wykorzystany obszar ma długość 768 bajtów dla rozdzielczości jednowierszowej graczy i 384 bajty dla rozdzielczości dwuwierszowej — wystarczająca ilość miejsca dla przechowania procedur w kodzie maszynowym.

Pierwsza procedura — o długości 21 bajtów — umieszczona jest w nie używanym obszarze od PMBASE w górę, druga procedura rozpoczyna się od PMBASE+21.

Procedury w takiej postaci będą sterować graczem lub pociskiem o wysokości do 20 punktów elementarnych. W celu użycia tej procedury dla wyższych graczy, musimy zmienić liczbę 22 w linii 340 na większą o 2 od wysokości gracza i liczbę 21, w linii 360, na większą o 1.

Przemieszczanie poziome obsługiwane jest w taki sam sposób jak w programie nr 2. W liniach 220 i 230 wstawiona została procedura śledzenia błędów, która zatrzymuje program, kiedy wartość zmiennej H stanie się mniejsza niż 48 lub większa niż 202. To znaczy, że gracz nie może opuścić obszaru pola gry i zapobiega pojawieniu się komunikatu błędu, który występuje, jeśli wartość H stanie się mniejsza od 0 lub większa od 255.

Linie 250 i 260 sterują przemieszczaniem pionowym poprzez wywoływanie procedur w kodzie maszynowym. V, zmienna współrzędnej pionowej, sprawdzana jest najpierw dla upewnienia się, że nie jest większa niż 32 lub mniejsza niż 212 — wtedy gracz jest na górnym lub dolnym brzegu pola gry — przed przekazaniem sterowania do odpowiedniej procedury maszynowej.

```
DM 5 REM ******************
MD 6 REM *                *
FP 7 REM *  Program nr 3   *
MF 8 REM *                *
DQ 9 REM ******************
AY 10 REM
SG 50 GRAPHICS 0:SETCOLOR 2,0,0:P
   OKE 755,0
NA 60 PMB=PEEK(106)-16
SN 70 POKE 54279,PMB
AB 80 PMBASE=PMB*256
LU 90 FOR J=0 TO 255:POKE PMBASE+
   1024+J,0:NEXT J
YE 100 FOR J=0 TO 12:READ A:POKE
   PMBASE+1024+140+J,A:NEXT J
RH 110 POKE 559,62
QK 120 POKE 704,204
MV 130 POKE 712,34
HU 140 POKE 53248,130
VM 150 POKE 53277,3
RF 160 POSITION 6,18:? "UZYJ MANI
   PULATORA DO PRZESUWANIH
          GRACZA"
BO 170 FOR I=0 TO 41:READ A:POKE
   PMBASE+I,A:NEXT I
XD 180 H=130:V=140
ZO 190 S=STICK(0):RESTORE
BA 200 IF S=11 THEN H=H-1
HN 210 IF S=7 THEN H=H+1
FP 220 IF H<48 THEN H=48
GN 230 IF H>202 THEN H=202
HK 240 POKE 53248,H
HW 250 IF S=14 AND V>32 THEN A=US
   R(PMBASE,PMBASE+1023+V):V=V-1
OJ 260 IF S=13 AND V<212 THEN A=U
   SR(PMBASE+21,PMBASE+1023+V):V=
   V+1
VV 270 POSITION 6,21:? "POZYCJA P
   OZIOMA =   ";CHR$(254);CHR$(25
   4);CHR$(254);H
JN 280 POSITION 6,22:? "POZYCJA P
   IONOWA =   ";CHR$(254);CHR$(25
   4);CHR$(254);V
PU 290 GOTO 190
JD 310 DATA 0,40,146,214,254,238,
   254,238,186,170,56,16,0
RM 330 DATA 104,104,133,204,104,1
   33,203,160,1,177
OR 340 DATA 203,136,145,203,200,2
   00,192,22,208,245,96
EE 360 DATA 104,104,133,204,104,1
   33,203,160,21,177
FR 370 DATA 203,200,145,203,136,1
   36,192,255,208,245,96
```

Wzór używania procedury jest następujący:
A=USR (adres kodu maszyn., adres gracza plus pozycja pionowa)
wykorzystując ten wzór, wywołanie procedury w linii 250 staje się następujące:

A=USR(PMBASE, PMBASE+1023+V)

Dla poprawnego działania procedury, V musi być zwiększane lub zmniejszane nie więcej niż o jeden naraz.

Po wywołaniu procedury i ponownym wyświetleniu gracza, V jest zmienne o 1, aby było gotowe do następnej operacji.

Inne własności

Opisany mechanizm umożliwia tworzenie bardzo szybkiej animacji. Istnieją czterej niezależni gracze. Mają oni własne zawartości rejestrów sterujących i obszary RAM; zatem ich działanie jest całkowicie niezależne. Oznaczeni są P0 do P3. Mogą być używani razem, jeden obok drugiego, dla otrzymania rozdzielczości 32-bitowej, lub mogą być używani niezależnie dla otrzymania czterech ruchomych obiektów.

Każdy gracz ma własny rejestr koloru, jest on całkowicie niezależny od rejestrów kolorów pola gry. Rejestry kolorów graczy nazywane są COLP(X). Umożliwia to otrzymanie większej ilości kolorów na ekranie. Jednak każdy gracz ma tylko jeden kolor; bez wykorzystania przerwań programu wyświetlania nie można uzyskać wielokolorowych graczy.

Każdy gracz ma zmienną szerokość, możemy ją ustawić normalną, podwójną lub poczwórną za pomocą rejestrów SIZEP(X). Jest to przydatne dla nadawania graczom różnych rozmiarów. Mamy także opcję wybierania rozdzielczości pionowej graczy. Możemy używać rozdzielczości jednowierszowej, w której każdy bajt w tablicy gracza zajmuje jedną linię ekranu, lub rozdzielczości dwuwierszowej, w której każdy bajt zajmuje dwie linie ekranu. Jest to jedyny przypadek, kiedy własności graczy nie są niezależne, wybór rozdzielczości pionowej dotyczy wszystkich graczy (steruje nią bit D4 rejestru DMACTL).

Pociski

Kolejną zaletą są pociski — obiekty graficzne o szerokości 2 bitów, skojarzone z graczami. Jeden pocisk przydzielony jest do każdego gracza. Informacje o kształcie pocisku pochodzą z tablicy umieszczonej tuż przed tablicami graczy. Cztery pociski upakowane są w tej samej tablicy. Mogą one poruszać się niezależnie od graczy; mają własne rejestry pozycji poziomej. Mają także własny rejestr wielkości, SIZEM, który określa ich szerokość, podobnie jak rejestry SIZEP(X) dla graczy. Jednak pociski nie mogą być ustawione na różne wielkości; są ustawiane razem. Na żądanie pociski mogą być połączone razem do postaci piątego gracza, w tym przypadku przyjmują one kolor rejestru 3 pola gry. Realizuje się to poprzez ustawienie bitu D4 rejestru sterowania priorytetem (PRIOR). W dalszym ciągu mogą one przemieszczać się niezależnie. Ustawienie tego bitu zmienia tylko kolor pocisków.

Pociski przesuwamy pionowo podobnie jak graczy: poprzez przemieszczenie danych obrazu w obszarze RAM pocisku. Jest to dosyć trudne w realizacji, ponieważ pociski umieszczone są w jednej tablicy.

Priorytety graczy i pola gry

Ważną własnością grafiki G—P jest to, że utworzone obiekty są całkowicie niezależne od pola gry. Możemy używać ich w dowolnym trybie graficznym. Powoduje to powstanie innego problemu: co się zdarzy, jeśli gracz zachodzi na obraz pola gry? Który obraz ma priorytet? Na szczęście mamy możliwość zdefiniowania priorytetów używanych do wyświetlania graczy. Jeśli zechcemy, wszyscy gracze mogą mieć priorytet nad rejestrami kolorów pola gry. Możemy ustalić, aby wszystkie rejestry pola gry (za wyjątkiem tła) miały priorytet nad wszystkimi graczami. Możemy również określić priorytety między poszczególnymi graczami i polem gry. Priorytety wybierane są z rejestru sterowania priorytetem — PRIOR. Ta własność umożliwia obiektom przechodzenie przed frontem jednego obrazu i poza innym, dają złudzenie efektu trójwymiarowości.

Układowe wykrywanie kolizji

Jedną z podstawowych zalet grafiki G—P, używaną w grach, jest możliwość układowego wykrywania kolizji. Możemy sprawdzić, czy jakiś obiekt graficzny (gracz lub pocisk) zderzył się z innym. Można wykryć następujące kolizje: pocisk-gracz, pocisk-pole gry, gracz-gracz, gracz-pole gry. Łącznie istnieje 54 możliwych kolizji, każda z nich ma przydzielony jeden bit, który może być sprawdzany. Jeśli bit jest ustawiony, wystąpiła kolizja. Bity te są odwzorowane w 15 rejestrach w GTIA (używane są tylko młodsze 4 bity). Rejestry te można tylko odczytywać, nie można ich oczyścić poprzez zapisanie zer. Aby je wyzerować (w celu dalszego wykrywania kolizji) musimy zapisać dowolną liczbę do rejestru HTCLR. Czyści to wszystkie rejestry kolizji.

Kolizja występuje wtedy, gdy obraz gracza nakłada się z innym obrazem; zatem bit kolizji nie będzie ustawiony dopóki kreślona jest część ekranu pokazująca kolizję. To znaczy, że wykrycie kolizji może nastąpić po około 20 ms od chwili przesunięcia gracza. Zalecanym rozwiązaniem jest wykonanie ruchu gracza i wykrywanie kolizji w czasie procedury przerwania pionowego. W tym przypadku najpierw należy sprawdzić, czy nie wystąpiła kolizja, następnie oczyścić rejestr kolizji i wykonać ruch graczem. Innym rozwiązaniem jest odczekanie 20 ms, po przesunięciu gracza, przed sprawdzeniem czy nie uległ on kolizji.

Zastosowania grafiki gracz—pocisk

Oprócz animacji, grafika G—P oferuje pewne dodatkowe możliwości. Gracze są wyjątkowym sposobem zwiększenia ilości kolorów obrazu. Cztery dodatkowe rejestry kolorów umożliwiają użycie czterech dodatkowych kolorów w każdej linii obrazu. Rozdzielczość 8-bitowa ogranicza oczywiście zakres ich zastosowań. Istnieje sposób obejścia tego ograniczenia. Weźmy gracza o poczwórnej szerokości i wstawmy go na ekran. Ustawmy tak priorytety, aby gracz miał niższy priorytet, niższy kolor pola gry. Następnie zamieńmy kolory pola gry i tła. Zatem widoczny kolor tła będzie w rzeczywistości kolorem pola gry. Gracz zniknie poza tym fałszywym tłem. Teraz wycinamy otwór poprzez wykreślenie prawdziwego koloru tła. Gracz pokaże się przed kolorem prawdziwego tła, lecz tylko w obszarze, gdzie było wykreślone prawdziwe tło. W ten sposób gracz może mieć więcej niż 8 bitów rozdzielczości poziomej.

Innym zastosowaniem grafiki G—P są znaki specjalne. Istnieje wiele znaków, które przekraczają pionowe granice w normalnych zestawach znaków. Jednym ze sposobów jest utworzenie zestawów znaków specjalnych, które rozwiążą ten problem. Innym sposobem jest użycie gracza. W ten sposób mogą być tworzone indeksy, znaki całek i inne symbole specjalne.

Szczególnie przydatnym zastosowaniem graczy jest wykorzystanie ich w charakterze kursora. Ze swoją zdolnością płynnego przemieszczania się po całym ekranie, bez naruszania jego zawartości, są oni idealnie dostosowani do takich zastosowań. Taki kursor może zmieniać kolor w miarę przemieszczania się po ekranie.

Grafika gracz—pocisk dostarcza wielu możliwości. Wykorzystanie jej w grach jako ruchomych obiektów jest oczywiste. Ma również zastosowanie poważne. Może zwiększyć ilość kolorów lub rozdzielczość obrazu. Może prezentować znaki specjalne. Może być używana w charakterze kursora. A zatem wykorzystujmy ją.

Ludwik PIELA,
Tomasz MROWIEC

Emulator BASIC 1.1 dla CPC 464 (1)

W niedługim czasie po ukazaniu się na rynku mikrokomputera CPC 464, pojawili się jego „młodsi bracia" — 664 i 6128. Do podstawowych różnic między CPC 464, a nowszymi modelami należy zaliczyć:

- zamianę pamięci zewnętrznej z magnetofonu na stację dyskietek
- powiększenie pamięci RAM z 64K do 128K (w przypadku 6128)
- rozszerzenie listy rozkazów interpretera BASIC o szereg instrukcji, głównie graficznych.

O ile wyeliminowanie dwóch pierwszych różnic, wymaga dosyć dużego wydatku, o tyle likwidację trzeciej możemy dokonać dzięki emulatorowi opublikowanemu w piśmie Schneider International 10/86. Celem zainstalowania rozszerzonej wersji interpretera należy wprowadzić krótki program LOADER. Program ten ładuje do pamięci RAM program maszynowy EMU.BIN, zawierający rozszerzenia. Nie są to RSX'y (tzn. komendy zewnętrzne względem interpretera — poprzedzane każdorazowo znakiem), lecz instrukcje stosowane na tych samych zasadach co komendy BASIC 1.0.

Program EMULATOR, składuje w pamięci zewnętrznej (kaseta lub dyskietka) program maszynowy EMU.BIN, ładowany w następnej fazie LOADER'em. Instrukcje BASIC 1.1 nie występujące w wersji 1.0 to:

- CURSOR
- ON BREAK i ON BREAK STOP
- MASK
- GRAPHICS PEN
- GRAPHICS PAPER
- FRAME
- FILL
- COPYCHR$
- CLEAR INPUT

Dodatkowo komendy graficzne MOVE, MOVER, PLOT, PLOTR, DRAW, DRAWR zawierają dodatkowy parametr: tryb tuszu. Powyższe komendy zostaną omówione i zilustrowane przykładami w kolejnym numerze „IKS'a".

J. R.

```
10 REM ************************************
20 REM *   664/6128 - Emulator dla CPC 464  *
30 REM ************************************
40 REM *      autor: M. Uphoff   1986       *
50 REM *                                    *
60 REM ************************************
70 adr=&A1FF:MEMORY adr
80 FOR bl=1 TO 9
90 FOR i=1 TO 128
100 READ byte$:IF byte$="EOF" THEN 150
110 v=VAL("&"+byte$):s=s+v
120 adr=adr+1:POKE adr,v
130 NEXT i
140 NEXT bl
150 SAVE"emu.bin",b,&A200,&439
160 END
170 '
180 '********************** Block 1
190 DATA 3E,C3,32,16,AC,21,83,A2
200 DATA 22,17,AC,32,19,AC,21,8E
210 DATA A2,22,1A,AC,32,07,AC,21
220 DATA 9A,A2,22,08,AC,32,04,AC
230 DATA 21,B4,A4,22,05,AC,32,0D
240 DATA AC,21,2A,A3,22,0E,AC,32
250 DATA 01,AC,21,AB,A2,22,02,AC
260 DATA 32,90,BB,21,D8,A2,22,91
270 DATA BB,32,DE,BB,21,11,A3,22
280 DATA DF,BB,32,C0,BB,21,F5,A2
290 DATA 22,C1,BB,32,C3,BB,21,FA
300 DATA A2,22,C4,BB,21,53,A3,22
310 DATA E3,BD,21,62,A3,22,E9,BD
320 DATA 3A,80,BC,FE,C3,C8,32,36
330 DATA A6,2A,81,BC,22,37,A6,3E
340 DATA C3,32,80,BC,21,B2,A2,22
350 '********************** Block 2
360 DATA 81,BC,C9,11,E1,A5,CD,27
370 DATA E3,D0,F1,C3,61,DF,21,E1
380 DATA A5,CD,13,E3,D0,7E,23,C1
390 DATA C1,C9,FE,C5,D0,D6,BA,D1
400 DATA EB,4F,06,00,21,1C,A6,09
410 DATA C3,BB,DD,21,91,B2,75,C3
420 DATA 68,A2,E5,3A,36,A6,32,80
430 DATA BC,2A,37,A6,22,81,BC,CD
440 DATA 80,BC,21,80,BC,36,C3,21
450 DATA B2,A2,22,81,BC,E1,D8,C8
460 DATA FE,1A,37,3F,C0,B7,37,C9
470 DATA CD,E5,A5,F3,09,CB,D9,D9
480 DATA F3,D0,E1,E3,CD,55,BD,30
490 DATA DA,3E,02,CD,FD,D1,E5,CD
500 DATA 95,BB,E1,E3,E9,CD,DB,A5
510 DATA 18,03,CD,DE,A5,F3,D9,CB
```

KRESKA (BASIC CPC 464) — zakończenie

Przedstawione w poprzednim odcinku komendy !SCREENCOPY i !SCREENSHOW posłużyły mi do zapamiętania w dowolnym momencie rysunku tworzonego przy wykorzystaniu programu KRESKA, a także umożliwiają one powrót do określonego etapu rysowania.

Ostatnią funkcją, którą proponuję dołączyć do programu, jest wprowadzanie tekstu do grafiki. Lokalizację tekstu na ekranie determinuje w tym wypadku bieżące położenie kursora graficznego. W programie umieściłem instrukcję TAG, która pozwala na mieszanie tekstu z grafiką. Rozkaz TAGOFF kasuje działanie rozkazu TAG (przywrócony jest druk tekstu na bieżącej pozycji kursora tekstowego).

Na tym zakończyłbym rozszerzanie programu KRESKA. Mam nadzieję, że kontynuowany przez wiele wydań „IKS-a" cykl artykułów stał się dla wielu czytelników okazją do własnych przemyśleń i inspiracją do poszukiwań.

Janek

```
190 LOCATE #2,2,2:PRINT #2,"(Z)AK. PRACY
 (O)DCZYT (W)PISANIE";
231 REM*********************************

232 REM      ODCZYT RYSUNKU Z ROM
233 REM*********************************

234 IF INKEY(34)=0 THEN !SCREENSHOW:GOTO
 180
235 REM*********************************

236 REM      ZAPIS RYSUNKU DO ROM
237 REM*********************************

238 IF INKEY(59)=0 THEN !SCREENCOPY:GOTO
 180
715 IF INKEY(2)=0 THEN GOTO 5000
4997 REM ********************************
4998 REM       WPROWADZANIE TEKSTU
4999 REM ********************************
5000 CLS #2:PRINT #2,"(T)EKST
      LUB ("+CHR$(240)+")";
5010 IF INKEY(0)=0 THEN GOTO 650
5020 IF INKEY(51)=0 THEN 5050
5040 GOTO 5010
5050 rb%=0:CLS #2:PRINT #2,"WPROWADZ. TE
KSTU WYLACZONE LUB P";
5060 LOCATE #2,1,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+"
 COPY LUB JOYSTICK"
5070 GOSUB 1070:GOSUB 1210:GOSUB 1120:GO
SUB 1010:GOSUB 1070
5080 IF INKEY(27)=0 THEN GOTO 5000

5090 IF INKEY(F%)=0 AND rb%=0 THEN rb%=1
:FOR i=1 TO 200:NEXT i:LOCATE #2,18,1:PR
INT #2,"WLACZONE ":FOR i=1 TO 1000:NEXT
i:GOTO 5110 ELSE IF INKEY(F%)=0 AND rb%=
1 THEN rb%=0:LOCATE #2,18,1:PRINT #2,"WY
LACZONE":FOR i=1 TO 200:NEXT i:GOTO 5070

5100 GOTO 5070
5110 CLS #2:PRINT #2,"WCISNIJ klawisz -
ENTER !":PEN #3,kolor:INPUT #3,tekst$:CL
S #3:PEN #3,1:PRINT #3,CHR$(211)+CHR$(32
)+CHR$(211)+CHR$(32);:CLS #2:INPUT #2,"P
odaj tekst:",tekst$
5120 TAG #1
5125 PLOT XPOS,YPOS,kolor
5130 PRINT #1,tekst$;
5140 rb%=0:TAGOFF #1
5150 GOTO 5050
```

```
520 '********************** Block 3
530 DATA 59,D9,FB,C0,E1,E3,CD,55
540 DATA DD,DC,0E,A3,E3,E9,CD,4B
550 DATA C2,CD,D8,A5,F3,B9,CB,59
560 DATA D9,FB,C0,CD,55,DD,D0,3E
570 DATA 04,CD,FB,C1,E5,CB,59,BC
580 DATA E1,C9,2B,7E,FE,20,28,FA
590 DATA FE,2C,C0,2B,E5,2A,34,AE
600 DATA CD,3F,DD,21,28,A6,CD,AA
610 DATA FF,E1,38,08,FE,BB,C0,CD
620 DATA 93,BB,18,03,CD,E1,BB,D1
630 DATA C3,4F,00,3A,2F,A6,2F,32
640 DATA 33,A6,CD,3C,18,AF,32,33
650 DATA A6,C9,3A,33,A6,B7,20,0F
660 DATA 3A,39,B3,B8,C2,CC,B1,3A
670 DATA 30,A6,B7,CA,CC,B1,C9,3A
680 DATA 46,B3,B7,3A,2F,A6,20,11
690 '********************** Block 4
700 DATA 07,32,2F,A6,3A,38,B3,47
710 DATA DA,CC,B1,3A,39,B3,47,18
720 DATA DE,E5,D5,C5,21,CF,B1,5F
730 DATA 3A,07,B2,ED,44,47,4F,CB
740 DATA 01,30,05,CB,03,30,01,B6
750 DATA 23,10,F4,4F,7B,32,2F,A6
760 DATA 3A,30,A6,B7,20,0A,2A,38
770 DATA B3,7D,AC,A1,AC,C1,18,01
780 DATA F1,47,D1,E1,C3,CC,B1,CA
790 DATA A3,FE,CD,86,0C,32,3A,B3
800 DATA CD,1A,16,CD,FF,16,B2,68
810 DATA A4,E5,FD,E1,D5,DD,E1,CD
820 DATA A9,0B,3A,3A,B3,57,AE,A1
830 DATA 28,7E,3A,38,B3,5F,AE,A1
840 DATA 28,76,06,00,FD,23,CD,2D
850 DATA 0C,3A,34,B3,FD,95,38,52
860 '********************** Block 5
870 DATA 7A,AE,A1,28,4D,7B,AE,A1
880 DATA 20,EA,18,46,AE,77,CB,20
890 DATA E5,CB,01,DC,05,0C,7A,AE
900 DATA A1,28,12,7B,AE,A1,28,0D
910 DATA CB,C0,CB,48,20,07,DD,2B
920 DATA DF,8A,A4,DD,23,CB,A0,CB
930 DATA 09,E1,E5,CB,09,DC,F9,0B
940 DATA 7A,AE,A1,28,12,7B,AE,A1
950 DATA 28,0D,CB,E0,CB,68,20,07
960 DATA DD,23,DF,8A,A4,DD,2B,CB
970 DATA 01,E1,3A,36,B3,FD,95,30
980 DATA 0F,FD,2B,CD,13,0C,7B,AE
990 DATA A1,28,05,7A,AE,A1,20,A4
1000 DATA 2A,34,A6,7D,B4,C8,2B,22
1010 DATA 34,A6,DF,78,A4,C3,D3,A3
1020 DATA 7B,A4,FD,2A,89,AE,28,56
1030 '********************** Block 6
1040 DATA 2B,5E,2B,22,89,AE,6E,26
1050 DATA 00,C9,8,A4,FD,CB,D5,2A
1060 DATA 89,AE,01,03,00,CD,18,F6
1070 DATA 38,17,EB,22,89,AE,DD,E5
1080 DATA D1,FD,E5,C1,2B,72,2B,73
1090 DATA 2B,71,2A,34,A6,23,22,34
1100 DATA A6,D1,C1,C9,FE,03,28,41
1110 DATA FE,AB,28,57,FE,9F,CA,4A
1120 DATA A5,FE2C,28,03,FE,28,C0
1130 DATA 3E,02,B9,20,64,F1,C1,F5
1140 DATA 3E,F8,B8,C0,79,FE,EE,28
1150 DATA 15,FE,F5,C0,7E,2B,FE,72
1160 DATA 20,FA,7E,3C,20,F6,23,CD
1170 DATA 3F,DD,CD,37,DD,28,F1,CD
1180 DATA FB,CE,CD,37,DD,2C,C3,F5
1190 DATA F8,3E,0D,DB,C0,2A,34,AE
1200 '********************** Block 7
1210 DATA CD,3F,DD,FE,7C,28,03,FE
1220 DATA 83,C0,E1,2A,C2,B0,CD,21
1230 DATA FB,37,C9,3E,05,BB,C0,3E
1240 DATA 1A,91,C0,B8,C0,F1,CD,F5
1250 DATA C1,CD,D3,C1,CD,37,DD,29
1260 DATA CD,60,BB,E5,CD,19,FA,E1
1270 DATA C9,3E,49,B9,C0,F1,C1,F5
1280 DATA 3E,D0,B8,C0,3E,93,B9,C0
1290 DATA F1,E5,00,00,AF,CD,0A,FF
1300 DATA E1,C9,F1,C1,F5,3E,C8,B8
1310 DATA C0,3E,D6,B9,C0,3E,8B,BE
1320 DATA C0,F1,11,61,A5,1B,C3,DA
1330 DATA C8,06,00,FF,FF,C9,00,CD
1340 DATA 55,DD,38,10,3E,02,CD,FB
1350 DATA C1,B7,CC,84,BB,C4,81,BB
1360 DATA CD,55,DD,D0,3E,02,CD,FB
1370 '********************** Block 8
1380 DATA C1,B7,CA,7E,BB,C3,7B,BB
1390 DATA 23,01,E4,BB,FE,BA,CA,18
1400 DATA C2,FE,BB,C2,C6,DD,CD,55
1410 DATA DD,38,0A,CD,4B,C2,CD,D8
1420 DATA A5,CD,55,DD,D0,3E,02,CD
1430 DATA FB,C1,32,30,A6,C9,CD,4B
1440 DATA C2,E5,DF,C7,A3,E1,C9,CD
1450 DATA 55,DD,38,0A,CD,67,CE,32
1460 DATA 2F,A6,CD,55,DD,D0,3E,02
1470 DATA CD,FB,C1,32,31,A6,C9,CD
1480 DATA 09,BB,38,FB,C9,CF,A9,92
1490 DATA CF,F6,97,CF,F4,95,CF,F1
1500 DATA 95,46,49,4C,CC,DD,47,52
1510 DATA 41,50,48,49,43,D3,DE,4D
1520 DATA 41,53,CB,DF,46,52,41,4D
1530 DATA C5,E0,43,55,52,53,4F,D2
1540 '********************** Block 9
1550 DATA E1,43,4C,45,41,52,20,49
1560 ATA 4E,50,55,D4,E2,43,4F,50
1570 DATA 59,43,48,52,A4,7E,44,45
1580 DATA 52,D2,49,00,AE,A5,B8,A5
1590 DATA B7,A5,19,BD,67,A5,CF,A5
1600 DATA 94,95,AE,AF,BC,BD,00,FF
1610 DATA 00,FF,FF,00,00,00,00,00
1620 DATA 00,EOF
```

Mikrokomputery IBM PC (4)

Jacek WOJTALA

Każdy z mikrokomputerów osobistych wyposażony jest w system operacyjny czyli zespół programów umożliwiających wykorzystanie zasobów sprzętowych mikrokomputera, organizujących i kontrolujących jego pracę.

Mikrokomputery IBM PC/XT mogą pracować pod następującymi systemami operacyjnymi: DOS, CP/M-86, CONCURRENT DOS, CONCURRENT CP/M. Mikrokomputery IBM PC/AT ponadto mogą pracować pod systemem XENIX, wersją systemu UNIX.

System operacyjny DOS (Disk Operating System) jest najpopularniejszym z systemów wykorzystywanych w mikrokomputerach IBM PC, większość oprogramowania pracuje właśnie pod tym systemem.

Polecenia DOS dzielą się na dwa typy:

Polecenia wewnętrzne i **polecenia zewnętrzne. Polecenia wewnętrzne** wykonywane są natychmiast ponieważ wbudowane są w DOS. Jeżeli system został wczytany do pamięci operacyjnej, wówczas dyskietka ze zbiorami DOS nie jest potrzebna do wykonania tych poleceń.

Polecenia zewnętrzne umieszczone są na dyskietce jako zbiory programowe i przed ich wykonaniem muszą być przeczytane.

Dowolne zbiory z rozszerzeniem nazwy zbioru .BAT, .COM, .EXE są traktowane jako polecenia zewnętrzne. Pozwala to na tworzenie nowych, unikalnych poleceń i dołączenie ich do systemu operacyjnego.

Struktura systemu DOS

BOOT STRAP jest to program ładowania, umieszczony na ścieżce 0, w sektorze 1, na stronie 0 dyskietki, zajmuje 512 bajtów (1 sektor). Powoduje odszukanie w katalogach i wczytanie do pamięci operacyjnej programów IBMBIO.COM, IBMDOS.COM.

IBMBIO.COM jest to program pośredniczący między procedurami umieszczonymi w BIOS (Basic Input Output System), a programem IBMDOS.COM. Program dokonuje również obróbki zbioru konfiguracyjnego CONFIG.SYS. Zajmuje 9 sektorów.

IBMDOS.COM jest to program z rezydentem systemu operacyjnego, zarządza zbiorami na dyskach, wpisuje adresy procedur użytkowych, wczytuje do pamięci operacyjnej interpreter poleceń COMMAND.COM. Program zajmuje 34 sektory.

COMMAND.COM służy do interpretacji i wykonywania poleceń wewnętrznych oraz do wczytywania do pamięci operacyjnej poleceń zewnętrznych (lub programów użytkowych) i ich wykonywania. Zajmuje 36 sektorów.

PROGRAMY OBSŁUGI POLECEŃ ZEWNĘTRZNYCH

umieszczone są na dyskietce systemowej DOS lub w katalogu systemowym na dysku stałym.

Praca systemu DOS

Po włączeniu zasilania (start zimny) mikrokomputera w pierwszej kolejności wykonywane są testy sprawdzające procesor, pamięć, klawiaturę itp. Następnie jedna z proce-

➡

dur BIOS-u powoduje ściągnięcie BOOT STRAP-u, ten z kolei wczytuje do pamięci IBMBIO.COM i IBMDOS.COM. Ostatni z programów wczytuje interpreter poleceń COMMAND.COM.

W dowolnej chwili praca systemu może być przerwana, można dokonać tzw. startu gorącego czyli spowodować ponowną inicjację systemu operacyjnego.

Zbiory wsadowe

Polecenia zawarte w zbiorze wsadowym, który musi mieć rozszerzenie nazwy zbioru .BAT, wykonywane są tak samo, jak gdyby zostały wprowadzone bezpośrednio. Istnieje siedem poleceń: ECHO, FOR, GOTO, IF, SHIFT, PAUSE, REM, które pozwalają kontrolować wykonywanie zbioru wsadowego. Ostatnim poleceniem zbioru może być nazwa innego zbioru wsadowego.

Przy każdej inicjacji systemu DOS poszukuje w katalogu systemowym dysku, z którego był inicjowany, specjalnego zbioru wsadowego o nazwie AUTOEXEC.BAT. Zbiór ten jest automatycznie wykonywany po uruchomieniu systemu.

Zbiór wsadowy może zawierać parametry, które będą zastępowane argumentami w trakcie jego wykonywania.

Konfiguracja systemu

Zbiór CONFIG.SYS jest specjalnym zbiorem, zawierającym listę poleceń konfiguracyjnych pozwalających m.in.: podać liczbę buforów systemowych; podać format daty i czasu; instalować programy obsługi urządzeń; podać liczbę zbiorów, które mogą być otwarte przez blok kontroli zbiorów.

Podczas każdego startu systemu, DOS szuka — w katalogu systemowym dysku, z którego był inicjowany — zbioru CONFIG. SYS. Polecenie zawarte w tym zbiorze: BREAK, BUFFERS, COUNTRY, DEVICE, FCBS, FILES, LASTDRIVE, SHELL powodują zmianę standardowych poleceń konfiguracyjnych systemu. Przykładowo posiadając program obsługi urządzenia, które symuluje stację dysków i po umieszczeniu odpowiedniego polecenia DEVICE w zbiorze CONFIG.SYS, można wykorzystując część pamięci operacyjnej komputera zainstalować **dysk wirtualny**.

Katalogi strukturalne

Katalog systemowy dyskietki czy dysku stałego może zmieścić ograniczoną liczbę tzw. **katalogów wejściowych**. Katalogiem wejściowym może być zbiór, katalog niższego poziomu lub etykieta. Katalog systemowy dyskietki jednostronnej (160 lub 180 KB) może przechować 64 wejścia; katalog dyskietki dwustronnej (320 lub 360 KB) 112 wejść; katalog dyskietki dużej pojemności (1,2 MB) 222 wejścia; natomiast katalog dysku stałego 512 wejść. Aby uniknąć tych ograniczeń oraz aby skrócić czas poszukiwania konkretnego zbioru przez DOS, zbiory przechowywane są w grupach zwanych **katalogami niższych poziomów**. Katalogi te nie posiadają ograniczeń na ilość zbiorów w nich zapisanych, jedynym ograniczeniem jest wielkość dostępnego obszaru dyskowego.

Polecenia DOS

Wersja 3.10. DOS posiada, łącznie z poleceniami do kontroli wykonywania zbioru AUTOEXEC. BAT, a bez poleceń konfiguracyjnych zbioru CONFIG.SYS, 55 poleceń wewnętrznych i zewnętrznych. Polecenia mogą być wykorzystane, m.in. do następujących operacji:

- porównywania, kopiowania, wyświetlania, usuwania i zmiany nazw zbiorów;
- formatowania dysku stałego i dyskietek;
- uruchamiania programów użytkowych i systemowych, takich jak EDLIN, DEBUG i LINK;
- ustawiania różnych opcji pracy drukarki i monitora;
- żądania przerwania;
- skierowania wyjścia drukarki na urządzenie komunikacji asynchronicznej;
- odzyskiwania konkretnego zbioru z uszkodzonego dysku;
- przesłania zawartości ekranu graficznego na drukarkę;
- drukowania zbiorów na drukarce podczas wykonywania przez system innego zadania;
- wykonywania kopii zapasowych;
- definiowania konsoli pomocniczej;
- sortowania danych tekstowych;
- poszukiwania łańcuchów znaków w tekście;
- ustawiania znaku gotowości systemu;
- określania środowiska systemu;
- zabezpieczania zbiorów przed skasowaniem;
- zmiany etykiety dysku;
- tworzenia, usuwania i zmieniania katalogów niższego poziomu;
- wyświetlania struktury katalogu;
- ustawiania daty i czasu oraz ich formatu;
- sprawdzania błędów;
- przyłączania urządzeń do katalogu innego urządzenia.

Większość poleceń systemu DOS wersji 3.10 może być wykorzystana do pracy w sieci.

Do systemu operacyjnego dołączone są program EDLIN (edytor tekstów), program LINK (do łączenia programów), program DEBUG (do uruchamiania i testowania programów).

Jacek WOJTALA

Kompilator Hisoft — Pascal

Programy ilustrujące kurs języka Pascal prowadzonego w „IKS-ie" pisane są przy użyciu kompilatora TURBO PASCAL. Jest on bardzo wygodny w użyciu, lecz dostępny tylko dla komputerów pracujących pod CP/M. Nie jest więc dostępny na większości użytkowanych przez naszych Czytelników komputerach. Znacznie bardziej rozpowszechniony jest kompilator **HISOFT-PASCAL.** Trzeba przyznać, że kompilator ten jest mniej wygodny w użyciu, ale umożliwia uruchomienie tych programów na komputerach nie posiadających systemu CP/M. Omawiana poniżej wersja kompilatora dotyczy P.C. „Spectrum", lecz różnice występujące w innych wersjach są nieznaczne. Poprawnie wczytany kompilator rozpoczyna działanie od zadania pytań dotyczących lokalizacji poszczególnych bloków, dla większości programów wystarczy odpowiedzieć wciskając klawisz ENTER, co powoduje standardową organizację pamięci.

Po zakończeniu wczytywania możemy zacząć wpisywać nasz program. Linie programu należy ponumerować tak jak w języku Basic. **Po wpisaniu każdej linii wciskamy klawisz ENTER.**

W celu wprowadzenia poprawek w dowolnej linii przywołujemy ją instrukcją E*n* (*n* — numer edytowanej linii).

Na dole ekranu ukazuje się zapis linii w wersji pierwotnej, a poniżej numer tej linii, od którego należy wpisać zapis poprawny.

Istnieją trzy możliwości wprowadzania poprawek:

- wprowadzić pełny nowy zapis z klawiatury;
- przesunąć kursor za pomocą klawisza spacji do miejsca, w którym tekst ma być poprawiony, nacisnąć „I", w wyniku czego pojawi się znak kursora „x". Wpisywanie nowych znaków spowoduje rozsunięcie się pozostałych znaków;
- po ustawieniu kursora we właściwe miejsce nacisnąć klawisz „C", kursor staje się znakiem „+". Wprowadzany nowy tekst zapisywany jest na istniejącym.

Naciśnięcie klawisza „X" powoduje przejście kursora na koniec linii ze spisaniem pozostałych znaków linii poprawionej. Kursor zmienia się na znak „x". Pierwsze naciśnięcie ENTER powoduje powrót do normalnego kursora, drugie naciśnięcie — wprowadzenie skorygowanej linii programu.

Maszyna do pisania

Zadaniem programu jest wykorzystanie drukarki ATARI 1029 do druku tekstu z polskimi literami. Tekst nie może zawierać więcej niż 80 znaków w wierszu.

Wprowadzony zestaw polskich znaków otrzymuje się wciskając klawisz CONTROL i "A" (uzyskuje się literę ą), „C" (ć), "E" (ę), "L" (ł), "N" (ń), "O" (ó), "S" (ś), "Z" (ż), "," (ź).

Przed przystąpieniem do pisania należy ustawić lewy i prawy margines. W czasie pisania istnieje możliwość (w dowolnym momencie) zmiany marginesów, w tym celu należy nacisnąć klawisz CONTROL i "Q". Jeżeli w trakcie pisania użytkownik pomyli się, powinien wtedy nacisnąć klawisz CONTROL i "P". Wpisany fragment wiersza jest wtedy kasowany, a piszący może wprowadzić go ponownie.

Przy dochodzeniu pisanego wiersza do prawego marginesu, trzy ostatnie znaki są sygnalizowane dźwiękiem. Po wprowadzeniu ostatniego znaku wyłączana jest automatycznie klawiatura i następuje wydruk. Jeśli piszący decyduje się wydrukować krótszy wiersz niż przewiduje.to prawy margines, należy nacisnąć RETURN. Po wykonaniu przez drukarkę wydruku komputer oczekuje wprowadzenia następnego wiersza. Odstęp między wierszami wynosi cztery skoki wałka maszyny do pisania. Zwiększenie tego odstępu można uzyskać wciskając klawisz RETURN.

W celu stworzenia wizualnej kontroli nad wprowadzanym tekstem, w górnej części ekranu podane są kolejno: wartość lewego i prawego marginesu, zliczanie znaków w wierszu, zliczanie wierszy.

W programie nie zostały uwzględnione duże polskie znaki.

J. JANIEC

```
10 ? CHR$(125)
400 REM ************************
410 REM Wykorzystanie drukarki
420 REM ATARI 1029 jako maszyny
430 REM do pisania z zestawem
440 REM polskich liter
450 REM
460 REM Janusz J.
470 REM Warszawa 1986,12


480 REM ************************
500 GRAPHICS 2:COLOR 1
510 POSITION 8,3:? #6;"MINI"
520 POSITION 7,5:? #6;"EDYTOR"
525 FOR A=0 TO 100:NEXT A
530 POKE 656,1:POKE 752,1:? "START
"
560 IF PEEK(53279)=6 THEN 600
570 POSITION 8,3:? #6;CHR$(205);CH
R$(201);CHR$(206);CHR$(201)
575 POSITION 7,5:? #6;CHR$(197);CH
R$(196);CHR$(217);CHR$(212);CHR$(2
07);CHR$(210)
580 FOR A=0 TO 100:NEXT A
590 GOTO 510
600 GRAPHICS 0:?
620 ? "       ** PROSZE CZEKAC **
"


1000 REM ***********************
1010 N=57344:M=32768
1020 FOR I=0 TO 1023:POKE M+I,PEEK
(N+I):NEXT I
1040 A=M+776:B=M+520
1050 FOR I=0 TO 207:POKE B+I,PEEK(
A+I):NEXT I
1070 A=M+976:B=M+512
1080 FOR I=0 TO 7:POKE B+I,PEEK(A+
I):NEXT I
1100 FOR X=1 TO 21:READ MA,AA:POKE
 M+MA,AA:NEXT X
1140 POKE 756,128
1150 DATA 513,12,514,24,515,124
1160 DATA 527,3,537,12,538,24
1170 DATA 539,60,559,6,611,28
1180 DATA 612,56,625,12,626,24
1190 DATA 627,124,633,12,634,24
1200 DATA 635,60,664,6,665,12
1210 DATA 721,24,722,0,723,124
1230 DATA 0,9,11,45,73,1,0
1231 DATA 4,42,42,30,3,1,0
1233 DATA 0,6,9,41,73,9,0
1235 DATA 28,42,42,42,19,1,0
1242 DATA 0,0,4,127,17,0,0
1244 DATA 0,8,7,40,72,7,0
1245 DATA 0,6,9,41,73,6,0
1249 DATA 0,9,21,53,85,2,0
1256 DATA 0,17,19,85,25,17,0
1320 C=0:DIM A$(80):OPEN #3,4,0,"K
":OPEN #1,8,0,"P:":OPEN #2,8,0,"P
"
1380 ? CHR$(125):IF A=16 THEN 1440
1400 POSITION 2,2:? "LEWY MARGINES
:";:INPUT Y:IF Y<1 OR Y>80 THEN 14
00
1420 POSITION 2,4:? "PRAWY MARGINE
S:";:INPUT YY:IF YY<1 OR YY>80 THE
N 1420
1440 REM ************************
1442 W=5
1450 FOR X=1 TO Y:A$(X,X)=CHR$(32)
:NEXT X
1470 ? CHR$(125):POKE 752,1:B=Y:C=
C+1:POSITION 29,1:? "WIERSZ:";C
1500 GOTO 1520
1510 B=B+1:POKE 730,5:IF B>YY-3 TH
EN POKE 730,0
1515 IF B=YY OR A=155 THEN A$(B,B)
=CHR$(155):POKE 621,255:GOTO 1630
1520 POSITION 2,1:? "MARG.L:";Y;"
P:";YY;"   ZNAK:"
1525 K=B-Y:IF K<40 THEN 1530
1527 K=K-40:W=7
1530 POKE 621,0:POSITION K,W:? CHR
$(160);:GET #3,A
1540 IF A=16 OR A=17 THEN A$="":C=
C-1:GOTO 1380
1555 A$(B,B)=CHR$(A):POSITION 24,1
:? B+1:POSITION K,W:? CHR$(A);
1580 IF B<YY-2 THEN 1510
1590 FOR X=1 TO 5:SOUND 0,50,10,15
:NEXT X:SOUND 0,0,0,0:GOTO 1510
1620 REM ************************
1630 FOR BB=1 TO B:A=ASC(A$(BB,BB)
)
1650 IF A=0 OR A=1 OR A=3 OR A=5 O
R A=12 OR A=14 OR A=15 OR A=19 OR
A=25 THEN 1700
1660 ? #2;CHR$(A);
1670 NEXT BB
1675 IF A$(B-1,B-1)=CHR$(155) THEN
 1690
1680 IF B=YY THEN LPRINT
1690 A=125:GOTO 1442
1700 RESTORE 1230+A
1710 ? #1;CHR$(27);CHR$(54);CHR$(2
7);CHR$(65);CHR$(0);CHR$(7);
1720 FOR X=0 TO 6:READ A
1740 ? #1;CHR$(A);:NEXT X
1760 GOTO 1670
```

Dodatkowo w tej fazie można używać dyrektyw:
D*n.m* — kasuje linie między *n* a *m*:
O — zakończenie edycji z pominięciem zmian;
L — wypisanie całej linii wraz z niewidoczną częścią za kursorem.
Po wyjściu z trybu edycji możemy uzyskać:
L*n.m* — listuje na ekranie linie od n do m;
M*n.m* — zastąpienie linii m tekstem z linii n (kopiowanie linii);
N*n.m* — przenumerowanie linii od linii n do końca programu, nr linii następnej jest o m większy od poprzedniej.

Gotowy program należy wyprowadzić na taśmę. Radzę robić to stosunkowo często, gdyż kompilator ma tendencje do „zawieszania się" (komputer przestaje reagować na naciskanie jakichkolwiek klawiszy). W tym przypadku jedyną radą jest wyłączenie go na moment i ponowne wczytanie kompilatora i programu.

Wyprowadzenia na taśmę lub dysk dokonujemy instrukcją **P***n.m.s*, instrukcją tą wyprowadzamy na taśmę linie o numerach od *n* do *m* w bloku o nazwie *s*. Ponowne wczytanie dokonujemy instrukcją **G.***s* (*s* — nazwa bloku).

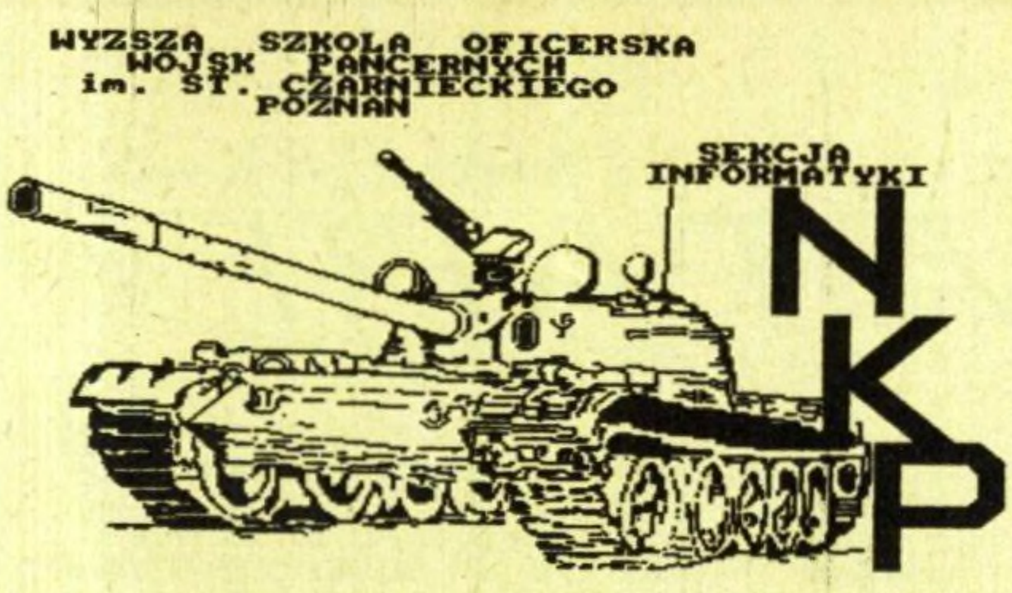
Teraz możemy przejść do kompilacji i uruchomienia programu. Instrukcja **C***n* powoduje kompilacje programu od linii *n* do końca. W razie wykrycia błędu kompilacja jest przerywana i drukowana informacja o błędzie.

Po pomyślnym zakończeniu kompilacji drukowane jest pytanie **"RUN?"**. Naciśnięcie klawisza **"Y"** powoduje uruchomienie programu wynikowego. Ponowne uruchomienie uzyskujemy naciskając klawisz **"R"**.

W celu stworzenia kompletnego programu wynikowego składa się go z podprogramów standardowych, podprogramów wspomagających i przetłumaczonego programu używamy instrukcji TN (n-numer początkowej linii programu).

Do podstawowych błędów wykrywanych w trakcie kompilacji należą:
1 — zbyt duża liczba;
2 — oczekiwano średnika;
3 — wykryto niezadeklarowany identyfikator;
4 — oczekiwano identyfikatora;
5—6 — oczekiwano "=";
8—9 — oczekiwano ":=", ")";
10 — niewłaściwy typ;
11 — oczekiwano ".";
12 — oczekiwano wskaźnika;
13—14 — oczekiwano stałej;
25—28 — oczekiwano "THEN", "DO", "TO", "(";
20—23 — oczekiwano "OF", ".", ":" "PROGRAM"
25 — oczekiwano "BEGIN"

opracował **CEZARY DOBROWOLSKI**



— *Zaczęło się to 5 lat temu* — wspomina por. **Mieczysław Skonieczny**, opiekun KNP — *szkoła dysponowała wówczas komputerem ze standardowym zestawem urządzeń zewnętrznych. Praca sekcji skupiała się na poznawaniu możliwości maszyny i podstaw informatyki. Dla nieprofesjonalistów były to sprawy chyba zbyt mało frapujące. Mieliśmy bowiem problemy z naborem chętnych.*

Punktem zwrotnym w działalności koła stało się wyposażenie szkoły w zestawy mikrokomputerowe. Zaczęli od trzech mikrokomputerów ZX Spectrum +. Rok temu otrzymali 8 zestawów Amstrad CPC 6128. Wówczas to zainteresowanie pracą sekcji informatyki, wśród podchorążych i kadetów znacznie wzrosło. Po ścisłej selekcji zaszczytu bycia jej członkiem dostąpiło 40 chętnych. Autorzy koncepcji uaktywnienia sekcji zadali sobie wiele trudu, opracowali zasady działalności, wzór statutu oraz prawa i obowiązki członków. Rozpisano również cele i zadania w szczegółach. Pierwsze tematy są łatwe: omówienie mikrokomputera Amstrad CPC 6128, ogólne uwagi o programowaniu, krótka charakterystyka języka BASIC czy omówienie zasad opracowania i edycji programu na mikrokomputerze. Później wzrasta stopień trudności, realizacja instrukcji w języku BASIC i instrukcje skoku, projekt rozwiązań graficznych ekranu, algorytm realizacji programu. Takie programowe zajęcia odbywają się 3—4 razy w miesiącu. Poza tym, członkowie sekcji uczestniczą w organizowaniu ważniejszych imprez na terenie uczelni. Ponad 4 miesiące trwał otwarty turniej szachowy, gdzie przeciwnikiem zawodników był mikrokomputer. Startowało w nim 60 szachistów. W wyniku pucharowego systemu wyłoniono pięciu zwycięzców. I tak to zabawa edukacyjna wpleciona została w proces nauczania.

Sekcja informatyki KNP była również współorganizatorem ogólnokrajowego sympozjum naukowego nt. wykorzystania mikrokomputerów w systemie dydaktycznym. Członkowie różnych sekcji prezentowali wybrane programy dydaktyczne wspomagające proces nauczania. Pięć z nich opracowała sekcja informatyki. Trzeba przyznać, że podchorążym i kadetom stworzono bardzo dobre warunki do pracy naukowej. Posiadają odpowiednią salę komputerową, bibliotekę. Dobrym duchem sekcji jest jej kierownik st. kpr. kdt **Grzegorz Siankowski**. — *Moje zainteresowania* — mówi — *zrodziły się w czasie mikrokomputerowych gier. Najpierw była fascynacja samą grą, a później dochodzenie do tego, jak ona powstaje. Zaczęła się żmudna praca od podstaw: poznanie języka, komend, pisanie programu. Teraz najciekawsze jest oglądanie efektu programowania.*

Na przykład program „budowa i zasady działania mikrokomputera" tworzyli wspólnie z opiekunem sekcji prawie miesiąc. Bywało, że po ośmiu godzinach ślęczenia, oczy i głowa bolały, ale coś z tej pracy pozostało. I to cieszy, to wciąga bez reszty. Mieliśmy okazję obejrzenia kilku programów dydaktycznych i trzeba powiedzieć, że są to udane dzieła i o to przecież chodzi.

Jerzy RAUBE

Czołgiści wśród mikrokomputerów

Foto: Jan Zelman

Komputer u kierowców

Ten program opracowano w Wyższej Szkole Oficerskiej Wojsk Pancernych im. S. Czarnieckiego w Poznaniu. Przeznaczony jest do badania kierowców, a zastąpić może niekiedy urządzenia wykorzystywane w Wojskowej Poradni Psychotechnicznej.

W tym przypadku program symuluje pracę aparatu PIORKOWSKI służącego do oceny koordynacji wzrokowo-ruchowej i stopnia koncentracji uwagi kierowców.

Pakiet programów użytkowych obejmuje 4 segmenty:

— menu główne, które informuje użytkownika o możliwościach pakietu;
— program badający czas reakcji prostej i z wyborem kierowców;
— symulator aparatu PIORKOWSKI do badania koordynacji wzrokowo-ruchowej i stopnia koncentracji uwagi kierowców;
— program badający organizację przestrzenną i szybkość podejmowania decyzji przez kierowcę.

Kierowca poddawany jest testowi koordynacji wzrokowo-ruchowej i stopnia koncentracji uwagi w 2 etapach;

— etap próbny: emisja dowolnej ilości prób (w zależności od czasu reakcji badanego (w czasie 30 sekund);
— etap właściwy: emisja 150 prób w jednakowych odstępach czasu.

Etap próbny ma na celu przygotowanie badanego kierowcy do właściwej części testu i oswojenia użytkownika za sprzętem mikrokomputerowym. Na tym etapie nie jest badany czas reakcji. Kolejne losowe wyświetlenie lampki jest uzależnione od reakcji kierowcy.

Etap właściwy polega na losowym zapalaniu lampek w równych odstępach czasowych. Mikrokomputer zalicza ilość:

— poprawnych reakcji (badany kierowca zdążył nacisnąć właściwy klawisz);
— błędnych reakcji (badany kierowca nie zdążył nacisnąć właściwego klawisza lub nacisnął klawisz ze złym odpowiednikiem cyfrowym).

Po zakończeniu etapu właściwego są wyświetlane wyniki badania, które zawierają:

— sumaryczną ilość prób dobrych;
— sumaryczną ilość prób złych;
— ilość prób złych w 1 i 2 części etapu właściwego.

Etap właściwy podzielono na dwie części (każda po 75 prób), aby obiektywniej ocenić użytkownika uwzględniając proces zmęczenia badaniem. W dolnej części ekranu wyświetlana jest ocena słowna, określająca stopień przydatności do wykonywania zawodu kierowcy. Poniżej pokazuje się ocena w skali czterostopniowej.

Do segmentów menu głównego i programu właściwego dołączono podprogram, który umożliwia wypisywanie tekstów na ekranie monitora z uwzględnieniem liter charakterystycznych dla pisowni w języku polskim (np. ą, ę, ń, ł). Wpływa to na zwiększenie czytelności prezentowanych informacji przez użytkownika mikrokomputera. W programie wykorzystano duże możliwości kolorystyczne, dźwiękowe i graficzne mikrokomputera AMSTRAD CPC 6128.

Danuta KWASIŻUR, Mieczysław SKONIECZNY, Robert ZUGEHÖR

```
10 MODE 1:CLEAR INPUT:INK 0,0:INK 1,5:INK 2,24:BO
RDER 0:PAPER 0:CLS:GOSUB 900:WINDOW #1,1,40,10,16:
PAPER #1,2:CLS #1
20 PEN #1,0:LOCATE #1,2,4:PRINT#1,"APARATU 'PIORKO
WSKI' DO BADANIA KOORDY-":LOCATE #1,2,5:PRINT#1,"N
ACJI WZROKOWO-RUCHOWEJ I STOPNIA KON-"
30 LOCATE #1,2,6:PRINT#1."CENTRACJI UWAGI KIEROWCÓ
```

```
W":SPEED INK 10,10
40 GRAPHICS PAPER 2:TAG:FOR x=550 TO -450 STEP -16
50 GRAPHICS PEN 1:PAPER 2:FRAME:MOVE x,240:PRINT "
 SYMULATOR MIKROKOMPUTEROWY ";
60 IF INKEY$=" " THEN 90
70 NEXT
80 GOTO 40
90 TAGOFF:MODE 1:GOSUB 800:PEN 2:PAPER 0:CLS:PRINT
 "Zadaniem programu jest symulacja badaniakierowc5
w na aparacie typu 'PIORKOWSKI' wykorzystujlca mik
rokomputer AMSTRAD    CPC 6128.
100 WINDOW #1,1,40,6,14:PAPER #1,2:CLS #1:PEN #1,0
:LOCATE #1,2,1:PRINT #1,"Na ekranie monitora wy7wi
etli si[ obraz":LOCATE #1,2,2:PRINT#1,"10 ponumero
wanych lampek. Zadaniem ba-"
110  LOCATE #1,2,3:PRINT #1,"danego jest naci7ni[c
ie po zapaleniu":LOCATE #1,2,4:PRINT#1,"dowolnej l
ampki klawisza numerycznego":LOCATE #1,2,5:PRINT#1
,"z odpowiadaj!c! lampce cyfr! w g5rnym":LOCATE #1
,2,6:PRINT#1,"rz[dzie klawiatury mikrokomputera.Ki
e-"
120 LOCATE #1,2,7:PRINT #1,"rowca poddawany jest t
estowi koordyna-":LOCATE #1,2,8:PRINT #1,"cji wzro
kowo-ruchowej i stopnia koncen-":LOCATE #1,2,9:PRI
NT#1,"tracji uwagi w dw5ch etapach:"
130 LOCATE 1,16:PRINT"1. Etap pr5bny - emisja dowo
lnej ilo7ci    pr5b (w zale9no7ci od czasu reakcji
     badanego) w czasie 30 sek."
140 LOCATE 1,19:PRINT"2. Etap wla7ciwy - emisja 15
0 pr5b w je-   dnakowych odst[pach czasowych"
150 WINDOW #2,1,40,22,24:PAPER #2,2:PEN #2,0:CLS #
2:LOCATE #2,2,1:PRINT #2,"Na zako3czenie badania k
ierowcy b[dzie":LOCATE #2,2,2:PRINT #2,"wy7wietlan
a jego ocena w skali cztero-":LOCATE #2,2,3:PRINT#
2,"stopniowej"
160 IF INKEY$="" THEN 160
170 MODE 0:INK 1,26:WINDOW #1,4,17,12,14:PAPER #1,
2:CLS #1:PEN #1,0:LOCATE #1,2,2:PRINT#1,"ETAP PR6B
NY":INK 2,24,26
180 IF INKEY$="" THEN 180
190 INK 2,24:MODE 1:INK 2,14:INK 3,6:CLS:WINDOW #1
,2,39,2,7:PAPER #1,2:CLS #1:PEN #1,0:WINDOW #2,1,3
6,10,13:PAPER #2,3:CLS #2:PEN #2,0:WINDOW #3,2,39,
16,20:PAPER #3,2:CLS #3:PEN #3,0
200 LOCATE #1,2,2:PRINT #1,"Etap pr5bny ma na celu
 przygotowanie":LOCATE #1,2,3:PRINT#1,"badanego ki
erowcy do wla7ciwej cz[7ci":LOCATE #1,2,4:PRINT#1,
"testu i oswojenie u9ytkownika ze":LOCATE #1,2,5:P
RINT#1,"sprz[tem mikrokomputerowym."
210 LOCATE #2,2,2:PRINT#2,"NA TYM ETAPIE NIE JEST
BADANY CZAS ":LOCATE #2,2,3:PRINT#2,"REAKCJI.":LOC
ATE #3,2,2:PRINT#3,"Kolejne, losowe wy7wietlenie l
ampki":LOCATE #3,2,3:PRINT#3,"jest uzale9nione od
czasu reakcji ":LOCATE #3,2,4:PRINT#3,"kierowcy."
220 PEN 3:LOCATE 1,23:PRINT"Ten etap zako3czy si[
po up1ywie 30 sek."
230 IF INKEY$=""THEN 230
240 GOSUB 1110
250 GOSUB 710
260 AFTER 1500,1 GOSUB 330
270 FOR g=1 TO 200:NEXT:GOSUB 790
280 a$=INKEY$:IF f=1 GOTO 340
290 IF a$="" THEN 280
300 IF nr$="10" THEN nr$="0"
310 IF NOT(nr$=a$) THEN GOTO 280
320 PAPER 0:INK nr,14:GOTO 270
330 f=1:s=REMAIN(1):RETURN
340 MODE 0:INK 0,0:PAPER 0:BORDER 0:CLS:GOSUB 800:
INK 2,24:WINDOW #1,3,17,12,14:PAPER #1,2:CLS #1:PE
N #1,0:LOCATE #1,2,2:PRINT #1,"ETAP W2A8CIWY":SPEE
D INK 20,20:INK 2,24,26
350 IF NOT(INKEY$=" ") THEN 350
360 INK 2,24:MODE 1:GOSUB 800:INK 0,0:INK 1,26:INK
 2,14:INK 3,6:WINDOW #1,1,40,1,4:PAPER #1,2:CLS #1
:PEN #1,0:LOCATE #1,1,2:PRINT#1,"Etap wla7ciwy pol
ega na losowym zapala-":LOCATE #1,1,3:PRINT#1,"niu
 lampek w r5wnych odst[pach czasowych"
370 PAPER 0:PEN 1:LOCATE 1,6:PRINT"Mikrokomputer z
licza ilo7~:":LOCATE 1,8:PRINT"- poprawnych reakcj
i (badany kierowca     zd!9y1 nacisn!~ wla7ciwy kl
awisz)     - b![dnych reakcji (badany kierowca nie
   zd!9y1 nacisn!~ wla7ciwego klawisza"
380 LOCATE 1,12:PRINT"  lub nacisn!1 klawisz ze z1
ym odpowie-   dnikiem cyfrowym)"
390 LOCATE 4,15:PEN 3:PRINT "UWAGA!":WINDOW #2,1,4
0,17,25:PAPER #2,3:CLS #2:PEN #2,0:LOCATE #2,1,2:P
RINT#2,"Mikrokomputer generuje numer wy7wietla- ne
j lampki losowo tzn badany kierowca   nie mo9e prz
ewidzie~ kt5ra lampka zapa-"
400 LOCATE #2,1,5:PRINT#2,"li si[ w nast[pnej kole
jno7ci. Jest mo- 9liwe 9e w nast[pnej pr5bie wy7wi
etli   si[ ta sama lampka. Efektem takiej sy- tua
cji na ekranie jest mignilcie tej    samej cyfry."
410 IF NOT (INKEY$=" ") THEN 410
420 GOSUB 1110
430 P=0:PEN 13:GOSUB 710
440 FOR r=1 TO 2:b=0
450 FOR n=1 TO 75:GOSUB 790
460 AFTER 70,1 GOSUB 700
470 CLEAR INPUT
480 a$=INKEY$:IF f=1 GOTO 540
490 IF a$="" THEN 480
500 IF nr$="10" THEN nr$="0"
510 IF nr$=a$ GOTO 530
520 b=b+1:GOSUB 690:GOTO 550
530 GOSUB 690:P=P+1
540 F=0
550 NEXT:bl(r)=b:NEXT
560 S=REMAIN(1):INK 1,26:INK 0,0:INK 2,24:INK 3,0:
PAPER 2:BORDER 24:MODE 1:CLS:GOSUB 800:WINDOW #1,3
,17,3,5:PAPER #1,1:CLS #1:PEN #1,0:LOCATE #1,2,2:P
RINT#1,"WYNIKI TESTU":INK 0,0,24
570 PEN 3:LOCATE 3,7:PRINT "PRAWIDLOWO - ";:PRINT
USING "###";P:LOCATE 3,9:PRINT "BLEDNIE    - ";:PR
INT USING "###";Bl(1)+bl(2);
580 LOCATE 3,11:PRINT"1 CZ08] - ";USING "###";BL(1
);:PRINT" BLEDOW":LOCATE 3,12:PRINT"2 CZ08] - ";US
ING "###";BL(2);:PRINT" BLEDOW"
590 INK 1,6:PEN 1:LOCATE 9,15:PRINT"***** OCENA **
***":PEN 3
```

```
600 IF p>140 GOTO 660
610 IF p>124 GOTO 650
620 IF p>100  GOTO 640
630 LOCATE 1,17:PRINT"TWOJE WYNIKI S) BARDZO S2ABE
.CZAS BADA- NIA KOORDYNACJI WZROKOWO-RUCHOWEJ I ST
O-PNIA KONCENTRACJI UWAGI DYSKWALIFIKUJE  CIO JAKO
 KIEROWCO WOJSKOWEGO. ZDECYDOWA-NIE MUSISZ )WICZY)
.":PRINT:PEN 1:PRINT"UZYSKA2E8 OCENO  2":GOTO 670
640 LOCATE 1,17:PRINT"TWOJE WYNIKI S) S2ABE. CZAS
BADANIA     KOORDYNACJI WZROKOWO-RUCHOWEJ I STOPNI
A KONCENTRACJI UWAGI KWALIFIKUJE CIO JAKO KIEROWCO
 WOJSKOWEGO.ALE MUSISZ ZDECYDO- WANIE )WICZY).":PR
INT:PEN 1:PRINT"UZYSKA2E8 OCENO  3":GOTO 670
650 LOCATE 1,17:PRINT"TWOJE WYNIKI S) DOBRE. CZAS
BADANIA     KOORDYNACJI WZROKOWO-RUCHOWEJ I STOPNI
A KONCENTRACJI UWAGI KWALIFIKUJE CIO JAKO KIEROWCO
 WOJSKOWEGO.NIE ZASZKODZI JEDNAKJAK BODZIESZ )WICZ
Y2.":PRINT:PEN 1:PRINT"UZYSKA2E8 OCENO  4":GOTO 67
0
660 LOCATE 1,17:PRINT"TWOJE WYNIKI S) BARDZO DOBRE
. CZAS BADA-NIA KOORDYNACJI WZROKOWO-RUCHOWEJ I ST
O-PNIA KONCENTRACJI UWAGI KWALIFIKUJE CIO JAKO KIE
ROWCO WOJSKOWEGO. NIE MUSISZ    )WICZY).":PRINT:PE
N 1:PRINT"UZYSKA2E8 OCENO  5":GOTO 670
670 IF NOT(INKEY$="s" OR INKEY$="S") THEN 670
680 GOTO 10
690 s=REMAIN(1):PAPER 0:INK nr,14:RETURN
700 SOUND 1,2000:B=B+1:GOSUB 690:F=1:RETURN
710 MODE 0:INK 11,0:INK 0,1:GRAPHICS PEN 11:INK 12
,14:PEN 13,0:BORDER 1:PAPER 0:CLS
720 FOR a=1 TO 10
730 INK a,14:x=64*(a-1):PLOT x,172:DRAW x+62,172:P
LOT x,174:DRAW x+62,174:DRAW x+62,238:DRAW x,238:D
RAW x,174:PLOT x,240:DRAW x+62,240:MOVE x+4,180:FI
LL a
740 NEXT
750 PEN 11:c=1:b=0:FOR a=1 TO 19 STEP 2
760 PAPER c:LOCATE a,12:PRINT CHR$(125+b);CHR$(126
+b)
770 LOCATE a,13:PRINT CHR$(127+b);CHR$(128+b):b=b+
4:c=c+1
780 NEXT:f=0:RETURN
790 nr=1+INT(10*RND):nr$=RIGHT$(STR$(nr),1):INK nr
,6:RETURN
800 SYMBOL AFTER 170
810 SYMBOL 171,0,0,120,12,124,204,118,3
820 SYMBOL 172,24,60,102,102,126,102,102,3
830 SYMBOL 173,24,0,60,102,96,102,60,0
840 SYMBOL 174,24,60,102,192,192,102,60,0
850 SYMBOL 175,0,0,60,102,126,96,60,6
860 SYMBOL 176,254,98,104,120,104,98,254,3
870 SYMBOL 177,56,24,28,24,56,24,60,0
880 SYMBOL 178,240,96,112,96,226,102,254,0
890 SYMBOL 179,24,0,220,102,102,102,102,0
900 SYMBOL 180,24,198,230,246,222,206,198,0
910 SYMBOL 181,24,0,60,102,102,102,60,0
920 SYMBOL 182,12,56,108,198,198,108,56,0
930 SYMBOL 183,24,0,60,96,60,6,124,0
940 SYMBOL 184,24,60,96,60,6,102,60,0
950 SYMBOL 185,24,0,126,76,24,48,126,0
960 SYMBOL 186,254,198,140,126,48,102,254,0
970 SYMBOL 187,12,24,126,76,24,48,126,0
980 SYMBOL 188,48,254,172,24,50,102,254,0
990 SYMBOL 189,0,0,0,0,0,12,12,24
1000 KEY DEF 10,1,55,171,172
1010 KEY DEF 11,1,56,173,174
1020 KEY DEF 3,1,57,175,176
1030 KEY DEF 20,1,52,177,178
1040 KEY DEF 12,1,53,179,180
1050 KEY DEF 4,1,54,181,182
1060 KEY DEF 13,1,49,183,184
1070 KEY DEF 14,1,50,185,186
1080 KEY DEF 5,1,51,187,188
1090 KEY DEF 15,1,189
1100 RETURN
1110 SYMBOL AFTER 125
1120 SYMBOL 125,129,129,131,131,129,129,129,129
1130 SYMBOL 126,129,129,129,129,129,129,129,129
1140 SYMBOL 127,129,129,129,129,131,131,128,128
1150 SYMBOL 128,129,129,129,129,193,193,1,1
1160 SYMBOL 129,131,131,134,134,128,128,131,131
1170 SYMBOL 130,193,193,97,97,97,97,193,193
1180 SYMBOL 131,134,134,134,134,135,135,128,128
1190 SYMBOL 132,1,1,97,97,225,225,1,1
1200 SYMBOL 133,131,131,132,132,128,128,131,131
1210 SYMBOL 134,193,193,97,97,97,97,193,193
1220 SYMBOL 135,128,128,134,134,131,131,128,128
1230 SYMBOL 136,97,97,97,97,193,193,1,1
1240 SYMBOL 137,129,129,131,131,133,133,137,137
1250 SYMBOL 138,129,129,129,129,129,129,129,129
1260 SYMBOL 139,143,143,129,129,131,131,128,128
1270 SYMBOL 140,225,225,129,129,193,193,1,1
1280 SYMBOL 141,135,135,134,134,134,134,135,135
1290 SYMBOL 142,225,225,33,33,1,1,193,193
1300 SYMBOL 143,128,128,134,134,131,131,128,128
1310 SYMBOL 144,97,97,97,97,193,193,1,1
1320 SYMBOL 145,131,131,134,134,134,134,135,135
1330 SYMBOL 146,193,193,97,97,1,1,193,193
1340 SYMBOL 147,134,134,134,134,131,131,128,128
1350 SYMBOL 148,97,97,97,97,193,193,1,1
1360 SYMBOL 149,135,135,132,132,128,128,128,128
1370 SYMBOL 150,225,225,97,97,97,97,193,193
1380 SYMBOL 151,129,129,129,129,129,129,128,128
1390 SYMBOL 152,129,129,129,129,129,129,1,1
1400 SYMBOL 153,131,131,134,134,134,134,131,131
1410 SYMBOL 154,193,193,97,97,97,97,193,193
1420 SYMBOL 155,134,134,134,134,131,131,128,128
1430 SYMBOL 156,97,97,97,97,193,193,1,1
1440 SYMBOL 157,131,131,134,134,134,134,131,131
1450 SYMBOL 158,193,193,97,97,97,97,225,225
1460 SYMBOL 159,128,128,134,134,131,131,128,128
1470 SYMBOL 160,97,97,97,97,193,193,1,1
1480 SYMBOL 161,131,131,134,134,134,134,134,134
1490 SYMBOL 162,193,193,97,97,97,97,97,97
1500 SYMBOL 163,134,134,134,134,131,131,128,128
1510 SYMBOL 164,97,97,97,97,193,193,1,1
1520 RETURN
```

Grafika na ekranie (1)

Grafika komputerowa fascynuje większość użytkowników mikrokomputerów. Konstrukcja obrazu wymaga znajomości zasad jego definiowania i przekształcania. W prezentowanym cyklu przedstawimy programy przydatne do przedstawiania i przekształcania położenia punktów i linii. Konstrukcja tych programów oparta będzie na zastosowaniu matematyki w dziedzinie grafiki komputerowej. Każdy program poprzedzony będzie krótkim opisem zastosowanej metody matematycznej. Wyboru metod matematycznych stosowanych w grafice komputerowej dokonano na podstawie pracy D. F. Rogers'a i I. A. Adams'a: „MATHEMATICAL ELEMENTS FOR COMPUTER GRAPHICS" (New York, McGraw-Hill Book Company).

PRZEDSTAWIANIE PUNKTÓW

Na płaszczyźnie punkt reprezentowany jest przez dwie współrzędne. Można je traktować jako elementy macierzy wierszowej [x y]. W przestrzeni trójwymiarowej każdy punkt przedstawia macierz wierszowa [x y z]. Można je również przedstawić w postaci macierzy kolumnowej $\begin{bmatrix} x \\ y \end{bmatrix}$ lub $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$.

PRZEKSZTAŁCENIA I MACIERZE

Macierzowy zapis współrzędnych punktu jest wygodny do:
— reprezentacji maszynowej,
— zastosowania algebry macierzowej, co umożliwia wykonywanie transformacji obrazu.

Wiele fizycznych zadań można sprowadzić do następującej formuły. Niech będą dane macierze A i B oraz zależność AT = B. Łatwo znaleźć macierz transformacji $T = A^{-1}B$,
gdzie A^{-1} jest macierzą odwrotną do macierzy kwadratowej A.

Interpretacja podanej formuły w grafice komputerowej:
A — macierz zdefiniowanych współrzędnych punktów (współrzędne pierwotne),
B — macierz przekształconych współrzędnych punktów (współrzędne po transformacji).
T — macierz transformacji.

Interpretacja mnożenia macierzy jako operatora geometrycznego stanowi podstawę matematycznych przekształceń wykorzystywanych w grafice komputerowej.

PRZEKSZTAŁCENIA PUNKTÓW

Rozważmy rezultat macierzowego pomnożenia macierzy [xy] odpowiadającej punktowi *p* oraz macierzy transformacji [2 x 2].

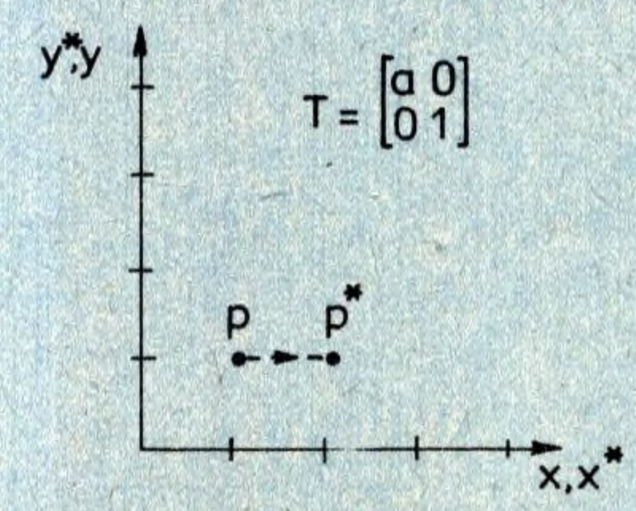
$$[xy]\begin{bmatrix} a & b \\ c & d \end{bmatrix} = [(ax + cy)\ (bx + dy)] = [x^*\ y^*]$$

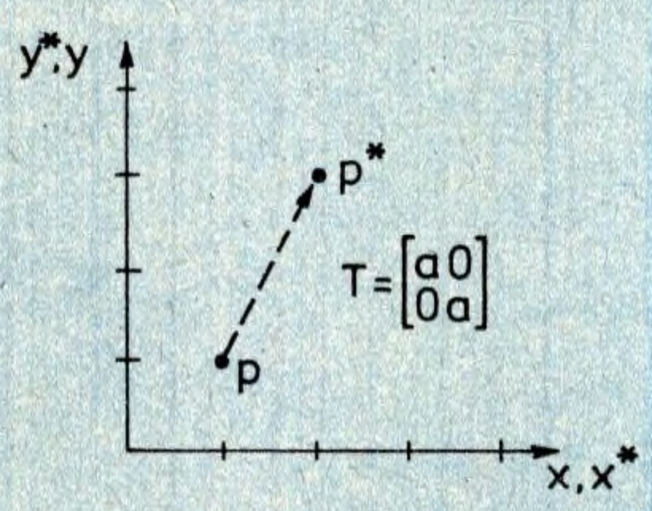
Ten matematyczny zapis oznacza, że współrzędne x i y przekształcone zostały w x^* i y^*, gdzie: $x^* = (ax + cy)$, $y^* = (bx + dy)$. Przeprowadźmy analizę otrzymanych wyników, rozpatrując x^* i y^* jako przekształcone (przetransportowane) współrzędne. Analiza ta pozwoli wyłonić podstawowe przekształcenia obrazów (zbiorów punktów).

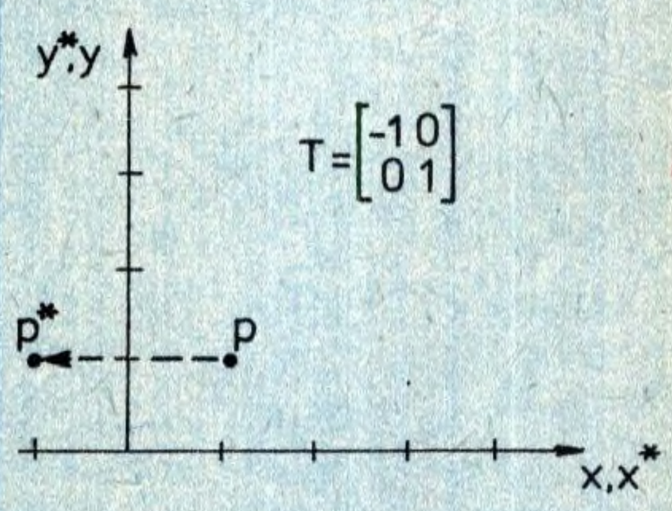
PRZYPADEK 1

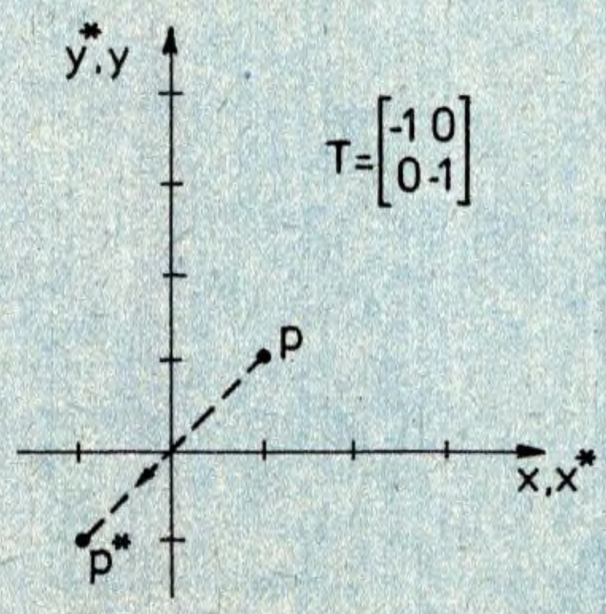
Niech d = 1, b = c = o.

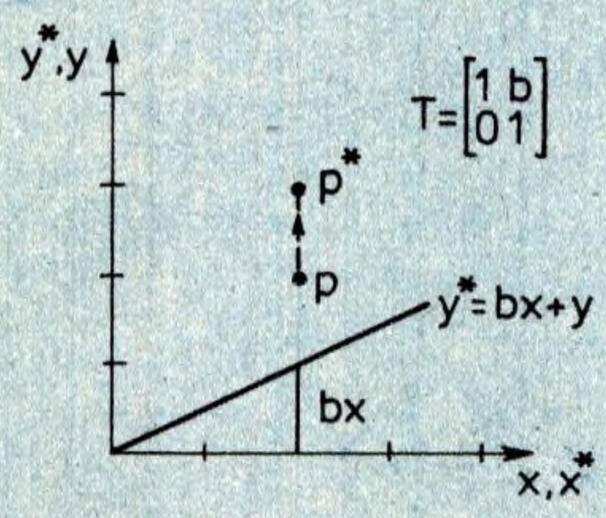
$$[xy]\begin{bmatrix} a & o \\ o & 1 \end{bmatrix} = [ax\ y] = [x^*y^*]$$

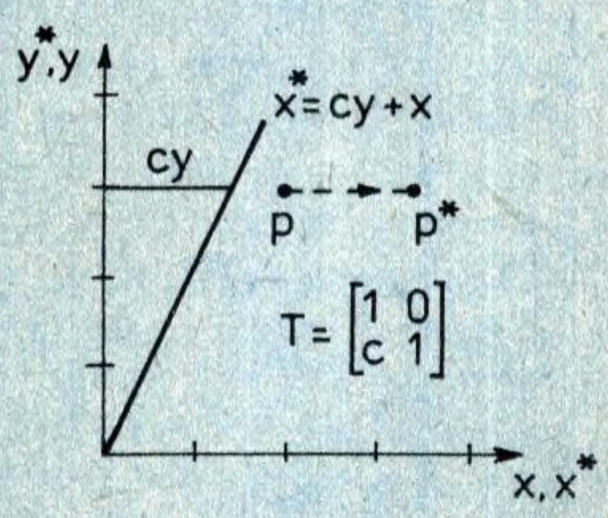












Przekształcenie takie — przedstawione na rys. 1 — nosi nazwę powinowactwa prostokątnego. W wyniku następuje przesunięcie punktu wzdłuż osi x.

PRZYPADEK 2

Niech $b = c = 0$

$[xy]\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = [ax\ dy] = [x^*y^*]$

Przekształcenie takie — przedstawione na rys. 2 — nazywamy skalowaniem w kierunku osi x i y.

Jeśli $a = d > 1$ to następuje zwiększenie współrzędnych punktu *p*.

Jeśli $0 < a = d < 1$ to następuje zmniejszenie współrzędnych punktu *p*.

Gdy $a \neq b$ to skalowanie współrzędnych punktu *p* jest nierównomierne wzdłuż osi x i y.

PRZYPADEK 3

Niech $a = -1, b = c = 0, d = 1$

$[xy]\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = [-x\ y] = [x^*y^*]$

Przekształcenie takie — przedstawione na rys. 3 — nosi nazwę lustrzanego odbicia względem osi y.

PRZYPADEK 4

Niech $a = d = -1, b = c = 0$.

$[xy]\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = [-x\ -y] = [x^*y^*]$

Przekształcenie takie — przedstawione na rys. 4 — nazywamy symetrią środkową.

PRZYPADEK 5

Niech $a = d = 1, c = 0$

$[xy]\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} = [x\ (bx + y)] = [x^*y^*]$

Przekształcenie takie — przedstawione na rys. 5 — nosi nazwę skręcenia. Współrzędna y^* zależy liniowo od współrzędnych pierwotnych punktu *p*.

Analogiczne przekształcenie zachodzi, gdy $a = d = 1$ i $b = 0$. Ilustruje to rysunek 6.

Przyjmując, że położenie dowolnego odcinka może być przekształcone w dowolne nowe położenie prostą transformacją jego granicznych punktów, możemy wszystkie przedstawione powyżej transformacje wykorzystać do przekształcenia odcinków oraz figur z nich utworzonych.

Definiowanię figury złożonej z odcinków umożliwia nam zapis macierzowy; w kolejnych wierszach macierzy zapisujemy współrzędne punktów granicznych odcinków kreślących figurę.

UWAGA

Wszystkie powyżej opisane przekształcenia z zastosowaniem ogólnej macierzy transformacji 2 x 2 dokonywane były względem początku układu współrzędnych określonego macierzą [∅, ∅]. Jest to ograniczenie, które zostanie wyeliminowane w kolejnych odcinkach niniejszego cyklu. Przedstawiony na przykładzie punktu zbiór prostych przekształceń na płaszczyźnie obejmuje:

— powinowactwo prostokątne,
— skalowanie,
— lustrzane odbicie,
— symetrię środkową,
— skręcenie.

Zaprezentujemy teraz program umożliwiający wykonanie podstawowego przekształcenia na płaszczyźnie jakim jest skalowanie równomierne wzdłuż osi x i y. Dodatkowo program umożliwia realizację symetrii środkowej.

Program skalowania napisany został w języku Basic mikrokomputera Amstrad 6128. Użyto Basica ze względu na powszechną jego znajomość. Konstrukcja programu pozwoli łatwo przetłumaczyć go na inny język programowania. Pożądana jest jednak duża rozdzielczość ekranu z uwagi na zakres współrzędnych elementów obrazu. Program napisany został w sposób konwersacyjny. Jednakże ze względu na udostępnienie użytkującym jak największego zakresu współrzędnych obrazu, komunikacja programu z użytkownikiem sprowadzona została do jednego 40-znakowego wiersza. Stąd kilka słów komentarza. Rysowanie (definiowanie) obrazu odbywa się poprzez podawanie współrzędnych kolejnych punktów obrazu. Punkty te łączone są odcinkami. Po wprowadzeniu każdej współrzędnej punktu wypisywany jest komunikat *„koniec wprowadzania współ. —k, nie —n"*. Gdy chcemy zakończyć wprowadzanie współrzędnych naciskamy klawisz *K* oraz *ENTER*. W przeciwnym wypadku *N* oraz *ENTER*. Komunikat *„podaj skalę"* wypisywany jest po zakończeniu definiowania obrazu. Gdy wprowadzimy liczbę z zakresu (0, 1) to uzyskamy zmniejszenie skali. Liczba większa od 1 powoduje zwiększenie skali. Wprowadzając jako skalę liczbę ujemną uzyskujemy przekształcenie obrazu nazwane **SYMETRIĄ ŚRODKOWĄ**. Po wyskalowaniu obrazu, sygnalizowanym komunikatem *„wyskalowano: ponownie —r, koniec —k"*, uzyskujemy na ekranie monitora dwie figury. Jedną zdefiniowaną przez nas oraz drugą przekształconą operacją skalowania. Gdy chcemy kontynuować skalowanie naciskamy klawisz **r**. Wciśnięcie klawisza **K** kończy działanie programu. Wybierając kontynuację zauważymy komunikat *„kasowanie obrazu—k, rysowanie—r"*. Wybór rysowania powoduje definiowanie obrazu od początku. Kasowanie obrazu oznacza, że będziemy mogli pozostawić na ekranie jedną z obrazowanych figur. Wyboru dokonujemy po komunikacie *„nowego obrazu—n, starego—s"*. Nowy obraz oznacza figurę uzyskaną po przekształceniu uprzednio zdefiniowanej (starej). Po wyborze figury do skasowania wyłania się nam kolejna możliwość sygnalizowana komunikatem *„skalowanie—s, dorysowanie—d"*. Wybierając skalowanie dokonujemy ponownego skalowania obrazu. Dorysowanie umożliwia kontynuację definiowania kolejnych elementów obrazu.

Andrzej CETERA

```
30 REM SKALA
60 DIM wsp(101,2),mat(2,2),nwsp(101,2)
70 MODE 1:INK 0,13:INK 1,3:INK 2,12:INK 3,0
80 BORDER 0:PEN 1:PAPER 2
90 CLS
100 PRINT "Skalowanie obrazu w ukladzie"
110 PRINT "wspolrzednych:-319<x<319,-191<y<191"
120 PRINT
130 INPUT "Gdy jestes gotow to wcisnij t"; nk$
140 IF nk$="t" THEN 160
150 GOTO 130
160 GRAPHICS PAPER 1:REM Tlo grafiki
170 ORIGIN 320,208,0,640,8,400
180 CLEAR INPUT
190 WINDOW 1,40,25,25
200 GOSUB 1200:REM Rysowanie ukladu wspolrzed.
210 i=0 :REM  Licznik punktow
220 INPUT "Podaj wsp. punktu x,y";x,y
230 i=i+1
240 IF i>100 THEN GOTO 370
250 wsp(i,1)=x
260 wsp(i,2)=y
270 GOSUB 790:REM wykreslanie punktu lub odcinka
280 INPUT "koniec wprowadzania wspol.-k,nie-n";nk$
290 IF nk$="n" THEN 220
```

```
300 IF nk$="k" THEN 320
310 GOTO 280
320 INPUT "Podaj skale";p
330 GOSUB 860 :REM Ustaw macierz przeksz.
340 GOSUB 1000 :REM Mnozenie macierzy
350 GOSUB 1110 :REM Zobrazowanie
360 GOTO 390
370 INPUT "Juz za duzo punktow:skala";p
380 GOTO 330
390 INPUT "wyskalowano:ponownie-r,koniec-k";nk$
400 IF nk$="r" THEN 430
410 IF nk$="k" THEN 1400
420 GOTO 390
430 INPUT "kasowanie obrazu-k,rysowanie-r";nk$
440 IF nk$="k" THEN 470
450 IF nk$="r" THEN 680
460 GOTO 430
470 INPUT"nowego obrazu-n,starego-s";nk$
480 IF nk$="n" THEN 630
490 IF nk$="s" THEN 510
500 GOTO 470
510 REM Kasowanie starego obrazu
520 GOSUB 1200:REM Rys. ukladu wspolrzednych
530 FOR k=1 TO i
540 wsp(k,1)=nwsp(k,1)
550 wsp(k,2)=nwsp(k,2)
560 NEXT k
570 GRAPHICS PEN 3
580 GOSUB 1300
590 INPUT "skalowanie-s,dorysowanie-d";nk$
600 IF nk$="s" THEN 320
610 IF nk$="d" THEN 220
620 GOTO 590
630 REM Kasowanie nowego obrazu
640 GOSUB 1200
650 GRAPHICS PEN 3
660 GOSUB 1300
670 GOTO 590
680 FOR n=1 TO i:REM Rysowanie od poczatku
690 FOR k=1 TO 2
700 wsp(n,k)=0
710 nwsp(n,k)=0
720 NEXT k
730 NEXT n
740 GOTO 200
750 REM ########################################
760 REM     PODPROGRAMY
770 REM ########################################
780 REM
790 REM Kreslenie punktu lub odcinka
800 REM
810 GRAPHICS PEN 3:REM Stare wspolrzedne
820 IF i=1 THEN PLOT wsp(i,1),wsp(i,2)
830 IF i>1 THEN DRAW wsp(i,1),wsp(i,2)
840 RETURN
850 REM
860 REM Ustawianie macierzy przeksztalcenia
870 REM
880 FOR k=1 TO 2
890 mat(k,k)=p
900 NEXT k
910 mat(1,2)=0
920 mat(2,1)=0
930 FOR k=1  TO i:REM Zerowanie nwsp
940 FOR l=1 TO 2
950 nwsp(k,l)=0
960 NEXT l
970 NEXT k
980 RETURN
990 REM
1000 REM Mnozenie macierzy
1010 REM
1020 FOR n=1 TO i
1030 FOR k=1 TO 2
1040 FOR m=1 TO 2
1050 nwsp(n,k)=nwsp(n,k)+wsp(n,m)*mat(m,k)
1060 NEXT m
1070 NEXT k
1080 NEXT n
1090 RETURN
1100 REM
1110 REM Rysowanie nowej figury
1120 REM
1130 GRAPHICS PEN 0:REM Nowe wspolrzedne
1140 PLOT nwsp(1,1),nwsp(1,2)
1150 FOR n=2 TO i
1160 DRAW nwsp(n,1),nwsp(n,2)
1170 NEXT
1180 RETURN
1190 REM
1200 REM Rysowanie ukladu wspolrzednych
1210 REM
1220 CLG 1
1230 GRAPHICS PEN 2:REM Pioro ukladu wspol.
1240 PLOT 0,-208
1250 DRAW 0,208
1260 PLOT -320,0
1270 DRAW 320,0
1280 RETURN
1290 REM
1300 REM Rysowanie starej figury
1310 REM
1320 PLOT wsp(1,1),wsp(1,2)
1330 FOR n=2 TO i
1340 DRAW wsp(n,1),wsp(n,2)
1350 NEXT n
1360 RETURN
1370 REM
1380 REM Koniec
1390 REM
1400 WINDOW 1,40,1,25
1410 CLS
1420 CLG 1
1430 END
```

Spectrum trwa nadal

Po Spectrum +2 na rynek wypuszczono kolejną wersję tego komputera Spectrum +3. Od poprzednika różni się przede wszystkim stacją dysku, która podobnie, jak w Amstradzie CPC 6128 jest integralną częścią klawiatury. Nowe Spectrum ma pamięć RAM 128 Kb. ROM 64 Kb. Oczywiście tak jak wszyst kie komputery posiada możliwość dołą czenia kolejnej stacji dysków, drukarki i dwóch joysticków. Tyle danych katalogowych. Innych walorów użytkowych nie znamy.

Sztuki i sztuczki (10)

Zmienne systemowe (ROM–SPECTRUM)

Krzysztof MAMCARZ

Kalkulator

MEM	23656/7
MEMBOT	23698..23727
BREG	23655

Kalkulator możemy uważać za odrębną część systemu operacyjnego wykonującą działania arytmetyczne i operacje na łańcuchach tekstowych. Jest swoistym „państwem w państwie". Posługuje się własnym „językiem" (tzw. literały— ang. literals) i korzysta z własnego stosu. Literały to 8-bitowe rozkazy wskazujące odpowiednią procedurę w ROM-ie. Umieszcza się je sekwencyjnie po instrukcji RST 40, która inicjuje pracę kalkulatora, sygnałem kończącym listę jest literał 56. Wszystkie operacje kalkulatora są przeprowadzone na 5-bajtowych reprezentacjach liczb zwanych liczbami zmiennoprzecinkowymi (ang. floating-point numbers). W dalszej części tak właśnie będzie rozumiane pojęcie liczby.

Operacje, wykonywane na liczbach nazywa się arytmetyką zmiennoprzecinkową. Aby w skuteczny sposób posługiwać się nią, nie trzeba wcale umieć przekształcać liczbę rzeczywistą na jej 5-bajtową postać. Poniższe przykłady wyjaśnią kilka podstawowych sztuczek z kalkulatorem.

Najważniejsza sprawa to kontrola stosu. Jego dno wskazuje zawsze zmienna **STKBOT** a szczyt **STKEND**. Poszczególne operacje przeprowadza się zgodnie z notacją Łukasiewicza, co w praktyce sprowadza sią do tego, że wysłanie kodu operacji powinno być poprzedzone umieszczeniem na stosie odpowiedniej ilości parametrów. Przykładowo, aby dodać dwie liczby do siebie należy wysłać na stos pierwszą, potem drugą, a na końcu literał dodawania. Można wyróżnić następujące typy operacji:

1) pojedyncze, gdy liczba z wierzchołka stosu podlega obliczeniom i jest zastępowana przez wynik np. *sin, abs, not, truncate*;

2) podwójne, gdy w obliczeniach uczestniczy para liczb z wierzchołka stosu i jest zastępowana przez wynik np. *multiply, division* lub przez dwie liczby będące wynikiem np. *modulo*;

3) manipulacyjne, gdy liczba ze szczytu stosu jest przekazywana do obszaru pamięci kalkulatora lub dorzucana na stos np. *st-mem-1, get-mem-1, stk-1*;

4) wieloczynnościowe (ang. multi-purpose) — w przypadku generowania ciągów liczbowych np. współczynników wielomianów Czebyszewa.

Pierwszym narzucającym się problemem jest przekazanie liczby z poziomu BASIC'A na stos kalkulatora. Można to zrobić przez zdefiniowanie funkcji użytkownika i odczyt **DEFADD** przez oznaczenie zmiennej (LET) i odczyt **DEST**. Ten drugi sposób sprawia nieco kłopotów, lecz zaczniemy właśnie od niego. Poniższy zestaw instrukcji asemblera GENS3M2 umieszcza na stosie kalkulatora liczbę rzeczywistą przekazaną ze zmiennej Basic'a. Literały są podawane dziesiętnie.

```
  5 ; ** DEST '87
 10                 ORG   62000
 20 DEST            EQU   23629
 30 CHADD           EQU   23645
 40 MEMBOT          EQU   23698
 50 MEM             EQU   23656
 60 SEED            EQU   23670
 70 LOOKVS          EQU   #28B2
 80                 LD    HL,(CHADD)
 90                 LD    (SEED),HL
100                 LD    HL,(DEST)
110                 BIT   1,(IY+55)
120                 JR    NZ,NEW
130                 BIT   5,(HL)
140                 JR    Z,NONS
150                 BIT   6,(HL)
160                 JR    NZ,OLD
170 NEW             LD    (CHADD),HL
180                 CALL  LOOKVS
190                 JR    C,VARNOT
200 OLD             INC   HL
210                 LD    (MEM),HL
240                 RST   40
250 getmem          DEFB  224
260 end             DEFB  56
270                 LD    HL,MEMBOT
280                 LD    (MEM),HL
290                 POP   BC
300                 LD    HL,(SEED)
310                 LD    (CHADD),HL
320                 RET
330 NONS            RST   8
340                 DEFB  11
350 VARNOT          RST   8
360                 DEFB  1
```

Po wczytaniu instrukcji np.:
LET liczba = 9.81: **PRINT USR** 62000
nastąpi uruchomienie procedury, która wykona następujące czynności:

linie 80..90	— przechowanie adresu interpretowanego wyrażenia w **SEED**;
linie 100..110	— odczyt **DEST** i flagi charakteryzującej wczytaną po **LET** zmienną;
linie 120..160	— test typu zmiennej BASIC'a, jeśli już istnieje;
linie 170..190	— odszukanie adresu zmiennej w pamięci, jeśli jest nowa lub wieloliterowa oznaczona wcześniej;
linia 210	— przełączenie obszaru pamięci kalkulatora na początek liczby
linie 240..260	— wejście do kalkulatora i przekazanie liczby z pamięci na stos;
linie 270..280	— odtworzenie oryginalnej wartości zmiennej MEM;
linie 290..320	— odtworzenie adresu przerwanej interpretacji i powrót do BASIC'a, jeśli to potrzebne;

Zmienna systemowa **MEM**, która odgrywa tu główną rolę wskazuje obszar pamięci kalkulatora. Zajmuje on 30 bajtów od **MEMBOT** w górę, co oznacza, że mieści 6 liczb. W pamięci mogą być składane liczby ze stosu, które będą potrzebne w innym momencie. Operacje tego typu wykonują literały *st-mem* i *get-mem* zaopatrzone w numer sektora. Sektorów jest 6 (gdyż tyle może zmieścić pamięć) ponumerowanych od 0 do 5.

Korzystając z tego, że wśród literałów znajduje się procedura modulo, spróbujmy zaimplementować funkcję m-modulo-n. Określmy zatem:

10 **DEF FN** m (m,n) = **USR** 64000

Chcąc obliczyć wartość 5 mod 2 należy przykładowo umieścić linie:

20 **LET** mod=**FN** m (5,2)
30 **PRINT** mod

Tekst źródłowy asemblera wygląda następująco:

```
10 ; ** MODULO '87
20 ;GENS3M2 assembler
30                ORG   64000
40 DEFADD         EQU   23563
50 STK_NR         EQU   #33B4
60                LD    HL,(DEFADD)
70                INC   HL
80                INC   HL
90                CALL  STK_NR
100               CALL  TEST
110               LD    HL,(DEFADD)
120               LD    BC,10
130               ADD   HL,BC
140               CALL  STK_NR
150               CALL  TEST
160               RST   40
170 mod           DEFB  50
180 delete        DEFB  2
190 end           DEFB  56
200               POP   HL
210               RET
220 TEST          RST   40
230 duplic        DEFB  49
240 less0         DEFB  54
250 jptrue        DEFB  0
260               DEFB  8 ;do 340
270 duplic        DEFB  49
280 duplic        DEFB  49
290 trunc         DEFB  58
300 subtr         DEFB  3
310 not           DEFB  48
320 jptrue        DEFB  0
330               DEFB  4 ;do 370
340 end           DEFB  56
350               RST   8
360               DEFB  9 ;inv. arg.
370 end           DEFB  56
380               RET
```

Opis:

linie 60..90 — przekazanie parametru m na stos kalkulatora (adres 5-bajtowej liczby wskazuje para HL);

linia 100 — sprawdzenie bezpośrednio na stosie, czy liczba jest dodatnia i całkowita;

linie 110..150 — przekazanie parametru *n* na stos;

linie 160..190 — operacje na stosie; mod odkłada dwie liczby, z których pierwsza to m-mod-n, a druga wykazuje ile razy n mieści się w m (stąd operacja delete);

linie 200..210 — powrót do BASIC'a z pominięciem procedury umieszczającej na stosie zawartość rejestrów **BC**;

Skompilowany kod nie jest relokowalny w przeciwieństwie do poprzedniego. Etykiety składające się z małych liter są rozkazami kalkulatora i spełniają rolę poglądową. Nie trzeba ich wpisywać.

Kolejny przykład również wymaga określenia funkcji użytkownika

13 **DEF FN** s (a) = **USR** 60000

Rzecz dotyczy procedury obliczającej silnię liczby całkowitej nieujemnej. Po instrukcji np.

PRINT FN s (9)

nastąpi umieszczenie parametru na stosie kalkulatora (linie 60..90) oraz obliczenia (linie 100..270).

```
10 ; ** SILNIA '86
20 ;GENS3M2 assembler
30                ORG   60000
40 DEFADD         EQU   23563
50 STK_NR         EQU   #33B4
60                LD    HL,(DEFADD)
70                INC   HL
80                INC   HL
90                CALL  STK_NR
100               RST   40
110 duplic        DEFB  49
120 not           DEFB  48
130 jptrue        DEFB  0
140               DEFB  13 ;do 270
150 duplic        DEFB  49
160 stk1          DEFB  161
170 subtr         DEFB  3
180 stmem0        DEFB  192
190 not           DEFB  48
200 jptrue        DEFB  0
210               DEFB  11 ;do 300
220 gtmem0        DEFB  224
230 multip        DEFB  4
240 gtmem0        DEFB  224
250 jump          DEFB  51
260               DEFB  -10 ;do 150
270 end           DEFB  56
280               LD    BC,1
290               RET
300 end           DEFB  56
310               POP   BC
320               RET
```

Na początku parametr jest sprawdzany. Jeśli jest zerem, następuje powrót do BASIC'a z jedynką na stosie. Jeśli nie jest zerem wykonuje się pętla w liniach 150..260. Na dnie stosu zawsze znajduje się wynik kolejnych operacji mnożenia go przez parametr pomniejszony o jeden. W chwili, gdy doszłoby do mnożenia przez zero procedura kończy pracę (linie 300..320). Ponieważ możliwości Spectrum w zakresie przechowywania liczb są ograniczone, więc najwyższa liczba, której silnię można poznać wynosi 33. Poruszając się tylko w języku wewnętrznym również można dokonywać operacji na liczbach rzeczywistych i to w sposób całkiem łatwy. Oto przykład podniesienia stałej *e* do potęgi *pi*:

```
10 ; ** E^PI '87
20                ORG   61200
30 CHADD          EQU   23645
40 DEC_FP         EQU   #2CB8
50                LD    HL,E_CON
60                LD    (CHADD),HL
70                LD    A,(HL)
80                CALL  DEC_FP
90                LD    HL,PI_CON
100               LD    (CHADD),HL
```

```
110                 LD    A,(HL)
120                 CALL  DEC_FP
130                 RST   40
140                 DEFB  6
150                 DEFB  56
160                 POP   HL
170                 RET
180 PI_CON          DEFM  "3.14159265"
190                 DEFB  13
200 E_CON           DEFM  "2.71828183"
210                 DEFB  13
```

Przekazanie obu stałych na stos odbywa się na zasadzie interpretacji kolejnych cyfr umieszczonych w postaci znaków ASCII. Listę znaków kończy bajt 13 lub inny, różny od kodów cyfr, kropki lub litery *e*. Część interpretacyjną wykonuje procedura ROM umieszczając jednocześnie liczby na stosie. Po wejściu do kalkulatora wykonywana jest tylko operacja potęgowania (linia 140). Chcąc sprawdzić wynik należy wpisać:

PRINT USR 61200

Zmienna systemowa **BREG** pełni dość ciekawą rolę w działaniu kalkulatora. Analogia do rejestru **B** mikroprocesora nie jest przypadkowa, gdyż **BREG** wspólnie z literałem *dec-jr-nz* stanowi symulację pętli. Wejście do kalkulatora w czasie pętlowania powinno odbywać się przez **CALL** 3362 (hex.), **a nie RST 40**.

Na zakończenie wykaz użytych w programach rozkazów kalkulatora:

literał	nazwa	znaczenie
224	*get-mem-0*	przesyła na stos kalkulatora liczbę z obszaru pomięci (sektor 0);
50	*modulo*	oblicza m-moduło-n;
2	*delete*	obniża wierzchołek stosu o 5 bajtów powodując pozbycie się liczby;
56	*end-calc*	wskaźnik końca listy literałów; wyjście z kalkulatora;
49	*duplicate*	zdublowanie liczby na szczycie stosu;
54	*less 0*	test znaku liczby; wynikiem operacji jest wartość logiczna 1, jeśli liczba jest ujemna;
0	*jptrue*	skok, jeśli liczba na stosie nie jest zerem (fałszem); po literale umieszcza się adres względny, od którego mają być kontynuowane operacje kalkulatora;
58	*truncate*	powoduje pozbawienie liczby jej części ułamkowej (tzw. zaokrąglenie w kierunku zera);
3	*subtract*	odejmowanie dwu liczb na stosie;
48	*not*	wynikiem tej operacji jest 0, jeśli liczba na stosie nie jest zerem lub 1, jeśli jest;
161	*stk-1*	umieszczenie na stosie liczby 1;
192	*stk-mem-0*	odłożenie do pamięci (sektor 0) liczby ze szczytu stosu (stos nie ulega obniżeniu);
4	*multiply*	mnożenie dwu liczb na stosie;
51	*jump*	skok bezwarunkowy do adresu względnego podanego za literałem;
6	*to-power*	podniesienie liczby pod wierzchołkiem stosu do potęgi liczby znajdującej się na szczycie stosu.

Oczywiście podane przykłady, ani wyjaśnienia nie wyczerpują wszystkich możliwości kalkulatora, natomiast kończą listę opisywanych zmiennych systemowych ROM, które były pretekstem do zagłębienia się we wnętrze Spectrum...

Krzysztof MAMCARZ

PROGRAM 23

ATARI

Migający kursor w mikrokomputerze

Przedstawiony program uruchamia pulsowanie kursora w mikrokomputerze ATARI 800 XL. Dodatkowo pulsują również napisy wykonane w technice INVERSE VIDEO, co może być przydatne w innych programach w celu zwrócenia uwagi na pewne fragmenty tekstu.

Program po wpisaniu i uruchomieniu można wykasować instrukcją NEW. W pamięci mikrokomputera pozostanie wtedy procedura w języku maszynowym i program nadal będzie działał. W przypadku użycia klawisza RESET program można ponownie wywołać instrukcją:
I = USR (30 000).

Drugi z przedstawionych programów jest napisany w asemblerze i analizując go można łatwo zrozumieć, co powoduje miganie kursora.

Tomasz WOJTCZAK

```
DM 10 REM ... MIGAJACY KURSOR ...
BI 15 REM
RS 20 FOR A=30000 TO 30036
YH 25 READ D
UU 30 POKE A,D
GH 35 NEXT A
BP 40 I=USR(30000):REM UAKTYWNIENIE
HM 45 REM PROCEDURY W JEZYKU MASZYN
   OWYM
RT 90 DATA 104,169,59,141,40,2,169,
   117
UN 91 DATA 141,41,2,169,18,141,26,2
EC 92 DATA 173,243,2,201,2,240,7,23
   8
BK 93 DATA 243,2,238,243,2,96,206,2
   43
AP 94 DATA 2,206,243,2,96
BU 97 REM
AT 98 REM ... napisane w HACKER-STU
   DIO,
XV 99 REM .   a dziala ?
```

Grafika C-64 (cz. IV)

Sławomir WASIELEWSKI

Opisane zostały już podstawowe tryby pracy mikrokomputera Commodore 64: znakowy w części II („IKS" nr 3, 1987) i graficzny w części III („IKS" nr 6, 1987). SPRITE, duszek, MOB, blokowy obiekt ruchomy są to nazwy używane dla określenia kolejnego trybu pracy C-64. W trybie tym mamy możliwość niezależnego definiowania i prostego sterowania ośmioma obrazkami graficznymi. Możemy łączyć go z dwoma pozostałymi trybami pracy. Cecha ta staje się bardzo ważna przy programowaniu specjalnych efektów graficznych (szczególnie w grach). Blokowy obiekt ruchomy należy traktować, jako pewnego rodzaju narzędzie graficzne wspomagające programistę.

SPRITE

SPRITE jest obiektem graficznym definiowalnym na powierzchni o wymiarach 24 x 21 punktów świetlnych ekranu. Nie wgłębiając się na razie w szczegóły wykonajmy następujące instrukcje (w trybie bezpośredniej pracy C-64) i zaobserwujmy ich efekt na ekranie:

POKE 53248, 150: POKE 53249, 150: POKE 53269, 1

Pokazał się nam obrazek o dziwacznych kształtach. Jest to jeden z ośmiu dostępnych ruchomych obiektów nie posiadający na razie sensownej postaci. Zmieniając wartości bajtów 53248 i 53249 możemy go przemieszczać po ekranie. Wykonanie instrukcji

POKE 53269, 0

powoduje zniknięcie obiektu.

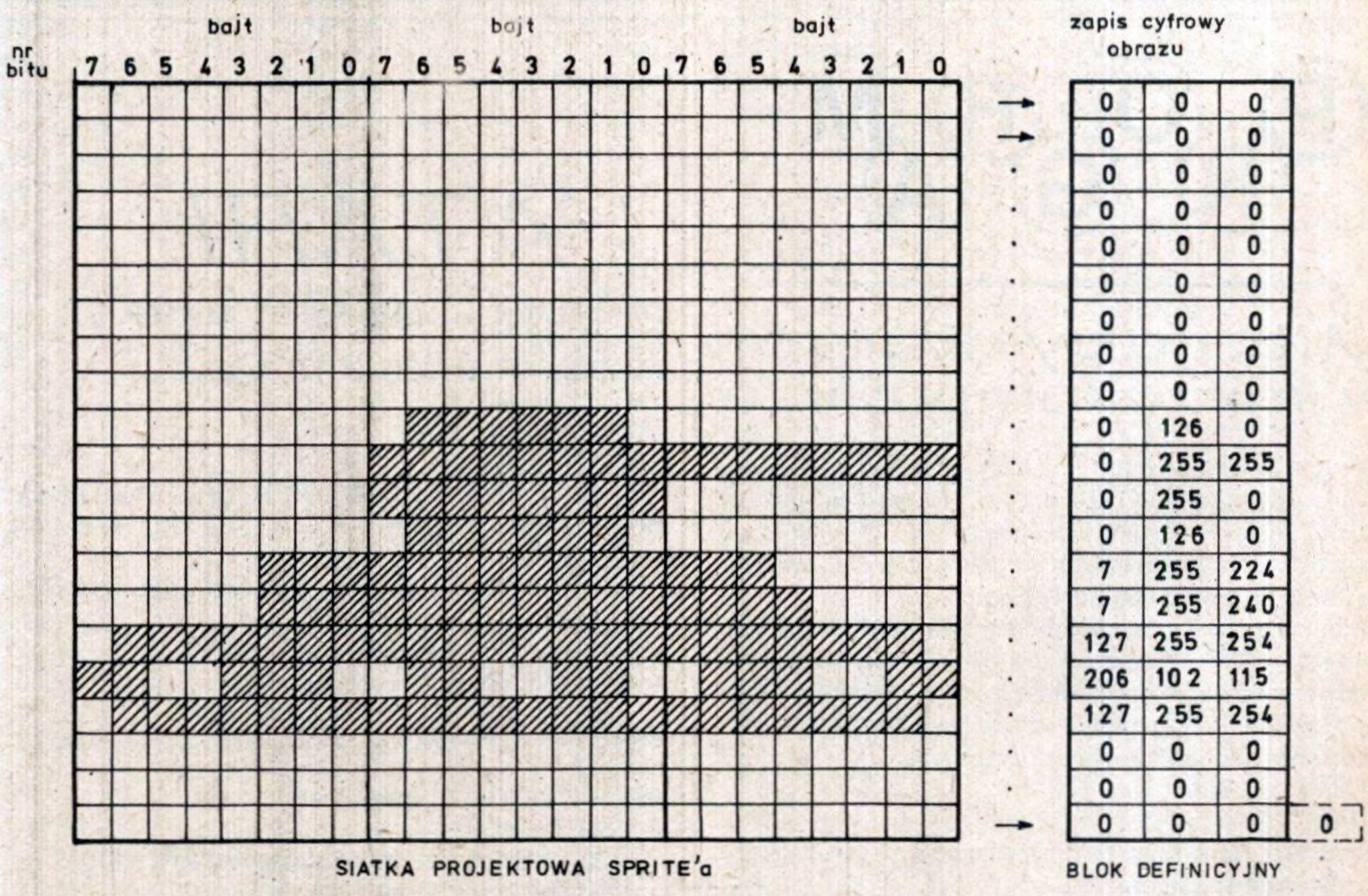
Do konstrukcji obrazu SPRITE'a na ekranie VIC wykorzystuje: blok definicyjny wskazany przez wskaźnik bloku oraz rejestry sterujące.

Blok definicyjny jest zapisanym w pamięci mikrokomputera obrazem SPRITE'a. Sposób tworzenia bloku definicyjnego ilustruje rys. 1.

Na siatce odpowiadającej jednemu obiektowi wrysowany został czołg. Wyodrębnianie kolejnych bajtów pamięci i wyznaczanie ich wartości odbywa się na ogólnych zasadach opisanych w części II. Blok definicyjny powinien składać się z 63 bajtów. Przyjmuje się jednak wielkość 64 bajtów, przy czym ostatni bajt nie jest brany pod uwagę przez VIC-a. Zapiszmy utworzony blok definicyjny w programie za pomocą instrukcji DATA:

```
1 DATA 0, 0, 0, 0, 0, 0
2 DATA 0, 0, 0, 0, 0, 0
3 DATA 0, 0, 0, 0, 0, 0
4 DATA 0, 0, 0, 0, 0, 0
5 DATA 0, 0, 0, 0, 126, 0
6 DATA 0,255, 255, 0, 255, 0
7 DATA 0, 126, 0, 7, 255, 224
8 DATA 7, 255, 240, 127, 255, 254
9 DATA 206, 102, 115, 127, 255, 254
10 DATA 0, 0, 0, 0, 0, 0
11 DATA 0, 0, 0, 0
```

Tworzenie bloku definicyjnego jest najbardziej żmudnym zajęciem podczas projektowania SPRITE'a. **Tym, których zniechęca ręczne przeliczanie i wprowadzanie do pamięci mikrokomputera wymaganych danych polecam wykorzystanie do tego celu programu użytkowego „PROJEKTANT-SPRITES" zamieszczonego w „IKS-ie" nr 3, 1986.**



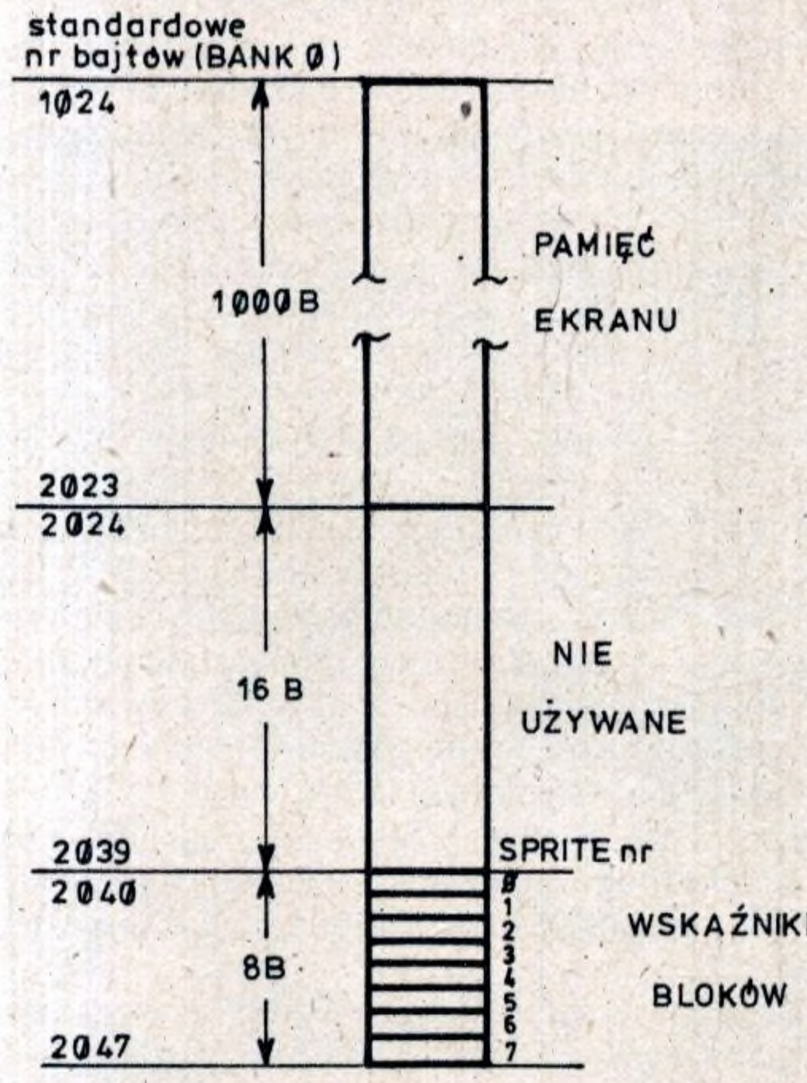
Rys. 1 Tworzenie bloku definicyjnego SPRITE'a.

VIC w każdym BANKU pozwala na 256 (256 x 64 =16 384) różnych lokalizacji bloków definicyjnych (od Ø do 255). Umieśćmy nasz blok definicyjny jako ostatni w BANKU Ø:

```
2Ø FOR I=Ø TO 63: READ A: POKE 255*64+I, A: NEXT I
```

W przypadku pracy VIC-a w BANKU innym niż zerowy musimy do adresu początkowego wybranego bloku (255*64) dodać adres początkowy użytego BANKU.

Jak poinformować VIC-a, w którym bloku umieściliśmy postać SPRITE'a? Do tego celu służą wskaźniki bloków definicyjnych. W każdym BANKU jest ich ósiem, gdyż można wyświetlać jednocześnie do ośmiu ruchomych obiektów. Lokalizacja wskaźników zależy od umiejscowienia PAMIĘCI EKRANU wewnątrz BANKU. Zajmują one bowiem ostatnie osiem bajtów w 1K obszarze wyznaczonym adresem początkowym PAMIĘCI EKRANU (rys. 2).



Rys. 2 Lokalizacja wskaźników bloków definicyjnych w pamięci mikrokomputera

Ponieważ nie zmienialiśmy standardowego położenia PAMIĘCI EKRANU, w naszym przypadku wskaźniki te mieszczą się między bajtami 2Ø4Ø — 2Ø47. Wprowadzenie do odpowiedniego wskaźnika liczby z zakresu Ø — 255 wskazuje VIC-owi, który blok definicyjny ma być interpretowany. Teraz już wiemy dlaczego uzyskany początkowo na ekranie SPRITE miał nieokreślony kształt. Wykonajmy ponownie te same trzy instrukcje POKE oraz dodatkowo

POKE 2Ø4Ø, 255

SPRITE nadal ma tę samą postać. Wprowadźmy do 255 bloku definicyjnego zaprojektowaną postać obiektu ruchomego komendą RUN. Na ekranie uzyskaliśmy czołg.

W celu przemieszczania obiektów ruchomych po ekranie i uzyskiwania pewnych ich cech służą niektóre z rejestrów sterujących VIC-a. Wykaz interesujących nas rejestrów wraz z krótkim opisem realizowanych funkcji zamieszczony jest w tabeli.

bajt pamięci	realizowana funkcja
53248	SPRITE Ø współrzędna X
53249	SPRITE Ø współrzędna Y
5325Ø	SPRITE 1 współrzędna X
53251	SPRITE 1 — " — Y
53252	SPRITE 2 — " — X
53253	— " — — " — Y
53254	SPRITE 3 — " — X
53255	— " — — " — Y
53256	SPRITE 4 — " — X
53257	— " — — " — Y
53258	SPRITE 5 — " — X
53259	— " — — " — Y
5326Ø	SPRITE 6 — " — X
53261	— " — — " — Y
53262	SPRITE 7 — " — X
53263	— " — — " — Y
53264	znacznik pozycji X
53269	aktywność SPRITE'a (1 — aktywny)
53271	powiększenie w pionie (na wysokość)
53275	priorytet SPRITE — podłoże (Ø — SPRITE ma wyższy)
53276	wprowadzenie trybu wielobarwnego (1 — wprowadzony)
53277	powiększenie w poziomie (na szerokość)
53278	sygnalizacja kolizji SPRITE — SPRITE
53279	— " — — " — SPRITE — podłoże
53281	kolor punktu Ø (ekran, SPRITE) w t. dwukolorowym; kolor punktu ØØ w t. wielobarwnym
53285	kolor punktu Ø1 SPRITE w t. wielobarwnym
53286	kolor punktu 11 SPRITE — " —
53287	SPRITE Ø: kolor punktu 1 w t. dwukolorowym lub kolor punktu 1Ø w t. wielobarwnym
53288	SPRITE 1: — " —
53289	SPRITE 2: — " —
53290	SPRITE 3: — " —
53291	SPRITE 4: — " —
53292	SPRITE 5: — " —
53293	SPRITE 6: — " —
53294	SPRITE 7: — " —

TABELA:
Wykaz rejestrów sterujących obiektami ruchomymi.

Niektóre z wymienionych rejestrów określają jakąś cechę SPRITE'a zawartością pojedynczego bitu odpowiadającego jego numerowi. Bit nr Ø jest odpowiedzialny za zaistnienie tej cechy u SPRITE'a nr Ø, bit nr 1 u SPRITE'a nr 1, itd. aż do ostatniego bitu (nr 7). Ogólne postacie instrukcji zmieniających zawartość pojedynczych bitów w rejestrze są następujące:

— ustawienie bitu na 1

POKE rejestr, **PEEK** (rejestr) **OR** 2↑bit

— ustawienie bitu na Ø

POKE rejestr, **PEEK** (rejestr) **AND** (255-2↑bit)

gdzie: rejestr — adres rejestru (bajtu pamięci),
bit — numer bitu odpowiadający wybranemu obiektowi.

Zapoznamy się teraz bliżej z wymienionymi w tabeli funkcjami. Pozwolą one nam w sposób efektywny wykorzystać możliwości VIC-a do programowania obiektów ruchomych.

AKTYWNOŚĆ

Każdy z ośmiu obiektów ruchomych może być niezależnie wprowadzany w stan aktywności. Jest to stan, w którym VIC interpretuje wszystkie niezbędne informacje doty-

➡

czące danego SPRITE'a i na ich podstawie wyświetla jego obraz na ekranie.

Do określenia stanu aktywności poszczególnych obiektów służy rejestr sterujący VIC-a umieszczony pod adresem 53269. Ustawienie jakiegoś bitu na „1" w tym rejestrze oznacza stan aktywny odpowiadającego mu obiektu. Jeśli chcemy, na przykład uaktywnić tylko obiekty numer Ø i 1 musimy wykonać instrukcję:

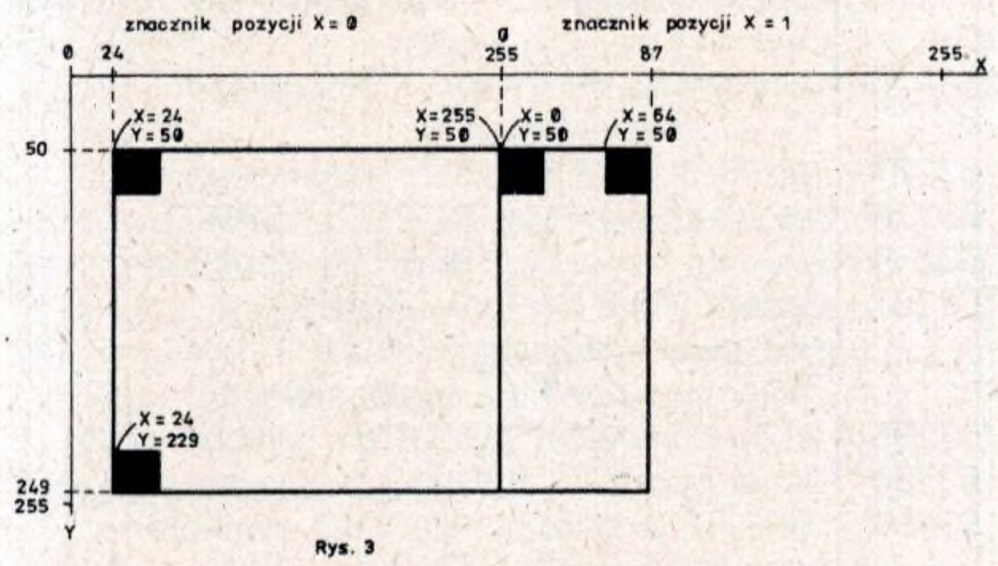
POKE 53269, **PEEK** (53269) **OR** 3

SPRITE może być widoczny na ekranie lub nie, zależnie od stanu pozostałych odpowiadających mu rejestrów VIC-a. Powrót do stanu nieaktywnego uzyskujemy zerując odpowiednie bity. W naszym przypadku:

POKE 53269, **PEEK** (53269) **AND** 252

POZYCJONOWANIE

Do ustalenia pozycji ruchomych obiektów na ekranie służy grupa rejestrów o adresach od 53248 do 53264. Pozycja obiektu wyznaczana jest na podstawie wartości współrzędnej X na osi poziomej i współrzędnej Y na osi pionowej (rys. 3).



Rys. 3

Rys. 3 Pozycjonowanie SPRITE'a na ekranie.

Współrzędne X i Y ustalają pozycję lewego górnego punktu SPRITE'a. Na rys. 1 jest to lewy górny kwadrat siatki projektowej. Każdy SPRITE ma przydzieloną parę rejestrów do wprowadzania wartości obu współrzędnych. Ponieważ największa liczba jaką można zapisać w jednym bajcie wynosi 255, do pozycjonowania na osi poziomej (X) wykorzystuje się dla każdego obiektu dodatkowy jeden bit w rejestrze o adresie 53264. Bit ten nazywany jest znacznikiem pozycji X. Ustawienie znacznika na Ø oznacza, że wartość współrzędnej X dotyczy. lewego odcinka osi poziomej. Znacznik równy 1 określa prawy odcinek osi poziomej X.

Wykonajmy następujący przykład w trybie bezpośrednim:

— utwórzmy blok definicyjny obiektu w kształcie prostokąta

FOR I=Ø **TO** 63: **POKE** 254*64+I, 255: **NEXT** I

— postać tę nadamy obiektowi nr Ø

POKE 2Ø4Ø, 254: **POKE** 53269, 1

— wyświetlimy prostokąt w lewym górnym rogu ekranu

POKE 53248, 24: **POKE** 53249, 5Ø: **POKE** 53264, Ø

Zmieniając edytorem wartość ostatnich trzech rejestrów (współrzędne X, Y oraz znacznik pozycji X) możemy zbadać położenie SPRITE'a w różnych punktach ekranu.

Wiedząc już jak zmieniać położenie obiektu spróbujmy uruchomić nasz czołg w realizowanym programie i przejechać nim po ekranie. Wprowadźmy niezbędne instrukcje i wykonajmy komendę RUN.

```
3Ø POKE 2Ø4Ø, 255: POKE 53269, 1: POKE 53249, 1ØØ
4Ø POKE 53264, Ø: FOR X=Ø TO 255: POKE 53248, X: NEXT X
5Ø POKE 53264, 1: FOR X=1 TO 88: POKE 53248, X: NEXT X
```

POWIĘKSZANIE

Normalnej wielkości SPRITE ma 24 punkty szerokości i 21 punktów wysokości. Istnieje możliwość dwukrotnego powiększenia SPRITE'a. Powiększenie na szerokość uzyskujemy ustawiając odpowiedni bit na 1 w rejestrze o adresie 53277, a na wysokość w rejestrze 53271. Powrót do normalnych wymiarów uzyskamy po wyzerowaniu odpowiednich bitów.

Ostatnio używaliśmy czołgu jako obiektu ruchomego o nr. Ø. Wyświetlmy więc go ponownie w centralnej części ekranu:

POKE 53248, 15Ø: **POKE** 53249, 15Ø: **POKE** 53264,Ø

a teraz powiększymy go na szerokość

POKE 53277,1

i na wysokość

POKE 53271,1

Powracamy do normalnych wymiarów:

POKE 53277,Ø: **POKE** 53271,Ø

PRIORYTET

Przy użyciu większej liczby obiektów ruchomych mogą zaistnieć sytuacje, gdy obiekty będą nachodzić na siebie. Należy zatem wcześniej zadbać o to, który SPRITE ma być widziany „przed" innym lub „za" innym. Do tego celu wykorzystuje się priorytety przyporządkowane na stałe wszystkim obiektom ruchomym. SPRITE Ø ma największy priorytet i jest widziany zawsze „przed" wszystkimi pozostałymi. SPRITE 7 ma najmniejszy priorytet i jest widziany „za" wszystkimi pozostałymi. Inaczej mówiąc: SPRITE o numerze niższym ma zawsze wyższy priorytet.

W trakcie łączenia różnych trybów graficznych należy ustalić priorytet każdego używanego SPRITE'a względem pozostałych elementów graficznych (podłoża). Do ustalenia tego priorytetu służy rejestr o adresie 53275. Wartość zerowa bitu odpowiadającego wykorzystywanemu obiektowi oznacza, że ma on wyższy priorytet względem podłoża. Na ekranie widoczny będzie SPRITE, a za nim schowane będą występujące w tym samym miejscu elementy graficzne zrealizowane w innym trybie pracy VIC-a.

Należy zaznaczyć, że przesłaniane są tylko szczegóły obrazu graficznego, które w reprezentacji bitowej są jedynkami (np. zaczernione kwadraty w siatce projektowej na rys. 1).

Zmieniając priorytet czołgu (SPRITE Ø) względem podłoża i przesuwając kursor w miejsce, gdzie znajduje się czołg na ekranie możemy zaobserwować opisane efekty. Zmianę priorytetu uzyskamy instrukcjami:

— czołg ma niższy priorytet

POKE 53275,1

— czołg ma wyższy priorytet

POKE 53275,Ø

KOLIZJA

Sytuacja kolizji występuje, gdy nastąpi zetknięcie się kilku niezerowych (w reprezentacji bitowej) elementów graficznych. Do kolizji może dojść między obiektami ruchomymi oraz między podłożem a obiektem ruchomym. VIC sam wykrywa sytuacje kolizji i informuje nas o tym za pomocą rejestrów o adresach 53278 i 53279. Rejestr 53278 sygnalizuje wystąpienie kolizji SPRITE — SPRITE ustawiając na 1 bity odpowiadające obiektom ruchomym, między którymi wystąpiła kolizja. Rejestr 53279 sygnalizuje wystąpienie kolizji typu SPRITE — podłoże ustawiając na 1 bity odpowiadające obiektom, które zetknęły się z jakimkolwiek elementem graficznym podłoża.

Sygnały o sytuacjach kolizyjnych są dostarczane ciągle przez VIC-a. Zerowanie rejestrów kolizji następuje automatycznie po odczytaniu ich zawartości instrukcją PEEK. Należy pamiętać o tym, że SPRITE nieaktywny może również powodować kolizje.

KOLORY

Podobnie, jak i w pozostałych trybach pracy VIC-a, można w obiektach ruchomych wykorzystywać grafikę dwukolorową i wielobarwną. Za pomocą rejestru o adresie 53276 możemy wybrać rodzaj grafiki dla każdego obiektu oddzielnie. Wartość zerowa bitu oznacza grafikę dwukolorową. Ustawienie na 1 jakiegoś bitu wprowadza odpowiadający mu SPRITE w grafikę wielobarwną.

W grafice dwukolorowej VIC pobiera kod kolorów (Ø — 15) każdego z obiektów ruchomych z dwóch rejestrów. Rejestr odpowiedzialny za kolor podłoża SPRITE'a (zera w reprezentacji bitowej, puste kwadraty w siatce projektowej) jest ten sam dla wszystkich obiektów i mieści się pod adresem 53281. Rejestr ten równocześnie określa kolor punktów o zerowej reprezentacji bitowej na ekranie. Kolor grafiki SPRITE'a (jedynki w reprezentacji bitowej) jest określany niezależnie dla każdego z nich za pomocą odpowiedniego rejestru od 53287 do 53294.

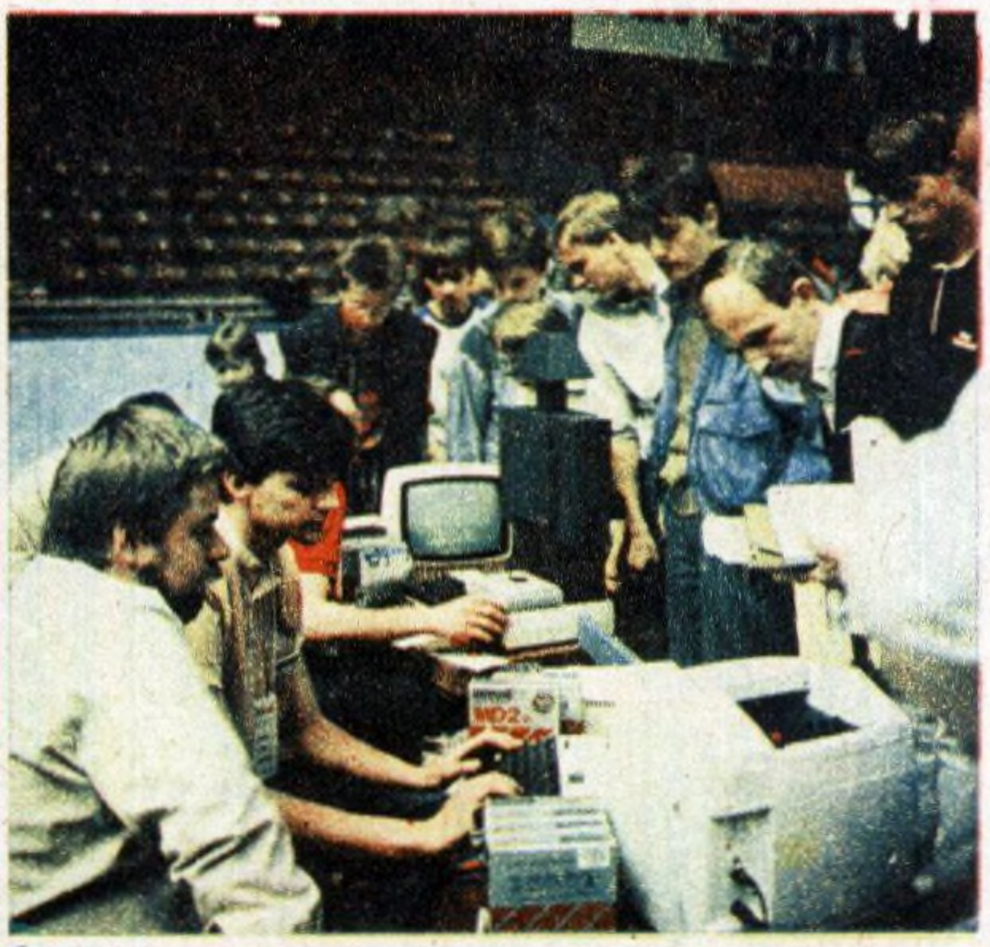
W grafice wielobarwnej, podobnie jak w trybie tekstowym czy trybie wysokiej rozdzielczości jeden punkt w obrazie SPRITE'a na ekranie odpowiada dwóm bitom pamięci. Kolory są kodowane w następujący sposób:

punkt	rejestr zawierający kod koloru	
ØØ	53281	} wspólne dla wszystkich obiektów ruchomych
Ø1	53285	
11	53286	
1Ø	rejestr koloru odpowiedniego SPRITE'a (53287 — 53294)	

I to już wszystko. Przekazany zakres wiedzy na temat grafiki C-64 powinien wystarczyć aby samodzielnie pisać programy graficzne. Należy jednak zdawać sobie sprawę z tego, że uzyskanie wyrafinowanych efektów graficznych wiąże się z wykorzystaniem dodatkowych technik programowania mikrokomputerów (np. technika przerwań), a w szczególności programowania w języku wewnętrznym mikrokomputera Commodore 64.

Sławomir WASIELEWSKI

ATAROWISKO



(Foto: Jan ZELMAN)

PROGRAM 24

ZX SPECTRUM

BOMBA

Hasło *„zagrajmy w bombę"* kierujemy do adeptów języka LOGO, którzy w wolnej chwili chcieliby zostać bohaterami. Otrzymujecie informację (oczywiście pewną), że w jednym z 357 911 pokoi wieżowca podłożono bombę. Należy ją rozbroić. Zabieracie specjalny czujnik i wchodzicie do budynku w miejscu o umownych współrzędnych 0,0,0. Budynek ma kształt sześcianu. Pokój na drugim końcu przekątnej ma współrzędne 70,70,70.

Czujnik natychmiast podaje pewien sygnał, który należy wykorzystać. Jedyną wskazówką, jaką można podać jest zakres sygnału wynoszący 0...1000. Im większy wskaźnik, tym bliżej bomby. Macie 15 szans na wybranie takich trzech współrzędnych, które wskażą poszukiwany pokój. Współrzędne należy wczytywać oddzielając je spacją. Na końcu wprowadzania naciskamy ENTER.

Gra nie wymaga intensywnego angażowania wielu szarych komórek pod warunkiem, że szybko znajdziecie trochę reguł.

Krzysztof MAMCARZ

```
?
TO REKORD
IF :numer < ITEM 2 :best [type :
imie type char 32 pr :numer ( ma
ke "best list :imie :numer )] [t
ype ( item 1 :best ) type char 3
2 pr ( item 2 :best )]
END

TO WPROW
SETCUR SE 22 :line
MAKE "dane RL
MAKE "ilpar COUNT :dane
IF :ilpar = 3 [] [wprow]
MAKE "wys FIRST :dane
IF :wys < 0 [pr " pr [wpadles do
 piwnicy!!!] TOPLEVEL]
IF :wys > 70 [pr " pr [spadles z
 dachu!!!] TOPLEVEL]
MAKE "sz ITEM 2 :dane
IF OR :sz < 0 :sz > 70 [pr " pr
[wypadles przez okno!!!] TOPLEVE
L]
MAKE "dl LAST :dane
IF OR :dl < 0 :dl > 70 [pr " pr
[wypadles przez okno!!!] TOPLEVE
L]
MAKE "czujnik ( SUM :wys / 10 :s
z :dl * 10 ) - ( SUM :a / 10 :b
:c * 10 )
IF :czujnik < 0 [make "czujnik -
 :czujnik]
MAKE "syg 1000 - :czujnik
IF :syg = 1000 [pr " pr [BRAVO!
BOMBA ROZBROJONA!] REKORD TOPLEV
EL]
IF :numer = 15 [pr " pr [NIESTET
Y! BOMBA WYBUCHA!] TOPLEVEL]
MAKE "line :line + 1
END

TO WYDRUK
SETCUR SE 0 :line
TYPE :numer
SETCUR SE 4 :line
TYPE :wys
SETCUR SE 8 :line
TYPE :sz
SETCUR SE 12 :line
TYPE :dl
SETCUR SE 16 :line
TYPE :syg
MAKE "numer :numer + 1
END

TO INIT
HT TS CS SETBR 4
SETCUR [0 0]
PR [wpisz imie...]
MAKE "imie RL
MAKE "ilpar COUNT :imie
IF :ilpar = 1 [] [toplevel]
CS SETCUR [0 0]
REPEAT 32 [type char 42]
PR [nr..wys.sz..dl..sygnal]
REPEAT 32 [type char 42]
MAKE "a RANDOM 71
MAKE "b RANDOM 71
MAKE "c RANDOM 71
MAKE "wys 0
MAKE "sz 0
MAKE "dl 0
MAKE "syg 1000 - ( SUM :a / 10 :
b :c * 10 )
MAKE "numer 0
MAKE "line 3
WYDRUK
END

TO BOMBA
IF ( .EXAMINE 65535 ) = 0 [make
"best [EMIL 6] .DEPOSIT 65535 1]
INIT
REPEAT 15 [wprow wydruk]
END
```

```
**********************************
nr..wys.sz..dl..sygnal
**********************************
0    0    0    0    0    0    0
1    7    0    0    0    0    0
2    7    5    0    0    0    0
3    7    4    0    0    0    0
4    7    4    0    0    0    0
5    6    4    2    0    0    0
6    6    4    2    0    0    0

BRAVO! BOMBA ROZBROJONA!
krzysztof 6
```

GIEŁDA POMYSŁÓW

Meritum umie generować ludzki głos

Można się o tym przekonać wpisując poniższy program, który koduje w pamięci dźwięk wprowadzany z magnetofonu.

Głos do pamięci mikrokomputera wprowadza się tak jak przy wczytywaniu programów. Najlepszy efekt osiąga się stosując firmowy magnetofon dołączany do zestawu.

Najpierw należy przygotować około piętnastosekundowy fragment nagrania (ciągłej mowy, z dość wysokim poziomem zapisu), następnie połączyć magnetofon z mikrokomputerem, włączyć odtwarzanie i w chwili usłyszenia dźwięku nacisnąć klawisz „Z". Dźwięk będzie wpisywany do pamięci, a po jej wypełnieniu zostanie odtworzony. Prostota wprowadzenia dźwięku okupiona jest zniekształceniami, które powoduje interface magnetofonu. Odtwarzany dźwięk przypomina dźwięki z radiostacji krótkofalarskich. Na wydruku kodu maszynowego wyróżnione są bajty regulujące częstotliwość próbkowania sygnału wejściowego (pierwszy) i prędkość odtwarzania (dwa następne). Można zmieniać ich wartości (heksadecymalnie).

Piotr BIEDRZYCKI

```
10      FOR A=32528 TO 32622
20    READ A$
30    N=ASC( LEFT$(A$,1))-48
40    N=N-INT(N/12)*7
50    M=ASC(RIGHT$(A$,1))-48
60    M=M-INT(M/12)*7
70     B=N*16+M: POKE A,B :W=W+B
75       NEXT A
80     ' KOD MASZYNOWY PROGRAMU
90   DATA CD,17,7F,CD,48,7F,C9,C5
100  DATA D5,E5,11,FF,7E,21,00,7F
110  DATA 36,00,01,01,38,ED,B8,21
120  DATA 00,47,0E,FF,11,80,00,06
130  DATA 08,ED,78,A3,B6,07,77,16
140  DATA 0B,15,20,FD,ED,51,10,F1
150  DATA 23,7C,FE,7F,20,E9,18,23
160  DATA C5,D5,E5,21,00,47,0E,FE
170  DATA 16,00,7E,1E,08,07,ED,79
180  DATA 06,07,10,FE,ED,51,06,08
190  DATA 10,FE,1D,20,F0,23,7C,FE
200  DATA 7F,20,E7,E1,D1,C1,C9
210  IF W<>10207 THEN PRINT
     "BLAD W ZBIORZE DANYCH ":END
230   CLS : POKE 16527,127
240      PRINT@ 448,
"/Z/ - ZAPIS GLOSU Z MAGNETOFONU
/O/ - ODTWARZANIE DZWIEKU":PRINT:
PRINT"       WYBIERZ OPCJE  /Z,O/"
250  A$=INKEY$:IF A$="O" OR A$="Z"
     THEN 260 ELSE GOTO 250
260  IF A$="O" THEN POKE 16526,19
270  IF A$="Z" THEN POKE 16526,16
280  PRINT@ 782,"-";A$;"-";:A$=""
290  U=USR(0):PRINT@ 782,"- -";
400       GOTO 250
```

LIGA MYŚLĄCYCH

Zadanie 1

Zapytano rybaka, ile waży złowiona przez niego ryba. Rybak odpowiedział: $\frac{2}{5}$ kg i jeszcze 2 razy po $\frac{2}{5}$ swego ciężaru. Ile ważyła złowiona przez rybaka ryba?

Zadanie 2

Pamiętny w historii rok pewnego odkrycia wyraża się liczbą czterocyfrową. Należy znaleźć tę liczbę wiedząc, iż suma cyfr tej liczby wynosi 16, cyfra tysięcy jest cztery razy mniejsza od cyfry setek, cyfra setek jest dwa razy większa od cyfry jedności, a liczba trzycyfrowa utworzona z szukanej liczby przez skreślenie w niej cyfry jedności jest o 100 większa od liczby dwucyfrowej utworzonej z szukanej liczby przez skreślenie w niej cyfry jedności i cyfry tysięcy.

Zadanie 3

Wieśniaczka sprzedała pierwszej osobie $\frac{1}{2}$ całej ilości jajek i jeszcze 2 jajka. Drugiej osobie sprzedała połowę reszty jajek i jeszcze jedno jajko. Po drugiej sprzedaży zostało wieśniaczce jeszcze 8 jajek. Ile wszystkich jajek przyniosła wieśniaczka na targ? Ile jajek kupiła pierwsza osoba, a ile druga?

Zadanie 4

Codziennie z miejscowości A, w której mieści się urząd pocztowy, wyjeżdża listonosz do miejscowości B, C, D, by dostarczyć listy.

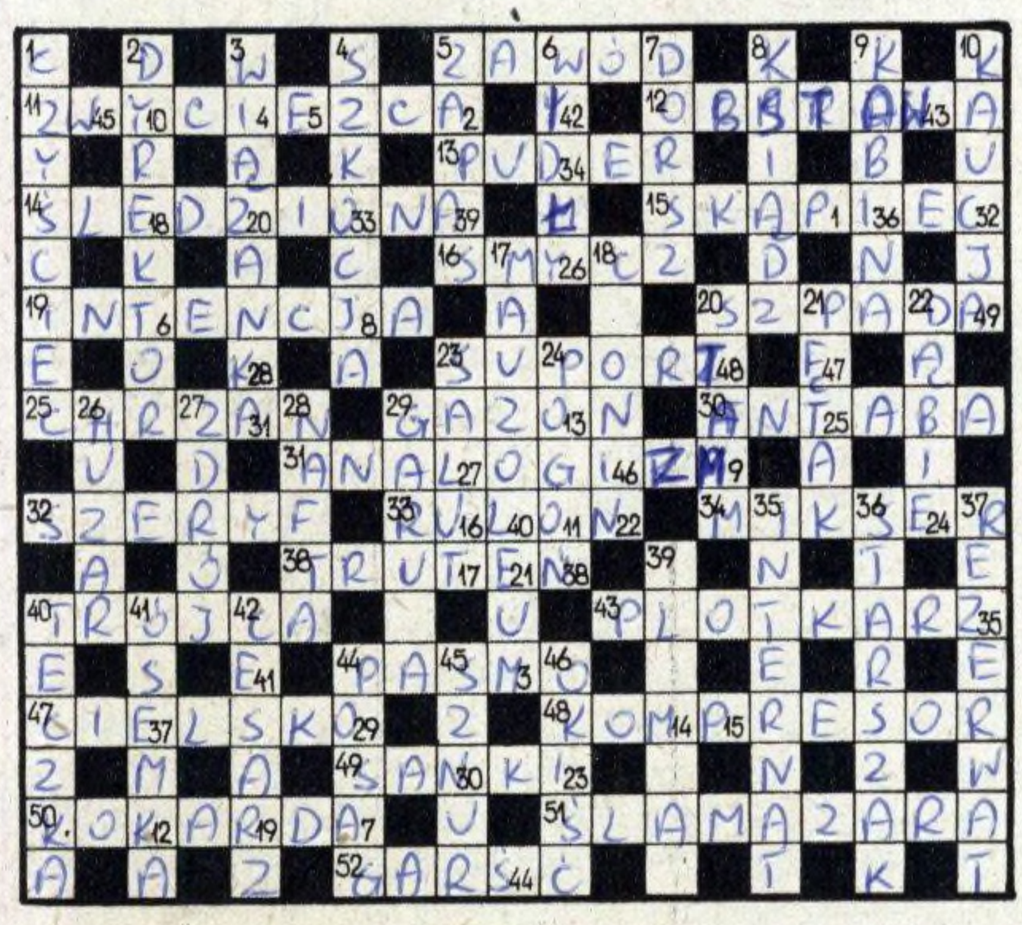
W jakiej kolejności powinien listonosz objeżdżać te miejscowości aby trasa objazdu z A przez pozostałe miejscowości i z powrotem do A była możliwie najkrótsza, jeśli długość drogi z A do B wynosi 5 km, od A do C — 7 km, od A do D — 7 km, od B do D — 8 km, od B do C — 10 km, od C do D — 6 km?

Zadanie 5

Gdy Jan zapytał Andrzeja ile ma lat, usłyszał odpowiedź: gdy ja byłem w twoim wieku byłeś ode mnie cztery razy młodszy, a gdy ty będziesz w moim wieku ja będę miał 40 lat. Ile lat ma Jan, a ile Andrzej?

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

Krzyżówka nr 8



POZIOMO: 5) profesja, 11) triumfator, 12) ludzie stanowiący ubezpieczenie, zapewniający ochronę kogoś lub czegoś, 13) środek kosmetyczny lub leczniczy, 14) wydłużony narząd kręgowców, stanowiący swego rodzaju magazyn krwi, 15) sknera, kutwa, 16) do prowadzenia psa, 19) motyw działania; zamiar, zamysł, 20) broń sportowa używana w szermierce, 23) ruchomy zespół obrabiarki (np. tokarki, strugarki), 25) składnik ćwikły, 29) ozdobny trawnik lub kwietnik, 30) metalowa sztaba umacniająca zamknięcie drzwi, bramy, 31) w logice; wnioskowanie przez analogię, 32) w USA urzędnik lokalny o uprawnieniach adm.-sądowych, 33) zwój o cylindrycznym kształcie, 34) zmechanizowane urządzenie używane przy sporządzaniu różnych potraw, deserów i napojów cocktailowych, 38) samiec pszczoły miodnej, 40) rel. grupa trzech bóstw występująca w wielu wierzeniach, 43) siewca kłamliwych pogłosek, 44) ... nieszczęść lub sukcesów, 47) opasłe, olbrzymie ciało, 48) sprężarka, 49) popularny zimowy sprzęt sportowy, 50) ozdoba warkocza, 51) ktoś powolny, pozbawiony energii, 52) dłoń złożona w sposób umożliwiający zaczerpnięcie, uchwycenie czegoś.

PIONOWO: 1) wg religii katolickiej: miejsce pozagrobowej okresowej pokuty dusz ludzi zmarłych, 2) stoi na czele instytucji, przedsiębiorstwa, 3) ślubna lub nagrobna, 4) kraj z Edynburgiem, 5) zasób, rezerwa, 6) gable, 7) pospolita ryba morska, 8) duchowny katolicki, 9) kajuta okrętowa, 10) suma pieniężna złożona jako zastaw, 17) monumentalny grobowiec w formie wolno stojącej budowli, 18) pisarz szkocko-angielski (1896 — 1981), autor „Cytadeli", 20) najwybitniejszy trener polskich pięściarzy, 21) brzdąc, smarkacz, 22) dzielnica Szczecina lub jezioro, nad którym leży to miasto, 23) uroczyste oddanie salw artyleryjskich, 24) klub sportowy ze Szczecina, 26) dawny żołnierz lekkiej jazdy, ubrany i uzbrojony na sposób węgierski, 27) źródło, 28) łatwo zapalna ciecz, produkt destylacji ropy naftowej, 29) żegl. wiatr wiejący z boku, 35) bursa, 36) dziecko z grupy starszych w przedszkolu, żłobku, 37) teren wyłączony spod eksploatacji, znajdujący się pod specjalną ochroną, 39) dawny zakład, w którym uczniowie otrzymywali darmo naukę i utrzymanie, 40) aktówka, 41) liczna osada wioślarska, 42) np. Napoleon Bonaparte, 44) wiano, 45) cienki powróz, 46) śnieg na gałęziach drzew, zwieszający się w kształcie kiści.

Po prawidłowym rozwiązaniu krzyżówki litery z kratek ponumerowanych dodatkowo w prawym dolnym rogu, czytane w kolejności odpowiadających im liczb od 1 do 49, dadzą hasło, które wystarczy nadesłać jako rozwiązanie zadania pod adresem redakcji do końca października 1987 r. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

MIKROCIEKAWOSTKI:

Rysunek z kolekcji GEM wykonany na zestawie firmy ICL
komputer — DRS 300
drukarka — Ink Yet Coluor Printer
(drukarka wykorzystuje miniaturowe działka z różnymi kolorami tuszu)

W naszym komputerlandzie

Spektruś postanowił zwiększyć w swoim zespole inicjatywę w dziedzinie racjonalizacji. Zainspirował więc do działania swojego młodego i zdolnego dzieloroba Jaśka Atari. Jaśko przeszedł się po pracowni i rozejrzał po swojemu. Ujrzał, że jeden z półautomatów wyrabia 20 części na godzinę. — To stanowczo za mało — pomyślał. Zabrał się do roboty.

Po tygodniu specjalna komisja stwierdziła, że pomysł jest dobry — dokumentacja w porządku, a korzyści finansowe udowodnione. W maszynie wymieniono mechanizm napędowy. Półautomat wyrabiał teraz 40 części na godzinę, Jaśko zaś otrzymał grubą kopertę.

Do Spektrusia przyszedł jednak brakarz, który stwierdził, że półautomat wyrabia rzeczywiście dużo, ale z jakością jest gorzej. Tylko jedna trzecia część nadaje się do użytku.

Jeszcze tego samego dnia Jaśko stanął przed półautomatem. Pomyślał i zabrał się do roboty. Po dwóch tygodniach komisja przyjęła projekt, wszyscy klaskali, a Jaśko otrzymał grubą kopertę.

Po wymianie mechanizmu napędowego półautomat wyrabiał 20 części na godzinę, brakarz nie mógł się nachwalić tych części, lecz roboty nie chciały pracować, bo urządzenie hałasować poczęło potwornie.

Racjonalizator Jaśko znowu był na posterunku. Teraz przygotowywał projekt rozwiązania z zakresu bhp. Zaproponowaną przez niego wymianę układu napędowego komisja przyjęła bez uwag. Jaśko otrzymał grubą kopertę. Półautomat wyrabiał 20 dobrych części, a co najważniejsze, można było wyjąć watę z uszu, bo pracował cicho.

Zadowolony był też Spektruś, gdy od dyrektora generalnego otrzymał grubą kopertę za wdrożenie w ostatnim czasie trzech wybitnych projektów.

Podglądał: Eugeniusz MLECZAK

Podano program...

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Janusz Janiec, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, 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 offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 9189. Nr. ind. 361682. K-65