IKS

INFORMATYKA KOMPUTERY SYSTEMY

CENA 80 zł

DODATEK „ŻOŁNIERZA WOLNOŚCI" Nr 1/1988 ISSN 0860—2794

V Krajowa Konferencja Informatyków

V Krajowa Konferencja Informatyków odbyła się 8—10 grudnia 1987 roku w Poznaniu (hotel Poznań). Zaplanowanym tematem spotkania (262 uczestników) był Rozwój Metod i Zastosowań Informatyki. Na konferencję nadesłano 108 referatów. Zostały one opublikowane w 534 stronicowym tomie. Organizatorów Konferencji wspomagali: Komitet Wspomagający oraz Rada Programowa, w sumie 48-osobowy zespół fachowców.

Tyle spraw formalnych, jeśli idzie o merytoryczne to warto podkreślić, iż posiedzenia odbywały się z udziałem wszystkich uczestników piewszego dnia (około 30—50% drugiego, trzeciego bywało różnie...).

Każdemu z posiedzeń przewodniczyło kilku Panów Profesorów lub Panów Docentów. Wygłaszali oni słowo wstępne, a następnie zachęcali do dyskusji (referaty nie były wygłaszane).

Pierwszego dnia zwrócono między innymi uwagę na rangę sieci komputerowych, które są obecnie powszechną formą wykorzystania informatyki niemal we wszystkich dziedzinach życia wysokorozwiniętych krajów. Wiąże się z tym konieczność dysponowania dużymi komputerami — superkomputerami. Tworzą one tak zwane węzły sieci. Krajowe wysiłki w tej kwestii reprezentowały wypowiedzi na temat doświadczeń i perspektyw sieci komputerowych w Polsce. Najbardziej zaawansowane są prace nad Międzyuczelnianą Siecią Komputerową (MSK) realizowana od 1985 roku oraz nad Krajową Akademicką Siecią Komputerową (KASK), która budowana jest w ramach CPBR nr 8.13. Warto w tym miejscu podkreślić, że już od kilku lat pracuje z powodzeniem u nas w kraju komputerowa sieć GABRIEL LOT-u. Obsługujący ją komputer jest w.. Nowym Jorku.

Poza kwestiami technicznymi budowy sieci, zwrócono uwagę na bardzo istotny problem zachowania standaryzacji sprzętu i oprogramowania. Podkreślono także, że w naszych warunkach, posiadanie nawet doskonałej sieci, to dopiero początek długiej, bardzo długiej drogi do jej efektywnego wykorzystania — bowiem sieć tę trzeba „wypełnić odpowiednimi informacjami" i chcieć (umieć) z nich korzystać. Jest to problem ogromny, tym bardziej że nasze doświadczenia w tym względzie są ciągle pomijalne.

Drugiego dnia konferencji zwrócono między innymi uwagę na centralne komputery — stanowiące serce (mózg?) sieci. Jeden z prowadzących posiedzenie przedstawił bardzo ciekawe zestawienie czasu liczenia problemu XYZ.

Typ komputera	Czas liczenia problemu XYZ w latach
VAX II/780	30 000
Cosmic cube	3 000
Cray I	100
Columbia	10
GF 11	1

Ostatni z wymienionych — GF 11 — w chwilach „dobrych" (tak zwanych pikach) liczy około 11 gigaoperacji w czasie jednej sekundy (11 miliardów operacji/sek.) Innymi słowy w czasie, gdy komputer wykonuje jedną operację (złożoną) np.: mnożenie dwóch wielocyfrowych liczb — światło przebędzie zaledwie nieco ponad 2 cm (dwa)! Jak to możliwe? — zadanie to (operację) wykonuje jednocześnie setki procesorów.

Cena superkomputera wynosi aż 5—20 mln dolarów. Reprezentantami tego sprzętu są np.: Cyber 205, Cray I, Cray II, a w Japonii urządzenia tego typu produkuje NEC i parę innych firm. Ciekawym osiągnięciem naukowców Kraju Kwitnącej Wiśni jest mikroprocesor (liczy szeregowo), który rachuje z szybkością przekraczającą 1 gigaoperację na sekundę (światło, w czasie wykonywania przez ten procesor jednej operacji, przebiega 30 cm).

Dostęp do superkomputerów np.: w Holandii (są to Cyber 205 i Cray) realizowany jest poprzez sieci połączone komputerami VAX.

Ciekawymi spostrzeżeniami jednego z uczestników konferencji jest to, że Fortran jest obecnie podstawowym językiem programowania superkomputerów.

Aby nie popadać w kompleksy zauważono, że odpowiednie „uzbrojenie" IBMa PC (których mamy w kraju sporo) w „przyspieszacze" tworzy nam komputery o tempie liczenia tylko 50—100 razy mniejszym od Cray'a.

Odmienny podział komputerów zaproponował inny z uczestników konferencji:

Mikrokomputery	— IBM, PS/2
Superminikomputery	— VAX, SCS-40
Superkomputery	— Cray, CD6, NEC
Hiperkomputery	— ETA-10

Architektura superkomputerów została zaprezentowana przez najmłodszych uczestników konferencji — członków studenckiego Koła Zainteresowań Cybernetycznych z WATu.

Warto w tym miejscu podkreślić wątpliwość, którą wywołała wypowiedź jednego z Panów Profesorów — *„Potrzebujemy w kraju superkomputera, a jest to jedyny nakład techniczny — sprzęt ten nie wymaga konserwacji"*... czyżby?

Na świecie pracuje obecnie około 260 superkomputerów, w tym, ponad 50 eksploatowanych jest w Japonii.

Powstało zatem pytanie — co robić w naszych warunkach? Pytanie to stało się retorycznym, a nas zaproszono do Ośrodka Krajowej Państwowej Komunikacji Samochodowej w Poznaniu. Tyle sprzętu informatycznego w jednym niewielkim budynku jeszcze nie widziałem: kilkadziesiąt klonów IBMa PC, w tym ELWRO 801At (dwie sztuki) oraz RIAD 34 we wspaniałym zestawie: dwie drukarki wierszowe, pięć napędów dyskowych ze sterownikami (2X100 Mb plus 3X217 Mb z możliwością dołączenia innych napędów, także 600 Mb), cztery przewijaki („samoładujące się") i adapter do wielu, bardzo wielu terminali (także JSG). Najwięcej zainteresowania budziła jednostka centralna — zbudowana u nas w kraju — wygląda niepozornie — solidny „słup" trochę wyższy od piszącego te słowa. Jednostka mieści w sobie 8 Mb RAM. Wykonuje około 0,5 mln operacji na sekundę. Sprzęt ten jest w trakcie uruchamiania.

Kończąc spisane na gorąco niektóre wrażenia, chciałbym podkreślić nieprzypadkowe wyliczanie we wstępie: 262 uczestników, 48 organizatorów! Wydaje się, że gdyby uzupełniono tytuł konferencji o magiczne słowo „naukowa", mniej było by osób zawiedzionych, które oczekiwały informacji bliższych naszej rzeczywistości i przyszłości — tej najbliższej.

Uczestnik

— Kocha, lubi, szanuje, nie dba, nie chce, żartuje...

Laser za złotówki

Rozmowa z Zygmuntem Klukiem — kierownikiem sklepu Centralnej Składnicy Harcerskiej Oddział Warszawa na ulicy Mokotowskiej 26. Jednym z 7 sklepów CSH specjalizujących się w sprzedaży sprzętu komputerowego i wideo.

Czego mogą spodziewać się klienci CSH w tym roku?

— Utrzymania dotychczasowych zasad. Będziemy nadal sprowadzać sprzęt komputerowy firm Spectra Video, Timex i Bondwell. W sprzedaży będą modele Spectra video SV 738, Timex, klony IBM z Dalekiego Wschodu — Bondwell 38-2-XT, Bondwell Portable (przenośny). Zamierzamy sprowadzić Bondwell 63 AT, uzupełnieniem do modeli XT i AT będą karty (pakiety) programowe. Nowością na polskim rynku będzie Spectra video X-PRESS 16 kompatybilny z IBM PC. Oczywiście nadal będziemy oferować urządzenia zewnętrzne i różne drobiazgi komputerowe: interfejsy, joystiki, dyskietki.

Od pewnego czasu poszerzyliśmy ofertę o sprzęt wideo — magnetowidy VHS, kamwidy, kasety, telewizory. Melomanów z pewnością ucieszy wiadomość, że w tym roku zamierzamy sprowadzić kompact dyski i odtwarzacze laserowe.

Jednym słowem klient będzie mógł kupić w CSH nowoczesny sprzęt. Czy nie boi się Pan konkurencji? W tym roku z pewnością powstaną liczne firmy ze zbliżonej branży.

— Jak do tej pory, to konkurencja boi się CSH. Bondwell XT sprzedawany był po cenach znacznie niższych niż w innych firmach, w związku z czym musiały je obniżyć. Jesteśmy na tyle solidną firmą, że „dorobiliśmy" się własnych klientów przyjeżdżających z odległych miast, nawet takich, gdzie są podobne sklepy. Z pewnością w jakimś stopniu przyciągają ich dwie literki w nazwie „PP" — przedsiębiorstwo państwowe.

Kto najczęściej korzysta z pańskiego sklepu?

— Blisko 50 procent sprzętu zakupił resort oświaty. Na drugim miejscu są kluby osiedlowe, miejskie, zakładowe, wiejskie. Z pewnością proporcje wyglądałyby inaczej gdybyśmy mieli więcej towaru, klienci prywatni kupują tylko „nadwyżki".

Jakie wymagania stawia Pan pracownikom?

— Muszą być uczciwi, znać się na komputerach, lubić handel. Tu przerzuca się tony sprzętu.

Ile zarabiają?

— W naszym oddziale jesteśmy w czołówce, powyżej średniej krajowej. Zarobki są ważne, bo mobilizują pracowników i przywiązują do firmy.

Kto wybiera dostawców sprzętu?

— Zarząd przedsiębiorstwa. Oceniamy ofertę potencjalnych kooperantów, ich wyroby, programy. Coraz więcej polskich firm z nami współpracuje — rzemieślnicy, spółdzielnie, zakłady państwowe. Firmy zagraniczne nie dostarczą nam polskich programów użytkowych, edukacyjnych czy gier.

Pańskie życzenia?

— Od początku uruchomienia sklepu to samo. Chciałbym sprzedawać polski komputer.

Czego Panu w imieniu redakcji życzę.

Dziękuję za rozmowę.

Dariusz OTTO

NAUKA. TECHNIKA ...

Naukowcom amerykańskim z laboratorium fizycznego w Waltham (stan Massachusetts) udało się opracować technologię produkcji tranzystorów. które niemal w całości wytwarza ... natura. Proces budowy tranzystorów rozpoczyna się od stopienia krzemu i tantalu powodując ich całkowite wymieszanie. Gdy tak przygotowaną mieszaninę powoli ochłodzić. to w kryształkach krzemu zaczynają powstawać wtrącenia tantalu. Występują one jednak tylko w tych punktach. w których uprzednio zanurzone były cienkie krzemowe pałeczki pełniące rolę zarodników. W trakcie krzepnięcia pałeczki te są stopniowo usuwane. a na ich miejscu rosną nitki tantalu o średnicy około 1 μm. Każda z tantalowych nici zamieniana zostaje w tranzystor w wyniku nałożenia wokół niej dwóch koncentrycznych warstw przewodnika prądu. Ten nowy sposób budowy tranzystorów pozwala na uniknięcie podstawowych trudności wiążących się z produkcją układów scalonych rozmieszczanych na płytkach krzemowych. jak również umożliwia obciążanie układów prądem o wyższym natężeniu bez możliwości uszkodzenia elementu elektronicznego.

■ ■ ■

W ZSRR prowadzone są ostatnio próby aparatury. króra w lipcu br.. w ramach międzynarodowej ekspedycji "PHOBOS" poleci w kierunku Marsa. W odstępie kilku dni wystartują dwa aparaty kosmiczne zawierające ponad 20 różnych urządzeń naukowych przeznaczonych do badania "Czerwonej Planety". jego satelity Phobosa. a także Słońca i przestrzeni międzyplanetarnej. W ekspedycji tej po raz pierwszy w praktyce kosmonautyki zostanie przeprowadzone badanie składu gruntu ciała niebieskiego przy pomocy promienia laserowego.

■ ■ ■

W roku 1988 planuje się w Wielkiej Brytanii założenie banku informacji o zwierzętach znajdujących się w ogrodach zoologicznych. Centralny bank zlokalizowany ma być w Londynie. Przewiduje się zainstalowanie terminali komputerowych w 15 ogrodach umożliwiających dostęp do danych o 40000 różnych zwierząt należących do ponad 1500 gatunków. Pozwoli to na opracowywanie nowych programów hodowli rzadkich i ginących zwierząt jak również wydawanie zaleceń dotyczących hodowli zwierząt obecnie żyjących w sztucznych warunkach. Należy oczekiwać. że do 1994 .roku powstanie międzynarodowy bank informacji. oparty na systemie opracowanym w Wielkiej Brytanii. z którego będą mogły korzystać ogrody zoologiczne w Ameryce.Australii i Europie.

■ ■ ■

Znana amerykańska firma elektroniczna TEXAS INSTRUMENTS opracowała nowy specjalizowany moduł mikroprocesorowy VHSIC oznaczony symbolem 1750A.

Będzie on. od 1988 r., podstawowym układem sterowania i kontroli w wojskowych samolotach firmy LOCKHEED.

Systemy doradcze

Janusz MORBITZER

Jednym z praktycznych rezultatów badań w dziedzinie sztucznej inteligencji jest powstanie tzw. systemów doradczych, zwanych również systemami eksperckimi (ang. expert systems).

System doradczy to program symulujący pracę grupy ekspertów rozwiązującej problem z zakresu jednoznacznie sprecyzowanej dziedziny. System taki jest bardzo wąsko specjalizowany.

Jaki jest cel stosowania systemów doradczych? Otóż praca eksperta-człowieka jest zwykle bardzo droga. Nie zawsze możliwe jest szybkie przybycie eksperta na miejsce, gdzie trzeba postawić diagnozę. Ponadto zdania poszczególnych ekspertów rozwiązujących ten sam problem mogą być różne. Opłacalne jest więc kilkakrotne zorganizowanie spotkań ekspertów w celu „wydobycia" od nich wiedzy z zakresu wybranej dziedziny. Wiedza ta jest podstawą budowy systemu doradczego. Zakłada się, że suma wiedzy uzyskanej od ekspertów jest większa od wiedzy każdego z nich z osobna. Raz zbudowany system można powielać w dowolnej liczbie egzemplarzy i dostarczać tam, gdzie jego stosowanie jest celowe i opłacalne.

Dla jakich problemów tworzy się systemy doradcze? Ogólnie istnieją dwie grupy takich problemów. Pierwsza, to problemy, dla których nie jest znany algorytm rozwiązania. Drugą grupą są problemy, dla których algorytm rozwiązania jest wprawdzie znany, ale czas jego realizacji znacznie przekracza rozsądne, dopuszczalne granice. Przykładem takiego problemu jest gra w szachy.

Algorytm rozwiązania jest tutaj znany. Wszystkie możliwe warianty gry można przedstawić w postaci drzewa (rys. 1), przypisując końcowym węzłom atrybuty: partia wygrana, przegrana i remis. Wystarczy teraz tylko wybierać te ruchy, które prowadzą do wygranej lub w najgorszym wypadku (gdy przeciwnik również realizuje tę samą strategię) do remisu.

sytuacja startowa

20 możliwych ruchów białych

20 możliwych odpowiedzi czarnych na każdy ruch białych

Wyczerpująca analiza takiego drzewa nie jest jednak możliwa. Liczba wszystkich dróg do przebadania wynosi około 10^{120}. Zakładając, że analiza każdego ruchu trwa 1 ns i wiedząc, że 1 rok = 3.15×10^{13} (ns), łatwo można obliczyć, że analiza całego drzewa trwałaby około 10^{99} lat. Warto podkreślić, że istotne ograniczenia, dyskwalifikujące przedstawiony algorytm, nie tkwią w sprzęcie, gdyż nawet przyjęcie większej o kilka rzędów prędkości obliczeń nie prowadzi do zadowalających rezultatów. W jaki sposób można więc rozwiązać problem gry w szachy? Problem ten można rozwiązać stosując tzw. programowanie heurystyczne. Heurystyka to ogólna zasada, wskazówka lub taktyka oparta na doświadczeniu, drastycznie ograniczająca poszukiwanie rozwiązań w dużych obszarach problemowych.

Heurystyka prawie zawsze daje wystarczająco dobre rozwiązanie. Oznacza to, że czasami na skutek drastycznego ograniczania poszukiwań, najlepsze rozwiązanie (a nawet wszystkie rozwiązania) mogą zostać pominięte. Programowanie heurystyczne odgrywa dominującą rolę w systemach doradczych. Systemy te nie działają bowiem na podstawie algorytmów, lecz wiedzy empirycznej, a więc heurystyki.

System doradczy składa się z następujących modułów funkcjonalnych:

— bazy danych,
— bazy reguł,
— układu kontroli.

Baza danych zawiera fakty z danej dziedziny. Relacje pomiędzy faktami przechowywane są w bazie reguł. Są to najczęściej reguły typu

IF warunek THEN akcja — 1
ELSE akcja — 2

przy czym warunek nie musi mieć określonej wartości logicznej lub wartość ta może być określona z pewnym prawdopodobieństwem.

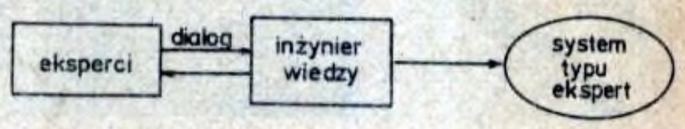
Układ kontroli określa sposób korzystania z relacji. Służy on do:

— selekcjonowania reguł.
— dopasowywania reguł
— kolejkowania reguł (gdy kilka z nich rokuje sukces)
— wykonywania reguł.

Jakość systemu doradczego zależy głównie od jakości wiedzy. Wiedza obejmuje fakty, związki między faktami oraz metody korzystania z tych związków. Istotnym zagadnieniem jest również zupełność wiedzy, tj. objęcie wszystkich możliwych problemów z danej dziedziny. Jeżeli wiedza jest niezupełna, to część problemów nie może być rozwiązana, gdyż brak jest powiązań z wiedzą zawartą w systemie.

W jaki sposób buduje się system doradczy? Proces ten przebiega w kilku po sobie następujących etapach:

1 — pozyskanie wiedzy (ang. knowledge acquisition process) — np. 2
2 — identyfikacja (określenie problemów i zakresu działania systemu)
3 — konceptualizacja (określenie sposobu reprezentowania wiedzy i przepływu informacji)
4 — formalizacja (wyborów języka, opis)
5 — implementacja systemu (programowanie)
6 — testowanie.



Rys. 2. Etap pozyskiwania wiedzy

Systemy doradcze tworzone są zwykle w języku LISP, PROLOG, C, PASCAL lub innym zorientowanym na przetwarzanie logiczne. Systemy te nie tylko powinny dostarczyć rozwiązania problemu, ale również odtworzyć na żądanie sposób dojścia do tego rozwiązania. Jest to tzw. odpowiedź na pytanie WHY? (dlaczego system zrobił to, co zrobił).

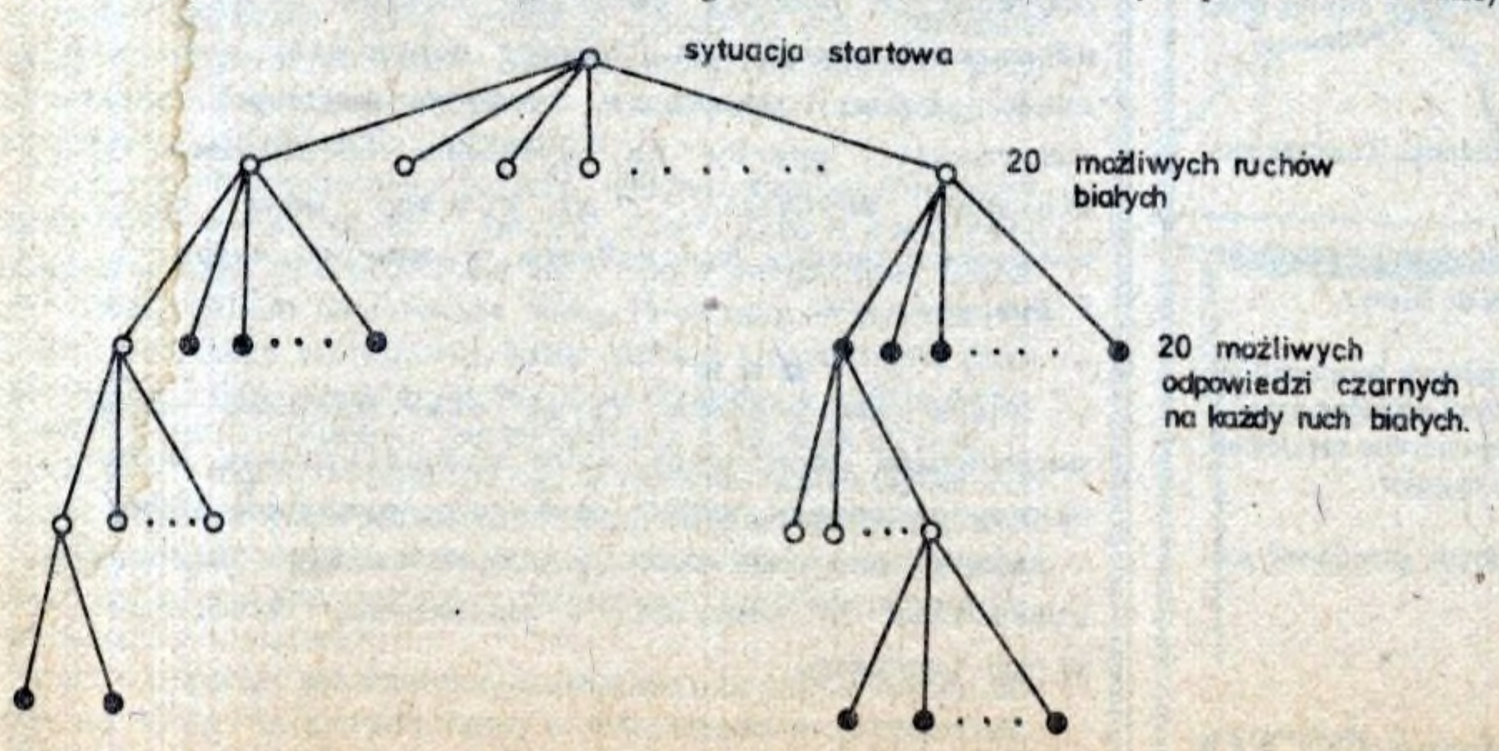
Systemy doradcze są systemami otwartymi — mogą przyjmować nowe reguły wzbogacając tym samym bazę wiedzy. System automatycznie sprawdza, czy nowa reguła nie jest sprzeczna z istniejącymi lub czy z nich w oczywisty sposób nie wynika.

Systemy doradcze znajdują obecnie zastosowanie w wielu dziedzinach:

— w medycynie (np. MYCYN, ARAMIS, INTERNIST)
— w rozpoznawaniu struktur chemicznych (np. DENDRAL)
— w diagnozowaniu chorób roślin
— w klasyfikowaniu roślin i zwierząt
— w diagnozowaniu uszkodzeń elektrycznych i mechanicznych (np. silników)
— w ocenie strat wynikłych z trzęsienia ziemi (system SPERILL)
— w systemach biurowych.

Jednym z najbardziej znanych systemów jest MYCYN. Służy on do diagnozowania chorób zakaźnych krwi. System ten składa się z 450 reguł.

Zbudowane dotychczas systemy są stosunkowo prymitywne. Zawarta w nich wiedza jest dużym uproszczeniem rzeczywistości. Podstawowymi problemami są sposoby reprezentowania wiedzy oraz sekwencyjność pracy komputerów (człowiek myśli równolegle). O ile drugi problem może być rozwiązany na drodze sprzętowej (wieloprocesowość, transputery), o tyle rozwiązanie pierwszego wymaga opracowania nowych metod reprezentacji wiedzy — jej rozumienia i organizacji, tj. powiązań między poszczególnymi informacjami. Jest to podstawowy kierunek badań sztucznej inteligencji. Duże nadzieje na postęp w tej dziedzinie związane są z komputerami piątej generacji.



Rys. 1. Fragment drzewa gry w szachy

W szponach Atari (11)

„Przewijanie" obrazu

Tomasz MROWIEC
Ludwik PIELA

Często zdarza się, że chcemy wyświetlić taką ilość informacji, która przekracza pojemność ekranu. Jednym ze sposobów rozwiązania tego problemu jest przesuwanie informacji po ekranie. Przykładem takiego rozwiązania jest przeglądanie programu po podaniu polecenia **LIST**. Poszczególne linie programu przesuwają się po ekranie od dołu do góry. Ten rodzaj przesuwania stosowany jest we wszystkich mikrokomputerach. Komputery Atari dysponują dwoma dodatkowymi sposobami przesuwania, oferującymi niezwykłe możliwości: zgrubnym i płynnym.

W większości komputerów używane jest przesuwanie zgrubne: pola przechowujące znaki mają ustalone położenie na ekranie i przesuwanie tekstu polega na przemieszczaniu bajtów w obszarze pamięci ekranu. Takie przesuwanie odbywa się skokowo, o jeden wiersz lub kolumnę. Rozdzielczością przesuwania jest wysokość lub szerokość znaku. Ruch jest szarpany i dosyć nieprzyjemny. Poza tym uzyskiwany jest przez przemieszczanie w pamięci tysięcy bajtów, co odbywa się stosunkowo wolno.

W pewnych mikrokomputerach można uzyskać nieco płynniejszy ruch przez kreślenie obrazów w trybie graficznym o wyższej rozdzielczości. Podczas przesuwania uzyskuje się większą rozdzielczość, jednak kosztem znacznego spowolnienia działania programu. Cała trudność polega na tym, że przesuwanie informacji na ekranie realizowane jest przez przemieszczanie danych w obszarze pamięci ekranu.

W komputerach Atari przesuwanie zgrubne można uzyskać w nieco inny sposób: przesuwając ekran nad danymi, które mają być wyświetlone. Umożliwia to rozkaz LMS, który występuje w programie wyświetlania i informuje ANTIC, skąd mają być brane dane do wyświetlania. Normalnie występuje jeden rozkaz LMS na początku programu wyświetlania. Wskazuje on początek obszaru RAM zawierającego dane, które mają być wyświetlone na ekranie. A zatem zgrubne przesuwanie możemy uzyskać przez manipulowanie bajtami argumentu rozkazu LMS. Przez zmianę zawartości dwóch bajtów adresu możemy osiągnąć efekt podobny do przemieszczania setek bajtów w obszarze pamięci ekranu. **Program nr 1** demonstruje wykorzystanie tego sposobu. Przesuwa on ekran nad obszarem pamięci zawierającym wydruk programu oraz procedury systemu operacyjnego. W rezultacie otrzymujemy przesuwanie poziome i pionowe. Czyste przesuwanie pionowe możemy uzyskać przez dodawanie lub odejmowanie długości linii w bajtach do argumentu rozkazu LMS. Wykonuje to **program nr 2.**

```
FK 1 REM ****************
FJ 2 REM *                *
XA 3 REM * Program nr 1 *
FL 4 REM *                *
FO 5 REM ****************
NL 6 REM
ST 10 DL=PEEK(560)+256*PEEK(561)
HE 20 LMSL=DL+4
GP 30 LMSH=DL+5
KS 35 ? CHR$(125):LIST
JU 40 FOR I=PEEK(89) TO 255
IG 50 POKE LMSH,I
JO 60 FOR J=PEEK(88) TO 255
KS 70 POKE LMSL,J
ES 80 FOR W=1 TO 160:NEXT W
JG 90 NEXT J
FR 100 NEXT I
NS 110 END
```

```
FK 1 REM ****************
FJ 2 REM *                *
XU 3 REM * Program nr 2 *
FL 4 REM *                *
FO 5 REM ****************
NL 6 REM
HG 10 LIST 1,140
SU 20 DL=PEEK(560)+256*PEEK(561)
HF 30 LMSL=DL+4
GQ 40 LMSH=DL+5
ZR 50 EL=24:EH=PEEK(561)
PY 60 EL=EL+40
HO 70 IF EL<256 THEN 110
QZ 80 EL=EL-256
SP 90 EH=EH+1
FF 100 IF EH=256 THEN END
MQ 110 POKE LMSL,EL
IG 120 POKE LMSH,EH
LF 125 FOR W=1 TO 250:NEXT W
RG 130 GOTO 60
NY 140 END
```

Czyste przesuwanie poziome nie jest tak łatwe do uzyskania jak pionowe. Powodem jest szeregowa organizacja pamięci ekranu. Bajty danych umieszczone są kolejno, bajty jednej linii występują bezpośrednio za bajtami poprzedniej linii. Możemy przesuwać linie poziomo w lewo przez zwiększanie argumentu LMS. W tym wypadku jednak, skrajne lewe bajty każdej linii będą przesuwane do skrajnej prawej pozycji następnej, wyższej linii. Było to widoczne po uruchomieniu programu nr 1.

Rozwiązaniem może być rozszerzenie obszaru danych ekranu i rozbicie go na ciąg niezależnych obszarów poziomych linii danych.

Musimy pamiętać, że RAM jest jednowymiarowa i bajty umieszczone są kolejno, lecz teraz używana jest inaczej. Dla każdej linii poziomej rozszerza się obszar RAM dużo dalej niż może być pokazane na ekranie. Nie jest to przypadek, wszak chcemy wyświetlić większą ilość danych niż możemy zmieścić na ekranie. Aby pokazać tę dodatkową informację, musimy ją wpierw przechować w pamięci. Przy takim uporządkowaniu możemy stosować prawdziwe przesuwanie poziome. Możemy przesuwać ekran nad danymi bez niepożądanego przesuwania pionowego.

Pierwszym krokiem w uzyskaniu czystego przesuwania poziomego jest określenie całkowitej długości linii poziomej i przydzielenie odpowiedniego obszaru RAM. Następnie musimy napisać całkowicie nowy program wyświetlania z rozkazami LMS w każdej linii obrazu. Program wyświetlania będzie oczywiście dłuższy niż zwykle. Jako argumentu LMS najwygodniej jest użyć adresu pierwszego bajtu każdej poziomej linii danych ekranu. Dla każdej linii obrazu będzie jeden taki adres.

W celu uzyskania przesuwania każdy argument rozkazu LMS w programie wyświetlania musi być zwiększony, aby uzyskać ruch w lewo, albo zmniejszony dla ruchu w prawo. Program musi być tak skonstruowany, aby obraz nie przesunął się poza granice przydzielonych obszarów RAM; w przeciwnym przypadku mogą być wyświetlone „śmieci". Podczas tworzenia programu należy pamiętać, że argument LMS wskazuje pierwszy bajt danych ekranu w wyświetlanej linii. Maksymalna wartość argumentu równa jest adresowi ostatniego bajtu w długiej linii poziomej zmniejszonemu o liczbę bajtów w wyświetlanej linii. Pamiętajmy także, że wartość argumentu LMS nie powinna, wewnątrz jednej długości linii ekranu (w bajtach), przekraczać granicy adresu 4KB.

Opisany proces jest dosyć skomplikowany, dlatego przedstawimy go na przykładzie. Najpierw wybieramy długość linii poziomej. Najwygodniej będzie użyć linii o długości 256B, uprości to nam obliczanie adresu. Każda linia będzie zatem wymagać jednej strony pamięci. Używając trybu 2 BASIC-a zmieścimy na ekranie 12 linii obrazu; zatem potrzebne będzie 12 stron (3KB) RAM. Dla uproszczenią i zagwarantowania, że pamięć ekranu nie będzie zawierać samych zer, użyjemy dolnych 3KB RAM. Obszar ten jest używany przez OS i DOS, dlatego powinien zawierać interesujące dane. Dla uczynienia eksperymentu bardziej interesującym, umieścimy program wyświetlania na stronie 6, dzięki temu możemy wyświetlić go na ekranie podczas przesuwania. Początkowe wartości argumentów LMS będą zatem łatwe do obliczenia: wszystkie mniej znaczące bajty będą zerami, a bardziej znaczące będą miały kolejno wartości 0,1,2,...itd. **Program nr 3** wykonuje opisane operacje i przesuwa program poziomo. Przegląda on dane z prawa na lewo. Po osiągnięciu końca strony zaczyna przeglądanie od początku. Program wyświetlania można znaleźć w szóstej linii od dołu (tj. na stronie 6).

```
FK 1 REM ****************
FJ 2 REM *                *
YO 3 REM * Program nr 3 *
FL 4 REM *                *
FO 5 REM ****************
NL 6 REM
QY 10 REM ZAPIS PROGRAMU WYSWIETL
   ANIA
WO 20 FOR I=1536 TO 1538
LD 30 POKE I,112
IS 40 NEXT I
HL 50 FOR I=1 TO 12
XI 60 POKE 1536+3*I,71
ZV 70 POKE 1536+3*I+1,0
TU 80 POKE 1536+3*I+2,I
IX 90 NEXT I
WU 100 POKE 1575,65
NH 110 POKE 1576,0
RI 120 POKE 1577,6
IT 130 REM PODANIE ADRESU PROGRAM
   U WYSWIETLANIA
HS 140 POKE 560,0
LM 150 POKE 561,6
ZB 160 REM PRZESUWANIE POZIOME
DV 170 FOR I=0 TO 235
SS 180 FOR J=1 TO 12
LW 190 POKE 1536+3*J+1,I
GC 200 NEXT J
JJ 205 FOR W=1 TO 50:NEXT W
FU 210 NEXT I
OK 220 GOTO 170
NX 230 END
```

Następnym krokiem jest połączenie przesuwania pionowego i poziomego dla uzy-

skania ruchu ukośnego. Przesuwanie poziome otrzymujemy przez dodanie lub odjęcie 1 od argumentu LMS. Przesuwanie pionowe — przez dodanie lub odjęcie długości linii od argumentu. Wykonanie obu operacji daje w efekcie ruch ukośny.

Poprzez różnorodne manipulowanie bajtami argumentu LMS możemy uzyskać różne rodzaje uporządkowań. Linie mogą przesuwać się względem siebie lub wzajemnie przeskakiwać. Część z tych efektów można uzyskać za pomocą konwencjonalnego manipulowania obrazem, lecz dla ich osiągnięcia musi być przemieszczonych więcej danych. Zaletą przesuwania za pomocą argumentów LMS jest jego szybkość. Zamiast manipulowania całym ekranem danych, o wielkości wielu tysięcy bajtów, wystarcza manipulowanie dwoma lub kilkoma tuzinami bajtów.

Drugą interesującą zaletą komputerów Atari jest możliwość płynnego przesuwania, tj. przesuwania w odstępach mniejszych niż wielkość znaku. Odbywa się ono w krokach jednej linii ekranu pionowo oraz jednego taktu koloru poziomo. Ten typ przesuwania nie może iść zbyt daleko; dla otrzymania w pełni płynnego przesuwania na dużych odległościach, musimy połączyć płynne przesuwanie ze zgrubnym.

Istnieją dwa kroki dla implementacji płynnego przesuwania. Pierwszy, ustawiamy bity zezwolenia na płynne przesuwanie w bajtach rozkazów programu wyświetlania dla linii obrazu, w których chcemy mieć taki ruch. W większości przypadków interesuje nas cały ekran, zatem ustawiamy odpowiednie bity we wszystkich rozkazach programu wyświetlania. Bit D5 rozkazu zezwala na przesuwanie pionowe, a bit D4 — na poziome. Następnie zapamiętujemy żądaną odległość przesuwania w odpowiednim rejestrze. Istnieją dwa, jeden dla przesuwania poziomego, drugi dla pionowego. Rejestr przesuwania poziomego (HSCROL) jest pod adresem $D404(54276), rejestr przesuwania pionowego (VSCROL) — pod adresem $D405(54277). Dla ruchu poziomego musimy zapamiętać w HSCROL liczbę taktów koloru, o którą chcemy przesunąć linie obrazu. Dla ruchu pionowego zapamiętujemy w VSCROL liczbę linii ekranu, o którą chcemy przesunąć linie obrazu. Zapamiętane wartości przesuwania będą używane dla każdej linii, dla której dozwolone jest płynne przesuwanie.

Podczas używania płynnego przesuwania możemy napotkać dwa problemy. Oba wynikają z faktu, że częściowo przesunięty obraz pokaże więcej informacji niż normalny. Rozważmy, na przykład, co się zdarzy, kiedy przesuniemy poziomo linię o pół znaku w lewo. W linii jest 40 znaków. Połowa pierwszego znaku zniknie za lewym brzegiem ekranu. Czterdziesty znak przesunie się w lewo. Co zajmie jego miejsce? Powinna przesunąć się połowa nowego znaku i zająć miejsce przesuniętego teraz czterdziestego znaku. Mógłby to być 41 znak. Lecz co się zdarzy, jeśli w normalnej linii jest tylko 40 znaków?

Jeśli zastosujemy przesuwanie zgrubne, wtedy 41 znak nieoczekiwanie pojawi się na ekranie po zniknięciu pierwszego znaku za lewym brzegiem. Rozwiązanie tego problemu zostało wbudowane w sprzęt. Istnieją trzy możliwe szerokości ekranu: wąski (o szerokości 128 taktów koloru), normalny (o szerokości 160 taktów koloru) i szeroki (o szerokości 192 taktów koloru). Opcje te wybierane są przez ustawienie odpowiednich bitów w rejestrze DMACTL. Podczas używania płynnego przesuwania poziomego ANTIC automatycznie pobiera więcej danych z RAM niż wyświetla. Jeśli, na przykład, DMACTL ustawiony jest dla normalnej szerokości ekranu, który w trybie 0 BASIC-u ma 40 bajtów na linię, to ANTIC będzie pobierał ilość danych odpowiednią dla szerokiego ekranu — 48 bajtów na linię. Jeśli nie będzie to brane pod uwagę, to spowoduje przesunięcie linii w poziomie. Problem ten nie wystąpi, jeśli zorganizowaliśmy już RAM w długie linie poziome.

Podobny problem dla przesuwania pionowego może być obsługiwany jednym z dwóch sposobów. Pierwszym, niezbyt starannym, jest zignorowanie go. Wtedy nie otrzymamy półobrazów na obu końcach obrazu. Zamiast tego obrazy u dołu ekranu nie będą przesuwać się właściwie; będą one pojawiać się nieoczekiwanie. Dla uzyskania właściwego płynnego przesuwania do i z rejonu ekranu musimy przeznaczyć jedną linię obrazu do działania jako bufor. Wykonamy to przez nieustawianie bitu przesuwania pionowego w rozkazie programu wyświetlania ostatniej linii strefy obrazu przesuwanej pionowo. Okno przesunie się teraz bez nieprzyjemnego szarpnięcia. Obraz ekranu będzie skrócony o jedną linię. Zaleta przesuwania obrazów staje się teraz widoczna. Możliwe jest tworzenie obrazów, które mają więcej niż 192 linie ekranu.

Płynne przesuwanie ma ograniczony zasięg. Pionowym ograniczeniem jest 15 linii ekranu, poziomym — 16 taktów koloru. Jeżeli spróbujemy przesunąć poza te granice, ANTIC zignoruje starsze bity rejestrów przesuwania. Dla uzyskania pełnego płynnego przesuwania, w którym cały ekran płynnie przesuwa się tak daleko, jak chcemy, musimy połączyć płynne przesuwanie ze zgrubnym. W tym celu najpierw płynnie przesuwamy obraz, śledząc jak daleko się przemieścił. Kiedy długość przesunięcia równa się wielkości znaku, zerujemy rejestr płynnego przesuwania i wykonujemy przesuwanie zgrubne.

Program nr 4 ilustruje proste płynne przesuwanie realizowane z małą szybkością. Demonstruje on kilka problemów, które wystąpią podczas przesuwania. Pierwszy, linie obrazu poniżej przesuwanego okna są przesunięte w prawo. Jest to zgodne z wcześniejszym opisem, że ANTIC automatycznie pobiera 48 bajtów zamiast 40. Problem ten występuje tylko w nierealistycznych programach demonstracyjnych, takich jak ten. W rzeczywistych zastosowaniach uporządkowanie danych ekranu wyklucza ten problem. Drugi, bardziej poważny, występuje, kiedy rejestry przesuwania są modyfikowane w czasie, gdy ANTIC jest w środku procesu wyświetlania. Wprowadza to pewne zamieszanie i powoduje, że obraz na ekranie drży. Rozwiązaniem jest zmiana rejestrów przesuwania tylko w czasie okresów przerwy pionowej. Może to być wykonywane tylko za pomocą procedur języka asemblera.

```
FK 1 REM ****************
FJ 2 REM *              *
ZI 3 REM * Program nr 4 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
TR 8 HSCR=54276
VU 9 VSCR=54277
BS 10 GRAPHICS 0:LIST 1,200
SU 20 DL=PEEK(560)+256*PEEK(561)
XZ 30 POKE DL+10,50
YN 40 POKE DL+11,50
KR 50 FOR Y=0 TO 7
UN 60 POKE VSCR,Y
OT 70 GOSUB 200
OK 80 NEXT Y
WO 110 FOR X=15 TO 0 STEP -1
ZN 120 POKE HSCR,X
RD 130 GOSUB 200
LT 140 NEXT X
QG 150 GOTO 30
VG 200 FOR J=1 TO 200:NEXT J:RETU
   RN
JJ 205 FOR W=1 TO 50:NEXT W
NT 210 END
OK 220 GOTO 170
NX 230 END
```

Zastosowania pełnego płynnego przesuwania są niezliczone. Oczywistym jest, na przykład, wykorzystanie go do przeglądania dużych map, utworzonych za pomocą grafiki znakowej. Używając trybu 2 BASIC-u można utworzyć dużą mapę, która zawiera np.: 10 pełnoekranowych obrazów. Ekran staje się oknem dla mapy. Użytkownik może przesuwać całą mapę za pomocą manipulatora. Taki system jest bardzo efektywny pod względem zajętości pamięci.

W ten sposób mogą być również prezentowane duże schematy elektroniczne. Manipulator może być używany zarówno do poruszania się po schemacie, jak i wskazywania określonych składników, które użytkownik chce wybrać. Za pomocą tej techniki może być prezentowany dowolny duży obraz, który nie musi być oglądany w całości.

W podobny sposób mogą być przeglądane duże bloki tekstu. Wprawdzie czytanie ciągłych bloków tekstu przez przesuwanie obrazu nie jest zbyt wygodne, jednak może być używane, na przykad, do sprawdzenia, czy tekst został właściwie zredagowany. Wykorzystano to w programie ATARI WRITER. Na ekranie dysponującym 40 znakami w linii możemy obejrzeć zredagowaną stronę (taką, jak otrzymamy po wydrukowaniu) zawierającą np.: 80 znaków w linii. Technika ta bardziej przydaje się do prezentowania oddzielnych bloków tekstu. Szczególnie interesujące może być zastosowanie jej do tworzenia rozwijających się menu.

Program nr 5 prezentuje jeden z możliwych do uzyskania efektów. Ma on zmodyfi-

```
FK 1 REM ****************
FJ 2 REM *              *
AC 3 REM * Program nr 5 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
BO 8 DIM SC$(17):FOR I=1 TO 17:RE
   AD A:SC$(I,I)=CHR$(A):NEXT I
MW 9 GRAPHICS 0:POKE 752,1:POKE 6
   22,255
NU 10 POKE 1536,112
BF 20 POKE 1537,66:POKE 1538,64:P
   OKE 1539,156
KM 25 FOR I=0 TO 23:POKE 1540+I,2
   :NEXT I:POKE 1564,112
BD 30 POKE 1565,82:POKE 1566,0:PO
   KE 1567,100:POKE 1568,65:POKE
   1569,0:POKE 1570,6
KD 40 POKE 560,0:POKE 561,6:POKE
   54276,7:LIST 0,70
HR 50 FOR I=0 TO 920:POKE 25640+I
   ,PEEK(40000+I):NEXT I
MV 55 X=0:Y=100:POKE 1567,Y:POKE
   1566,X
OK 56 FOR J=0 TO 230
VG 60 FOR I=7 TO 0 STEP -1:POKE 5
   4276,I:GOSUB 100:NEXT I
```

```
PL 62 X=X+2:IF X>255 THEN X=0:Y=Y
   +1
JU 63 A=USR(ADR(SC$),7,X,Y)
YL 64 REM POKE 54276,7:POKE 1566,
   X:POKE 1567,Y
JP 66 NEXT J
UW 70 GOTO 55
VU 100 FOR T=0 TO 5:NEXT T:RETURN
QU 200 DATA 104,104,104,141,4,212
   ,104,104,141,30,6,104,104,141,
   31,6,96
JJ 205 FOR W=1 TO 50:NEXT W
OK 220 GOTO 170
NX 230 END
```

kowany program wyświetlania, dzięki czemu została wyraźnie wydzielona jedna linia u dołu ekranu. Otrzymujemy w niej LIST programu z przesuwaniem w poziomie, a nie w pionie. Podprogram w języku maszynowym (umieszczony w zmiennej SC$) umożliwia zrealizowanie płynnego przesuwania poziomego na dużej odległości.

Program nr 6 zamienia ekran naszego telewizora w przesuwający się papier, na którym dwa pisaki kreślą linie. Jeden pisak stoi w miejscu w celu oznaczenia linii środkowej.

```
FK 1 REM ****************
FJ 2 REM *              *
RW 3 REM * Program nr 6 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
QZ 50 GOSUB 800
RK 60 GOSUB 900
SX 70 POKE X2,80
OS 80 POKE 1675,0:REM USTAWIENIE
   SZYBKOSCI
BW 90 START=100
VU 100 ? :? :? :? :? "NACISNIJ <S
   TART> DLA NOWEGO PRZEBIEGU"
RW 110 IF PEEK(53279)<>6 THEN 110
TE 120 GOSUB 700
HP 130 W=INT(PEEK(53770)/32)
AG 140 WYNIK=W:PHI=T(W)
UK 150 Q=-16:PROBA=100
IK 160 FOR PR=1 TO PROBA-1:Q=-Q:S
   =Q+16
ON 170 R=PEEK(53770)
VT 180 IF PEEK(53279)=3 THEN R=25
   6-R
FU 190 IF R>127 THEN 220
ZN 200 W=W+1:IF W=8 THEN W=7
NA 210 GOTO 230
OM 220 W=W-1:IF W=-1 THEN W=0
TT 230 WYNIK=WYNIK+W:PHI=T(W)
BC 240 FOR Z=32+S TO 63+S
TG 250 POKE X1,PEEK(88+Z-PHI)+PEE
   K(88+Z+PHI-1)
UJ 260 NEXT Z:NEXT PR
FN 270 GOSUB 780:? :? :? :?
ES 280 ? "   WARTOSC JEST DODATNI
   A LUB UJEMNA"
UA 290 ? "   JESLI SREDNIA AMPLIT
   UDA BYLA"
NG 300 ? "   POWYZEJ LUB PONIZEJ
   OCZEKIWANEJ."
GE 310 ? :? :? "     WARTOSC = ";W
   YNIK-PROBA*7/2
DJ 320 ? :? :? :GOTO START
XL 700 GRAPHICS 7+16:POKE 559,0
DI 710 SETCOLOR 4,0,0:SETCOLOR 2,
   1,8:SETCOLOR 0,8,4
KP 720 PG=PEEK(106)
TT 730 POKE 203,56:POKE 204,PG-1
CJ 740 POKE 205,PEEK(560):POKE 20
   6,PEEK(561)
LF 750 KEEPL=PEEK(546):KEEPH=PEEK
   (547)
NZ 760 POKE 546,135:POKE 547,6
XP 770 POKE 559,34:RETURN
CO 780 POKE 559,0:GRAPHICS 7+16:P
   OKE 546,KEEPL:POKE 547,KEEPH
EO 790 GRAPHICS 0:RETURN
VZ 800 X1=1664:X2=1665
VX 810 DATA 200,200,40,192,48,12,
   3,216,165,20,41,0,24,208,66,16
   5
YO 811 DATA 203,109,130,6,133,203
   ,144,26,165,204,105,0,41,15,72
   ,13
AC 812 DATA 175,6,133,204,104,208
   ,11,174,130,6,169,0,202,157,0,
   112
OR 813 DATA 208,250,172,130,6,169
   ,0,136,145,203,208,251,160,4,1
   77,205
RP 814 DATA 109,130,6,145,205,200
   ,177,205,105,0,41,15,109,175,6
   ,145
WV 815 DATA 205,169,255,141,250,6
   ,173,128,6,32,234,6,169,85,141
   ,250
GW 816 DATA 6,173,129,6,32,234,6,
   76,95,228,201,180,176,17,72,74
WM 817 DATA 74,168,104,41,3,170,1
   89,131,6,41,255,17,203,145,203
   ,96
HE 820 DATA 559,0
UW 830 FOR N=1664 TO 1791
FM 840 READ DAT:POKE N,DAT:NEXT N
DS 850 POKE 1711,PEEK(106)-16
XO 860 POKE 559,34:RETURN
HY 900 DIM T(8)
LN 910 POKE 559,0
MG 920 DEG :SB=1536
FR 930 FOR N=0 TO 127
QV 940 POKE SB+N,40*SIN(N*45/8)+4
   0:NEXT N
AP 950 T(0)=16:T(1)=15:T(2)=14:T(
   3)=13
VG 960 T(4)=11:T(5)=9:T(6)=6:T(7)
   =0
RP 970 POKE 559,34
ZV 980 RETURN
```

a drugi wykonuje ruch sinusoidalny ze zmieniającą się losowo amplitudą. Przy każdym przekraczaniu linii środkowej podejmowana jest losowo decyzja o zwiększeniu lub zmniejszeniu amplitudy dla następnego półokresu (z uwzględnieniem dolnego i górnego ograniczenia amplitudy).

Każda binarna decyzja (zwiększenie lub zmniejszenie) wynika z kombinacji decyzji człowieka i komputera. W komórce o adresie 53770(RANDOM) występuje szybko zmieniający się ciąg liczb pseudolosowych z przedziału 0—255. Podczas każdego przecinania linii środkowej komputer decyduje o zwiększeniu lub zmniejszeniu amplitudy pisaka w zależności od tego, czy aktualna liczba losowa jest większa, czy mniejsza od 127. Określa to przebieg zdarzeń do czasu naciśnięcia przez użytkownika klawisza OPTION. Jeśli jest on naciśnięty w czasie przecinania linii środkowej, wtedy decyzja komputera, o zwiększeniu lub zmniejszeniu, jest odwracana.

Po pewnej liczbie okresów program kończy działanie i wyświetlany jest wynik. Wartość dodatnia lub ujemna wskazuje, że średnia amplituda ruchomego pisaka była większa lub mniejsza od oczekiwanej w tym przebiegu.

Opisując manipulowanie obrazem za pomocą rozkazów LMS programu wyświetlania nie można pominąć przesuwania „bardzo zgrubnego", czyli wymiany całych obrazów na ekranie.

Konwencjonalny sposób, stosowany w większości komputerów, polega na kopiowaniu zawartości wybranego obszaru RAM do obszaru pamięci wyświetlanego na ekranie. Czynność taka może być czasochłonna, dlatego musi być realizowana przez procedurę w języku maszynowym. Pomimo tego, podczas przesyłania kilku tysięcy bajtów, może być zauważalne opóźnienie zmiany obrazu.

Zamiast przesyłania setek i tysięcy bajtów, w Atari wystarczy zmienić zawartość dwóch (lub czterech) bajtów, aby uzyskać taki sam efekt w znacznie krótszym czasie. Możliwe jest również przygotowanie kilku obrazów i szybkie wyświetlanie ich na ekranach w określonej kolejności.

Opisany sposób demonstruje **program nr 7**. Po uruchomieniu następuje krótka przerwa, w czasie której komputer zapisuje ekran nr 2. Następnie widoczne jest zapisywanie informacji na ekranie nr 1. Naciskając klawisze 1 lub 2 otrzymujemy na ekranie jeden z dwóch obrazów.

```
FK 1 REM ****************
FJ 2 REM *              *
BQ 3 REM * Program nr 7 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
FL 100 DL=PEEK(560)+256*PEEK(561)
ZW 110 MEMTOP=PEEK(742)
TO 120 EKRAN1=PEEK(89):EKRAN2=MEM
   TOP-5
PT 130 POKE 89,EKRAN2:POKE 106,EK
   RAN2+4:? CHR$(125)
WB 140 FOR X=1 TO 22:? "*********
      EKRAN 2     *********":NE
   XT X
NX 150 POKE 89,EKRAN1:POKE 106,EK
   RAN1+4:? CHR$(125)
XL 160 FOR X=1 TO 22:? "---------
      ekran 1     ---------":NE
   XT X
WG 180 OPEN #1,4,0,"K:":GET #1,X:
   CLOSE #1:X=X-48:IF X<1 THEN 18
   0
TO 190 ON X GOTO 210,220
OR 200 GOTO 180
UA 210 GOSUB 250:POKE DL+5,EKRAN1
   :GOTO 180
MG 220 GOSUB 250:POKE DL+5,EKRAN2
   :GOTO 180
IX 230 END
 M 250 SOUND 0,100,10,15:FOR X=1
   TO 20:NEXT X
IK 260 SOUND 0,0,0,0:RETURN
```

Proszę zwrócić uwagę na linie 130 i 150. Pierwsza z nich umożliwia zapisanie obrazu w obszarze pamięci, który nie jest wyświetlany, tak jakby to był obszar pamięci ekranu. W taki sposób możemy oszukać system operacyjny i przygotować jeden lub więcej obrazów, które później wyświetlimy na ekranie.

Przygotowując ten artykuł korzystaliśmy z opisów i programów zamieszczonych w: ***De Re Atari, Second book of Atari*** *oraz* ***Second book of Atari graphics.***

Tomasz MROWIEC
Ludwik PIELA

Uwaga!

Dwa znaki z lewej strony numeru linii nie są częścią programu, lecz sumą kontrolną linii dla Edytora BASIC-u („IKS" nr 4 z 1987 r.). Nie wprowadzamy ich do komputera.

Występujące w programach podkreślone znaki należy wprowadzić w negatywie.

Czy komputery lubią kolejki?

Kolejki już od dawna wrosły w naszą rzeczywistość. Nie można wyobrazić sobie pracy jakiejkolwiek kasy bez kolejek.

Postawmy nasz komputer np. w budce, w której sprzedają lody. Przychodząc wrzucasz do automatu pieniądze i wtedy możesz wpisać się do kolejki. Ponieważ w kolejce nie ma wielu klientów wystarczą tylko inicjały i np. wiek. Komputer musi wiedzieć, ile można danej osobie sprzedać kulek lodów.

Gdy przyjdzie pora, automat zawoła cię i wyda lody. Oczywiście natychmiast zapomina o tym, że kupowałeś lody. Nawet komputer *„nie ma głowy"* do zapamiętania wszystkich klientów.

Dla zainteresowanych opowiem, jak tworzymy taką kolejkę. Stojąc w kolejce, musimy pamiętać, za kim stoimy. Warto również zapamiętać, kto stoi z tyłu. W zasadzie inne wiadomości nie są ważne. Komputer również tak czyni. Pozostaje jeszcze problem przed kim stoi pierwszy klient i za kim ostatni. W tym celu tworzymy dodatkowego sztucznego klienta (tzw. głowę kolejki), który stoi przed pierwszym, a za ostatnim klientem. Przy tak zorganizowanej kolejce można pozwolić sobie np. na wejście w środek kolejki, choć nie staliśmy w niej od początku. Wystarczy wmówić panu, za którym chcemy stanąć, że staliśmy za nim, a panu, przed którym stajemy, że staliśmy przed nim. Oczywiście nasza kolejka nie przewiduje takich możliwości, lecz możecie w ramach treningu dopisać taką procedurę.

Prezentowany program napisany w języku Pascal operuje na strukturze nie występującej w Basicu. Jest nią REKORD. Rekord jest podobny do wektora (tablicy o jednym wierszu), lecz o ile w wektorze każdy jego element jest jednego typu, np. liczby rzeczywiste, o tyle w rekordzie kolejne jego elementy mogą być różnych typów. W naszym programie inicjały są typu znakowego (char), zaś wiek i liczba kulek typu całkowitego.

```
{ ****************************
  *    PROGRAM KOLEJKA       *
  *         autor:           *
  *   Cezary Dobrowolski     *
  **************************** }

Program kolejka;
label 10,40;
type
  t1=record nazw,imie: char;
            lat,il:integer;
            end;
  t2=array[1..100] of t1;
var
  i,ik:integer;
  kol:t2;
  a:char;
  procedure wejscie(var ik:integer;var kol:t2);
  label 15,20;
  var
    a:char;
    cena,wartosc,ilosc,reszta:integer;
 begin
15:
  if ik=100 then
  begin
   writeln('brak miejsc w kolejce');
   writeln('nacisnij ENTER');readln(a);
   goto 20;
  end;
  cena:=15;
  write('podaj wplate? ');readln(wartosc);
  write('ile chcesz kupic? ');readln(ilosc);
  reszta:=wartosc-cena*ilosc;
  writeln('wyplata reszty',reszta);
  if reszta<0 then begin writeln ('za malo wplaciles');readln(a); goto 20;end;
  ik:=ik+1;
  kol[ik].il:=ilosc;
  write('podaj pierwsza litere nazwiska klienta:  ');readln(kol[ik].nazw);
  write('podaj pierwsza litere imienia klienta:  ');readln(kol[ik].imie);
  write('ilosc lat:  ');readln(kol[ik].lat);
  write('czy jeszcze ktos chce kupic lody (T/N)');readln(a);
  if a='t' then goto 15;
20:
  end;
procedure obsluga (var ik:integer; var kol:t2);
label 30;
 var i:integer;
 begin
  if ik=0 then
  begin
   writeln('nie ma klientow do obslugi');
   writeln('nacisnij ENTER');
   goto 30;
  end;
  writeln( obsluga klienta: ',kol[1].nazw,'  ',kol[1].imie,'  ',kol[1].lat);
  for i:=2 to ik do
  begin
   kol[i-1].nazw:=kol[i].nazw;
   kol[i-1].imie:=kol[i].imie;
   kol[i-1].lat:=kol[i].lat;
  end;
  ik:=ik-1;
  writeln('nacisnij enter');readln(a);
30:
 end;
procedure wydruk (ik:integer;kol:t2);
var
 i:integer;
 begin
  for i:=1 to ik do
  begin
   writeln(i,'  ',kol[i].nazw,'  ',kol[i].imie,'  lat',kol[i].lat);
  end;
  write('nacisnij ENTER');readln(a);
 end;
 begin {programu glownego}
  ik:=0;
10:
   for i:=1 to 25 do begin writeln; end;
   writeln('wybiez opcje: ');
   writeln('1-wejscie klienta');
   writeln('2-obsluga klienta ');
   writeln('3-lista klientow w kolejce');
   writeln('4-koniec pracy ');
   readln(i);
   if i=1 then wejscie(ik,kol);
   if i=2 then obsluga(ik,kol);
   if i=3 then wydruk(ik,kol);
   if i=4 then begin writeln('Koniec pracy przepraszamy '); goto 40; end;
   goto 10;
40:
 end.
```

„Lilavati" — Okopy (1)

Mieczysław SKONIECZNY

Większości Czytelników nie trzeba tłumaczyć czym jest „Lilavati". Są to interesujące rozrywki matematyczne zebrane w ciekawą całość przez Szczepana Jeleńskiego. Jak zaznacza autor w przedmowie, celem tej publikacji jest *„...służyć przez dłuższy czas i wielokrotnie za podręcznik dostarczający Czytelnikowi okazji do pożytecznej rozrywki w chwilach wolnych, spędzanych czy to samotnie, czy też w gronie przyjaciół i znajomych"*. Chcę zaproponować, aby w tej zabawie wziął udział także mikrokomputer. Artykuł ten rozpocznie cykl poświęcony rozwiązywaniu anegdot matematycznych zaczerpniętych w „Lilavati" z wykorzystaniem mikrokomputera.

Ze względu na charakter czasopisma na początku proponuję anegdotę prezentowaną przez S. Jeleńskiego pod numerem 55 o nazwie „W okopach".

Do okopu na froncie weszło 8 żołnierzy i oficer, który zajął stanowisko na lewym skrzydle. Z pewnych względów oficer musi znaleźć się na prawym skrzydle, co nie jest proste, gdyż okop jest ciasny, co uniemożliwia przesuwanie się poza żołnierzami. Na zewnątrz nie można nawet wysunąć głowy, bo nieprzyjaciel prowadzi ciągły ogień na nasze pozycje. Ale w okopie są trzy wgłębienia, w których może chwilowo stanąć jeden żołnierz. Problem do rozwiązania jest następujący: jak powinni poruszać się żołnierze, aby oficer przeszedł wzdłuż całego okopu i stanął na prawym skrzydle?

Charakterystyka programu

Wersja źródłowa programu, opracowanego w BASIC-u, jest następująca:

W instrukcjach 12—16 są wypełniane wartościami początkowymi 2 pomocnicze tablice 13-elementowe:

a) td — zajętość określonej pozycji
   td(i)=0 — i-ta pozycja wolna
   td(i)=j — żołnierz nr j stoi na pozycji i.

b) w — współrzędne znakowe x dla rysowania głowy żołnierza w i-tej pozycji.

Program pracuje w 2 wariantach:

1) Samodzielne poszukiwanie rozwiązania problemu.
2) Prezentacja rozwiązania modelowego (75 przesunięć).

Program w instrukcjach 20—180 zapoznaje użytkownika z problemem i wybiera żądany wariant pracy.

Instrukcje 300—499 to realizacja wariantu 1. Po zapoznaniu z zasadami gry, użytkownik samodzielnie rozwiązuje anegdotę, wprowadzając przed każdym posunięciem numer przesuwanego żołnierza (1—9) oraz numer pozycji, na której ma stanąć żołnierz (1—3).

Instrukcje 460—480 badają czy ruch jest możliwy. Informacje o każdym przesunięciu są wyświetlane w górnym okienku ekranu.

Instrukcja 500—550 to realizacja wariantu 2. Parametry przesunięcia są pobierane z DATA i automatycznie wykonuje się ruch. W ten sposób odbywa się 75 przesunięć.

```
5 DIM td(13):DIM w(13)
10 kod=0:MODE 1:INK 0,26:INK 1,0:INK 2,9:INK 3,6:PAPER 2:BORDER 9:PEN 1
11 WINDOW £1,1,40,1,6:PAPER £1,0:CLS £1
12 CLS:RESTORE 12:FOR i=1 TO 13:READ w(i):NEXT i
13 DATA 3,7,11,15,19,23,27,31,35,39,15,23,31
15 td(1)=0:td(10)=1:FOR i=2 TO 9:td(i)=i:NEXT i
16 FOR i=11 TO 13:td(i)=0:NEXT i
20 GOSUB 9000:PRINT"Do okopu na froncie, w styczno7ci z nie-przyjacielem, weszlo 8 9olnierzy (kolor
zielony, numery 2-9) i oficer (k
olor    czerwony, numer 1).":GOSUB 1000
80 PAPER 2 :LOCATE 1,5:PRINT"Oficer musi z pewnych wzgl[d5w znale;^  si[ na prawym skrzydle. Okop je
st ciasnynie mo9e wi[c przesuni^
 si[ poza 9olnie-rzami.Na zewn!trz nie mo9na nawet wysu-^
90 PRINT"ni^ glowy, bo nieprzyjaciel prowadzi    ci!gly ogie3. Ale w okopie s! trzy wg![bienia, w k
t5rych mo9e chwilowo stan!^ 19ol
nierz."
100 PEN 0:LOCATE 1,25:PRINT"NACIBNIJ DOWOLNY KLAWISZ"
110 IF INKEY$="" GOTO 110
120 LOCATE 10,12:FOR I=1 TO 541:PRINT" ";:NEXT I
130 PEN 1:LOCATE 1,13:PRINT"Jak powinni porusza^ si[ 9olnierze, aby oficer przeszedl wzdlu9 calego o
kopu i  stanll na prawym skrzydl
e?"
140 LOCATE 1,18:PRINT"Wybierz jeden z wariant5w:                    1.Sam szukasz rozwi!zania problemu.
   2.Chcesz obejrze^ rozwi!zani
e modelowe    (75 przesuni[^)."
150 LOCATE 1,23:INPUT"Podaj kod wariantu (1 lub 2):",kod
160 IF kod=1 GOTO 300
170 IF kod=2 GOTO 500
180 LOCATE 1,23:PRINT"Blld - zly kod wariantu          ":FOR i=1 TO 1000:NEXT i:GOTO 150
300 CLS:PRINT"Przesuni[cia dokonujesz podaj!c!          -nr przesuwanego 9olnierza!                -nr p
ozycji, na kt5rej ma stan!^."
310 PRINT"Po podaniu ka9dego numeru nale9y wcisni^klawisz ENTER. W bialym oknie na g5rze  ekranu s!
wv7wietlane informacje doty-  cz
!ce ka9dego przesuni[cia."
320 PRINT:PRINT"Gra ko3czy si[ sukcesem po przesuni[ciu dow5dcy (kolor czerwony - nr1) na pozy- cj[
nr 1(lewy skraj okopu) i ustawie
niu 9olnierzy w poczltkowym szyku tzn. 9ol- nierz nr2 na pozycji nr2,...,9olnierz   nr9 na pozycji n
r9.
330 PRINT:PRINT"!olnierze maj! swoje numery wypisane na piersi, natomiast numery pozycji s! wi- docz
ne na zewn!trz okopu.":PRINT
340 PRINT"!yczymy powodzenia w ilo7ci ruch5w      mniejszej ni9 75 (rozwi!zanie modelowe).
350 PEN 0:LOCATE 1,25:PRINT"NACIBNIJ DOWOLNY KLAWISZ"
360 IF INKEY$="" GOTO 360
370 CLS:GOSUB 1000:ruch=1
380 PAPER 2:LOCATE 1,8:INPUT "Podaj nr przesuwanego 9olnierza: ",nr2
390 IF nr2>0 AND nr2<10 GOTO 420
400 LOCATE 1,8:PRINT"Blld - podano zly numer 9olnierza    "
410 FOR t=1 TO 1000:NEXT t:GOTO 380
420 LOCATE 1,9:INPUT "Podaj nr pozycji na,kt5rej ma stan!^:",nr1
430 IF nr1>0 AND nr1<14 GOTO 455
440 LOCATE 1,9:PRINT"Blld - podano zly numer pozycji.      "
450 FOR t=1 TO 1000:NEXT t:GOTO 420
455 CLS £1:IF td(nr1)=0 GOTO 460
456 LOCATE £1,13,4:PEN £1,3:PRINT £1,"POZYCJA ZAJOTA"
457 FOR t=1 TO 1500:NEXT t:CLS £1
459 LOCATE 1,8:FOR i=1 TO 80:PRINT" ";:NEXT i:GOTO 380
460 FOR i=1 TO 13:IF td(i)=nr2 GOTO 462
461 NEXT i
462 IF nr1=1 AND i=2 GOTO 490
463 FOR j=2 TO 9
464 IF nr1=j AND i=j-1 GOTO 490
465 IF nr1=j AND i=j+1 GOTO 490
466 NEXT j
467 IF nr1=10 AND i=9 GOTO 490
468 IF nr1=4 AND i=11 GOTO 490
469 IF nr1=11 AND i=4 GOTO 490
470 IF nr1=12 AND j=6 GOTO 490
471 IF nr1=6 AND i=12 GOTO 490

472 IF nr1=8 AND i=13 GOTO 490
473 IF nr1=13 AND i=8 GOTO 490
480 LOCATE £1,11,4:PEN £1,3:PRINT £1,"RUCH NIEDOZWOLONY":GOTO 457
490 GOSUB 5000:PAPER 2:ruch=ruch+1
495 IF td(1)<>1 GOTO 459
496 FOR p=2 TO 9:IF td(p)<>p GOTO 459
497 NEXT p
499 LOCATE 1,8:FOR p=1 TO 80:PRINT" ";:NEXT p:GOTO 600
500 CLS:GOSUB 1000:FOR ruch=1 TO 75
505 PEN 0:PAPER 2:LOCATE 14,8:PRINT"GRA POKAZOWA"
510 READ nr1,nr2
520 DATA 1,2,2,3,3,4,4,5,11,5,5,6,4,6,6,7,5,7,7,8,6,8,8,9,7,9,9,1,8,1,13,1,8,9,9,9,10,9,7,8,8,8,9,8,
8,1,7,1,6,1,12,1,6,7,7,7,8,7,13,
7,5,6,6,6,7,6,8,6,4,5,5,5,6,5,7,5
530 DATA 6,1,5,1,4,1,11,1,4,4,5,4,6,4,12,4,3,3,4,3,5,3,6,3,2,2,3,2,4,2,5,2,4,1,3,1,2,1,1,1,4,2,3,2,2
,2,5,3,4,3,3,3,6,4,5,4,4,4,6,5,5
,5,7,6,6,6,8,7,7,7,8,8,9,9
540 GOSUB 5000
550 NEXT ruch
600 PEN 0:LOCATE 12,8:PRINT"OSI)GN)2EB SUKCES"
610 FOR i=0 TO 26:FOR t=1 TO 100:NEXT t:BORDER i:NEXT i
620 GOTO 10
```

Charakterystyka podprogramów

Poszczególne czynności wykonywane w trakcie realizacji programu są zorganizowane w postaci następujących podprogramów:

1) rysowanie okopu na ekranie i początkowego rozmieszczenia żołnierzy (instrukcje 1000—1170).

```
1000 PEN 1:PAPER 2:FOR i=1 TO 40:LOCATE i,24:PRINT CHR$(131);:NEXT i
1020 PAPER 0: FOR i=1 TO 37 STEP 4
1030 LOCATE i,18:PRINT CHR$(135);CHR$(131);CHR$(131);CHR$(131);
1040 FOR j=19 TO 23
1050 LOCATE i,j:PRINT CHR$(133);"   ":NEXT j:NEXT i
1060 FOR i=13 TO 29 STEP 8
1070 PAPER 0:LOCATE i,12:PRINT CHR$(135);CHR$(131);CHR$(131);CHR$(131);:PAPER 2:PRINT CHR$(133)
1080 FOR j=13 TO 17
1090 PAPER 0:LOCATE i,j:PRINT CHR$(133);"   ";:PAPER 2:PRINT CHR$(133):NEXT j:NEXT i
1100 a=2:y=19:FOR x=39 TO 7 STEP -4
1105 FOR t=1 TO 1000:NEXT t
1110 GOSUB 2000:NEXT x
1120 a=3:x=39:GOSUB 2000
1130 PEN 1: FOR i=2 TO 9:PAPER 2:LOCATE 4*i-1,20:PRINT USING "£";i;:NEXT i
1140 PAPER 3:LOCATE 39,20:PRINT"1"
1145 IF kod=0 GOTO 1170
1150 PAPER 2:PEN 1:FOR i=1 TO 10:LOCATE 4*i-2,25:PRINT USING "££";i;:NEXT i
1160 FOR i=0 TO 2:LOCATE 14+i*8,11:PRINT USING "££";11+i;:NEXT i
1170 RETURN
```

2) Rysowanie jednego żołnierza w wybranej pozycji okopu (instrukcje 2000—2100).

Przed wywołaniem podprogramu należy nadać wartość następującym parametrom:

a — kolor żołnierza (a==2 kolor zielony, a=3 kolor czerwony — dowódca)

x, y — współrzędne znakowe głowy żołnierza.

```
2000 PAPER 0:PEN a:LOCATE x,y:PRINT CHR$(190)
2010 LOCATE x-1,y+1:PRINT CHR$(191);CHR$(143);CHR$(192)
2020 LOCATE x-1,y+2:PRINT CHR$(193);CHR$(143);CHR$(194)
2030 LOCATE x-1,y+3:PRINT CHR$(195);CHR$(196);CHR$(197)
2040 LOCATE x-1,y+4:PRINT CHR$(198);CHR$(199);CHR$(200)
2050 PLOT (x-1)*16+3,(26-y)*16-10:DRAWR 0,-4:DRAWR 2,-2:DRAWR 6,0:DRAWR 2,2:DRAWR 0,4
2060 PLOT (x-1)*16+1,(26-y)*16-40:DRAWR 14,0:DRAWR 0,-2:DRAWR -14,0
2070 PLOT (x-2)*16+1,(26-y)*16-34:DRAWR 0,-36:DRAWR 2,0:DRAWR 0,10:DRAWR 0,-20:DRAWR 2,0:DRAWR 0,12:
PLOTR 2,-6:DRAWR 0,-8
2100 RETURN
```

3) Wpisanie w okno informacji o przesunięciu (instrukcje 3000—3030)

Parametry:

ruch — numer kolejnego przesunięcia;

nr 1 — numer pozycji, na której ma stanąć żołnierz;

nr 2 — numer przesuwanego żołnierza;

```
3000 CLS £1:PEN £1,3:LOCATE £1,2,2:PRINT £1,"Przesunięcie nr:";ruch
3020 LOCATE £1,2,4:PEN £1,1:PRINT £1,"Przesuwam żołnierza nr:";nr2:PRINT £1," na pozycję nr
:";nr1
3030 RETURN
```

4) Opróżnienie pozycji zajmowanej do tej pory przez żołnierza (instrukcje 4000—4020).

Parametry:

par, py — współrzędne znakowe pola, w którym była głowa żołnierza

```
4000 PAPER 0:LOCATE par,py:PRINT" "
4005 FOR m=1 TO 4
4010 LOCATE par-1,py+m:PRINT"   ":NEXT m
4020 RETURN
```

5) Realizacja i przesunięcie na ekranie (instrukcje 5000—5110)

```
5000 FOR i=1 TO 13
5010 IF td(i)=nr2 GOTO 5030
5020 NEXT i
5030 GOSUB 3000
5040 par=w(i):py=19:IF i>10 THEN py=13
5050 GOSUB 4000
5060 a=2:IF nr2=1 THEN a=3
5070 y=19:IF nr1>10 THEN y=13
5080 x=w(nr1):GOSUB 2000
5085 PAPER 2:IF nr2=1 THEN PAPER 3
5086 PEN 1:LOCATE x,y+1:PRINT USING "£";nr2
5090 IF kod=2 THEN FOR t=1 TO 1500:NEXT t
5100 td(nr1)=nr2:td(i)=0
5110 RETURN
```

6) Definicje własnych symboli graficznych (instrukcje 9000—9980)

```
9000 SYMBOL AFTER 170
9010 SYMBOL 171,0,0,120,12,124,204,118,3
9020 SYMBOL 172,24,60,102,102,126,102,102,3
9030 SYMBOL 173,24,0,60,102,96,102,60,0
9040 SYMBOL 174,24,60,102,192,192,102,60,0
9050 SYMBOL 175,0,0,60,102,126,96,60,6
9060 SYMBOL 176,254,98,104,120,104,98,254,3
9070 SYMBOL 177,56,24,28,24,56,24,60,0
9080 SYMBOL 178,240,96,112,96,226,102,254,0
9090 SYMBOL 179,24,0,220,102,102,102,102,0
9100 SYMBOL 180,24,198,230,246,222,206,198,0
9110 SYMBOL 181,24,0,60,102,102,102,60,0
9120 SYMBOL 182,12,56,108,198,198,108,56,0
9130 SYMBOL 183,24,0,60,96,60,6,124,0
9140 SYMBOL 184,24,60,96,60,6,102,60,0
9150 SYMBOL 185,24,0,126,76,24,48,126,0
9160 SYMBOL 186,254,198,140,126,48,102,254,0
9170 SYMBOL 187,12,24,126,76,24,48,126,0
9180 SYMBOL 188,48,254,172,24,50,102,254,0
9190 KEY DEF 10,1,55,171,172
9200 KEY DEF 11,1,56,173,174
9210 KEY DEF 3,1,57,175,176
9220 KEY DEF 20,1,52,177,178
9230 KEY DEF 12,1,53,179,180
9240 KEY DEF 4,1,54,181,182
9250 KEY DEF 13,1,49,183,184
9260 KEY DEF 14,1,50,185,186
9270 KEY DEF 5,1,51,187,188
9280 SYMBOL 190,60,126,126,255
9290 SYMBOL 191,0,3,7,15,15,14,14,14
9300 SYMBOL 192,0,192,224,240,240,112,112,112
9310 SYMBOL 193,14,14,14,30,62,60,120,112
9320 SYMBOL 194,112,112,112,112,112,112,112,112
9330 SYMBOL 195,112,96:SYMBOL 196,255,255,231,231,231,231,231,231
9340 SYMBOL 197,112,112,112:SYMBOL 198,0,0,0,0,0,1,7,7
9350 SYMBOL 199,231,231,231,231,231,231,231,231:SYMBOL 200,0,0,0,0,0,128,224,224
9980 RETURN
```

Symbole o numerach 171—188 to definicje znaków charakterystycznych dla języka polskiego (ą, ę, ć, ś).

Litery małe otrzymujemy przytrzymując klawisz SHIFT i wciskając jeden z klawiszy funkcyjnych (f0—f9).

Litery duże otrzymujemy analogicznie przytrzymując klawisz CONTROL.

Symbole 190—200 to części składowe rysunku postaci żołnierza.

W trakcie samodzielnego szukania rozwiązania (wariant 1) należy unikać 2 sytuacji sygnalizowanych przez program:

POZYCJA ZAJĘTA — nie należy próbować przesuwać żołnierza na pozycję, na której stoi inny żołnierz.

RUCH NIEDOZWOLONY — dopuszczalne są tylko ruchy na pozycje sąsiadujące z pozycją zajmowaną przez żołnierza.

Na zakończenie życzę Czytelnikom udanych poszukiwań w liczbie ruchów mniejszych niż 75 (rozwiązanie modelowe proponowane przez autora „LILAVATI").

Komputer kontra miś

— *Mamusiu, dzisiaj jest wtorek* — pyta po przebudzeniu się pięcioletni Karol.

— *Tak syneczku. Dlaczego się ubierasz, przecież jesteś chory i nie idziesz do przedszkola.*

— *Ale ja muszę? Dzisiaj są zajęcia z komputerami.*

Gdy „zaczarowana skrzynka" pojawia się w przedszkolu, takie sceny mają miejsce w niejednym domu. Dzieci momentalnie przestają się ociągać, ba nawet ponaglają rodziców, byleby szybciej znaleźć się wśród rówieśników.

Wanda Kostrzyńska, dyrektorka Przedszkola nr 407 na osiedlu Gocław Lotnisko nie kryje swojego zadowolenia.

— *Dzięki pomocy i uprzejmości rodziców mogliśmy szybciej uruchomić tak bardzo na*

nowym osiedlu potrzebne przedszkole. Rodzice sami wnosili meble, zbijali szafeczki, zawieszali firanki. Również nawiązałam współpracę z kołem ZSMP działającym przy Spółdzielni Mieszkaniowej Gocław-Lotnisko. Zetesempowcy wraz z pracownikami administracji, w ramach czynu społecznego wykonali m.in. uchwyty przy balustradzie, skrzynki do kwiatów. Pojawiają się w każdej wolnej chwili, na każde moje wezwanie, bo niestety drobnych, ale uciążliwych awarii, tzw. niedoróbek nie brakuje.

Dobrze, że chociaż dzieci tymi przyziemnymi sprawami nie muszą się interesować. Od czasu, gdy jeden z rodziców raz w tygodniu właśnie we wtorek przynosi komputer, w grupach pięcio- i sześciolatków nikogo nie brakuje.

— W programie nauczania starszych grup przedszkolnych jeszcze nie uwzględniono zajęć z komputerem. Program jest stary sugerowały wychowawczynie, nie nastawiony na żadne nowinki. A przecież dzieci są tak chłonne i podatne na nową wiedzę. W mig chwytają wszelkie zawiłości związane z obsługą komputera. Wystarczy im po-

kazać, wytłumaczyć, by bezbłędnie powtórzyły podpatrzone czynności. Co tydzień włączają inną dyskietkę; z inną grą.

— *Co prawda, wolałybyśmy na ekranie monitora obserwować wraz z dziećmi programy ułatwiające m.in. poznanie cyferek czy liter niż gry komputerowe* — dodała pani dyrektor. — *Właściwie, to nie mam co narzekać. Przynajmniej dzieci od podstaw zagłębiają się w abecadło i tajniki komputera...*

— *Czy komputer zastąpi misia Puchatka* — głośno zastanawia się wychowawczyni pani **Jolanta Szewczyk.** — *Uważam, że raczej nie. Przecież nie można go ani przytulić, ani z nim się utożsamiać. Na pewno znajdzie swe miejsce obok misia, a nie zamiast niego. Zaobserwowałam, że niektóre gry wywołują u dzieci agresję. Tak nie powinno być. Może jestem w błędzie, może się nie znam, ale takie mam odczucie. Gdy nadejdzie taki dzień, że komputery zadomowią się na stałe w naszych mieszkaniach, to nie powinny być traktowane, jako wypełniacz czasu.*

Dla mnie przerażający jest fakt, iż przez telewizory emitowane programy tak bardzo absorbują naszych podopiecznych. Przedszkolaki w poniedziałki nic nie robią tylko komentują poszczególne pozycje telewizyjne, nawet filmy wyświetlane późnym wieczorem...

Jednak co by się powiedziało, to wtorek w przedszkolu jest wielkim świętem. Postępu nie da się zatrzymać. Dobrze się dzieje, że dzieci nie zostają w tyle za nowinkami techniki, że zaczarowana skrzynka traci magię, a staje się wspaniałym towarzyszem intelektualnej zabawy.

(a)

Giełda pomysłów

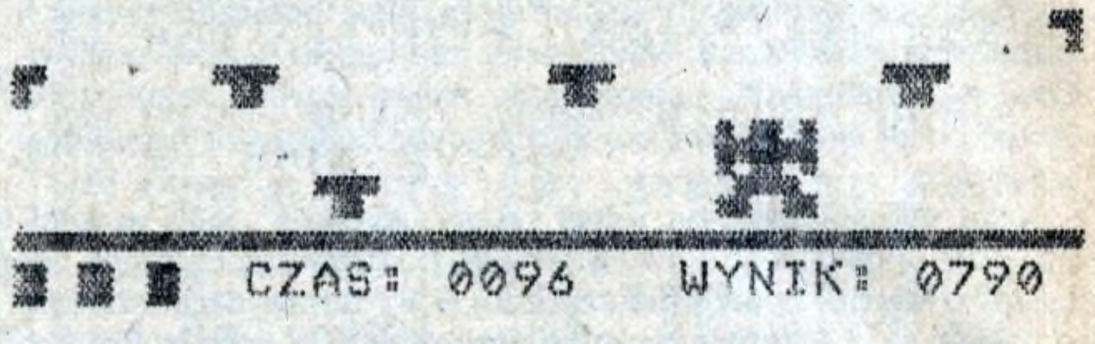
MERITUM „Nietoperze”

Program „Nietoperze” to prosta gra zręcznościowa. Zabawa polega na zestrzeleniu jak największej ilości latających obiektów przypominających nietoperze. Czas gry jest ograniczony, a najlepsze wyniki przedstawia lista rekordów. Na wydruku kody programu maszynowego podzielone są na bloki, co pozwala na stopniowe wpisywanie programu. Każdy blok posiada swoją sumę kontrolną, która wyklucza możliwość uruchomienia programu z błędnie wprowadzonymi wartościami. Linie 6 i 7 należy wpisać dopiero po uruchomieniu programu, gdyż blokują one klawisze BREAK i NMI.

Piotr BIEDRZYCKI

```
3       CLEAR 200  : GOSUB 3000
4       POKE 16396,175: POKE 16397,201
5       DIM R(5), R$(5): A=30000
6       OUT 250,0 'ZABLOKOWANIE NMI
7       POKE 16526,0: POKE 16527,100
9     '-----------------------------------
10      GOSUB  1000: GOSUB 2000
20      U=USR(0)
25      NW=PEEK(A)+100*PEEK(A+1)
30      GOSUB 620  : GOTO    10
35    '-----------------------------------
619     'SPRAWDZANIE WYNIKU
620       FOR H=1 TO 5
625       IF NW ) R(H) THEN 655
630       NEXT H
635     RETURN
654   'WYNIK JEDEN Z PIECIU NAJLEPSZYCH
655     FOR J=5 TO H STEP -1
660     R(J)=R(J-1): R$(J)=R$(J-1)
665     NEXT J
668     CLS: PRINT CHR$(23)
670     PRINT@ 512,"   TWOJE IMIE";
672     INPUT I$
673     I$=I$+"      "
675     R(H)=NW      : R$(H)=LEFT$(I$,6)
680   RETURN
999   'PIERWSZY EKRAN
1000    CLS: PRINT CHR$(23)
1005    PRINT: PRINT"          NIETOPERZE"
1010    PRINT: PRINT
1015    PRINT"     '1'-LEWO","PRAWO-':'"
1020    PRINT"       'SHIFT'-STRZAL"
1025    PRINT: PRINT"        LISTA REKORDOW"
1026    PRINT: FOR H=1 TO 5
1030    PRINT "    ";H;"  - ";R$(H);R(H)
1035    NEXT H
1036    A$=INKEY$ 'KASOWANIE KODU OSTATNIEGO KL.
1037    IF INKEY$="" THEN 1037
1040  RETURN
1999  'PLANSZA GRY
2000    CLS: PRINT CHR$(23)
2005    PRINT@ 960,CHR$(143);" ";
2010    PRINT CHR$(143);" ";CHR$(143);
2015    PRINT "  CZAS: 0000   WYNIK: 0000";
2020    FOR H=0 TO 127: SET(H,43):NEXT
2025  RETURN
2888  'KOD WEWNETRZNY ZAJMUJE OBSZAR
2889  'OD 6400H (25600)   DO 677FH (26495)
2890  'ADRES STARTOWY 6400H (25600)
2999      'LADOWANIE KODU MASZYNOWEGO
3000        S=0:A=25600:P=7:GOSUB 4000  'BLOK 1
3005 DATA E5,D5,C5,21,00,00,22,30,75,21,44,75,22,3A,75,21
3010 DATA 30,05,22,32,75,21,03,00,22,36,75,21,04,04,22,38
3015 DATA 75,21,60,3F,22,34,75,CD,2E,64,C1,D1,E1,C9,ED,73
3020 DATA 3E,75,3E,02,32,AF,40,3A,39,75,F5,21,7F,02,CD,9A
3025 DATA 0A,CD,C9,14,CD,37,0B,2A,21,41,11,00,3C,CB,85,A7
3030 DATA 19,EB,2A,3A,75,73,23,72,23,22,3A,75,F1,3D,20,DA
3035 DATA 21,44,75,22,3A,75,CD,DD,66,CD,04,65,3A,40,3B,CB
3040      IF S()10139  THEN PRINT "BLAD W BLOKU NR 1":END
3050        S=0: A=Z   :P=8:GOSUB 4000  'BLOK 2
3055 DATA 57,20,F9,CD,FD,66,21,38,75,3A,39,75,96,3C,06,A0
3060 DATA 10,FE,3D,20,F9,CD,04,65,CD,DE,65,06,4A,10,FE,10
3065 DATA FE,3D,20,F7,CD,46,67,CD,65,65,18,CD,3A,3B,75,F5
3070 DATA 3E,02,32,AF,40,21,0C,00,CD,9A,0A,CD,C9,14,CD,37
3075 DATA 0B,11,3A,00,2A,21,41,A7,19,CB,85,EB,2A,3A,75,4E
3080 DATA 23,46,C5,E1,3E,20,02,03,03,02,19,11,7C,3F,3E,02
3085 DATA 32,AF,40,CD,39,0A,FE,01,28,22,EB,2A,3A,75,73,23
3090 DATA 72,23,22,3A,75,3E,AC,D3,FE,12,13,13,3E,9C,12,D3
3095      IF S()12355  THEN PRINT "BLAD W BLOKU NR 2":END
3100        S=0: A=Z   :P=8:GOSUB 4000  'BLOK 3
3105 DATA FE,F1,3D,20,AA,21,44,75,22,3A,75,C9,11,7C,03,A7
3110 DATA ED,52,18,D6,2A,34,75,3A,10,38,CB,4F,28,0E,2B,2B
3115 DATA 11,42,3F,CD,39,0A,FE,FF,28,17,18,18,3A,20,38,CB
3120 DATA 57,28,0E,23,23,11,78,3F,CD,39,0A,FE,01,28,02,18
3125 DATA 03,2A,34,75,11,40,00,22,34,75,36,B9,A7,ED,52,36
3130 DATA B4,23,23,36,BC,23,23,36,B8,19,36,B6,2B,2B,36,3F
3135 DATA 2A,34,75,01,08,00,2B,2B,36,20,ED,52,36,20,09,36
3140 DATA 20,19,36,20,C9,2A,34,75,01,04,00,11,40,00,3E,B9
3145      IF S()9777   THEN PRINT "BLAD W BLOKU NR 3":END
3150        S=0: A=Z   :P=8:GOSUB 4000  'BLOK 4
3155 DATA BE,20,14,ED,52,3E,B4,BE,20,0D,09,3E,B8,BE,20,07
3160 DATA 19,3E,B6,BE,20,01,C9,E5,CD,33,66,E1,CD,04,65,21
3165 DATA 00,00,01,FE,C0,C5,ED,B3,C1,10,FA,3A,36,75,3D,32
3170 DATA 36,75,FE,02,20,09,3E,20,32,C0,3F,D1,C3,69,64,FE
3175 DATA 01,20,09,3E,20,32,C4,3F,D1,C3,69,64,FE,00,20,09
3180 DATA 3E,20,32,C8,3F,D1,C3,69,64,16,C8,CD,ED,66,CD,ED
3185 DATA 66,16,14,CD,ED,66,CD,ED,66,ED,7B,3E,75,C9,3A,80
3190 DATA 38,CB,47,20,03,3E,32,C9,3E,0C,2A,34,75,11,40,00
3195      IF S()13734  THEN PRINT "BLAD W BLOKU NR 4":END
3200        S=0: A=Z   :P=8:GOSUB 4000  'BLOK 5
3205 DATA ED,52,22,3C,75,F5,2A,3C,75,11,40,00,ED,52,11,04
3210 DATA 00,22,3C,75,3E,20,BE,C4,33,66,36,95,19,BE,C4,33
3215 DATA 66,36,AA,F1,F5,07,0E,5A,D3,FE,47,10,FE,3D,D3,FE
3220 DATA 3C,0D,20,F4,2A,3C,75,36,20,19,36,20,F1,3D,20,C5
3225 DATA 3E,03,C9,3E,9C,E5,D5,BE,20,02,2B,2B,EB,3A,38,75
3230 DATA F5,2A,3A,75,4E,23,46,23,22,3A,75,C5,E1,CD,39,0A
3235 DATA A7,28,0D,F1,3D,20,E9,21,44,75,22,3A,75,D1,E1,C9
3240 DATA 36,20,23,23,36,20,2A,3A,75,E5,D1,1B,1B,01,00,03
3245      IF S()12564  THEN PRINT "BLAD W BLOKU NR 5":END
3250        S=0: A=Z   :P=8:GOSUB 4000  'BLOK 6
3255 DATA ED,B0,CD,BB,66,2A,3C,75,36,20,11,40,00,19,36,20
```

```
3260 DATA 3A,38,75,3D,28,1D,32,38,75,3E,48,21,00,11,0E,FE
3265 DATA 57,5F,ED,69,43,10,FE,ED,61,15,20,F5,D6,09,20,F0
3270 DATA F1,18,B4,3A,39,75,3C,3C,32,38,75,21,44,75,22,3A
3275 DATA 75,32,39,75,ED,7B,3E,75,C3,2E,64,3A,30,75,C6,05
3280 DATA 32,30,75,FE,64,C0,AF,32,30,75,3A,31,75,3C,32,31
3285 DATA 75,FE,14,C0,3D,32,31,75,ED,7B,3E,75,C9,16,14,CD
3290 DATA ED,66,16,B4,CD,ED,66,16,14,CD,ED,66,C9,0E,C8,D3
3295      IF S()13083  THEN PRINT "BLAD W BLOKU NR 6":END
3300         S=0: A=Z   :P=9:GOSUB 4000  'BLOK 7
3305 DATA FE,42,10,FE,10,FE,3C,D3,FE,0D,20,F3,C9,CD,33,67
3310 DATA 11,32,75,21,E0,3F,CD,0F,67,11,30,75,21,FC,3F,06
3315 DATA 02,CD,1A,67,13,2B,2B,10,F8,C9,1A,0E,FF,CD,2A,67
3320 DATA F6,30,77,2B,2B,79,F6,30,77,C9,0C,D6,0A,F2,2A,67
3325 DATA C6,0A,C9,21,32,75,35,C0,36,63,23,35,3E,FF,BE,C0
3330 DATA CD,DD,66,C3,C9,65,C3,76,67,3A,40,75,FE,01,28,13
3335 DATA 3E,01,32,40,75,3E,AD,32,E6,64,3E,9E,32,ED,64,32
3340 DATA 34,66,C9,3E,02,32,40,75,3E,AC,32,E6,64,3E,9C,32
3345 DATA ED,64,32,34,66,C9,CD,49,67,CD,9C,64,C9,00,00,00
3350      IF S()15675  THEN PRINT "BLAD W BLOKU NR 7":END
3800    RETURN   'KONIEC DANYCH
3999        'PODPROGRAM LADUJACY
4000        FOR Z=A  TO  A+P*16-1
4005        READ A$:PRINTA$;" ";
4010        N=ASC( LEFT$(A$,1))-48
4015        M=ASC(RIGHT$(A$,1))-48
4020        N=N-INT(N/12)*7
4025        M=M-INT(M/12)*7
4030        B=N*16+M: S=S+B
4035        POKE Z,B : NEXT Z
4040      RETURN
```



AMSTRAD „Generator”

Ten krótki program umożliwia wyświetlenie na ekranie kolejnych bajtów zadanego obszaru pamięci (np. kodu maszynowego własnego podprogramu) w postaci kolejnych (o ustalonej przez siebie numeracji i odstępie) wierszy DATA oraz zeskładowanie go na dyskietce (kasecie) w określonym, również przez siebie, zbiorze.
Otrzymany przez siebie zbiór można załadować do pamięci komendą LOAD i dokonywać poprawek używając edytora. Można go również dołączać do własnego programu w BASIC-u komendą MERGE.

Bajty przedstawione są w postaci heksadecymalnej (linia 230), a chcąc przedstawić je np. dziesiętnie, należy linię tę zmienić na: a$ = PEEK (x).

Ilość liczb w jednym wierszu DATA ustalona jest na 10 (zmienna c) i chcąc ją zmienić należy poprawić linię 260.

Cezary SOBCZAK

```
10 REM ****************
20 REM Generator
30 REM wierszy 'DATA'
40 REM ****************
50 MEMORY &2FFF
60 MODE 2
70 LOCATE 20,1:WINDOW #
1,40,80,25,25
80 PRINT CHR$(24)"G E N
 E R A T O R   WIERSZY
  'DATA'"CHR$(24)
90 PRINT:PRINT:PRINT
100 CLEAR INPUT
110 LOCATE 1,5: INPUT "
Podaj poczatkowy adres
pamieci ( > &2fff ) ",x
120 IF x<&2FFF THEN PRI
NT #1,"Musi byc WIEKSZY
 niz &2fff":FOR i=1 TO
4000:NEXT:CLS #1:LOCATE
 1,5:PRINT SPACE$(79):G
OTO 110
130 LOCATE 1,6:INPUT "P
odaj koncowy adres pami
eci                  ",y
140 IF y<x THEN  PRINT
#1, "Musi byc WIEKSZY n
iz poczatkowy":FOR i=1
TO 4000:NEXT:CLS #1:LOC
ATE 1,6:PRINT SPACE$(79
):GOTO 130
150 LOCATE 1,7:INPUT "N
umerowac linie 'DATA' o
d                    ",z
160 IF z<1 OR z>=65536
THEN PRINT#1,"Numery mo
ga byc od 1 do 65535":F
OR i=1 TO 4000:NEXT:CLS
 #1:LOCATE 1,7:PRINT SP
ACE$(79):GOTO 150
170 LOCATE 1,8:INPUT "O
dstep miedzy liniami
                     ",dz
180 LOCATE 1,9:INPUT "N
azwa zbioru na kopie li
nii 'DATA'           ",n$
190 CLS
200 WINDOW #1,2,77,4,22
:PRINT "
     G E N E R A T O R
 'WIERSZY  'DATA'"
210 OPENOUT n$
220 FOR x=x TO y
230 a$=HEX$(PEEK(x),2)
240 c=c+1
250 IF LEN(z$)=0 THEN z
$=MID$(STR$(z),2)+" DAT
A "+a$  ELSE z$=z$+","+
a$
260 IF c=10 OR x=y THEN
 PRINT #1, z$:PRINT #9,
z$:z$="":c=0:z=z+dz
270 NEXT
280 CLOSEOUT
290 LOCATE 10,25:PRINT"
K O N I E C
                Kontynuacja
   -   S P A C J A"
300 WHILE INKEY(47)=-1:
WEND:GOTO 60
```

Sprzęg, łącznia, interfejs

Układ elektroniczny łączący urządzenia systemu cyfrowego i umożliwiający współpracę wg określonych zasad nazywany jest interfejsem. Jest to pewien standard techniczny rozwiązania problemu połączenia komunikacyjnego (do przesyłania informacji jedno- lub dwukierunkowo) między np. procesorem a monitorem ekranowym, lub monitorem ekranowym a drukarką.

W celu uniknięcia sytuacji wieży Babel w połączeniach dokonano pewnych światowych ustaleń w zakresie połączeń zwanych popularnie interfejsami. Najczęściej interfejs to wiązka przewodów o określonych parametrach elektrycznych, zakończona dwustronnie łączówkami, umożliwiającymi dołączenie go do dwu urządzeń.

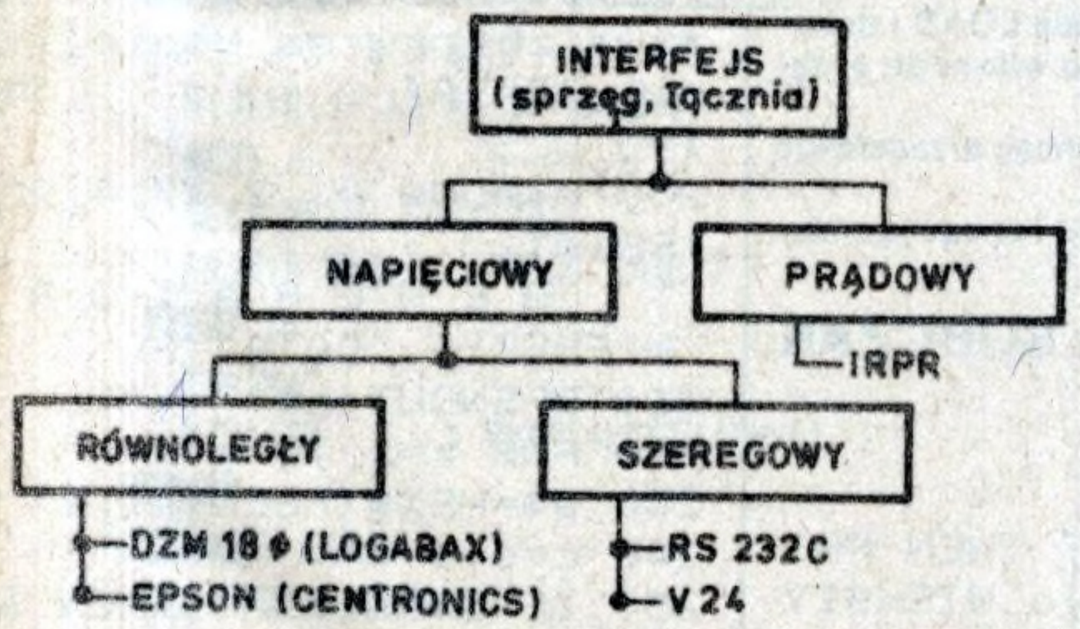
Kilka słów dygresji na temat nazwy „interfejs". Jest to spolszczenie angielskiego słowa **interface**, co oznacza dosłownie połączenie **między twarzami** (urządzeń), a w potocznym żartobliwym przekładzie — interfejs to **„międzymordzie"**. W lingwistycznych zapędach posunęliśmy się tak daleko, że w zamian mamy: sprzęg, łącznia, złącze, łącze, układ pośredniczący, układ sprzęgający, sprzężenie, styk.

Niezależnie od nadmiaru polskich „ersatzów" fachowcy uparcie używają obcego słowa interfejs, które się tak utrwaliło w informatyce, że nawet końmi go nie wyrwie.

Rodzaje interfejsów

W zależności od sygnału przesyłanego liniami między urządzeniami interfejsy dzieli się na prądowe (przesyłane są impulsy prądu elektrycznego) i napięciowe (przesyłane są impulsy napięcia elektrycznego). Interfejsy prądowe są mniej popularne, najbardziej rozpowszechnione są **interfejsy napięciowe**, wśród których z kolei rozróżnia się **szeregowe** i **równoległe** przesyłanie informacji. Interfejsem szeregowym informacje są przesyłane bit po bicie w jedną i w drugą stronę, a równoległym przesyłanych jest 8 bitów naraz, tworzących jeden znak.

Rysunek przedstawia podział interfejsów i nazwy najpopularniejszych ich przedstawicieli.



Najczęściej stosowane są standardy interfejsów napięciowych równoległych typu DZM-180, Centronics i szeregowych typu V 24 i RS 232C.

W tabelach połączeń tych interfejsów używane będą zapisy, do których odnoszą się następujące uwagi:

a) używane jest słownictwo i skróty pochodzące od określeń z języka angielskiego.
b) kreseczka nad nazwą sygnału oznacza jego negację, to znaczy, że działanie tego sygnału zaczyna się w chwili, gdy jego wartość spada do poziomu zera logicznego (Ø) — tak pisane, w odróżnieniu od literki „O" — duże.
c) pin oznacza element łączówki o określonym numerze, do którego dolutowujemy przewód.

Tabelka przedstawia standard interfejsów szeregowych V 24 i Rs 232C, które zasadniczo nie różnią się między sobą, jednakże zostały powołane do życia przez różne organizacje międzynarodowe.

Interfejs V 24 przez CCITT (fr. Comit'e Consultatif International Telegrafique et Telephonique / Międzynarodowy Doradczy Komitet Telegraficzny i Telefoniczny) natomiast RS 232C przez EIA (Electronic Industries Association).

Sygnały i wyprowadzenia interfejsów RS 232C i V 24

RS 232C	V 24	Pin = wyprowadzenie	Oznaczenie mnemoniczne	Nazwa linii
AA	101	1	—	ekran
AB	102	7	GND	masa sygnałowa
BA	103	2	TxD	dane nadawane
BB	104	3	RxD	dane odbierane
CA	105	4	RTS	żądanie nadawania
CB	106	5	CTS	gotowość nadawania
CC	107	6	DSR	gotowość modemu
CD	108.2	20	DTR	żądanie dołącz. modemu do linii
CE	125	22	RI	dzwonek
CF	109	8	DCD	nośna odbierana
CG	110	21	SQD	poziom sygnału odb.
CH	111	23	—	wybór prędk. trans.
DA	113	24	—	zegar transm. nad.
DB	114	15	—	zegar transm. nad.
DD	115	17	—	zegar transm. odb.
SBA	118	15	—	dane nad. — kanał. 2
SBB	119	14	—	dane odb. — kanał. 2
SCA	120	19	—	żąd. nad. — kanał 2
SCB	121	13	—	got. nad. — kanał 2
SCF	122	12	—	nośna odb. — kan. 2

Standard dopuszcza możliwość zredukowania niektórych linii kontrolnych i linii synchronizacji. W minimalnej postaci wystarczą trzy przewody: 1 — linia masy, 2 — dane nadawane, 3 — dane odbierane. Linie sterowane są sygnałami napięciowymi zmieniającymi się w zakresie od —12V do +12V.

Pozostały nam do omówienia dwa interfejsy równoległe. Są to DZM-180 (używa się również nazwy LOGABAX) i CENTRICS (używa się nazwy EPSON). Warto je znać, ponieważ większość mikrokomputerów ma wyjścia umożliwiające podłączenie drukarki przez jeden z tych standardów.

STANDARD DZM-180

Nr pinu	Nr pinu masy	Nazwa sygnału	Opis sygnałów
1	19	$\overline{SE}$	Impuls powodujący wpisanie danych do bufora drukarki
2	20	$\overline{ENT1}$	osiem linii danych, sygnały te reprezentują informację wejściową w postaci 8-bitowej danej równoległej
3	21	$\overline{ENT2}$	
4	22	$\overline{ENT3}$	
5	23	$\overline{ENT4}$	
6	24	$\overline{ENT5}$	
7	25	$\overline{ENT6}$	
8	26	$\overline{ENT7}$	
9	27	$\overline{ENT8}$	
10.	28	$\overline{ACK}$	Potwierdzenie przyjęcia informacji wejściowej
11	29	Nie używany	
12	30	FIN PAP	Brak lub koniec papieru
13	—	RELSEL	Gotowość drukarki do przyjmowania danych
14	—	Nie używany	
15	—	Nie używany	
16	—	Ø V	Ø V logiczne
17	—	GND	Masa obudowy drukarki
18	—	Nie używany	
31	—	Nie używany	
32	—	Nie używany	
33	—	Nie używany	
34	—	Nie używany	
35	—	FIN	Sygnalizacja pustego bufora drukarki
36	—	Nie używany	
37	—	CPLX	Zegar 2,5 μs

Standard CENTRONICS

Nr pinu	Nr pinu masy	Nazwa sygnału	Opis sygnałów
1	19	$\overline{\text{STROBE}}$	Impuls powodujący wpisanie danych do bufora drukarki
2	20	DATA 1	Osiem linii danych
3	21	DATA 2	Reprezentują 8-bitową
4	22	DATA 3	równoległą informację
5	23	DATA 4	wejściową
6	24	DATA 5	
7	25	DATA 6	
8	26	DATA 7	
9	27	DATA 8	
10	28	$\overline{\text{ACK}}$	Potwierdza przyjęcia inform.
11	29	BUSY	Drukarka zajęta
12	30	PAPER END	Brak lub koniec papieru
13		SELECTED	Gotowość drukarki do pracy
14		Nie używany	
15		Nie używany	
16		SIGNAL GND	Masa sygnałowa
17		CHASSIS GND	Uziemienie
18		+5V Dc	Nap. zasil. +5V (ok. 50 mA)
31		$\overline{\text{INPUT PRIMPE}}$	Zerowanie układów drukarki
32		ERROR	Sygnalizacja nieprawidłowości w przesyłanej informacji
33		EXT GND	Masa zewnętrzna
34		Nie używany	
35		Nie używany	
36		—	Poziom niski

Uwaga praktyczna: w celu zmniejszenia podatności na zakłócenia proponuje się, aby każdy przewód sygnałowy został skręcony z przewodem masy sygnałowej oznaczonej numerem pinu masy, który jest połączony elektrycznie z pozostałymi masami.

opr. inż. Edward WRZEŚNIEWSKI

Program stosuje procedurę przeniesienia blokowego dla zapełnienia rysunkiem całego ekranu (24 wiersze!).

W linii 999 wykorzystano możliwości CHR$ Ø — CHR$ 23 dla uzyskania nietypowego wydruku nazwy programu.

```
    1 REM   Program [redacted]
   10 BORDER 5: BEEP .02,14: CLEA
R 63999: FOR n=64000 TO 64011: R
EAD bt: POKE n,bt: NEXT n
   15 DIM a$(64)
   20 FOR n=7 TO 127 STEP 15: PLO
T 127,144: DRAW -n,-97: PLOT 128
,144: DRAW n,-97: NEXT N
   25 PLOT 0,143: DRAW 0,-95: DRA
W 255,0: DRAW 0,95
   30 PRINT #0;AT 0,0; PAPER 7;a$
   40 RANDOMIZE USR 64000
   50 FOR n=1 TO 8: PRINT a$: NEX
T n
   55 PLOT 0,48: DRAW 0,127: DRAW
 255,0: DRAW 0,-127
   60 FOR n=7 TO 127 STEP 15: PLO
T 127,80: DRAW -n,95: PLOT 128,8
0: DRAW n,95: NEXT N
   70 FOR n=7 TO 127 STEP 15: PLO
T 127,80: DRAW -n/3,-32: PLOT 12
8,80: DRAW n/3,-32: NEXT n
   80 FOR n=0 TO 175 STEP 16: PLO
T 127,80: DRAW -127,(n-80): PLOT
 128,80: DRAW 127,(n-80): NEXT n
   90 PAUSE 100: RUN
  100 DATA 33,0,72,17,0,80,1,0,8,
237,176,201
  999 CLEAR : LET b$=CHR$ 22+CHR$
 11+CHR$ 14+CHR$ 19+CHR$ 1+"24x3
2": PRINT b$: SAVE b$ LINE 1
```

PROGRAM 38

Spadająca piłka

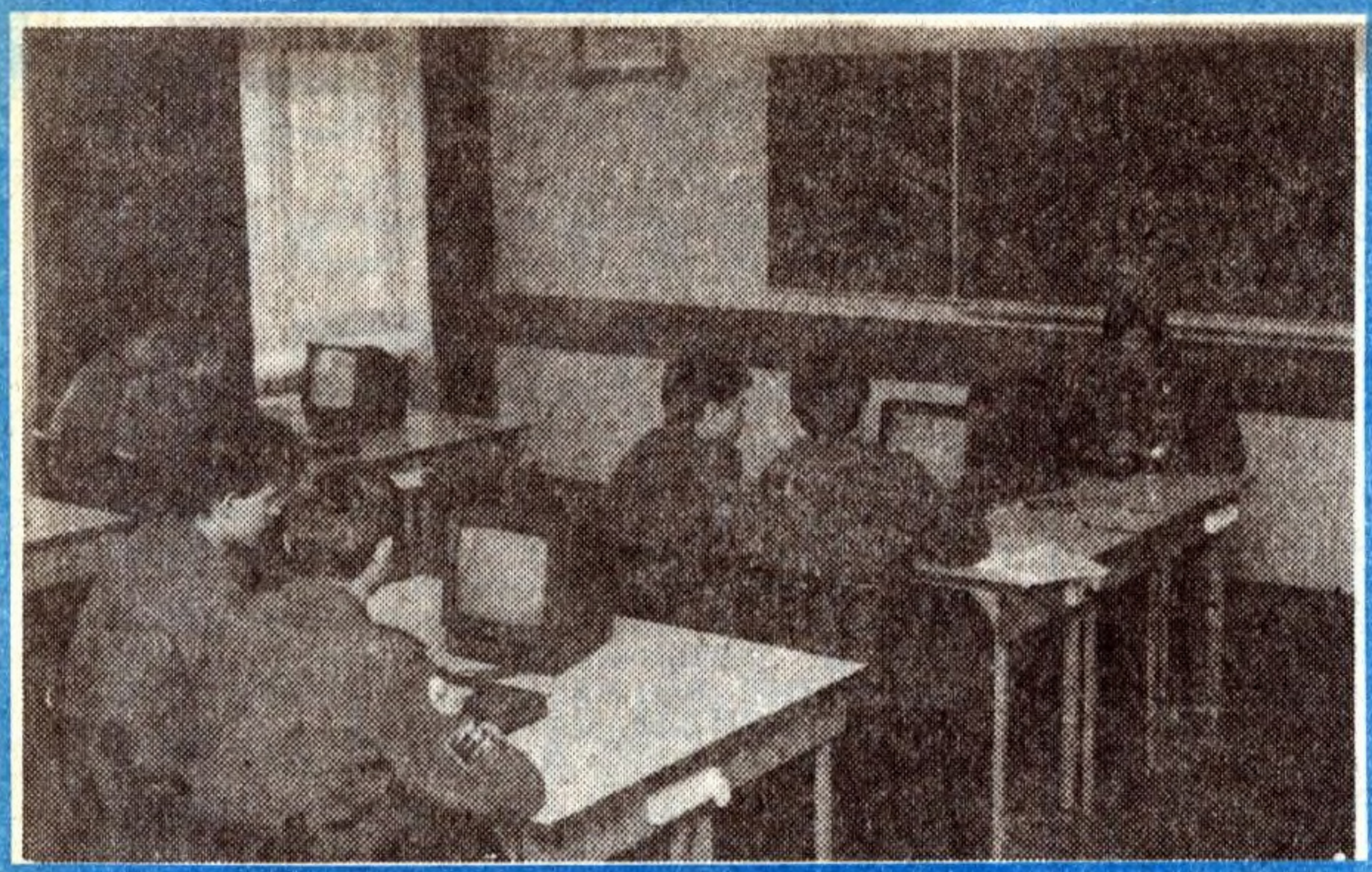
(Gra zręcznościowa)

Gra polega na złapaniu maksymalnej liczby piłek do pojemnika (litera U). Gracz może poruszać wózkiem za pomocą kursorów w lewo bądź w prawo. Piłka zmienia kierunek losowo, gdy natrafi na przeszkodę (litera O). Jest to możliwe dzieki kontroli ekranu przez SCREENS.

Nie zalecam zwalniać piłki za pomocą PAUSE, gdyż ta instrukcja nie jest wykonywana, jeżeli jest wciśnięty dowolny klawisz. Można natomiast wprowadzić jałową pętlę FOR-NEXT.

Krzysztof POŹNIAK
„Hobbyte"

```
    1 REM
*********************************
*                               *
*      Krzysztof  Poznlak       *
*                               *
*    Klub Mikrokomputerowy      *
*         HOBBYTE    © 1987     *
*                               *
*      SPADAJACA  PILKA         *
*                               *
*********************************
   10 PAPER 0: INK 7: BORDER 0: C
LS : FOR k=1 TO 6 STEP 2: FOR l=
0 TO 1: PRINT AT 2*(k+l),l;: FOR
 m=1 TO 15: PRINT "O ";: NEXT m:
 NEXT l: NEXT k
   15 LET p=1: LET zp=0
   20 PRINT #1;AT 0,0;"pilka :";p
 "zlapanych :";zp: LET k=6+INT (
10*RND): LET w=6+INT (10*RND)
   30 FOR l=0 TO 20: LET z=SCREEN
$ (l+1,k)<>" ": LET k1=k+z*SGN (
.5-RND)*(l<20)
   35 LET d=CODE INKEY$: LET w=w+
((d=9) AND (23-w))-((d=8) AND (w
-2)): PRINT AT 21,w;" _U_ "
   40 PRINT AT l+1,k1;"O";AT l,k;
" ": LET k=k1: NEXT l
   45 LET p=p+1: IF k1=w+2 THEN L
ET zp=zp+1
   50 BEEP 0.05,20: BEEP .05,30:
PRINT AT 21,w;"      ";AT 21,k1;"
": GO TO 20
```



W tym roku szkolnym chęć pobierania nauki w Liceum Lotniczym im. Żwirki i Wigury w Dęblinie wyraziło prawie dwa tysiące kandydatów. Po badaniach lekarskich do egzaminu wstępnego przystąpiło tylko 30 procent chętnych.

Dyrektor istniejącej już piętnaście lat szkoły, mjr dr **Janusz Ziółkowski** twierdzi, że Liceum Lotnicze, to zwykłe liceum zawodowe, a wyróżnia je się jedynie tym, że mieści się przy Wyższej Oficerskiej Szkole Lotniczej i od momentu powstania doskonale wywiązuje się z zadania zapewnienia stałego dopływu maturzystów do WOSL.

Na czym więc ta odrębność polega? Liceum kształci w specjalności „mechanik osprzętu lotniczego". W rzeczywistości jednak dęblińskie „orlęta" nie poprzestają na zdobywaniu tylko tych kwalifikacji. Są pilotami szybowców, skoczkami spadochrono-

Skomputery

wymi, znakomitymi modelarzami. Właśnie od tego roku funkcjonuje supernowoczesna modelarnia. A w ubiegłym roku uruchomiono nowoczesne, zautomatyzowane gabinety: elektroniki, osprzętu lotniczego i informatyki. W tym ostatnim gabinecie zatrzymajmy się na chwilę. Oddajmy głos uczniom. Oto kilka charakterystycznych wypowiedzi:

— *Od kilku lat interesuję się komputerami. Mam w domu nawet Atari i chyba sporo już umiem. Nie na tyle jednak, by nie nauczyć się więcej. Tutaj mam fachową opiekę, poznałem nowy typ komputera...*

— *Dla mnie komputery to nowość. Na początku nie byłem do nich przekonany. Myślałem, że będzie to przemijająca moda. Okazało się, że nie. Najpierw zaciekawiły mnie gry komputerowe, teraz uczę się sam układać programy...*

— *Chcę być pilotem i temu podporządkowuję wszystko. Czy możliwe jest, żeby bez*

...zowane „orlęta”

J. RAJCH

odpowiedniego przygotowania pilotować tak skomplikowane maszyny nafaszerowane elektroniką? Na pewno nie. Dlatego cieszę się, że tak wcześnie mamy okazję poznać komputery i pracować na nich. Zresztą jest to bardzo zajmujące i wciąga...

Obecnie Liceum Lotnicze dysponuje 10 mikrokomputerami typu Amstrad.

— *Są one dla nas najbardziej przydatne* — mówi zastępca dyrektora szkoły, mjr **Zenon Ziemkiewicz**. — *Wykorzystywane są przede wszystkim na lekcjach informatyki. Prócz tego używamy ich podczas zajęć informatycznego koła naukowego. Może w nich uczestniczyć każdy uczeń liceum. Wystarczą dobre wyniki w nauce i także chęci. Zajęcia prowadzą fachowcy, oficerowie z tutejszego Ośrodka Obliczeniowego.*

Plany są bogatsze. Kadra liceum stara się o zdobycie nowych mikrokomputerów. Będą one sukcesywnie wykorzystywane do nauki innych przedmiotów, pozwolą na zwiększenie oferty kół zainteresowań. Obecnie do pracy i nauki z komputerami przygotowywani są na kursach nauczyciele różnych zawodów. Może będzie można skomputeryzować i uatrakcyjnić na przykład lekcje języka polskiego?

A na razie uczą się języków, tych komputerowych, i programowania. Programy stosują coraz trudniejsze, o większym stopniu skompilkowania. Przystosowują je do swoich potrzeb. Zbierają fachową literaturę i wiadomości z komputerowego mikroświatka.

Na lekcjach informatyki i zajęciach koła zainteresowań uczniów dęblińskiego Liceum Lotniczego korzystających z komputerów podglądał **Jan Zelman**.

Komputer w dydaktyce

ZX — Spectrum

Komputer jest cennym narzędziem ułatwiającym uczniom poznawanie rzeczywistości. W dydaktyce szczególnie dużą wartość mają takie metody przekazu wiedzy, które wymagają od ucznia jego aktywnej, a nawet twórczej postawy. Dysponując odpowiednim programem komputerowym, uczeń może przeprowadzać różnorodne eksperymenty. Są one tanie, w pełni bezpieczne, niedestrukcyjne, umożliwiają dowolną zmianę skali czasu (przyspieszenie lub zwalnianie procesu) i zmianę nawet tych parametrów, którymi w realnym świecie nie można swobodnie manipulować (np. pogoda, gęstość zaludnienia, popyt na określone towary itp.). Zmieniając w dużym zakresie parametry symulowanego procesu i obserwując uzyskiwane rezultaty, uczeń może sam formułować pewne wnioski dotyczące badanego fragmentu rzeczywistości. Uczeń może też formułować hipotezy i sprawdzać je korzystając z odpowiedniego programu komputerowego. Uzyskana w ten sposób wiedza jest dużo głębsza i trwalsza niż wiedza przekazana metodami podawczymi. Jest ona wynikiem własnych dociekań ucznia i jego aktywnej postawy, stąd też zorientowana jest bardziej problemowo. Nie sposób także pominąć radości i satysfakcji ucznia z samodzielnego odkrywania nowych dla niego reguł, praw czy związków.

Prostym przykładem programu umożliwiającego samodzielne formułowanie wniosków o wybranym fragmencie rzeczywistości jest prezentowany RZUTUKOS (dla ZX Spectrum).

Program ten symuluje rzut ukośny. Zastosowany algorytm, oparty na równaniach rzutu ukośnego, jest bardzo prosty i nie wymaga komentarza. Uczeń podaje dwa parametry:

— kąt wyrzutu,

— prędkość początkową rzutu,

a program kreśli trajektorię lotu pocisku. Na ekranie można uzyskać całą rodzinę trajektorii dla różnych kątów i prędkości początkowych. Trajektorie kreślone są w dyskretnych momentach czasu, a stopień zagęszczenia punktów trajektorii jest odwrotnie proporcjonalny do prędkości chwilowej pocisku (najlepiej jest to widoczne przy rzucie pionowym — kąt = 90°).

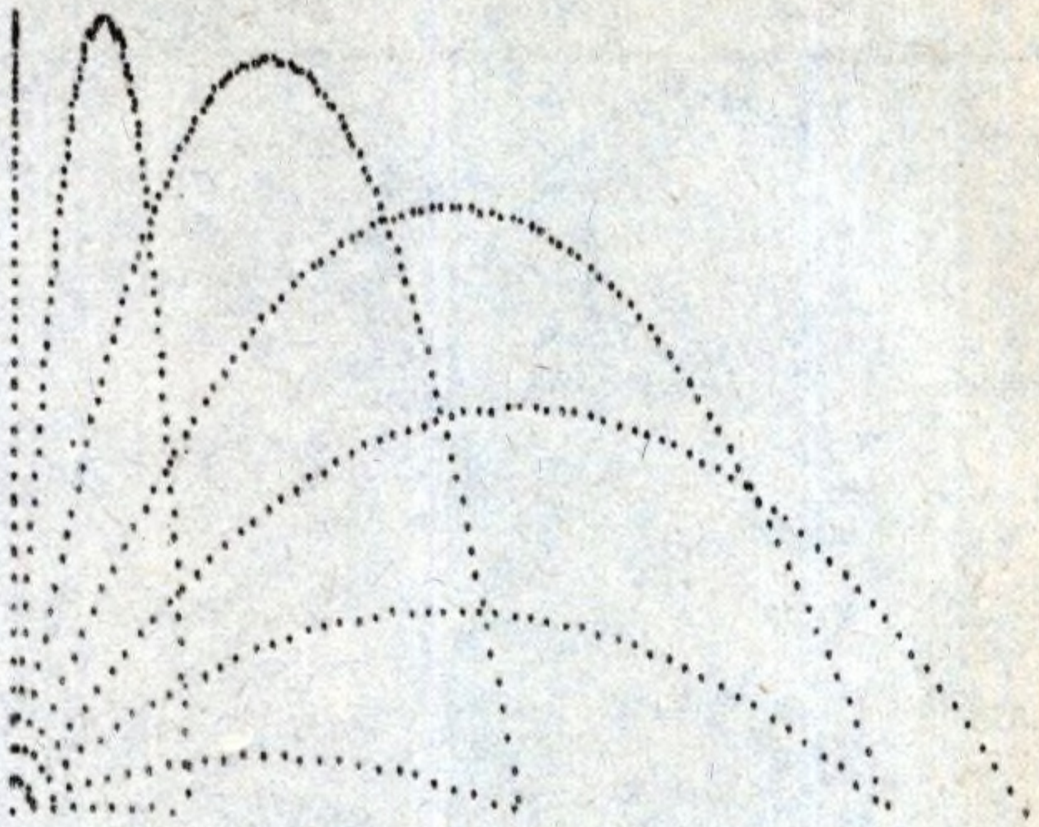
Na podstawie uzyskiwanych trajektorii uczeń może samodzielnie odpowiedzieć na pytania:

— od jakich parametrów i w jaki sposób zależy zasięg w rzucie ukośnym,

— dla jakiego kąta wyrzutu zasięg jest maksymalny,

— jaki jest związek między dwoma różnymi kątami, dla których zasięg jest taki sam.

Program może być również wykorzystany (po prostej modyfikacji) do prowadzenia gry, polegającej na trafieniu w określony punkt, przy jak najmniejszej liczbie rzutów. Przedstawiony program ma na celu zainspirowanie zainteresowanych Czytelników do własnych poszukiwań metod unowocześnienia i uatrakcyjnienia procesu dydaktycznego przez sensowne wdrażanie nowego technicznego środka nauczania, jakim jest mikrokomputer. Szczególnie duże pole do popisu istnieje w naukach matematyczno-przyrodniczych, ale ciekawe programy można napisać dla każdego przedmiotu.



```
  10 REM [redacted]
  20 INPUT "kąt rzutu (0-90): ";
k: IF k<0 OR k>90 THEN GO TO 20
  30 LET k=2*PI*k/360:
     REM [redacted]
  40 INPUT "prędkość początk. (0
-200): ";v0: IF v0<0 OR v0>200 T
HEN GO TO 40
  50 LET wx=.0625: LET wy=.085:
     REM [redacted]
  60 LET vx=v0*COS k: LET vy=v0*
SIN k: REM [redacted]
  70 LET dt=.4: LET g=9.81: LET
g=g/2:
     REM [redacted]
  80 FOR t=0 TO 50 STEP dt
  90 LET y=vy*t-g*t*t:
     REM [redacted]
 100 IF y<0 THEN GO TO 20:
     REM [redacted]
 110 LET x=vx*t
 120 PLOT x*wx,y*wy
 130 NEXT t
```

J.M.

Symulacja EDIT

Prezentowany program pozwala wprowadzać dowolne zmiany w zbiorze tekstowym, symulując swoim działaniem edycją linii. Mniej wtajemniczonym zwracam uwagę na kontrolę klawiszy **DELETE**, **EDIT**, **ENTER** i kursorów za pomocą INKEY$ oraz na uzyskanie wydruku na dole ekranu.

Można obejrzeć kody klawiszy wprowadzane przez INKEY$ dzięki linii 1000:

Krzysztof POŹNIAK
„Hobbyte"

```
  10 REM symulacja EDIT
  20 LET t$="ten tekst należy wy
edytować i w dowolny sposób popr
awić"
  30 PRINT t$
  40 IF INKEY$<>CHR$ 7 THEN GO T
O 40
  50 GO SUB 100: CLS : GO TO 30
 100 LET k=0: LET d=0
 110 BEEP .05,20: PRINT #1;AT 0,
0;t$( TO k); FLASH 1;"L"; FLASH
0;t$(k+1 TO );" " AND d: LET d=0
 120 LET inkey=CODE INKEY$
 130 IF inkey=13 THEN RETURN
 140 IF inkey=12 AND k AND LEN t
$ THEN LET d=1: LET t$=t$( TO k-
1)+t$(k+1 TO ): LET k=k-1: GO TO
 110
 160 IF inkey=8 AND k THEN LET k
=k-1: GO TO 110
 170 IF inkey=9 AND k<>LEN t$ TH
EN LET k=k+1: GO TO 110
 180 IF inkey<=31 THEN GO TO 120
 190 LET t$=t$( TO k)+CHR$ inkey
+t$(k+1 TO ): LET k=k+1: GO TO 1
10

1000 PRINT AT 0,0;CODE INKEY$;"
  ": GO TO 1000
```

Długość życia

Program oparty na badaniach naukowych pozwala na obliczenie przewidywanej długości życia osób dorosłych w wieku 20—65 lat. To jak długo będziemy żyć, zależy jak się okazuje od... wielu czynników, o które będziemy pytani w programie. Sprawdźmy, o ile dłużej możemy żyć, jeżeli w Nowym Roku 1988 rzucimy na przykład palenie, zaczniemy się gimnastykować i stracimy nieco na wadze. Różnica może być zaskakująca. Może zatem warto spróbować?

Tadeusz CISEK

```
1 REM DLUGOSC ZYCIA
2 PRINT CHR$(147)
3 PRINT'DLUGOSC ZYCIA'
5 PRINT
40 PRINT'OKRESLENIE PRZEWIDYWANEJ DLUGOS
CI ZYCIA DLA OSOB W WIEKU 20-65 LAT'
45 PRINT
50 PRINT'1=TAK,2=NIE':PRINT
60 A=72
70 INPUT'CZY JESTES MEZCZYZNA ?';B:PRINT

80 IF B=1 THEN A=A-3:GOTO 90
85 A=A+4
90 PRINT'CZY MIESZKASZ W MIESCIE O LICZB
IE'
95 INPUT'MIESZKANCOW > 1 MILIONA ?';B:PR
INT
100 IF B=1 THEN A=A-2:GOTO 110
105 PRINT'CZY MIESZKASZ W MIESCIE O LICZ
BIE '
107 INPUT'MIESZKANCOW (10000) ?';B:IF B=
1 THEN A=A+2:PRINT

110 PRINT'JEZELI PRACUJESZ ZA BIURKIEM W
PISZ '1'':PRINT
120 PRINT'JEZELI TWOJA PRACA WYMAGA REGU
LARNEGO, CIEZKIEGO'
125 INPUT'FIZYCZNEGO WYSILKU WPISZ '2'';
B :PRINT
130 IF B=1 THEN A=A-3:GOTO 140
135 IF B=2 THEN A=A+3
140 PRINT'JEZELI CWICZYSZ INTENSYWNIE WI
ECEJ NIZ'
145 PRINT'5 RAZY PO POL GODZ W TYGODNIU'

150 PRINT'WPISZ  '1'. JEZELI CWICZYSZ 2-
3 RAZY'
155 PRINT'W TYGODNIU, WPISZ  '2', JEZELI
 NIE'
157 INPUT' WPISZ  '3'';B:PRINT
160 IF B=1 THEN A=A+4:GOTO 170
165 IF B=2 THEN A=A+2
170 INPUT 'CZY JESTES ZONATY (ZAMEZNA) ?
';B:PRINT

180 IF B=1 THEN A=A+5:GOTO 190
185 PRINT'ILE DZIESIECIOLECI MIESZKALES
SAMOTNIE '
187 INPUT '(LICZAC OD 25 ROKU ZYCIA)?';B
:A=A-B:PRINT
190 PRINT' CZY SPISZ WIECEJ NIZ 10 GODZI
N '
195 INPUT'NA DOBE ?';B:PRINT
200 IF B=1 THEN A=A-4
210 PRINT'JEZELI JESTES OSOBA  AGRESYWNA
, ZYJACA INTENSYWNIE '
215 PRINT'WPISZ '1' , JEZELI JESTES OSOB
A'
220 INPUT 'SPOKOJNA, POGODNA WPISZ '2''
;B:PRINT
230 IF B=1 THEN A=A-3:GOTO 240
235 IF B=2 THEN A=A+3
240 PRINT'JEZELI JESTES SZCZESLIWY WPISZ
  '1';NIESZCZESLIWY,'
245 INPUT 'WPISZ  '2'';B:PRINT
250 IF B=1 THEN A=A+1:GOTO 260
255 IF B=2 THEN A=A-2

260 PRINT'CZY DOSTALES MANDAT ZA PRZEKRO
CZENIE PREDKOSCI'
265 INPUT 'W MINIONYM ROKU ?';B :PRINT
270 IF B=1 THEN A=A-1
280 PRINT 'CZY TWOJ DOCHOD W CIAGU ROKU'
:INPUT' PRZEKRACZA MILION ?';B :PRINT
290 IF B=1 THEN A=A-2
300 PRINT'JEZELI SKONCZYLES SZKOLE SREDN
IA WPISZ  '1';'
305 INPUT 'SZKOLE WYZSZA, WPISZ  '2':';B
:PRINT
310 IF B=1 THEN A=A+1:GOTO 320
315 IF B=2 THEN A=A+2
320 PRINT'JEZELI MASZ WIECEJ NIZ 65 LAT
I NADAL PRACUJESZ,'
325 INPUT' WPISZ  '1'';B :PRINT
330 IF B=1 THEN A=A+3
340 PRINT'JEZELI KTORES Z TWOICH DZIADKO
W ZYLO'
345 INPUT' 85 LAT WPISZ '1' ';B:PRINT
350 IF B=1 THEN A=A+2:INPUT 'CZY WSZYSCY
 4 ZYLI 80 LAT';B:IF B=1 THEN A=A+6

360 PRINT:PRINT'CZY KTORES Z TWOICH RODZ
ICOW ZMARLO'
365 PRINT 'NA ZAWAL LUB ATAK SERCA PRZED
':INPUT 'UKONCZENIEM 50 LAT';B :PRINT
370 IF B=1 THEN A=A-4
380 PRINT'CZY KTORES Z TWOICH RODZICOW L
UB RODZENSTWA PRZED 50'
385 PRINT'MIALO CHOROBE SERCA LUB RAKA,
LUB'
390 INPUT 'CUKRZYCE';B:PRINT
400 IF B=1 THEN A=A-3
410 PRINT'JEZELI PALISZ: >2 PACZKI/DZIEN
 WPISZ '1';'
415 PRINT '1-2 PACZKI/DZIEN, WPISZ '2';'

420 PRINT '1/2-1 PACZKI/DZIEN WPISZ '3''
:INPUT'JEZELI MNIEJ WPISZ '4'';B:PRINT
430 IF B=1 THEN A=A-8:GOTO 440
432 IF B=2 THEN A=A-6:GOTO 440
434 IF B=3 THEN A=A-3
440 PRINT'CZY PIJESZ CODZIENNIE CO NAJMN
IEJ'
445 INPUT '100 G ALKOHOLU ';B:PRINT
450 IF B=1 THEN A=A-1
460 PRINT'JEZELI MASZ NADWAGE POWYZEJ 25
 KG, WPISZ '1', 15-25 KG WPISZ '2''
470 PRINT ',5-15 KG , WPISZ '3'':INPUT '
W INNYM PRZYPADKU WPISZ '4' ';B:PRINT
480 IF B=1 THEN A=A-8:GOTO 490
482 IF B=2 THEN A=A-4:GOTO 490
484 IF B=3 THEN A=A-2
490 PRINT'CZY PRZECHODZISZ RAZ W ROKU OG
OLNE BADANIA'
495 INPUT'LEKARSKIE ?';B:PRINT
510 IF B=1 THEN A=A+2
520 PRINT'JEZELI MASZ 30-40 LAT WPISZ '1
';40-50, WPISZ'2'; 50-60, WPISZ'
530 INPUT ''3'; >70 WPISZ'4'';B:PRINT
540 A=A+B+1
550 PRINT ''
560 PRINT:PRINT' PRZEWIDYWANA DLUGOSC TW
OJEGO ZYCIA        WYNOSI ';A;' LAT !!!'
```

Sami piszemy gry

Cezary SOBCZAK

Zanim przystąpimy do pisania kolejnej gry, przedstawię Wam prosty program pozwalający uzyskać na ekranie efekt ruchu piłki bilardowej podstawę do dalszej zabawy.

```
10 REM ruch pileczki
20 MODE 1:INK 0,9:INK 1,26:INK 2,7
30 PAPER 2:BORDER 9:WINDOW #1,2,39,2,24
40 PAPER #1,0:CLS:CLS #1
50 x=20:y=12:dx=1:dy=-1
60 WHILE INKEY(47)=-1
70 LOCATE #1, x,y:PRINT #1, CHR$(231);
80 LOCATE #1,x,y:PRINT #1, " ";
90 x=x+dx:y=y+dy
100 IF x<=1 OR x>=38 THEN dx=-dx
110 IF y<=1 OR y>=23 THEN dy=-dy
120 LOCATE #1,x,y:PRINT #1, CHR$(231);
130 WEND
140 WHILE INKEY(60)=-1:WEND:GOTO 60
```

Po uruchomieniu programu komendą **RUN**, naciśnięcie spacji spowoduje zatrzymanie piłeczki, a klawisza „S" wprowadzenie jej znowu w ruch. Możemy teraz przystąpić do budowy gry. Zaczynamy, jak zwykle, od wymyślnego scenariusza. Na początek proponuję coś takiego: po ekranie poruszają się dwie piłeczki. Naszym zadaniem będzie sterowanie ruchem ludzika, tak aby w ustalonym czasie (np. 100 sekund) złapać jak najwięcej piłeczek. Za każde złapanie piłeczki otrzymujemy jeden punkt.

Wprowadźmy oznaczenia na zmienne

n — liczba piłeczek (n=2)
z, v — współrzędne ludzika (wart. początkowe z=20, v=12)
x(i), y(i) — współrzędne i-tej piłeczki, i=1,n
dx(i), dy(i) — przyrosty współrzędnych i-tej piłeczki
cg — łączny czas gry (cg=100)
t — bieżący czas
ta — czas początkowy
pt — liczba uzyskanych punktów
lewo, prawo, góra, dół — przechowują numery klawiszy sterujących ruchem ludzika. Przyjęto następujące klawisze: lewo=71(z), prawo=63(x), góra=19(l), dół=22(/).

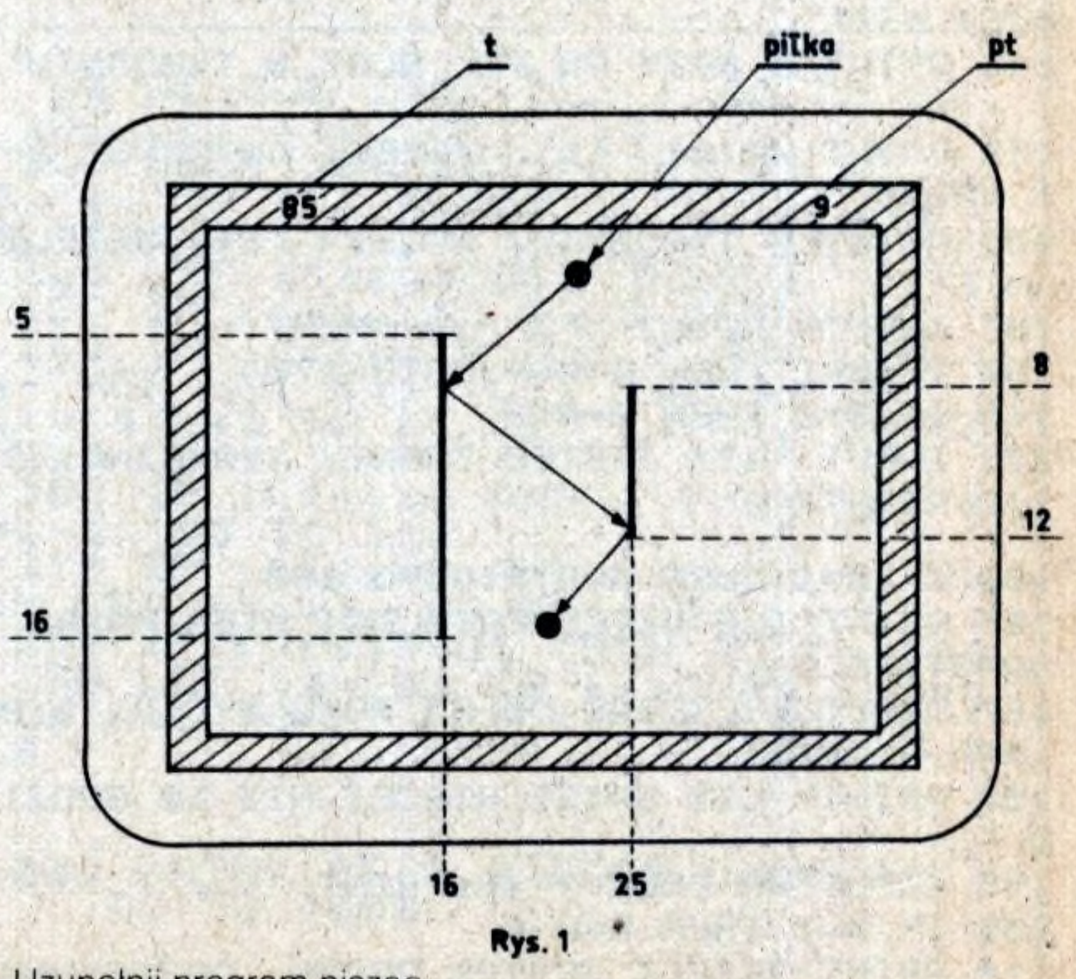
Wykorzystując przedstawioną wcześniej zasadę, piszemy:

```
10 REM zlap pilke
20 REM ---------------------------------
30 MODE 1:INK 0,0:INK 1,26:INK 2,7:INK 3
,11
40 BORDER 0:CLS
50 REM okreslenie tablic
60 n=2
70 DIM x(n):DIM y(n):DIM dx(n):DIM dy(n)
80 REM ramka
90 MOVE 12,12:DRAWR 0,360:DRAWR 615,0
100 DRAWR 0,-360:DRAWR -615,0
101 FOR y=370 TO 383:MOVE 12,y: DRAWR 61
5,0:NEXT
110 MOVE 12,399:FILL 2
140 REM nadanie wart. pocz. zmiennym
142 ENV 1,15,-1,10,15,1,10
150 ta=INT(TIME/300):pt=0:t=0:cg=100
160 z=20:v=12
170 lewo=71:prawo=63:gora=19:dol=22
180 x(1)=4:x(2)=30:y(1)=4:y(2)=20
190 FOR i=1 TO n
200 dx(i)=1:dy(i)=-1:NEXT
210 REM ---------------------------------
220 WHILE t<cg
230 t=INT(TIME/300)-ta:LOCATE 3,1
240 PEN 1:PRINT t:LOCATE 36,1:PRINT pt;
250 REM druk ludzika
260 LOCATE z,v:PEN 3:PRINT CHR$(248)
270 REM ruch pileczek
280 FOR i=1 TO n
290 IF x(i)<=2 OR x(i)>=39 THEN dx(i)=-d
x(i)
300 IF y(i)<=3 OR y(i)>=24 THEN dy(i)=-d
y(i)
310 IF x(i)=z AND y(i)=v THEN 460
320 LOCATE x(i),y(i):PRINT " ":x(i)=x(i)
+dx(i)
330 y(i)=y(i)+dy(i):LOCATE x(i),y(i):PEN
 1:PRINT CHR$(231)
340 NEXT
350 REM ruch ludzika
360 IF INKEY(lewo)=0 AND z>2 THEN GOSUB
1000:z=z-1
370 IF INKEY(prawo)=0 AND z<39 THEN GOSU
B 1000:z=z+1
380 IF INKEY(gora)=0 AND v>3 THEN GOSUB
1000:v=v-1
390 IF INKEY(dol)=0 AND v<24 THEN GOSUB
1000:v=v+1
400 WEND
410 REM koniec gry
420 INK 3,26,18:LOCATE 12,1:PEN 3:PRINT"
GRASZ JESZCZE (T/N)";
430 IF INKEY(51)=0 THEN RUN
440 IF INKEY(46)=0 THEN PEN 1:CLEAR INPU
T: CLS: END ELSE 430
460 REM zlapanie pilki
470 pt=pt+1:SOUND 3,340,50,15,1
472 FOR i=1 TO 150:BORDER 7,26:NEXT
473 BORDER 0
480 z=20:v=12:GOTO 230
1000 LOCATE z,v:PRINT " ":RETURN
```

Uruchom program i zagraj. Po pewnym czasie zapewne zauważysz, że piłeczki poruszają się po stałych torach i wystarczy ustawić ludzika w jednym miejscu, by piłeczka sama w niego trafiała. Zmodyfikuj program, generując początkowe współrzędne piłeczek w sposób losowy. Napisz:

```
145 randomize time
180 gosub 2000
475 gosub 2000
2000 for i=1 to n
2010 if x(i)<>Ø then locate x(i), y(i): print " "
2100 x(i)=int (RND★23+7)
2200 y(i)=int (RND★12+4): NEXT
2300 return
```

i uruchom program. Aby grę nieco uatrakcyjnić, wprowadzimy jeszcze niewidoczne dla nas przeszkody, od których piłeczki będą się odbijały, zmieniając niespodziewanie kierunek swojego ruchu. Proponuję ustawienie dwóch przeszkód (rys.1).



Rys. 1

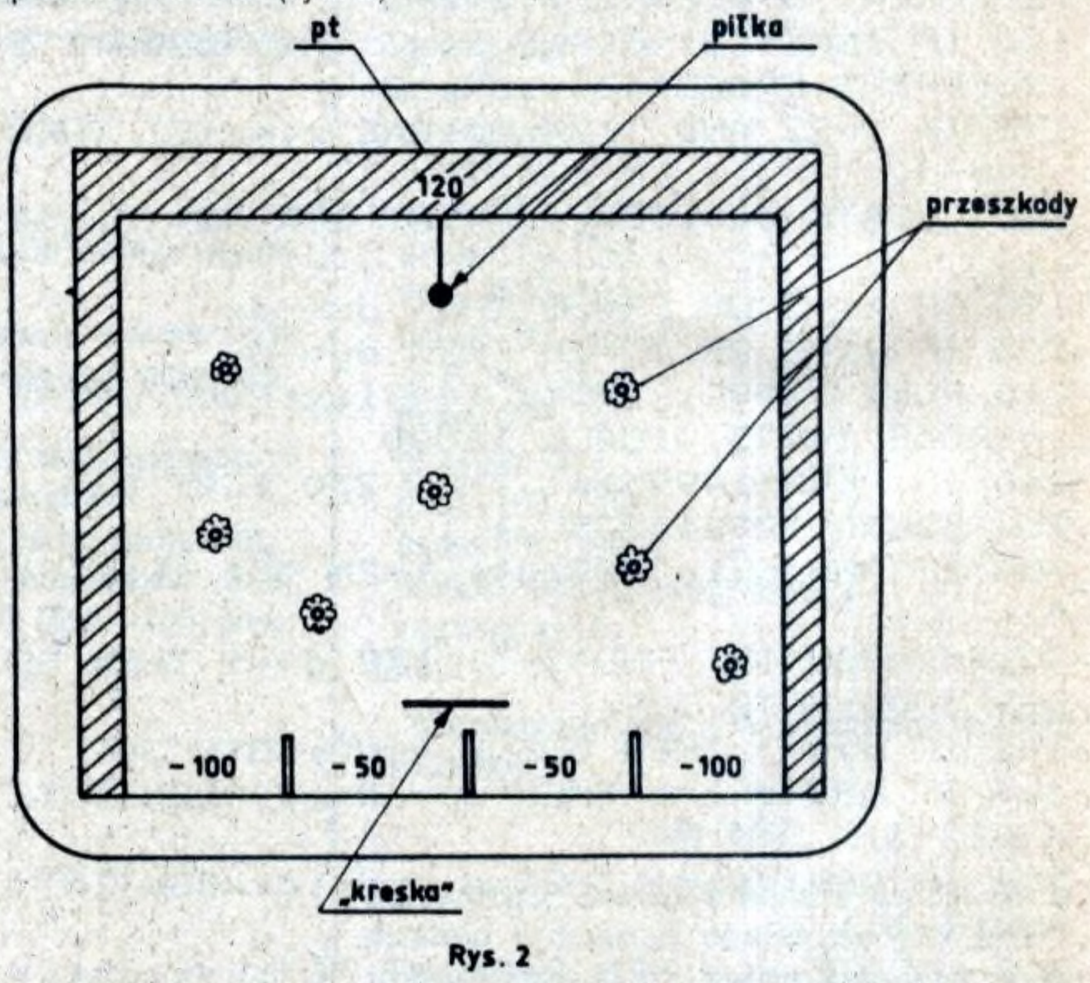
Uzupełnij program pisząc

```
285 if x(i)=16 and (y(i)<=16 and y(i)>=5) then dx(i)=—dx(i)
286 if x(i)=25 and (y(i)<=12 and y(i)>=8) then dx(i)=—dx(i)
```

Będzie nam na pewno przyjemniej grać słuchając muzyki. Dźwięki (ich okresy drgań) wprowadzone będą w wierszach DATA (od linii 5000) i chcąc zmienić melodię (czyli wprowadzić nowe liczby i nową ich ilość) trzeba pamiętać o uaktualnieniu zmiennej „mel" na bieżącą liczbę dźwięków, (wiersz 155).
Jako melodyjkę proponują fragment Partity h-moll na skrzypce solo J. S. Bacha.
Ostatecznie program ma postać:

```
10 REM zlap pilke
20 REM ----------------------
30 MODE 1:INK 0,0:INK 1,26:INK 2,7:INK 3
,16
40 BORDER 0:CLS
50 REM okreslenie tablic
60 n=2
70 DIM x(n):DIM y(n):DIM dx(n):DIM dy(n)
80 REM ramka
90 MOVE 12,12:DRAWR 0,360:DRAWR 615,0
100 DRAWR 0,-360:DRAWR -615,0
101 FOR y=370 TO 383:MOVE 12,y: DRAWR 61
5,0:NEXT
110 MOVE 12,399:FILL 2
140 REM nadanie wart. pocz. zmiennym
142 ENV 1,15,-1,10,15,1,10
145 RANDOMIZE TIME
150 ta=INT(TIME/300):pt=0:t=0:cg=100
155 RESTORE 5000:nt=1:mel=72
160 z=20:v=12
170 lewo=71:prawo=63:gora=19:dol=22
180 GOSUB 2000
190 FOR i=1 TO n
200 dx(i)=1:dy(i)=-1:NEXT
210 REM     gra
220 WHILE t<cg
230 t=INT(TIME/300)-ta:LOCATE 3,1
240 PEN 1:PRINT t:LOCATE 36,1:PRINT pt;
250 REM druk ludzika
260 LOCATE z,v:PEN 3:PRINT CHR$(248)
270 REM ruch pileczek
280 FOR i=1 TO n
285 IF x(i)=16 AND y(i)<16 AND y(i)>5 TH
EN dx(i)=-dx(i)
286 IF x(i)=25 AND y(i)<12 AND y(i)>8 TH
EN dx(i)=-dx(i)
290 IF x(i)<=2 OR x(i)>=39 THEN dx(i)=-d
x(i)
300 IF y(i)<=3 OR y(i)>=24 THEN dy(i)=-d
y(i)
310 IF x(i)=z AND y(i)=v THEN 460
320 LOCATE x(i),y(i):PRINT " ":x(i)=x(i)
+dx(i)
330 y(i)=y(i)+dy(i):LOCATE x(i),y(i):PEN
 1:PRINT CHR$(231)
340 NEXT
350 REM ruch ludzika
360 IF INKEY(lewo)=0 AND z>2 THEN GOSUB
1000:z=z-1
370 IF INKEY(prawo)=0 AND z<39 THEN GOSU
B 1000:z=z+1
380 IF INKEY(gora)=0 AND v>3 THEN GOSUB
1000:v=v-1
390 IF INKEY(dol)=0 AND v<24 THEN GOSUB
1000:v=v+1
391 READ nuta
392 SOUND 135,nuta,18,12,1
393 nt=nt+1
394 IF nt=mel THEN RESTORE 5000:nt=1
400 WEND
410 REM koniec gry
420 INK 3,26,18:LOCATE 12,1:PEN 3:PRINT"
GRASZ JESZCZE (T/N)";
430 IF INKEY(51)=0 THEN RUN
440 IF INKEY(46)=0 THEN PEN 1:CLEAR INPU
T: CLS: END ELSE 430
460 REM zlapanie pilki
470 pt=pt+1:SOUND 3,340,50,15,1
472 FOR i=1 TO 150:BORDER 7,26:NEXT
473 BORDER 0
475 GOSUB 2000
480 z=20:v=12:GOTO 230
1000 LOCATE z,v:PRINT " ":RETURN
2000 FOR i=1 TO n
2010 IF x(i)<>0 THEN LOCATE x(i),y(i):PR
INT" "
2100 x(i)=INT(RND*23+7)
2200 y(i)=INT(RND*12+4):NEXT
2300 RETURN
2350 REM  nuty
5000 DATA 127,106,84,80,95,106,113,71,80
5100 DATA 84,106,113,106,142,169,213,106
,113
5200 DATA 127,159,169,190,95,106,179,113
,127
5300 DATA 134,84,113,134,159,169,190,213
,225
5400 DATA 213,169,127,225,127,142,253,14
2,159
5500 DATA 284,225,190,159,113,95,134,84,
95
5600 DATA 106,127,142,159,95,106,113,134
,127
5700 DATA 169,134,113,84,80,84,95,106,11
3
```

Innym niemal już klasycznym przykładem wykorzystania efektu ruchu piłki bilardowej, jest „flipper". W grze tej wystrzelona piłeczka, trafiając w różnorodne przeszkody, zdobywa dla nas punkty do czasu, aż wypadnie poza pole gry. Sterując odpowiednio „flipperami", od których piłeczka się odbija, staramy się jak najdłużej utrzymać ją w ruchu na polu gry.
W naszej grze rolę dwóch „flipperów" przejmie poruszająca się poziomo kreska (rys. 2).



Rys. 2

Na początku piłeczka, wisząc na cienkiej nitce, mruga do nas zachęcająco. Po wystartowaniu (naciskając spację) piłka spada i krążąc po polu gry, odbija się od przeszkód. Za każdorazowe trafienie w przeszkodę otrzymujemy 10 punktów. Jeżeli nie uda nam się, poruszając „kreską" odbić piłeczki, wówczas wpada ona do jednej z czterech komór, „zdobywając" dla nas punkty ujemne.

```
9 SYMBOL AFTER 256
10 MEMORY 20000:SYMBOL AFTER 38:CLS
15 ENT 1,89,37,238:ENV 1,84,7,5
16 ENT -2,13,-9,1,1,121,1:ENV 2,1,14,1,1
4,-1,11
20 MODE 1:INK 0,0:INK 1,26:INK 2,24:INK
3,17:BORDER 0
30 FOR i=16000 TO 16028:READ a$:v=VAL("&
"+a$):POKE i,v:NEXT
```

```
40 n=15:pt=0
45 SYMBOL 38,60,66,153,189,189,153,66,60
50 LOCATE 1,1:PEN 2:PRINT STRING$(39,143
);:LOCATE 1,2:PEN 2:PRINT STRING$(39,143
);:LOCATE 19,1:PEN 1:PRINT pt;
60 FOR y=1 TO 25:LOCATE 1,y:PEN 2:PRINT
CHR$(143):LOCATE 39,y:PEN 2:PRINT CHR$(1
43):NEXT
70 LOCATE 3,25:PEN 3:PRINT"-100";
80 LOCATE 14,25:PEN 3:PRINT"-50";:LOCATE
 24,25:PEN 3:PRINT"-50";:LOCATE 32,25:PE
N 3:PRINT"-100";
85 FOR i=10 TO 30 STEP 10:LOCATE i,25:PE
N 2:PRINT CHR$(149);:LOCATE i,24:PEN 2:P
RINT CHR$(149);:NEXT
87 MOVE 0,0:DRAWR 620,0,2
90 MOVE 313,370:DRAWR 0,-45,2:LOCATE 20,
5:PEN 1:PRINT CHR$(231)
91 IF x<>0 THEN LOCATE x,24:PRINT" ":LOC
ATE 2,23:PRINT SPACE$(37);
100 z=20:v=23:x=20:y=5:dx=SGN(INT(RND*5-
3)):dy=1
105 LOCATE z,v:PEN 1:PRINT STRING$(3,154
);
110 WHILE INKEY(47)=-1:LOCATE 20,5:PEN 1
:PRINT CHR$(231):LOCATE 20,5:PRINT" ":WE
ND:MOVE 313,365:DRAWR 0,-45,2,1:LOCATE 2
0,5:PRINT" "
120 RESTORE 2000:FOR i=0 TO n-1:READ x,y
:LOCATE x,y:PEN 2:PRINT "&":POKE 15000+i
,x:POKE 15100+i,y:NEXT
125 x=20:y=5
130 WHILE y<24
140 LOCATE x,y:PRINT" ":x=x+dx:y=y+dy:LO
CATE x,y:PRINT CHR$(231)
160 IF INKEY(1)=0 AND z<36 THEN LOCATE z
,23:PRINT SPACE$(2);:z=z+2
170 IF INKEY(8)=0 AND z>=3 THEN LOCATE z
,23:PRINT SPACE$(3);:z=z-2
175 IF y=22 AND (x)>=z-1 AND x<=z+3) THEN
 dy=-dy
180 LOCATE z,v:PEN 1:PRINT STRING$(3,154
);
190 IF x<=3 OR x>=38 THEN dx=-dx
200 IF y<=3 OR y>=25 THEN dy=-dy
210 POKE 14990,x:POKE 14991,y:POKE 14992
,n:POKE 14995,0:CALL 16000
220 IF PEEK(14995)=1 THEN 250 ELSE 310
250 SOUND 129,142,50,1,1
300 IF ABS(dx)= ABS(dy) THEN 301 ELSE 30
4
301 dx=SGN(INT(RND*7-3)):IF dy=1 THEN GO
SUB 3000:GOTO 309
302 IF dy=-1 THEN GOSUB 4000:GOTO 309
304 IF ABS(dx)=1 THEN dx=-dx:dy=SGN(INT(
RND*7-3))
305 IF ABS(dy)=1 THEN dy=-dy:dx=SGN(INT(
RND*7-3))
309 LOCATE x,y:PEN 2:PRINT "&":x=x+dx:y=
y+dy:pt=pt+10:LOCATE 19,1:PEN 1:PRINT pt
310 WEND
312 SOUND 129,250,0,0,2,2
315 FOR a=1 TO 1000:NEXT
320 IF (x>=1 AND x<=10) OR (x>=31 AND x<
=40) THEN pt=pt-100:LOCATE 19,1:PRINT pt
;:GOTO 90
330  pt=pt-50:LOCATE 19,1:PRINT pt;:GOTO
 90
340 GOTO 90
1000 DATA 3a,8e,3a,21,98,3a
1100 DATA ed,4b,90,3a,ed,b1,c0,1,63,0,9
1200 DATA 3a,8f,3a,be,c0,1,93,3a,3e,1,2,
c9
2000 DATA 6,13,11,14,12,7,14,21,20,15,26
,9,31,16,34,19,35,12,25,6,18,17,21,8,13,
10,20,15,36,5
3000 IF RND>0.5 THEN dy=-1:RETURN
3100 dy=0:RETURN
4000 IF RND>0.5 THEN dy=1:RETURN
4100 dy=0:RETURN
```

Podprogram umieszczony w liniach 1000—1200 wczytywany jest w linii 30, pary współrzędnych (xp, yp) rozmieszczenia przeszkód podane w linii 2000 wczytujemy w linii 120.

Znak, który ma być drukowany jako przeszkoda jest zdefiniowany w wierszu 45.

Po wywołaniu podprogramu (linia 210) następuje badanie wskaźnika (linia 230). W razie trafienia piłeczki w przeszkodę następuje (w liniach 300—305) losowa zmiana kierunku ruchu piłeczki.

Efekty dźwiękowe występują tylko w wypadku trafienia w przeszkodę lub wyjścia z pola gry. Spróbuj rozszerzyć program, tak aby np. w czasie całej gry przygrywała muzyczka.

Podprogram wygląda następująco:

Kod mnemoniczny	Opis	Liczba bajtów rozkazu	Odpowiednik rozkazu w kodzie maszynowym (ósemkowo)
LD A, (14990)	Podstaw do akumulatora współrzędną x piłeczki	3	3A, 8E, 3A
LD HL, 15000	Podstaw do pary rejestrów HL adres początku tablicy współrzędnych xp przeszkód	3	21, 98, 3A
LD BC, (14992)	Podstaw do pary rejestrów BC ilość przeszkód (wartość n).	4	ED, 4B, 90, 3A
CPIR	Porównaj wartość współrzędnej x piłeczki (w akumulatorze) z kolejnymi elementami tablicy xp przeszkód	4	ED, B1
RET NZ	Powrót do programu BASICA, jeżeli nie znaleziono odpowiednika w tablicy	1	C0
LD BC, 99	Znaleziono. Współrzędna x=xp Podstaw do pary rejestrów BC wartość 99	3	1, 63, 0
ADD HL, BC	Zmodyfikuj adres tablicy wsp. przeszkód w celu porównania wsp. y piłeczki z odpowiadającą jej współrzędną yp przeszkody. Do pary rejestrów HL dodaj zawartość pary rejestrów BC. Wynik w HL	1	9
LD A, (14991)	Podstaw do akumulatora współrzędną y piłeczki	3	3A, 8F, 3A
CP (HL)	Porównaj wartość współrzędnej y piłeczki (w akumulatorze) z zawartością komórki o adresie umieszczonym w HL-element tablicy yp odpowiadający znalezionej wartości	1	BE
RET NZ	Powrót programu BASICA, jeżeli nie znaleziono odpowiednika	1	C0

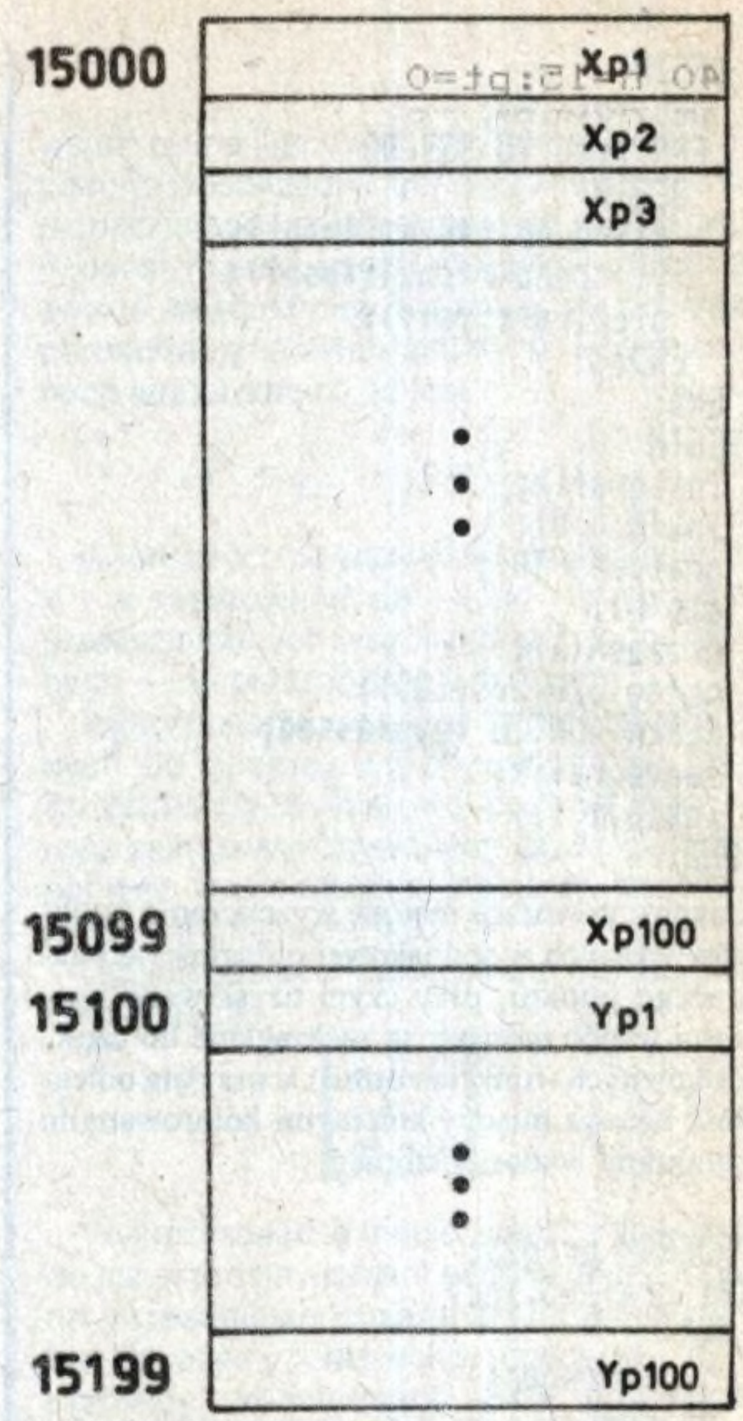
LD BC, 14995
LD A, 1
LD, BC, A
	Znaleziono, czyli x=xp i y=yp
	Ustawienie wskaźnika (komórki 14995) na jeden
		3
		2
		1
			1, 93, 3A
			3E, 1
			2

RET	Powrót do programu BASICA	1	C9

Oznaczenia zmiennych

n — liczba przeszkód
x, y — aktualne współrzędne piłki
xp, yp — współrzędne przeszkód
z, v — współrzędne „kreski" (v=23)
pt — punkty
lewo, prawo — numery klawiszy sterujących ruchem „kreski. Przyjęto lewo = 8 (←), prawo = 1 (→)

Współrzędne kolumn (xp) i linii (yp) przeszkód umieszczone zostaną w tablicach, które będziemy przeglądać po każdorazowej zmianie współrzędnych piłki. W tym celu napiszemy prosty podprogram w kodzie maszynowym, który będzie wywoływany w BASIC-u.
Przyjmijmy, że tablica współrzędnych xp zajmować będzie obszar pamięci od adresu 15000 (dziesiętnie) do 15099, a tablica odpowiadających im współrzędnych yp obszar od adresu 15100 do 15199 (rys. 3).



Rys. 3

Ustalmy ponadto, że:
— komórka o adresie 14990 zawierać będzie bieżącą wartość x piłeczki
— komórka o adresie 14991 zawierać będzie bieżącą wartość y piłeczki
— komórka o adresie 14992 zawierać będzie ilość przeszkód (wartość n)
— komórka o adresie 14995 będzie wskaźnikiem przyjmującym wartość 0 (zero), gdy współrzędne (x, y) piłeczki nie pokrywają się ze współrzędnymi (xp, yp) przeszkody, i wartość 1 (jeden) w przeciwnym wypadku.

Przed wywołaniem podprogramu wskaźnik ustawimy na zero.

Pascal (8)

Kontynuując rozważania na temat grafiki komputerowej, w kolejnej części przedstawiamy zasady wykorzystania pióra graficznego.

Z piórem graficznym związanych jest znacznie więcej procedur, niż z piórem tekstowym. Przed użyciem pióra graficznego niezbędne jest wywołanie bezparametrowej procedury

initgrafik

ustalającej początkowe warunki pracy. Po zakończeniu wykorzystywania pióra graficznego należy wywołać bezparametrową procedurę

leavegrafik

Odpowiednikiem procedury pen dla pióra graficznego jest procedura

grafpen (nrk)

a odpowiednikiem procedury **paper** — procedura

grafpaper (nrk)

gdzie **nrk** oznacza numer kałamarza.

Zasady wykorzystania kałamarzy przez pióro graficzne są identyczne, jak dla pióra tekstowego.

Z punktu widzenia pióra graficznego kartka podzielona jest na 400x600 punktów, (niezależnie od trybu wykorzystania ekranu) tj. 400 wierszy (numerowanych od 0 do 399) po 640 punktów (numerowanych od 0 do 639) w każdym z nich. Punkt można zakreślić kolorem tuszu za pomocą procedury

plot (x,y)

gdzie:
x — nr wiersza,
y — nr kolumny

Od ostatnio ustalonego położenia pióra można wykreślić odcinek do punktu wskazanego w procedurze

draw (x,y)

gdzie:
x,y — współrzędne punktu, do którego będzie kreślony odcinek.

Dodajmy, że punkt wskazany w **draw** staje się ostatnio ustalonym położeniem pióra graficznego.

Przedstawione procedury wykorzystano w programach kreślenia prostokątów:
— wyznaczającego ramy kartki (prog. 26),
— o współrzędnych wierzchołków (100, 100), (100,250), (550—250), (550—100) wraz z przekątnymi (prog. 27).

```
PROGRAM prog26;
 {$i grafik3.inc}
 BEGIN
  ink(0,0,0);
  ink(1,19,19);
  initgrafik;
  grafpaper(0);
  grafpen(1);
  plot(0,0);
  draw(0,399);
  draw(639,399);
  draw(639,0);
  draw(0,0);
  REPEAT UNTIL keypressed;
  leavegrafik;
  ink(0,1,1);
  ink(1,26,26);
END.
```

```
PROGRAM prog27;
 {$i grafik3.inc}
 BEGIN
  initgrafik;
  mode(0);
  ink(2,0,0);
  ink(3,12,26);
  ink(4,16,16);
  ink(5,11,11);
  grafpaper(2);
  clg(2);
  grafpen(5);
  plot(100,100);
  draw(100,250);
  draw(550,250);
  draw(550,100);
  draw(100,100);
  grafpen(3);
  draw(550,250);
  grafpen(4);
  plot(100,250);
  draw(550,100);
  REPEAT UNTIL keypressed;
  leavegrafik;
END.
```

W programie prog 27 wykorzystano ponadto procedurę

clg (nrk)

„czyszczącą" kartkę w kolorze ustalonym w ostatnio wykonanej procedurze grafpaper i ustanawiającą nowy numer kałamarza (nrk) dla tła pióra graficznego.

W kolejnych przykładach zilustrowaliśmy sposoby kreślenia figur graficznych oraz niektóre problemy związane z projektowaniem bardziej złożonych obrazów, ograniczając się do płaszczyzny.

W przykładzie 29 znajdziecie, program kreślenia okręgu o zadanym promieniu i współrzędnych środka. Wykorzystano w nim

```
PROGRAM prog28;
{$i grafik3.inc}
 PROCEDURE okrag(xs,ys,r:integer);
  VAR
   x,y,i:integer;
   wsp:real;
```

➡

```
 BEGIN
  wsp:=pi/180;
  FOR i:=0 TO 359 DO
   BEGIN
    x:=round(r*sin(i*wsp));
    y:=round(r*cos(i*wsp));
    plot(xs+x,ys+y);
   END;
 END;
BEGIN
 initgrafik;
 ink(0,0,0);
 grafpaper(0);
 clg(0);
 grafpen(1);
 okrag(320,200,100);
 REPEAT UNTIL keypressed;
 leavegrafik;
 ink(0,1,1);
END.
```

znaną z geometrii metodę wyznaczania punktów leżących w jednakowej odległości od zadanego punktu, przy czym uzyskiwane wartości współrzędnych są zaokrąglane do części całkowitych (funkcja **round**), a przyjęta odległość kątowa między kolejnymi kolorowanymi punktami wynosi 1 stopień.

```
PROGRAM prog29;
{$i grafik3.inc}
 VAR
  i,j:integer;
  ch:char;
  wsp:real;
 PROCEDURE okrag(xs,ys,r:integer);
  VAR
   x,y,i:integer;
   wsp:real;
  BEGIN
   wsp:=pi/180;
   FOR i:=0 TO 359 DO
    BEGIN
     x:=round(r*sin(i*wsp));
     y:=round(r*cos(i*wsp));
     plot(xs+x,ys+y);
    END;
 END;
BEGIN
 initgrafik;
 mode(1);
 ink(2,0,0);
 grafpaper(2);
 clg(2);
 ink(0,12,12);
 ink(3,10,10);
 grafpen(0);
 okrag(200,200,150);
 wsp:=pi/180;
 j:=0;
 FOR i:=0 TO 29 DO
  BEGIN
   grafpen(3);
   plot(200,200);
   draw(200+round(149*sin(6*j*wsp)),
        200+round(149*cos(6*j*wsp)));
   j:=j+1;
   grafpen(1);
   plot(200,200);
   draw(200+round(149*sin(6*j*wsp)),
        200+round(149*cos(6*j*wsp)));
   j:=j+1;
  END;
WHILE NOT keypressed DO
 BEGIN
  ink(1,10,10);
  ink(3,26,26);
  FOR i:=1 TO 4000 DO;
  ink(3,10,10);
  ink(1,26,26);
```

Wrażenie obracania się koła uzyskano w programie z przykładu 30 przez zmianę kolorów szprych koła. Efekt ten jest namiastką bardzo dynamicznie rozwijającej się animacji obrazów.

```
PROGRAM prog30;
 {$i grafik3.inc}
  VAR
   i,nr:integer;
   ch:char;
  BEGIN
   initgrafik;
   mode(0);
   ink(0,0,0);
   grafpaper(0);
   clg(0);
   grafpen(1);
   plot(100,100);
   draw(100,250);
   draw(550,250);
   draw(550,100);
   draw(100,100);
   REPEAT
    gotoxy(0,23);
    WRITE('podaj nr kalamarza  ');
    gotoxy(19,23);
    READ(nr);
    grafpen(nr MOD 16);
    FOR i:=100 TO 250 DO
     BEGIN
      plot(100,i);
      draw(550,i);
     END;
    REPEAT UNTIL keypressed;
    READ(kbd,ch);
   UNTIL ch=' ';
   ink(0,1,1);
   mode(2);
   leavegrafik;
END.
```

Inny problem to kolorowanie figur. W programie prog 30 zilustrowano jeden ze sposobów kolorowania pola prostokąta.

```
PROGRAM prog31;
 {$i grafik3.inc}
 VAR
  x11,y11,x12,y12,
  x21,y21,x22,y22:integer;
 BEGIN
  WRITELN('podaj wspolrzedne dolnego ',
          'prostokata');
  READLN(x11,y11,x12,y12);
  WRITELN('podaj wspolrzedne gornego ',
          'prostokata');
  READLN(x21,y21,x22,y22);
  initgrafik;
  mode(0);
  ink(0,0,0);
  clg(0);
  grafwindow(x11,y11,x12,y12);
  grafpaper(4);
  clg(4);
  grafwindow(x21,y21,x22,y22);
  grafpaper(8);
  clg(8);
  REPEAT UNTIL keypressed;
  mode(2);
  ink(0,1,1);
  leavegrafik;
END.
```

Występująca w programie instrukcja **gotoxy** (x,y) umożliwia ustawienie położenia pióra tekstowego w określonym miejscu ekranu (ekran jest tu widziany jako tablica znaków o wymiarach 25 wierszy i liczbie kolumn zależnej od trybu wykorzystania ekranu.

Wszędzie tam, gdzie zamieszcza się obrazy obiektów rozlokowanych w przestrzeni trójwymiarowej, pojawia się kwestia likwidowania rysunków niewidocznych części obiektów, ścian i krawędzi. Najprostszym przykładem może tu być kreślenie nakładających się częściowo prostokątów, zilustrowane w przykładzie 32.

W programie wykorzystano właściwości tzw. okien graficznych wyznaczonych za pomocą procedury

grafwindow (x_L, y_L, x_p, y_p)

gdzie:
x_L,y_L — współrzędne lewego dolnego wierzchołka okna
x_p,y_p — współrzędne prawego górnego wierzchołka okna.

Okno graficzne jest prostokątem. Przyjmuje się, że dla pióra graficznego kartka jest początkowym, standardowym oknem graficznym.

W podanym przykładzie każdy prostokąt reprezentowany jest przez okno graficzne, a do ich kolorowania wykorzystano znaną już procedurę **clg** (dotyczy ona faktycznie ostatnio określonego okna). Efekt przesłaniania uzyskano sterując kolejnością kolorowania prostokątów. W praktycznie spotykanych zadaniach uzyskanie tego efektu wymaga skomplikowanych obliczeń i należy do najtrudniejszych operacji w grafice komputerowej.

Okna graficzne wykorzystane są w programie wyświetlania kolorów tuszów, którymi standardowo napełniane są poszczególne kałamarze.

```
PROGRAM prog32;
 {$i grafik3.inc}
VAR
 x1,x2,y1,y2,
 i,j,k:integer;
BEGIN
 x1:=10;
 x2:=60;
 y1:=50;
 y2:=150;
 initgrafik;
 mode(0);
 ink(0,0,0);
 grafpaper(0);
 clg(0);
 k:=1;
 FOR i:=0 TO 1 DO
  FOR j:=0 TO 7 DO
   BEGIN
    grafwindow(x1+j*80,y1+i*125,
               x2+j*80,y2+i*125);
    clg(k);
    k:=k+1;
   END;
 REPEAT UNTIL keypressed;
 ink(0,1,1);
 ink(1,26,26);
 mode(2);
 leavegrafik;
END.
```

Podane przykłady na pewno nie oddają w pełni stopnia trudności grafiki komputerowej. Ilustrują jedynie wybrane, często spotykane problemy.

Stefan ROZMUS

Grafika na ekranie (3)

Przesunięcie oraz współrzędne jednorodne

Dotychczas w prezentowanym cyklu nie omawialiśmy przesunięcia punktów i linii na płaszczyźnie, a także figur składających się z punktów i linii. Spowodowane to było niemożliwością wprowadzenia stałych przesunięcia wewnątrz struktury ogólnej macierzy transformacji 2X2. Trudność tę można ominąć przez wprowadzenie trzeciego składnika do wektorów [*x y*] i [*x y*], przedstawiając je w postaci [*x y 1*] i [*x* y** 1]. Macierz transformacji po tej operacji staje się macierzą o rozmiarach 3X2:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ m & n \end{bmatrix}$$

Jest to konieczne, gdyż w celu wykonania mnożenia macierzy liczba kolumn w macierzy definiującej punkt winna równać się liczbie wierszy w macierzy transformacji. W ten sposób

$$[x\ y\ 1]\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ m & n \end{bmatrix} = [\,x+m\ \ y+m\,] = [\,x^*y^*\,]$$

Wynika z tego, że stałe *m* i *n* powodują przesunięcie *x** i *y** w stosunku do *x* i *y*. Aby istniała możliwość wyznaczenia macierzy odwrotnej (patrz część 1) do macierzy transformacji uzupełniamy tę macierz do kwadratowej o rozmiarach 3X3.

Na przykład:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

Trzeci składnik wektorów położenia punktów nie zmienia się po dodaniu trzeciej kolumny do macierzy transformacji. Wykorzystując tę macierz oraz uwzględniając (1) otrzymujemy przekształcony wektor [*x* y* 1*]. Dołożenie trzeciego elementu do wektora położenia i trzeciej kolumny do macierzy transformacji pozwala wykonać przesunięcie wektora położenia. Tak więc, wektor położenia [*x y 1*] po pomnożeniu przez macierz transformacji 3X3 staje się wektorem położenia w ogólnym przypadku postaci [*X Y H*]. Przedstawione przekształcenie było wykonane tak, że [*X Y H*] = [*x y 1*].

Przekształcenie zachodzące w przestrzeni trójwymiarowej w rozpatrywanym wypadku jest ograniczone do płaszczyzny gdyż H = 1. Jeżeli jednak trzecia kolumna

$$\begin{bmatrix} p \\ q \\ s \end{bmatrix}$$

macierzy transformacji T rozmiaru 3X3 nie jest równa

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

to w rezultacie przekształcenia otrzymamy [*x y 1*] = [*X Y H*], gdzie H ≠ 1. Płaszczyzna, na której obecnie znajduje się przekształcony wektor położenia, znajduje się w przestrzeni trójwymiarowej.

Zastąpienie dwuwymiarowego wektora trójwymiarowym lub w ogólnym przypadku n-wymiarowego (n + *1*) wymiarowym nazywa się wprowadzeniem tzw. współrzędnych jednorodnych. Opisane zostanie teraz dość obszernie zagadnienie współrzędnych jednorodnych. Współrzędne te mają duże zastosowanie, zwłaszcza w przestrzeni trójwymiarowej. Należy więc potraktować poniższy opis jako wprowadzenie do grafiki trójwymiarowej.

Idea tworzenia współrzędnych jednorodnych jest następująca. Punkt *p* przestrzeni *n*-wymiarowej charakteryzowany jest układem *n* liczb $p = (p_1, p_2, ..., p_n)$ będących jego współrzędnymi. Przedstawia się ten punkt za pomocą układu liczb *n+1* w postaci:

$$p = (p_1, p_2, ..., p_n, 1)$$

lub w ogólnym przypadku

$$p = (\lambda p_1, \lambda p_2, ..., \lambda p_n, \lambda) \text{ gdzie } \lambda \neq 0.$$

Liczby $\lambda p_1, \lambda p_2, ..., \lambda p_n, \lambda$ nazywa się współrzędnymi jednorodnymi punktu *p* określonego w przestrzeni *n*-wymiarowej.

Współrzędne jednorodne wykorzystuje się do wykonywania przekształceń w przestrzeni o jednym wymiarze większym niż ta, w której działamy. W ten sposób dwuwymiarowy wektor [*x y*] przedstawia się trójskładnikowym wektorem [*hx hy h*]. Związek pomiędzy współrzędnymi punktu na płaszczyźnie a jego współrzędnymi jednorodnymi wyraża zależność:

$$x = \frac{hx}{h} \qquad y = \frac{hy}{h}$$

Nie ma jednorodnego przekształcenia współrzędnych punktu w przestrzeni dwuwymiarowej. Na przykład jednorodne współrzędne (12, 3, 3), (8, 2, 2), (4, 1, 1) przedstawiają wyjściowy punkt (4, 1). Dla uproszczenia wyliczeń ustalamy, że [*x y 1*] przedstawia nieprzekształcony punkt w dwuwymiarowych współrzędnych jednorodnych.

Przekształcenie

$$[x^*\ y^*] = [x\ y]\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

we współrzędnych jednorodnych ma postać

$$[X\ Y\ H] = [x\ y\ 1]\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Wykonanie przedstawionego powyżej przekształcenia współrzędnych wykazuje, że *X = x*, Y = y*, H = 1*. Jeżeli dodatkowa współrzędna jest równa jedynce, to przekształcone współrzędne jednorazowe są równe przekształconym współrzędnym (zwykłym). W ogólnym przypadku *H ≠ 1* i przekształcone zwykłe współrzędne otrzymuje się poprzez normalizację przekształconych współrzędnych jednorodnych, to jest

$$x^* = \frac{X}{H} \quad i \quad y^* = \frac{Y}{H}$$

Geometrycznie wszystkie przekształcenia *x* i *y* realizowane są — po normalizacji przekształconych współrzędnych jednorodnych — na płaszczyźnie *h = 1*.

Wykorzystanie współrzędnych jednorodnych do operacji wykonywanych na płaszczyźnie dwuwymiarowej pozwala rozszerzyć zakres wykonywanych przekształceń, poprzez wprowadzenie macierzy transformacji o wymiarach 3X3.

Ogólna postać macierzy transformacji jest następująca:

$$\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$$

Wykorzystując tę macierz można dołączyć do już przedstawionych przekształceń inne, np. przesunięcie, zmianę skali i przesunięcie. Pokażemy teraz wpływ trzeciej kolumny macierzy transformacji 3X3 na wynik przekształceń na płaszczyźnie.

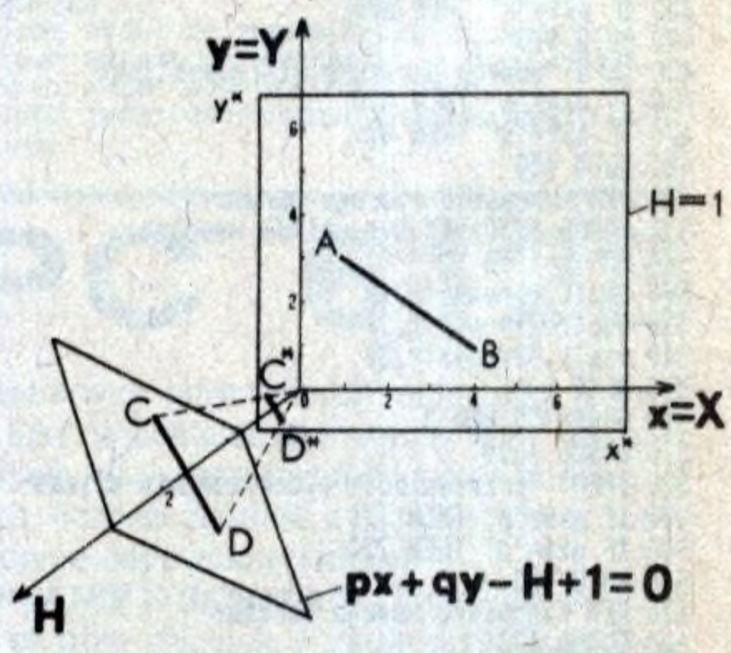
Prześledźmy wyniki następującej operacji:

$$[X\ Y\ H] = [x\ y\ 1]\begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= [x\ y\ (px + qx + 1)]$$

Teraz $X = x$, $Y = y$, $H = px + qx + 1$. Zmienna *H* określa płaszczyznę przekształcenia punktu (przedstawionego we współrzędnych jednorodnych) w przestrzeni trójwymiarowej.

Przekształcenie to pokazano na rys. 1.



Rys. 1 Przekształcenie we współrzędnych jednorodnych. Linia *AB* leżąca na płaszczyźnie *xy* jest rzutowana jako linia *CD* na płaszczyznę $pX+qY-H+1=0$

Na rys. 1 $p=q=1$. Wykonamy teraz normalizację w celu uzyskania zwykłych współrzędnych.

$$x^* = \frac{X}{H} = \frac{X}{pX+qY+1}$$

$$y^* = \frac{Y}{H} = \frac{Y}{pX+qY+1}$$

Przyjmując $p=q=1$ oraz współrzędne punktów A i B odpowiednio (1,3) i (4,1) otrzymujemy:

$$x^* = \frac{1}{1+3+1} = \frac{1}{5}, \qquad y^* = \frac{3}{5}$$

po przekształceniu A w C* oraz

$$x^* = \frac{4}{1+4+1} = \frac{2}{3}, \qquad y^* = \frac{1}{6}$$

po przekształceniu *B* w *D**.
Jednorodne współrzędne dla punktów *C** i *D** pokazanych na rys. 1 odpowiednio wynoszą $(\frac{1}{5}, \frac{3}{5}, 1)$ i $(\frac{2}{3}, \frac{1}{6}, 1)$.
W rezultacie normalizacji uzyskano przejście trójwymiarowej linii *CD* w jej rzut *C** *D**

```
10 REM ****************************************
20 REM
30 REM Przesuniecie
40 REM
50 REM ****************************************
60 DIM wsp(101,3),mat(3,3),nwsp(101,3)
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 "Przesuwanie obrazu w ukladzie";
110 PRINT "wspolrzednych:-319<x<319,-191<y<191"
120 PRINT
130 PRINT "Siatka wspolrzednych (t/n)"
140 z$=INKEY$
150 IF z$="" " THEN 140
160 IF z$="t" THEN fl=1:GOTO 190
170 IF z$="n" THEN fl=0:GOTO 190
180 GOTO 140
190 GRAPHICS PAPER 1:REM Tlo grafiki
200 ORIGIN 320,208,0,640,16,400
210 CLEAR INPUT
220 WINDOW 1,40,25,25
230 GOSUB 1230:REM Rys. ukladu wspol.
240 i=0 :REM Licznik punktow
250 INPUT "Podaj wsp. punktu x,y";x,y
260 IF x>998 THEN 330
270 i=i+1
280 wsp(i,1)=x
290 wsp(i,2)=y
300 wsp(i,3)=1
310 GOSUB 780:REM Wykresl. punkt lub odcinek
320 GOTO 250
330 INPUT "Podaj przesuniecie:x,y";p,q
340 GOSUB 850 :REM Ustaw macierz przeksz.
350 GOSUB 1030 :REM Mnozenie
360 GOSUB 1140 :REM Zobrazowanie
370 INPUT "Przesunieto:ponownie-r,koniec-k";nk$
380 IF nk$="r" THEN 410
390 IF nk$="k" THEN 1730
400 GOTO 370
410 INPUT "kasowanie obrazu-k,rysowanie-r";nk$
420 IF nk$="k" THEN 450
430 IF nk$="r" THEN 670
440 GOTO 410
450 INPUT"nowego obrazu-n,starego-s";nk$
460 IF nk$="n" THEN 620
470 IF nk$="s" THEN 490
480 GOTO 450
490 REM Kasowanie starego obrazu
500 GOSUB 1230:REM Rys.ukladu wspolrz.
510 FOR k=1 TO i
520 wsp(k,1)=nwsp(k,1)
530 wsp(k,2)=nwsp(k,2)
540 wsp(k,3)=nwsp(k,3)
550 NEXT k
560 GRAPHICS PEN 3
570 GOSUB 1630
580 INPUT "przesuniecie-p,dorysowanie-d";nk$
590 IF nk$="p" THEN 330
600 IF nk$="d" THEN 250
610 GOTO 580
620 REM Kasowanie nowego obrazu
630 GOSUB 1230
640 GRAPHICS PEN 3
650 GOSUB 1630
660 GOTO 580
670 FOR n=1 TO i:REM Rysowanie od poczatku
680 FOR k=1 TO 3
690 wsp(n,k)=0
700 nwsp(n,k)=0
710 NEXT k
720 NEXT n
730 GOTO 230
740 REM ****************************************
750 REM      PODPROGRAMY
760 REM ****************************************
770 REM
780 REM Rysowanie punktu lub odcinka
790 REM
800 GRAPHICS PEN 3:REM stare wspolrzedne
810 IF i=1 THEN PLOT wsp(i,1),wsp(i,2)
820 IF i>1 THEN DRAW wsp(i,1),wsp(i,2)
830 RETURN
840 REM
850 REM Ustawianie macierzy
860 REM
870 FOR k=1 TO 3
880 mat(k,k)=1
890 NEXT k
900 mat(1,2)=0
910 mat(2,1)=0
920 mat(1,3)=0
930 mat(2,3)=0
940 mat(3,1)=p
950 mat(3,2)=q
960 FOR k=1  TO i:REM Zerowanie nwsp
970 FOR l=1 TO 3
980 nwsp(k,l)=0
990 NEXT l
1000 NEXT k
1010 RETURN
1020 REM
1030 REM Mnozenie macierzy
1040 REM
1050 FOR n=1 TO i
1060 FOR k=1 TO 3
1070 FOR m=1 TO 3
1080 nwsp(n,k)=nwsp(n,k)+wsp(n,m)*mat(m,k)
1090 NEXT m
1100 NEXT k
1110 NEXT n
1120 RETURN
1130 REM
1140 REM Rysowanie nowej figury
1150 REM
1160 GRAPHICS PEN 0:REM PIORO nowych wspol.
1170 PLOT nwsp(1,1),nwsp(1,2)
1180 FOR n=2 TO i
1190 DRAW nwsp(n,1),nwsp(n,2)
1200 NEXT
1210 RETURN
1220 REM
1230 REM Rysowanie ukladu wspolrzednych
1240 REM
1250 CLG 1
1260 GRAPHICS PEN 2:REM Pioro ukladu wspol.
1270 FOR a=-200 TO 200 STEP 20
1280 PLOT -2,a
1290 DRAWR 4,0
1300 NEXT a
1310 PLOT 0,-208
1320 DRAW 0,208
1330 TAG
1340 MOVER -16,-20
1350 PRINT "Y";
1360 MOVER -310,0
1370 PRINT "20";
1380 TAGOFF
1390 PLOT -307,170 : REM Kreslenie podzialki
1400 DRAWR 0,-4
1410 DRAWR 0,2
1420 DRAWR 20,0
1430 DRAWR 0,2
1440 DRAWR 0,-4
1450 IF fl=1 THEN 1460 ELSE 1510
1460 FOR b=-200 TO 200 STEP 20
1470 FOR a=-320 TO 320 STEP 20
1480 PLOT a,b
1490 NEXT a
1500 NEXT b
1510 FOR a=-320 TO 320 STEP 20
1520 PLOT a,-2
1530 DRAWR 0,4
1540 NEXT a
1550 PLOT -320,0
1560 DRAW 320,0
1570 TAG
1580 MOVER -16,-4
1590 PRINT "X";
1600 TAGOFF
1610 RETURN
1620 REM
1630 REM:Rysowanie starej figury
1640 REM
1650 PLOT wsp(1,1),wsp(1,2)
1660 FOR n=2 TO i
1670 DRAW wsp(n,1),wsp(n,2)
1680 NEXT n
1690 RETURN
1700 REM
1710 REM Koniec
1720 REM
1730 WINDOW 1,40,1,25
1740 CLS
1750 CLG 1
1760 END
```

na płaszczyznę H=1. Jak pokazano na rysunku punktem centralnym rzutowania jest środek układu współrzędnych.

Podstawowa macierz transformacji rozmiaru 3×3 dla dwuwymiarowych jednorodnych współrzędnych może być podzielona na cztery części.

$$T = \left[\begin{array}{cc:c} a & b & p \\ c & d & q \\ \hdashline m & n & s \end{array}\right]$$

Jak uprzednio wykazano elementy *a, b, c, d* wpływają na szalowanie, obrót i skręcenie. Elementy m.in. umożliwiają przesunięcie, zaś p i q wpływają na rzutowanie. Pozostała część macierzy, element s powoduje ogólne skalowanie. Aby pokazać efekt ogólnego skalowania wykonamy następujące przekształcenie.

$$[X\ Y\ H] = [x\ y\ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix} = [x\ y\ s]$$

Teraz $X=x$, $Y=y$, $H=s$. Współrzędne punktu po przekształceniu wynoszą: $x^* = \frac{x}{s}$, $y^* = \frac{y}{s}$. W wyniku przekształcenia $[x\ y\ 1] \rightarrow [\frac{x}{s}\ \frac{y}{s}\ 1]$ ma miejsce ogólna zmiana skali wektora położenia. Gdy $s<1$ zachodzi powiększenie, a przy $s>1$ pomniejszenie skali.
Przykład programu realizującego przesunięcie figury na płaszczyźnie.

Po uruchomieniu programu „Przesunięcie" następuje przedstawienie zakresu umownych współrzędnych figur obrazowanych na ekranie monitora oraz pytanie o siatkę współrzędnych.
Na układzie współrzędnych przyjęto podziałkę wynoszącą 20 jednostek. W celu ułatwienia identyfikacji współrzędnych kreślonych figur wprowadza się siatkę współrzędnych, która pokrywa cały układ współrzędnych kropkami tworzącymi kwadraty o boku równym wielkości przyjętej podziałki. Wciskając klawisz „*t*" siatka współrzędnych będzie wyświetlana do końca działania programu. Naciśnięcie klawisza „*n*" powoduje efekt przeciwny. Algorytm działania programu jest następujący:

1) Definiowanie figury.
2) Określenie wartości przesunięcia.
3) Obrazowanie figury podstawowej oraz po przesunięciu.
4) Wykonanie spacji umożliwiających:
   a) definiowanie figury od początku,
   b) kasowanie jednej z otrzymanych figur,
   c) dorysowanie nowych elementów figury,
   d) wykonywanie przesunięcia,
   e) zakończenie działania programu.

Identyfikację poszczególnych części programu źródłowego z przedstawionym algorytmem umożliwiają komentarze zawarte w programie.

Uwaga eksploatacyjna

Koniec definiowania figury uzyskuje się wpisując wartość współrzędnej x>998 oraz dowolną wartość współrzędnej y.

Modyfikacje

Zmianę wielkości podziałki uzyskamy modyfikując linie 1270 oraz 1510 programu. Konsekwencją zmiany podziałki jest konieczność modyfikacji opisu podziałki (linia 1370 i 1420) oraz siatki współrzędnych (linie 1460 i 1470).

Uwaga praktyczna

Wpisując tekst instrukcją PRINT na pozycję kursora graficznego należy umieścić średnik na końcu instrukcji, gdyż jego pominięcie powoduje uzyskanie nieoczekiwanego efektu — **Sprawdź**. Jest to rezultat przeniesienia tekstu do nowej linii.

Andrzej CETERA

Biblioteka C 64

„Biblioteka". W naszej domowej bibliotece zapanował ogromny bałagan. Pora zatem na zrobienie porządków. Pomoże nam w tym prezentowany płogram. Wszystkie zbiory książkowe możemy umieścić na kasecie magnetofonowej (wg obliczeń autora ok. 2000 pozycji na 100 obrotów licznika magnetofonu). Nasz zbiór zostanie uporządkowany według haseł (nazwisk autorów) o długości do 20 znaków. Pod każdym hasłem (nazwiskiem) możemy umieścić różne dane, jak np.: liczbę tomów, rok wydania, liczbę stron, na której półce znajduje się pozycja itp. Dane te jednak nie mogą zawierać więoej niż 75 znaków. Może przepływ informacji w naszej skomputeryzowanej bibliotece nie będzie zbyt szybki (to tylko przecież magnetofon), niemniej program umożliwia wykorzystanie mikrokomputera do bardzo poważnych celów. Program można łatwo dostosować do wielkości naszej biblioteki, a także, wykorzystać go do innych celów.

Opracował:

Tadeusz CISEK, Warszawa

```
4040 IF LEFT$(B$(Q),LEN(B$))()B$ THEN 40
80
4050 PRINT'AUTOR:';B$(Q):PRINT

4060 PRINTTAB(5)C$(Q):PRINT
4070 N=N+1:IF INT(N/3)=N/3 THEN INPUT'WI
ECEJ--NACISNIJ RETURN';A$
4080 NEXT Q
4090 GOTO 40
5000 PRINTCHR$(147):PRINT
5010 INPUT'PODAJ AUTORA ';B$:PRINT
5020 FOR Q=0 TO A
5030 IF LEFT$(B$(Q),LEN(B$))()B$ THEN 50
90
5040 PRINTCHR$(147)'AUTOR:';B$(Q):PRINT
5050 PRINTC$(Q):PRINT
5060 INPUT'CZY CHCESZ USUNAC TEN ZAPIS (
T/N)';A$:PRINT
5070 IF A$()'T' THEN 5090
5080 B$(Q)='':C$(Q)='':D=D+1
5090 NEXT Q
5100 GOTO 40
6000 INPUT'COFNIJ KASETE, NACISNIJ RETUR
N';A$
6010 OPEN 1,1,2,'BIBLIOTEKA'
6020 PRINT# 1,(A-D)

6030 FOR Q=0TOA:IF B$(Q)=''THEN 6050
6040 PRINT# 1,B$(Q);CHR$(32)
6045 PRINT# 1,C$(Q)
6050 NEXT Q:PRINT# 1,'END'
6060 CLOSE 1: GOTO 40
7000 PRINT'SORTOWANIE...':FORQ=1 TO A
7005 D$(Q)=B$(Q):E$(Q)=C$(Q)
7010 A$(Q)=D$(Q)
7020 NEXT Q
7030 K=0
7040 Q=1
7050 FOR J=2 TO A
7060 IF A$(Q)(A$(J) THEN 7080
7070 Q=J
7080 NEXT J
7090 A$(Q)='ZZZ'
7100 K=K+1
7110 B$(K)=D$(Q):C$(K)=E$(Q)
7120 IF K=A+1 THEN 7140
7130 GOTO 7040

7140 GOTO 40
8000 REM PROCEDURA INICJUJACA
8002 A$=''
8005 OPEN 1,1,2,'BIBLIOTEKA'
8010 PRINT# 1,0;A$;A$
8020 CLOSE 1:RETURN

1060 IF A$='N' THEN 1080
1070 A=A+1
1071 IF A-D(300 THEN 1000
1072 IF A-D=300 THEN PRINT'OSTATNI ZAPIS
 W PLIKU'
1074 IF A-D)=300 THEN PRINT'BLAD, PLIK P
ELNY':PRINT'KONIEC PLIKU'
1076 PRINT'URUCHOM PONOWNIE PROGRAM, INIC
JUJAC NOWA SEKCJE NA TASMIE'
1078 GOTO 1000
1080 GOTO 40
2000 PRINT'PODAJ AUTORA DLA KTOREGO NAST
API ZMIANA '
2010 INPUT'REDAKCJI';B$ :PRINT
2020 FOR Q=0TOA
2030 IF LEFT$(B$(Q),LEN(B$))()B$ THEN 21
10
2040 PRINTCHR$(147)
2050 PRINT'BIBLIOTEKA':PRINT
2060 PRINTC$(Q):PRINT
2070 INPUT'CZY CHCESZ REDAGOWAC TEN ZAPI
S(T/N)';A$:PRINT
2080 IF A$='N' THEN 2110:PRINT
2090 INPUT'PONOWNIE WPROWADZ WSZYSTKIE D
ANE';C$(Q):INPUT A$:PRINT
2100 PRINT'CZY POPRAWNIE (T/N)?';C$(Q):I
NPUT A$ :PRINT

2105 IF A$()'T'THEN PRINT'NIE MOZE BYC D
LUZSZY NIZ 75 LITER':GOTO 2090:PRINT
2107 INPUT'PODANIE INNEGO(T/N)';A$:IF A$
()'T' THEN 2120
2110 NEXT Q
2120 PRINT'KONIEC PLIKU':GOTO 40
3000 PRINT'WYBIERZ:1)WYLISTOWANIE
                    KATALOGU ZAPISOW'
3005 PRINT'          2)WYLISTOWANIE ZAPISO
W                     WEDLUG AUTOROW'
3007 INPUT Y
3010 ON Y GOTO 3300,3020
3020 INPUT'PODAJ AUTORA ';H$ :PRINT
3040 PRINT'NUMER','AUTOR ':N=1:PRINT
3050 FOR Q=0 TO A
3060 IF LEFT$(B$(Q),LEN(H$))()H$ THEN 30
90
3070 PRINT N,B$(Q)
3080 N=N+1:IF INT(N/11)=N/11 THEN INPUT'
WIECEJ--NACISNIJ RETURN';A$

3090 NEXT Q
3100 PRINT'----------------':PRINT:GOTO
 40
3300 PRINT CHR$(147) :PRINT
3310 PRINT'NUMER','AUTOR ':N=1:PRINT
3320 FOR R=65 TO 90
3330 FOR Q=0 TO A
3340 IF LEFT$(B$(Q),1)()CHR$(R) THEN 336
0
3350 PRINTN,B$(Q):N=N+1:IF INT(N/11)=N/1
1THEN INPUT'WIECEJ--NACISNIJ RETURN';A$
3360 NEXT Q,R
3370 GOTO 40
4000 PRINTCHR$(147):PRINT
4010 INPUT'PODAJ AUTORA ';B$:PRINT
4020 N=1
4030 FOR Q=1TOA
```

```
5 PRINT'BIBLIOTEKA'
10 PRINTCHR$(147)TAB(55)'BIBLIOTEKA'
20 CLR:PRINT:DIM A$(300),B$(300),C$(300)
,D$(300),E$(300):FL=0:D=0
25 PRINT'WPROWADZ ODDZIELNA KASETE NA PL
IK DANYCH':PRINT
27 PRINT'COFNIJ KASETE;NACISNIJ RETURN '
;:INPUT'GDY JESTES GOTOW';A$:PRINT
30 PRINT'TEN PROGRAM MUSI BYC ZAINICJOWA
NY'
32 PRINT'KIEDY STOSOWANY JEST PO RAZ PIE
RWSZY, CZY CHCESZ GO '
33 INPUT'ZAINICJOWAC (T/N)';A$:IF A$()'T
' THEN 35
34 GOSUB 8000:PRINT:INPUT'PRZEWIN TASME,
 NACISNIJ RETURN ';A$
35 OPEN 1,1:INPUT# 1,Z:IF Z=0 THEN FL=1:
GOTO 40
36 IF Z)=295 THEN PRINT'PIEC REKORDOW DO
 KONCA PLIKU'
37 FOR A=0TOZ:INPUT# 1,B$(A):IF B$(A)='E
ND' THEN GOTO 40

38 INPUT# 1,C$(A):IF C$(A)='END'THEN GOT
O 40
39 NEXT
40 PRINT'WYBOR ROZKAZOW':PRINT
42 CLOSE 1,1
50 PRINT'ROZKAZ','FUNKCJA':PRINT:PRINT
60 PRINT'1','WPROWADZENIE NOWEJ POZYCJI'
:PRINT
70 PRINT'2','DRUKOWANIE LISTY POZYCJI':P
RINT
80 PRINT'3','SZUKANIE POZYCJI':PRINT
90 PRINT'4','ZMIANA REDAKCJI POZYCJI':PR
INT
100 PRINT'5','KASOWANIE POZYCJI':PRINT
110 PRINT'6','SORTOWANIE':PRINT
115 PRINT'7','NAGRANIE PLIKU':PRINT
120 INPUT A$
130 IF A$='1'THEN 1000
140 IF A$='2'THEN 3000
150 IF A$='3'THEN 4000
160 IF A$='4'THEN 2000
170 IF A$='5'THEN 5000

180 IF A$='6'THEN 7000
185 IF A$='7' THEN 6000
190 PRINT'ZLY ROZKAZ--PODAJ JEDEN Z '
195 PRINT'NASTEPUJACYCH:'
200 GOTO 50
1000 PRINT'PODAJ AUTORA (DLUGOSC  DO 20
':INPUT'LITER)';B$(A)
1010 IF LEN(B$(A))20 THEN 1000
1020 PRINT'PODAJ TYTUL LUB INNE DANE (NA
JWYZEJ ';
1025 PRINT'75 ZNAKOW)':INPUT C$(A)
1030 PRINT'CZY POPRAWNIE (T/N)?';C$(A):I
NPUT A$:IF A$()'T' THEN 1020
1050 INPUT'WPROWADZENIE NASTEPNEGO(T/N)'
;A$
```

Jak narysować CPC 6128

Ten program na AMSTRAD-a może służyć jako pomoc w nauce początków geometrii. Pokazuje, jak narysować symetralną odcinka, dwusieczną kąta i trójkąt w trzech wersjach, gdy mamy dane: trzy boki, dwa boki i kąt, bok i dwa kąty, posługując się cyrklem i linijką bez podziałki. Program wykorzystuje tryb wysokiej rozdzielczości — MODE 2 — 560x400 punktów.

W programie można wyróżnić 10 rozdzielnych modułów. Pierwszy z nich to moduł sterujący (linie 10—180), 5 następnych realizuje kolejno funkcje. Moduły te nie wymagają dodatkowych opisów — teksty w nich zawarte dokładnie opisują wykonywane czynności. Następne trzy moduły to podprogramy realizujące funkcje wykorzystywane przez 5 modułów podstawowych i im należy poświęcić więcej uwagi, tym bardziej że mogą być one wykorzystane w innych programach.

Podprogram „PRZESUWANIE ODCINKA" — linie 1890—2000. Podprogram przesuwa odcinek w 30 krokach z jednego miejsca na ekranie w drugie, pokazując jego ruch. Należy podać mu współrzędne odcinka wyjściowego (zmienne x11, y11; x12, y12) i końcowego (zmienne x21, y21; x22, y22).

Podprogram „RYSOWANIE ŁUKU" — linie 2000—2050. Rysuje na ekranie łuk o zadanych parametrach, pokazując jednocześnie ruch promienia łuku. Wykorzystuje zmienne: x, y — współrzędne środka, r — promień, k, l — współrzędne kątowe początku i końca łuku, w stopniach, liczone zgodnie z ruchem wskazówek zegara, przyjęto, że 0 stopnie odpowiada godz. 12.

Podprogram „RYSOWANIE ODCINKA" — linie 2060—2100. Podprogram rysuje na ekranie odcinek o zadanych współrzędnych (x11, y11; x12, y12) dzieląc go na 400 części, dużo wolniej niż instrukcja DRAW.

Ostatni podprogram to definicje polskich liter nie występujących na klawiaturze AMSTRADA. Wykorzystano klawiaturę numeryczną (f1—f9). Małe litery otrzymujemy naciskając SHIFT i wybrany klawisz F1—f9, duże KONTROL i f1—f9 — patrz rysunek. Jest to system w tej chwili znany wśród wielu użytkowników AMSTRADA i wygodny.

Wpisywanie programu najwygodniej rozpocząć od wpisania podprogramu „POLSKIE LITERY" — litery 2110—2390. Po wpisaniu należy podprogram uruchomić (tak jak każdy program w BASIC-u) i od tej chwili posługujemy się polskim alfabetem, co ułatwia wpisywanie tekstów zawartych w programie.

K. P.

```
10 MODE 1:DEG:CLS:FRAME
20 GOSUB 2110
30 LOCATE 9,1:PRINT "J A K   N A R Y S O W A Ć"
40 PRINT:PRINT:PRINT
50 PRINT:PRINT "   SYMETRALNA ODCINKA.....1"
60 PRINT:PRINT "   DWUSIECZNA KĄTA.....2"
70 PRINT:PRINT "   TRÓJKĄT (3 boki).....3"
80 PRINT:PRINT "   TRÓJKĄT (2 boki,kąt).....4"
90 PRINT:PRINT "   TRÓJKĄT (bok,2 kąty).....5"
100 LOCATE 1,20
110 PRINT "Do rysowania używamy linijki i cyrkla"
120 LOCATE 6,24:PRINT "Powrot do menu-dowolny znak"
130 a$=INKEY$:IF a$="1" GOTO 190
140 IF a$="2" GOTO 380
150 IF a$="3" GOTO 560
160 IF a$="4" GOTO 900
170 IF a$="5" GOTO 1330
180 GOTO 130
190 'rem "****** S Y M E T R A L N A ******"
200 MODE 2:PRINT CHR$(22);CHR$(1)
210 LOCATE 30,1:PRINT "SYMETRALNA ODCINKA"
220 LOCATE 30,1:PRINT "------------------"
230 a=200:LET b=400:c=170:PLOT a,c
240 DRAW b,c:k=30:l=60:x=a:y=c:r=155
250 PRINT :PRINT:PRINT
260 PRINT "1-Rysujemy 2 łuki z"
270 PRINT "  lewego końca odcinka"
280 GOSUB 2000:k=120:l=150:x=a:y=c
290 GOSUB 2000:PRINT :PRINT:PRINT
300 LOCATE 53,7:PRINT "2-Rysujemy 2 łuki z"
310 LOCATE 55,8:PRINT "prawego końca odcinka"
320 k=210:l=240:x=b:y=c:GOSUB 2000
330 k=300:l=330:x=b:y=c:GOSUB 2000
340 LOCATE 2,22:PRINT "3-Przez punkty przecięcia"
350 LOCATE 4,23:PRINT "przechodzi symetralna odcinka"
360 x11=300:y11=30:x12=300:y12=300
370 GOSUB 2060:CLEAR INPUT:GOTO 540
380 ' "***** D W U S I E C Z N A  K Ą T A *****"
390 MODE 2:PRINT CHR$(22);CHR$(1)
400 LOCATE 30,1:PRINT "DWUSIECZNA KĄTA"
410 LOCATE 30,1:PRINT "--------------"
420 PLOT 200,150:DRAW 400,100
430 MOVE 200,150:DRAW 400,200
440 LOCATE 3,14:PRINT "1-Z wierzchołka kąta"
450 LOCATE 5,15:PRINT "wykreślamy łuk"
460 x=200:y=150:l=120:k=70:r=150:GOSUB 2000
470 LOCATE 30,8:PRINT "2-Z punktów przecięcia wykreśl
amy dwa łuki"
480 x=346:y=187:k=90:l=130:r=150:GOSUB 2000
490 x=346:y=114:k=60:l=100:r=150:GOSUB 2000
500 LOCATE 3,23:PRINT "3-Przez wierzchołek kąta i pun
kt przecięcia"
510 LOCATE 46,23:PRINT " łuków przechodzi dwusieczna
kąta"
520 x11=170:y11=150:x12=510
530 y12=150:GOSUB 2060:CLEAR INPUT
540 IF INKEY$>=" " GOTO 10
550 GOTO 540
560 'rem "****** T R Ó J K Ą T (3 boki) ******"
570 MODE 2:DEG:PRINT CHR$(22);CHR$(1)
580 LOCATE 30,1:PRINT "T R Ó J K Ą T"
590 LOCATE 30,1:PRINT "-----------------"
600 LOCATE 40,5:PRINT "Mamy dane trzy boki trojkąta"
610 LOCATE 3,3:PRINT "a"
620 PLOT 50,356:DRAW 200,356
630 LOCATE 3,5:PRINT "b"
640 PLOT 50,324:DRAW 150,324
650 LOCATE 3,7:PRINT "c"
660 PLOT 50,296:DRAW 250,296
670 LOCATE 1,21:PRINT "1-Rysujemy odcinek np.'c'"
680 x11=50:x12=250:y11=296:y12=296
690 x21=200:x22=400:y21=100:y22=100
700 GOSUB 1890:LOCATE 39,20:PRINT "c"
710 LOCATE 3,15:PRINT "2-Z końca 'c' rysujemy"
720 LOCATE 3,16:PRINT "łuk o promieniu 'a'"
730 x11=50:x12=200:y11=356:y12=356
740 x21=200:x22=314:y21=100:y22=196
750 GOSUB 1890:LOCATE 35,16:PRINT "a"
760 x=200:y=100:r=150:k=55:l=70:GOSUB 2000
770 LOCATE 50,14:PRINT "3-Z drugiego końca 'c'"
780 LOCATE 52,15:PRINT "rysujemy łuk o promieniu 'b'"
790 x11=50:x12=150:y11=324:y12=324
800 x22=400:y22=100:x21=310:y21=142
810 GOSUB 1890:x=400:y=100:r=100
820 LOCATE 46,16:PRINT "b"
830 k=300:l=325:GOSUB 2000:LOCATE 15,24
840 PLOT 200,100:DRAW 314,196,1,1
850 PLOT 400,100:DRAW 310,142,1,1
860 PRINT "4-Łączmy punkt przecięcia łuków z końcami
odcinka 'c'"
870 x11=200:y11=100:x12=331:y12=172:GOSUB 2060
880 x11=400:y11=100:x12=331:y12=172:GOSUB 2060
890 CLEAR INPUT:GOTO 540
900 '******* T R Ó J K Ą T ( 2 boki,kąt) *******
910 MODE 2:DEG:PRINT CHR$(22);CHR$(1)
920 LOCATE 30,1: PRINT "T R Ó J K Ą T"
930 LOCATE 30,1:PRINT "-----------------"
940 LOCATE 40,3:PRINT "Mamy dane dwa boki trójkąta"
950 LOCATE 40,5:PRINT "i kąt miedzy nimi zawarty"
960 LOCATE 3,3:PRINT "a"
970 PLOT 50,356:DRAW 250,356
980 LOCATE 3,5:PRINT "b"
990 PLOT 50,324:DRAW 200,324
1000 LOCATE 1,10:PRINT "kąt"
1010 PLOT 50,244:DRAW 200,244
1020 PLOT 50,244:DRAW 161,308
1030 LOCATE 1,21:PRINT "1-Rysujemy odcinek np.'a'"
1040 x11=50:x12=250:y11=356:y12=356
1050 x21=200:x22=400:y21=100:y22=100
1060 GOSUB 1890:LOCATE 39,20:PRINT "a"
1070 LOCATE 1,15:PRINT "2-Przenosimy kąt"
1080 x=50:y=244:r=100:k=55:l=95:GOSUB 2000
1090 x11=50:y11=244:x12=149:y12=236:x21=200
1100 y21=100:x22=276:y22=164:GOSUB 1890
1110 MOVE 200,100:DRAW 276,164,1,1
1120 x=200:y=100:r=100:k=50:l=100:GOSUB 2000
1130 x11=150:y11=244:x12=136:y12=294:GOSUB 2060
1140 x11=136:y11=294:x12=150:y12=244:x21=264
1150 y21=136:x22=300:y22=100:GOSUB 1890
1160 MOVE 300,100:DRAW 264,136,1,1
1170 x=300:y=100:r=52:k=320:l=360:GOSUB 2000
1180 LOCATE 48,12:PRINT "3-Rysujemy pomocniczą półpro
stą"
1190 LOCATE 50,13:PRINT "i odkładamy na niej odcinek
'b'"
1200 x11=200:y11=100:x12=373:y12=200:GOSUB 2060
1210 MOVE 200,100:DRAW 373,200,1,1
1220 x11=50:x12=200:y11=324:y12=324:x21=200
1230 y21=100:x22=306:y22=206:GOSUB 1890
1240 x=200:y=100:r=150:k=50:l=70:GOSUB 2000

1250 MOVE 200,100:DRAW 306,206,1,1
1260 LOCATE 31,16:PRINT "b"
1270 x11=200:y11=100:x12=329:y12=175:GOSUB 2060
1280 LOCATE 57,16:PRINT "4-Łączymy końce odcinków"
1290 LOCATE 59,17:PRINT "-rysujemy bok 'c'"
1300 x11=400:y11=100:x12=329:y12=175:GOSUB 2060
1310 LOCATE 47,16:PRINT "c"
1320 CLEAR INPUT:GOTO 540
1330 '******* T R Ó J K Ą T ( bok,2 kąty) ********
1340 MODE 2:DEG:PRINT CHR$(22);CHR$(1)
1350 LOCATE 30,1: PRINT "T R Ó J K Ą T"
1360 LOCATE 30,1:PRINT "-----------------"
1370 LOCATE 40,3:PRINT "Mamy dane bok trójkąta"
1380 LOCATE 40,5:PRINT "i 2 kąty do niego przyległe"
1390 LOCATE 3,3:PRINT "a"

1400 PLOT 50,356:DRAW 250,356
1410 LOCATE 3,6:PRINT "kąt 1"
1420 PLOT 50,276:DRAW 200,276
1430 PLOT 50,276:DRAW 161,340
1440 LOCATE 3,14:PRINT "kąt 2"
1450 PLOT 50,140:DRAW 200,140
1460 PLOT 50,140:DRAW 156,246
1470 LOCATE 20,22:PRINT "1-Rysujemy odcinek 'a'"
1480 x11=50:x12=250:y11=356:y12=356
1490 x21=200:x22=400:y21=100:y22=100
1500 GOSUB 1890:LOCATE 39,20:PRINT "a"
1510 LOCATE 1,18:PRINT "2-Przenosimy kąty"
1520 x=50:y=276:r=100:k=55:l=95:GOSUB 2000
1530 x11=50:y11=276:x12=149:y12=264:x21=200
1540 y21=100:x22=276:y22=164:GOSUB 1890

1550 MOVE 200,100:DRAW 276,164,1,1
1560 x=200:y=100:r=100:k=50:l=100:GOSUB 2000
1570 x11=150:y11=276:x12=136:y12=326
1580 GOSUB 2060
1590 x11=150:y11=276:x12=136:y12=326
1600 x22=264:y22=136:x21=300:y21=100
1610 GOSUB 1890
1620 MOVE 300,100:DRAW 264,136,1,1
1630 x=300:y=100:r=52:k=320:l=360:GOSUB 2000
1640 '******DRUGI KĄT*************
1650 x=50:y=140:r=90:k=40:l=95:GOSUB 2000
1660 x11=50:y11=140:x12=139:y12=133:x22=400
1670 y22=100:x21=311:y21=93:GOSUB 1890
1680 MOVE 400,100:DRAW 311,93,1,1
1690 x=400:y=100:r=90:k=270:l=330:GOSUB 2000
1700 x11=140:y11=140:x12=113:y12=203:GOSUB 2060
1710 x11=140:y11=140:x12=113:y12=203:x21=310
1720 y21=100:x22=333:y22=164:GOSUB 1890
1730 MOVE 310,100:DRAW 333,164,1,1
1740 x=310:y=100:r=68:k=15:l=35:GOSUB 2000
1750 LOCATE 48,12:PRINT "3-Rysujemy 2 pomocnicze półp
roste"

1760 x11=200:y11=100:x12=373:y12=200:GOSUB 2060
1770 MOVE 200,100:DRAW 373,200,1,1
1780 x11=400:y11=100:x12=300:y12=200:GOSUB 2060
1790 MOVE 400,100:DRAW 300,200,1,1
1800 LOCATE 54,16:PRINT "4-Łączymy końce odcinka 'a'"
1810 LOCATE 57,17:PRINT "z punktem przecięcia"
1820 LOCATE 57,18:PRINT "półprostych - rysujemy"
1830 LOCATE 57,19:PRINT "boki 'b' i 'c'"
1840 x11=200:y11=100:x12=328:y12=174:GOSUB 2060
1850 LOCATE 34,16:PRINT "b"
1860 x11=400:y11=100:x12=328:y12=174:GOSUB 2060
1870 LOCATE 47,16:PRINT "c"
1880 CLEAR INPUT:GOTO 540
1890 'rem "****** PRZESUWANIE ODCINKA ******"

1900 dx1=(x21-x11)/30:dx2=(x22-x12)/30
1910 dy1=(y21-y11)/30:dy2=(y22-y12)/30
1920 FRAME:FOR i=1 TO 30
1930 x11=x11+dx1:x12=x12+dx2
1940 y11=y11+dy1:y12=y12+dy2
1950 MOVE x11,y11:DRAW x12,y12,1,1
1960 FOR p=0 TO 50:NEXT
1970 MOVE x11,y11:DRAW x12,y12,1,1
1980 NEXT:MOVE x11,y11
1990 DRAW x12,y12:RETURN

2000 'rem "******** RYSOWANIE ŁUKU *********"
2010 FOR i=k TO l STEP 5
2020 MOVE x,y:DRAW x+r*SIN(i),y+r*COS(i),1,1
2030 MOVE x,y:DRAW x+r*SIN(i),y+r*COS(i),1,1
2040 DRAW x+r*SIN(i-5),y+r*COS(i-5),1,3
2050 NEXT:RETURN

2060 ' "*** R Y S O W A N I E  O D C I N K A ***"
2070 dx=(x12-x11)/400:dy=(y12-y11)/400
2080 MOVE x11,y11:FOR i=1 TO 400
2090 x11=x11+dx:y11=y11+dy
2100 DRAW x11,y11,1,3:NEXT:RETURN
2110 'rem "****** P O L S K I E  L I T E R Y ******"
2120 SYMBOL AFTER 170

2130 SYMBOL 171,0,0,120,12,124,204,118,3
2140 SYMBOL 172,24,60,102,102,126,102,102,3
2150 SYMBOL 173,24,0,60,102,96,102,60,0
2160 SYMBOL 174,24,60,102,192,192,102,60,0
2170 SYMBOL 175,0,0,60,102,126,96,60,6
2180 SYMBOL 176,254,98,104,120,104,98,254,3
2190 SYMBOL 177,56,24,28,24,56,24,60,0
2200 SYMBOL 178,240,96,112,96,226,102,254,0
2210 SYMBOL 179,24,0,220,102,102,102,102,0
2220 SYMBOL 180,24,198,230,246,222,206,198,0
2230 SYMBOL 181,24,0,60,102,102,102,60,0
2240 SYMBOL 182,12,56,108,198,198,108,56,0
2250 SYMBOL 183,24,0,60,96,60,6,124,0
2260 SYMBOL 184,24,60,96,60,6,102,60,0
2270 SYMBOL 185,24,0,126,76,24,48,126,0
2280 SYMBOL 186,254,198,140,126,48,102,254,0
2290 SYMBOL 187,12,24,126,76,24,48,126,0
2300 SYMBOL 188,48,254,172,24,50,102,254,0
2310 KEY DEF 10,1,55,171,172
2320 KEY DEF 11,1,56,173,174
2330 KEY DEF 3,1,57,175,176
2340 KEY DEF 20,1,52,177,178
2350 KEY DEF 12,1,53,179,180
2360 KEY DEF 4,1,54,181,182
2370 KEY DEF 13,1,49,183,184
2380 KEY DEF 14,1,50,185,186
2390 KEY DEF 5,1,51,187,188
2400 RETURN
```

CELL - komórka, adres,
CELLAR - stos (w pamięci),
CELLULAR ARRAY - układ komórkowy,
CENTER - ośrodek,
CENTER FREQUENCY - częstotliwość nośna,
CENTRAL COMPUTER - komputer centralny,
CENTRAL PROCESSING UNIT (CPU) - jednostka centralna,
CENTRAL PROCESSOR - jednostka centralna,
CENTRAL STORAGE - pamięć operacyjna, pamięć centralna,
CERTIFIER - maszyna do regeneracji taśm magnetycznych,
CHAD(DED) TAPE - taśma dziurkowana całkowicie,
CHADLESS TAPE - taśma dziurkowana niecałkowicie,
CHADS - ścinki papierowe (powstałe przy dziurkowaniu taśmy lub karty),
CHAFF - konfetti, wycinki (kart lub taśm dziurkowanych),
CHAIN - łańcuch, szereg, zbiór liniowo uporządkowany,
CHAIN CODE - kod łańcuchowy,
CHAIN PRINTER - drukarka łańcuchowa,
CHAINED - także: doładowywany, dopinana (procedura),
CHAINED PROGRAM - program dołączany,
CHAINING - łączenie dotyczące adresowania pamięci masowych, łączenie (np. rozkazów), także: dzielenie programu na części, tworzenie łańcucha, wiązanie łańcuchowe,
CHANCHANGE - zmiana kanału,
CHANGE - zmiana, zmieniać,
CHANGE FILE - kartoteka zmian, kartoteka aktualizująca,
CHANGE OVER - przełączać,
CHANGE RECORD - zapis (rekord) zmian,
CHANGE TYPE - taśma zmian (dot. organizacji taśmy magnetycznej),
CHANGELABLE STORAGE - pamięć wymienna,
CHANGEOVER ONE-FOR-ONE - wprowadzenie nowego systemu pracy z jednoczesnym usunięciem starego,
CHANNEL - kanał,
CHANNEL ADDRESS WORD - słowo adresowe kanału,
CHANNEL CAPACITY - pojemność (zdolność przepustowa) kanału,
CHANNEL CONTROLLER - urządzenie sterujące kanałem,
CHANNEL INSTRUCTION - rozkaz kanałowy,
CHANNEL JUMP INSTRUCTION - rozkaz skoku kanałowy,
CHANNEL LOAD - zajętość kanału,
CHANNEL PROGRAM - program kanałowy,
CHAPTER - rozdział (programu),
CHARACTER - znak,
CHARACTER ADDRESSING - adresowanie pozycji, adresowanie znaku,
CHARACTER ADJUSTMENT - także: przesunięcie adresu,
CHARACTER ARITHMETIC - arytmetyka znakowa,
CHARACTER CODE MATRIX - matryca kodów znaków,
CHARACTER COMPUTER - komputer pozycyjny, komputer znakowy,
CHARACTER CONCENTRATOR - konwerter znaków,
CHARACTER CONVERTER - konwerter znaków,
CHARACTER DEFINITION TAG - znak definicji typu (na końcu nazwy zmiennej),
CHARACTER DENSITY - gęstość zapisu znaków,
CHARACTER ERROR PROBABILITY - stopa błędów znakowa,
CHARACTER ERROR RATE - wskaźnik błędów w znakach,
CHARACTER GENERATOR - generator znaków,
CHARACTER MACHINE - komputer pozycyjny, komputer znakowy,
CHARACTER PRINTER - drukarka znakowa, drukarka szeregowa,
CHARACTER READER - czytnik znaków,
CHARACTER RECOGNITION - rozpoznawanie znaków,
CHARACTER SET - zestaw znaków, także: zestaw przebitkowy,
CHARACTER STRING - ciąg znaków,
CHARACTER SUBSET - podzbiór znaków,
CHARACTER TRANSLATOR - konwerter znaków,
CHARACTER-AT-A-TIME PRINTER - drukarka znakowa,
CHARACTER-ORGANIZED STORAGE - pamięć o organizacji znakowej,
CHARACTERISTIC - cecha, cecha charakterystyczna, charakterystyka,
CHARGE-COUPLED DEVICE (CCD) - element ze sprzężeniem ładunkowym,
CHART - schemat, rysunek (techniczny), tabela (statystyczna), wykres,
CHARTING LAYOUT - wykres zamierzonej koncepcji (działania),
CHECK - także: analiza,
CHECK BIT - bit kontrolny,
CHECK CHARACTER - znak kontrolny,
CHECK COLUMN - kolumna kontrolna,
CHECK COMMAND - rozkaz weryfikacji,
CHECK DIGIT - cyfra kontrolna,
CHECK IN - wpisać się na listę, nawiązać łączność,
CHECK LIST - zestawienie (spis) kontrolne,
CHECK METER - przyrząd pomiarowy kontrolny,
CHECK NUMBER - numer kontrolny,
CHECK OUT - sprawdzać, kontrolować, test sprawdzający, także: wypisać się, zakończyć seans łączności,
CHECK POINT - punkt kontrolny,
CHECK READING - czytanie kontrolne, czytanie sprawdzające,
CHECK SUM - suma kontrolna,
CHECK TIME - czas sprawdzania,
CHECK TRACK - ścieżka kontrolna,
CHECKING CIRCUIT - układ kontrolny, układ sprawdzający,
CHECKING PROGRAM - program sprawdzający,
CHECKPOINT LABEL - etykieta punktu wznowienia,
CHIEF PROGRAMMER TEAM - zespół programisty wiodącego (zespół głównego programisty),
CHILD PROCESS - przebieg potomka (dot. ciągu programów),
CHIP - struktura półprzewodnikowa, moduł, kość (półprzewodnika),
CHIP BONDING - mikromontaż struktury,
CHIP BOX - pojemnik na ścinki (np. w dziurkarce kart, dziurkarce taśmy papierowej),
CHIP CIRCUIT - układ scalony pastylkowy,
CHIP ENABLE (CE) - sygnał wyboru struktury pamięci, uaktywnienie modułu,
CHIP ENABLE INPUT - wejście uaktywnienia,
CHIP SELECT - wybór modułu,
CHIP SELEKT INPUT - wejście wyboru,
CHIPPINGS - ścinki papierowe (powstałe przy dziurkowaniu taśmy lub karty),
CHOICE - wybór, dobór (np. maszyny),
CHROMA KEY - kluczowanie kolorem (TV),
CIM - patrz: COMPUTER INPUT MICROFILM,
CIPHER - cyfra, także: szyfr, szyfrować, kodować,
CIRCUIT - układ, obwód elektryczny,
CIRCUIT ALGEBRA - algebra schematów,
CIRCUIT ELEMENT - element przełączający,
CIRCULAR SHIFT - przesunięcie cykliczne,
CIRCULATING DECIMAL - ułamek dziesiętny okresowy,
CIRCULATING MEMORY - pamięć na liniach opóźniających,
CIRCULATING STORE - pamięć na liniach opóźniających ,
CISC - patrz: COMPLEX INSTRUCTION SET COMPUTER,
CIU - patrz: COMPUTER INTERFACE UNIT,
CLAMPING - obcinanie, zakończenie wykreślania wektorów na brzegu ekranu,
CLASS - rodzaj,
CLASSIFIED INFORMATION - informacja zastrzeżona,
CLAUSE - klauzula, warunek,
CLEANER - filtr,
CLEAR - kasować, wyzerować (np. licznik, stan rejestru), wymazać (np. pamięć maszyny),
CLEAR BAND - miejsce na dokumencie, które nie może być pokryte drukiem,
CLEAR STATE - start czysty,
CLEAR THE REGISTER - wyzerować rejestr,
CLEARANCE - kasowanie, wymazywanie, przywracanie do położenia wyjściowego,
CLEARING - kasowanie, wymazywanie, przywracanie do położenia wyjściowego,
CLEARING ROUTINES - programy wymazywania danych,
CLIPPING - obcinanie, zakończenie wykreślania wektorów na brzegu ekranu,
CLOCK - zegar, licznik, czujnik zegarowy,
CLOCK CIRCUIT - układ generacji sygnałów zegarowych,
CLOCK DRIVER - zasilacz zegarowy,
CLOCK FREQUENCY - częstotliwość cyklu zegarowego,
CLOCK GENERATOR - generator zegarowy,
CLOCK PERIOD - takt zegarowy,
CLOCK RATE - częstotliwość cyklu zegarowego,
CLOCK SIGNAL - sygnał zegarowy,
CLOCK TRACK - ścieżka synchronizująca (zegarowa),
CLOCKING - synchronizowanie, taktowanie (impulsów, sygnałów),
CLOSED LOOP - pętla zamknięta (rozkazów),
CLOSED SHOP - w żargonie: niedostępny (zamknięty) ośrodek obliczeniowy,
CLOSED SUBROUTINE - podprogram zamknięty,
CLOSED-LOOP SYSTEM - praca sprzężona z procesem w układzie zamkniętym,
CLOSELY COUPLED MICROCOMPUTER SYSTEMS - systemy mikrokomputerowe silnie powiązane (komunikacja przez wspólną pamięć operacyjną),
CLOSING A FILE - zamykanie pliku (zbioru),
CLUSTER - niepodzielna jednostka allokacji (porcja), fragment pliku (zbioru), klaster,
CLUSTER SET - pochodna zbioru, zbiór punktów skupienia,
CMC-7 FONT - pismo CMC-7,
CNC - patrz: COMPUTER NUMERICAL CONTROL,
CO-PROCESOR - procesor dodatkowy (wykonuje tylko rozkazy przeznaczone dla siebie, resztę pomija),
COAXIAL CABLE - kabel koncentryczny, kabel współosiowy,
COAXIAL TRANSCEIVER INTERFACE (CTI) - sprzężenie nadajnika-odbiornika z kablem,
CODASYL - patrz: CONFERENCE ON DATA LANGUAGE.

CODE - kod źródłowy, kod, także: program, kod programu,
CODE CHECK - kontrola kodu,
CODE COMBINATION - ciąg kodowy,
CODE CONVERTER - przetwornik kodu,
CODE NUMBER - liczba zakodowana,
CODE SEGMENT - segment programu,
CODE SET - lista kodu,
CODE SWITCHOVER - przełączanie kodu,
CODE SYMBOL - symbol kodowy,
CODE SYMBOL POSITION - pozycja kodowa, rząd kodowy,
CODE TRANSLATION - przekodowanie,
CODE TREE - drzewo kodowe, schemat kodowy,
CODED GRAPHIC - grafika konturowa,
CODED IMAGE - obraz konturowy (rysunek liniowy),
CODER - koder, urządzenie kodujące,
CODER NETWORK - tablica kodowania, matryca kodowania,
CODEWORD - słowo kodu,
CODIFY - kodować,
CODING - kodowanie,
CODING FORM - formularz programowy, formularz kodowy,
CODING LANGUAGE - język kodowania,
CODING MATRIX - tablica kodowania, matryca kodowania,
CODING SHEET - formularz programowy, formularz kodowy,
COEFFICIENT - współczynnik,
COEFFICIENT FIELD - ciało współczynników,
COINCIDE - pokrywać się z ..., zbiegać się z ...,
COINCIDENT-CURRENT MEMORY (CCM) - pamięć koincydencyjna,
COLD START - zimny start także: proces ładowania i inicjacji systemu operacyjnego,
COLLECT - gromadzić, zbierać (np. dane),
COLLECTING - gromadzenie, zbieranie (np. danych),
COLLECTION - zbiór, także: zbieranie, gromadzenie,
COLLECTION OF INFORMATION - zbiór informacji,
COLON - dwukropek,
COLOR SUPRESSED MODE - tryb z wygaszonym kolorem,
COLUMN - kolumna (np. karty dziurkowanej), także: rubryka pionowa w tabeli,
COLUMNAL LAYOUT - rozmieszczenie rubryk (np. na zestawieniu),
COM - patrz: COMPUTER OUTPUT MICROFILM,
COM DEVICE - rejestrator mikrofilmowy (rejestruje obraz z komputera na mikrofilmie lub mikrofiszy),
COMB ACCESS - dostęp grzebieniowy,
COMBINABILITY - zdolność tworzenia połączeń, łączenia się w kombinacje,
COMBINATION - łączenie, wiązanie, kombinacja,
COMBINATIONAL CIRCUIT - sieć przełączająca,
COMBINED (READ/WRITE) HEAD - głowica uniwersalna,
COMMA - przecinek,
COMMAND - rozkaz, także: instrukcja programu,
COMMAND CHAINING - wiązanie łańcuchowe rozkazów,
COMMAND CYCLE - cykl rozkazu,
COMMAND LANGUAGE - język rozkazów,
COMMAND LEVEL - poziom procesora,
COMMENCE - zacząć, rozpocząć,
COMMENSURABILITY - współmierność,
COMMENT - komentarz,
COMMENT FIELD - pole komentarza,
COMMERCIAL DATA PROCESSING - przetwarzanie danych gospodarczych,
COMMON BUSINESS ORIENTED LANGUAGE (COBOL) - wysokopoziomowy język programowania (proceduralny),
COMMON DENOMINATOR - wspólny mianownik,
COMMON NODE - punkt wspólny,
COMMON-COLLECTOR CONNECTION - układ (tranzystorowy) o wspólnym kolektorze,
COMMON-EMITTER CONNECTION - układ (tranzystorowy) o wspólnym emiterze,
COMMONLY-USED - zwykle używany,
COMMUNICATION INSTRUCTION - rozkaz komunikacyjny,
COMMUNICATION LAYOUT - rozmieszczenie przewodów, plan połączeń,
COMMUNICATION SESSION - seans łączności,
COMMUNICATIONS - komunikacja, przesyłanie,
COMMUNICATIONS CONNECTOR - łącze komunikacyjne,
COMPACT - gęsty, zbity, zwarty, małych rozmiarów, także: zagęszczać, zbijać,
COMPACTION - upakowanie,
COMPARABLE - porównywalny,
COMPARATIVE - porównawczy,
COMPARATIVELY - porównawczo, w porównaniu z ...,
COMPARE - porównywać,
COMPARE-AND-SWAP - porównaj i zamień,
COMPARER - układ porównywania,
COMPARING - porównywanie,
COMPARING BRUSH - szczotka porównująca,
COMPARING INSTRUCTION - rozkaz porównania,
COMPARING OPERATION - operacja porównania,
COMPARISON - porównanie,
COMPASS - granica, skala, zasięg,
COMPATIBILITY - wymienność, kompatybilność, zgodność,
COMPATIBLE - kompatybilny, wymienny,
COMPENSATE - kompensować, wyrównywać,
COMPILATION - tłumaczenie (programu zródłowego na język wynikowy),
COMPILE - kompilować (tłumaczyć) czyli przekształcać program ujęty w języku problemowym na język maszyny (wynikowy), także: opracowywać, zestawiać,
COMPILER - program kompilujący, kompilator,
COMPILER COMPILER - kompilator kompilatorów,
COMPILING - tłumaczenie (programu źródłowego na język wynikowy),
COMPILING STATEMENT - instrukcja translacji,
COMPLEMENT - uzupełnienie, dopełnienie,
COMPLEMENTATION - dopełnienie, uzupełnienie,
COMPLETE - kończyć, zakończyć wykonywanie czynności,
COMPLETE CARRY - przeniesienie natychmiastowe w dodawaniu równoległym,
COMPLETION - zakończenie (np. czynności, pracy),
COMPLEX - liczba zespolona także: skomplikowany, zawiły, złożony, kompleksowy,
COMPLEX INSTRUCTION SET COMPUTER - komputer o złożonej liście rozkazów,
COMPLEX NUMBERS - liczby zespolone,
COMPLEXITY - złożoność,
COMPONENT - część składowa, element, składnik,
COMPONENT PART - część składowa,
COMPOSITE COLOR MONITOR - zespolony monitor kolorowy,
COMPOSITE NUMBER - liczba złożona,
COMPOSITE VIDEO - zespolone wyjście video,
COMPOSITION OF A RECORD - budowa zapisu (rekordu),
COMPOUND - złożony, także: łączyć ze sobą,
COMPOUND COMPUTERS - komputery sprzężone,
COMPOUND CONDITION - warunek złożony,
COMPREHEND - zrozumieć,
COMPREHENSIVE - obszerny, wszechstronny, wyczerpujący,
COMPRESSED FORMAT - format zmniejszony, format zagęszczony,
COMPRESSION - ścieśnianie,
COMPRISE - zawierać,
COMPROMISING EMANATIONS - przypadkowe przekazywanie sygnałów odnoszących się do tajnych informacji,
COMPUNICATION - skomputeryzowane systemy łączności (computer communication),
COMPUTATION - liczenie, obliczanie,
COMPUTATION CENTER - centrum obliczeniowe,
COMPUTATIONAL - obliczeniowe,
COMPUTATIONAL LINGUISTICS - lingwistyka komputerowa,
COMPUTATIONAL SCHEME - schemat obliczeń,

COMPUTATIONS AIDS - środki obliczeniowe,
COMPUTE - liczyć, obliczać,
COMPUTED ENTRY TABLE - tablica wpisów o adresach obliczanych,
COMPUTER - komputer, maszyna matematyczna, maszyna licząca, elektroniczna maszyna cyfrowa,
COMPUTER-AIDED - skomputeryzowany,
COMPUTER AIDED DESIGN (CAD) - projektowanie układów wspomagane komputerem,
COMPUTER AIDED EDUCATION - nauczanie wspomagane komputerem,
COMPUTER AIDED INSTRUCTION - szkolenie wspomagane komputerowo,
COMPUTER AIDED LEARNING - nauczanie wspomagane komputerem,
COMPUTER AIDED MANUFACTURE - wytwarzanie wspomagane komputerowo,
COMPUTER AIDED MEASUREMENT AND CONTORL - komputerowo wspomagane pomiary i sterowanie,
COMPUTER ANALYST - analityk informatyk,
COMPUTER APPLICATION FOR MEASUREMENT AND CONTROL (CAMAC) - miedzynarodowy skomputeryzowany system modułowy pomiarów i sterowania,
COMPUTER BULLETIN BOARD - komputerowa tablica informacyjna,
COMPUTER BUREAU - usługowe biuro obliczeniowe,
COMPUTER CENTRE - ośrodek obliczeniowy,
COMPUTER CODE - lista rozkazów komputera,
COMPUTER COMPATIBILITY - kompatybilność komputerów, wymienność komputerów,
COMPUTER CONFIGURATION - zestaw komputera,
COMPUTER CONTROL - sterowanie komputerowe,
COMPUTER CONTROLLED SYSTEM - układ automatyczny sterowany komputerem,
COMPUTER DESIGN LANGUAGE (CDL) - język do opisywania komputerów cyfrowych,
COMPUTER DIAGNOSTICS - diagnostyka komputerowa,
COMPUTER DISPLAY - monitor ekranowy,
COMPUTER EVALUATION - ocena komputerów,
COMPUTER FAMILY - rodzina komputerów,
COMPUTER GENERATIONS - generacje komputerów,
COMPUTER GRAPHICS - grafika komputerowa,
COMPUTER HARDWARE - sprzęt komputerowy,
COMPUTER INDEPENDENT LANGUAGE - język niezależny od komputera,
COMPUTER INPUT MICROFILM - urządzenie wejściowe mikrofilmowe,
COMPUTER INSTRUCTION - rozkaz komputera,
COMPUTER INSTRUCTION CODE - kod wewnętrzny maszyny, kod instrukcji wewnętrznych,
COMPUTER LANGUAGE - język komputera, język wewnętrzny,
COMPUTER LITERACY - znajomość komputerowego abecadła,

Generowanie liczb pierwszych — sito Eratostenesa

Generowanie liczb pierwszych nie jest zagadnieniem banalnym. Do dnia dzisiejszego brak jest równania rekurencyjnego lub jakiejkolwiek funkcji generującej, która pozwoliłaby na otrzymanie liczb pierwszych wg jednolitego przepisu. Jedyną znaną metodą generowania liczb pierwszych jest słynne **sito Eratostenesa**, wymyślone już w II w. p.n.e. Metodę tę można stosować tylko wówczas, gdy problem sformułowany jest następująco:

Znaleźć wszystkie *liczby pierwsze* ≦N, gdzie N jest z góry zadaną liczbą.

Jeżeli problem sformułowany jest w postaci:

znaleźć N kolejnych liczb pierwszych,

trzeba stosować inny algorytm polegający na sprawdzaniu podzielności kolejnych liczb naturalnych nieparzystych i = 3, 5, 7, 9,... przez wiele podzielników ≦ $\sqrt{i}$. Wymaga to dużego nakładu obliczeń i jest bardzo czasochłonne (funkcja pierwiastkowania występująca wielokrotnie w algorytmie jest jedną z najdłużej obliczanych funkcji). Nie jest to właściwie metoda generowania liczb pierwszych, lecz metoda sprawdzenia, czy dana liczba jest liczbą pierwszą.

Algorytm sita Eratostenesa jest następujący:

1) tworzymy tablicę liczb naturalnych 1, 2, 3,..., N,
2) wiadomo, że 2 jest najmniejszą liczbą pierwszą, w związku z czym usuwamy z tablicy wszystkie wielokrotności liczby 2,
3) pierwsza po prawej niewykreślona liczba jest liczbą pierwszą,
4) usuwamy z tablicy wszystkie wielokrotności tej liczby,
5) powtarzamy czynności (3) i (4) do chwili, gdy niewykreślona liczba jest nie mniejsza od N/2,
6) wszystkie niewykreślone z tablicy liczby są liczbami pierwszymi.

Poniższy program napisany w języku BASIC na ZX Spectrum generuje liczby pierwsze metodą sita Eratostenesa.

```
399 REM "Sito Eratostenesa"
    **************************
      © Janusz MORBITZER
    **************************
 400 BEEP .2,10: LET v=3: CLS :
PRINT AT v,4;"Program znajduje
liczby";AT v+3,4;"pierwsze zawar
te w prze-";AT v+6,4;"dziale <1,
k> metoda sita";AT v+9,9;"Eratos
tenesa.": PRINT BRIGHT 1;AT v+13
,4;"##########################"
 401 INPUT "                Podaj k:
"; LINE k$: IF LEN k$=0 THEN GO
TO 400
 402 FOR i=1 TO LEN k$: IF k$(i)
>="0" AND k$(i)<="9" THEN NEXT i
: LET k=VAL k$: GO TO 405
 403 GO TO 400
 405 IF k<=1 THEN CLS : PRINT BR
IGHT 1;AT 10,3;"Brak liczb pierw
szych w tym": PRINT BRIGHT 1;AT
12,11;"przedziale": GO SUB 475:
GO TO 400
 408 IF k>7000 THEN CLS : PRINT
BRIGHT 1;AT 10,5;"Za duża liczba
!(>7000)": BEEP .4,0: GO SUB 475
: GO TO 400
 410 DIM b(k)
 411 GO SUB 485
 412 FOR i=1 TO k: LET b(i)=i: N
EXT i
 415 LET i=0: LET p=2
 417 BEEP .5,10: BORDER 7: PAPER
 7: INK 0: CLS : PRINT BRIGHT 1;
AT 0,8;"LICZBY PIERWSZE:": PRINT
 AT 1,8;"                ": PRIN
T
 420 FOR j=2*p TO k STEP p
 425 LET b(j)=0
 430 NEXT j
 432 LET l$=STR$ p: LET dl=LEN l
$
 433 LET pp=i+1-INT ((i+1)/2)*2
 435 IF pp=1 THEN PRINT TAB 8-dl
;p;: GO TO 438
 437 PRINT TAB 25-dl;p
 438 LET i=i+1
 440 LET p=p+1
 442 IF p>k THEN GO TO 460
 445 IF b(p)<>0 THEN GO TO 420
 450 GO TO 440
 460 PRINT : GO SUB 475: CLS : P
RINT BRIGHT 1;AT 9,6;"W przedzia
le <1,";k;">";AT 11,8;"znajduje
się ";i;AT 13,8;"liczb pierwszyc
h": GO SUB 475
 470 CLS : PAUSE 0: GO TO 400
 475 REM podprogram
 480 PRINT #1; BRIGHT 1; FLASH 1
;AT 0,4;"Naciśnij dowolny klawis
z": PAUSE 0: RETURN
 485 REM podprogram
 490 BORDER 0: PAPER 0: INK 7: C
LS : PRINT AT 8,9; FLASH 1;"PROS
ZĘ CZEKAĆ!": PRINT AT 12,7;"(tr
wają obliczenia)": RETURN
 495 REM koniec programu
```

Czas, jaki upływa od podania komputerowi wartości parametru N do pojawienia się na ekranie komunikatu LICZBY PIERWSZE jest niezbędny do inicjalizacji tablicy A/N/ i wypełnienia jej elementów kolejnymi liczbami naturalnymi /A/I/ = I dla I = 1, 2, ...,N/. Na ekranie obserwujemy pojawienie się kolejnych liczb pierwszych. Jest rzeczą charakterystyczną, że zaczynają się one pojawiać coraz szybciej. Wynika to z faktu, że rozmieszczenie liczb pierwszych w zbiorze liczb naturalnych jest nierównomierne: początkowo jest ich stosunkowo dużo, a następnie coraz mniej (np. w przedziale /1, 1000/ jest 168 liczb pierwszych, w przedziale /1001, 2000/ jest ich 135, a w przedziale /2001, 3000/ tylko 127). Większa ilość liczb pierwszych w początkowej fazie implikuje większy nakład obliczeń (większą liczbę wykreśleń, które w programie zostały zastąpione zerowaniem elementów tablicy A), a co za tym idzie dłuższy czas pomiędzy pojawieniem się na ekranie monitora kolejnych liczb pierwszych.

Największą dotychczas znaną liczbą pierwszą jest $2^{11213} - 1$. Odkryta została w roku 1963 przez **D.B. Gilliesa**, oczywiście przy użyciu maszyny cyfrowej. Znane są również tzw. liczby bliźniacze, tj. liczby pierwsze różniące się od siebie o 2, np. 5 i 7 czy 881 i 883. Oczywiście im większe liczby, tym trudniej znaleźć parę liczb bliźniaczych. Czytelnikom proponuję zmodyfikowanie przedstawionego algorytmu, tak aby program znajdował liczby bliźniacze.

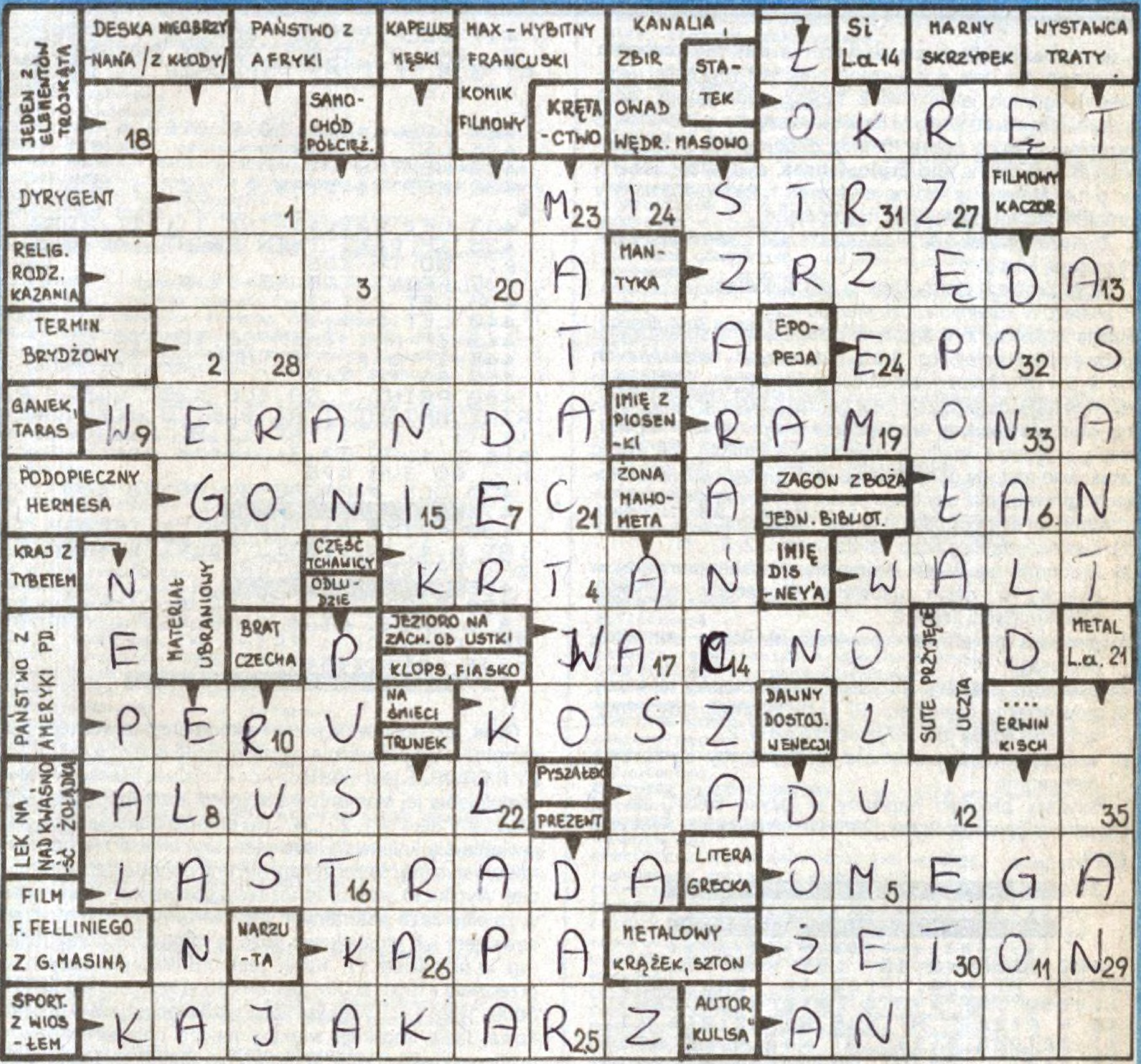
Janusz MORBITZER

Literatura

1) Corge C. „Elementy informatyki. Informatyka a myśl ludzka", PWN, 1981.
2) Tassel D. „Praktyka programowania", WNT, 1982.

KRZYŻÓWKA NR 1

KUPON



Litery z kratek ponumerowanych od 1 do 35 utworzą nazwy 7 producentów sprzętu komputerowego, które wystarczy nadesłać jako rozwiązanie zadania pod adresem redakcji na kartkach pocztowych w terminie do końca lutego naklejając kupon „IKS-a". Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, 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. 9554. Nr. ind. 361682.