INFORMATYKA
KOMPUTERY
SYSTEMY

Dodatek „Żołnierza Wolności" Cena 150 zł

zeszyty programów komputerowych nr 4

ATARI

SPECTRUM

SPIS TREŚCI

Ten zeszyt dedykujemy wszystkim tym, którzy postanowili programować samodzielnie. Wszystkie przykłady oparte zostały o dialekt Basica zastosowany w mikrokomputerach Atari. Jest ich w Polsce najwięcej. Łatwo też można zdobyć ich oprogramowanie. W koleżeński obieg ruszyły nie tylko gry, ale też i programy użytkowe. Czy zatem nasz zeszyt jest potrzebny?

Nie mamy wątpliwości — dobrej literatury informatycznej nadal brakuje. Oczywiście nie każdy musi umieć programować, nie każdy przecież zostanie informatykiem, ale wierzymy, że każdy kto chce bliżej poznać swój komputer sięgnie po niniejszy zeszyt.

Wprowadzanie długich programów zniechęca. Wiemy o tym, toteż tasiemcowych wydruków w zeszycie nie ma. Poza tym sumy kontrolne skutecznie zabezpieczają przed popełnieniem błędów podczas wprowadzania programu do pamięci komputera.

Od chwili informatycznej „eksplozji" w Polsce wysłuchaliśmy już wielu sporów na temat ułomności niektórych typów mikrokomputerów, krytykowano Juniora, Mazovię, Kraka 86; wiele czasu stracono na udowadnianie wyższości polskiego Logo nad **BASIC**-iem. Tymczasem czas upływał. Jak to często bywa dobre pomysły utopiono w potoku akademickich dyskusji.

Każdy język programowania ma swoje zalety, wyrabia dobre, bądź złe nawyki u programisty. BASIC obok swoich niewątpliwych wad ma tę wielką zaletę, że jest. W jego interpretery wyposażone są wszystkie występujące w kraju mikrokomputery. I czy ktoś chce czy nie to właśnie BASIC jest dla posiadaczy mikrokomputerów pierwszym językiem, w którym porozumiewają się ze swoim urządzeniem. Dlatego czwarty zeszyt programów to właśnie BASIC — teraz można krytykować.

Na końcu zeszytu zamieściliśmy również kilka programów dla posiadaczy starszego typu mikrokomputera — ZX Spectrum. Cierpliwość przy wprowadzaniu z pewnością będzie nagrodzona atrakcyjnością programu.

Życzymy powodzenia.

Aby wprowadzić programy do pamięci komputera, radzimy wykorzystać Edytor BASIC'u, zamieszczony w nr. 4 „IKS-a" z bieżącego roku. Pamiętajmy: dwa znaki z lewej strony numeru linii nie są częścią programu, lecz sumą kontrolną danej linii, dlatego nie wprowadzamy ich do komputera.

„IKS" — dodatek „Żołnierza Wolności". Redaguje: Wiesław Cetera (kierownik zespołu); stali współpracownicy: Włodzimierz Gogołek, Janusz Janiec, Krzysztof Mamcarz, Ireneusz Miernik, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-60 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem Wydawnictwa „Czasopisma Wojskowe". Nr zam. 9385. Nr ind. 390313 K-69

ATARI BASIC w przykładach

TOMASZ MROWIEC
LUDWIK PIELA

Po otrzymaniu propozycji przygotowania numeru specjalnego „IKS", w części poświęconego komputerom Atari, byliśmy w rozterce; co zaproponować czytelnikom, aby mogli jak najwięcej skorzystać? Wiedzieliśmy doskonale, czego nie chcemy umieścić, chociaż kosztowałoby to nas najmniej wysiłku. Spędzamy dużo czasu przy komputerach i wiemy, jak żmudną i męczącą czynnością jest „wklepywanie" programów, szczególnie wtedy, gdy nie bardzo wiemy, co mają robić. Dlatego odrzuciliśmy wszelkie propozycje umieszczania długich programów. Nie chcieliśmy również dostarczać gier polegających na zabiciu przeciwnika lub uniknięciu unicestwienia. Wystarczająca ich podaż jest we wszelkiego typu „wypożyczalniach" i na giełdach.

Jednocześnie zauważyliśmy, że wielu programom, realizującym nieraz bardzo interesujące pomysły, brakuje ostatecznego wykończenia, są mało przejrzyste i trudne dla ewentualnych modyfikacji. Dlatego zdecydowaliśmy się zaprezentować czytelnikom, szczególnie tym, którzy stawiają pierwsze kroki w programowaniu, a także bardziej zaawansowanym pewne ciekawe procedury, które mogą wykorzystać w swoich programach. Nie są to żadne „sztuczki" programowe, które są często efektowne, lecz mało czytelne. **Prezentowane moduły programowe, bo tak je można nazwać, są jasne i zrozumiałe dla każdego znającego nieco BASIC.** Jeśli chociaż niektórzy z was uznają je za przydatne i wykorzystają w tworzonych przez siebie programach, będzie to dla was wystarczającą satysfakcją.

Dodatkowo prezentujemy kilka programów narzędziowych, które przeznaczone są dla bardziej zaawansowanych programistów oraz przykłady gry edukacyjnej dla najmłodszych.

Najbardziej niedoświadczeni programiści szybko odkrywają, że sekcje wejścia i wyjścia programu są jego najtrudniejszymi częściami.

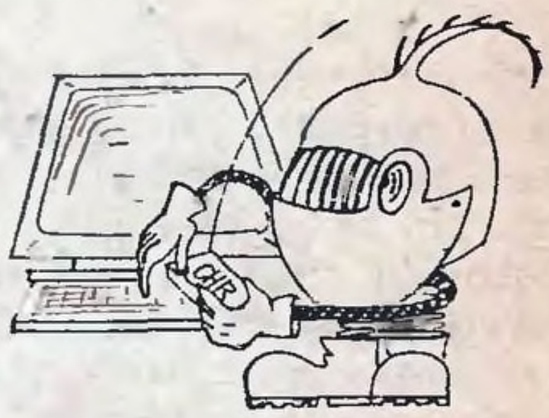
Prawie każdy program wymaga danych, które muszą być wprowadzane z klawiatury. Czy wystarczy kilka instrukcji INPUT? W większości wypadków nie. A co będzie, jeśli użytkownik przypadkiem naciśnie zły klawisz? Lub gorzej, co będzie, jeśli stwierdzi, po wprowadzeniu dwóch lub trzech pozycji danych, że są one niewłaściwe? Dobry programista musi przewidzieć, że użytkownik jego programu jest człowiekiem i popełnia nieprzewidziane, ludzkie błędy.

Podobna sytuacja jest z wynikami. Nie mogą one być po prostu wyświetlane lub drukowane przez grupę przypadkowych instrukcji PRINT. Będą przecież czytane przez człowieka. Dopóki ich postać nie jest starannie zaprojektowana, mogą być bardzo trudne do odczytania. W konsekwencji informacja może być źle odczytana lub całkowicie niewidoczna.

```
QO 1 REM ******************
ZR 2 REM * Program nr 1  *
QQ 3 REM ******************
TW 10 PRINT "Podaj cene za litr "
   ;
NK 20 INPUT C1L
HQ 30 PRINT "Ile litrow na 100 km
   . ";
CK 40 INPUT LKM
HQ 50 PRINT " Km.","  LITRY","CENA
   "
SS 60 PRINT "-----","--------","--
   ----"
TL 70 FOR KM=100 TO 1700 STEP 100
OU 80 LITR=KM*LKM/100
TR 90 CENA=INT(C1L*LITR*100)/100
NN 100 PRINT KM,LITR,CENA
NC 110 NEXT KM
TG 120 PRINT
VV 130 PRINT "L/100km =";LKM,C1L;
   " zl. za litr"
NY 140 END
```

```
QO 1 REM ******************
AL 2 REM * Program nr 2  *
QQ 3 REM ******************
QL 5 DIM N$(10),T$(10),BL$(40)
KF 6 REM Wypelnienie BL$ spacjami
AH 7 BL$=" ":BL$(40)=BL$:BL$(2)=B
   L$
TW 10 PRINT "Podaj cene za litr "
   ;
NK 20 INPUT C1L
HQ 30 PRINT "Ile litrow na 100 km
   . ";
CK 40 INPUT LKM
HQ 50 PRINT " Km.","  LITRY","CENA
   "
SS 60 PRINT "-----","--------","--
   ----"
TL 70 FOR KM=100 TO 1700 STEP 100
OU 80 LITR=KM*LKM/100
TR 90 CENA=INT(C1L*LITR*100)/100
SJ 100 NS=5:N=KM:GOSUB 11000:PRIN
   T N$(1,NS),
GR 102 NS=7:N=LITR:GOSUB 11000:PR
   INT N$(1,NS),
SI 104 NS=5:N=CENA:GOSUB 11000:PR
   INT N$(1,NS)
NC 110 NEXT KM
TG 120 PRINT
VV 130 PRINT "L/100km =";LKM,C1L;
   " zl. za litr"
NY 140 END
IK 10000 REM ******************
YT 10001 REM *   Podprogram   *
HY 10002 REM *wyrownania liczb*
KY 10003 REM *     w prawo     *
JE 10004 REM ******************
TC 11000 T$=STR$(N)
GS 11010 N$=BL$
TV 11020 N$(NS-LEN(T$)+1,NS)=T$
DL 11030 RETURN
```

```
QD 1 REM ********************
BF 2 REM * Program nr 3  *
QQ 3 REM ********************
QL 5 DIM N$(10),T$(10),BL$(40)
KF 6 REM Wypelnienie BL$ spacjami
AH 7 BL$=" ":BL$(40)=BL$:BL$(2)=B
   L$
TW 10 PRINT "Podaj cene za litr "
   ;
NK 20 INPUT C1L
HQ 30 PRINT "Ile litrow na 100 km
   . ";
CK 40 INPUT LKM
VK 50 PRINT " Km.","  LITRY","  CE
   NA"
KT 60 PRINT "------","--------","--
   ------"
TL 70 FOR KM=100 TO 1700 STEP 100
OU 80 LITR=KM*LKM/100
TR 90 CENA=INT(C1L*LITR*100)/100
TC 100 NS=6:DD=0:N=KM:GOSUB 11000
   :PRINT N$(1,NS),
NL 102 NS=7:DD=2:N=LITR:GOSUB 110
   00:PRINT N$(1,NS),
JV 104 NS=7:DD=1:N=CENA:GOSUB 110
   00:PRINT N$(1,NS)
NC 110 NEXT KM
TG 120 PRINT
VV 130 PRINT "L/100km =";LKM,C1L;
   " zl. za litr"
NY 140 END
IK 10000 REM ********************
YT 10001 REM *   Podprogram    *
HY 10002 REM *wyrowmania liczb*
NY 10003 REM *   w/g kropki    *
GT 10004 REM *   dziesietnej   *
JJ 10005 REM ********************
TC 11000 T$=STR$(N)
GS 11010 N$=BL$
DK 11030 DP=LEN(T$)+1
YZ 11040 FOR J=1 TO LEN(T$)
QY 11050 IF T$(J,J)="." THEN DP=J
   :J=NS
GV 11060 NEXT J
SG 11070 NL=DP+DD
GH 11080 N$(NS-NL+1,NS)=T$
EJ 11090 RETURN

FK 1 REM ********************
ZH 2 REM * Program nr 4 *
FM 3 REM ********************
OY 10 DIM R$(1),CU$(24),CR$(40)
QL 20 CU$=CHR$(28):CU$(24)=CU$:CU
   $(2)=CU$
YB 30 CR$=CHR$(31):CR$(40)=CR$:CR
   $(2)=CR$
YW 40 PRINT CHR$(125);"Przyszla w
   artosc wkladu w P K O"
VG 50 PRINT :PRINT
ZE 60 PRINT "Wklad pierwotny"
OR 70 PRINT "Stopa oprocentowania
   "
VP 80 PRINT "Ile lat";CU$(1,3)
ZU 100 PRINT CR$(1,15);:INPUT AMT
SI 110 PRINT CR$(1,20);:INPUT IR
CK 120 PRINT CR$(1,7);:INPUT YR
AF 130 CMP=1
OP 140 IP=IR/CMP/100
EB 150 FV=AMT*(1+IP)^(CMP*YR)
QY 160 PRINT "Przyszla wartosc: "
   ;INT(FV*100+0.5)/100;"          "
TQ 170 PRINT
UH 180 PRINT "Czy zmieniasz wklad
   ";
HE 190 INPUT R$
KJ 200 IF R$="T" THEN PRINT CU$(1
   ,6);:GOTO 100
NT 210 END
```



Wyprowadzenie informacji na ekran

Przedstawienie informacji na ekranie powinno być tak zorganizowane, aby byłą ona łatwa do zrozumienia lub przyjemna dla oczu. Podstawowym narzędziem wyświetlania informacji jest instrukcja **PRINT**.

Rozważmy wprowadzanie na ekran informacji przedstawionych w postaci tabel. Zwykle dużo łatwiej jest przeglądać wykaz pozycji, gdy są one zestawione w kolumny. Dotyczy to zarówno liczb, jak i danych tekstowych. Atari **BASIC** dysponuje dwoma sposobami wyprowadzania informacji w postaci kolumn. Jednym jest użycie przecinków między wartościami w instrukcjach **PRINT**. Drugim jest użycie klawisza **TAB**.

Jeśli komputer natrafi na przecinek po wartości (zmiennej) instrukcji **PRINT**, przesuwa kursor w prawo. Wypełnia spacjami obszar między końcem wyświetlanej wartości i początkiem następnej kolumny. Druga kolumna zaczyna się o dziesięć odstępów od lewego marginesu, dodatkowe kolumny występują co każde dziesięć odstępów. **PROGRAM NR 1 używa dwóch z trzech dostępnych kolumn**.

Używając przecinków należy zachować ostrożność. Dwie pozycje przed końcem kolumny muszą być puste. W przeciwnym razie następuje przejście do początku następnej kolumny. Możliwa jest zmiana szerokości kolumn za pomocą **POKE 201,*n***, gdzie n jest nową szerokością kolumn.

Oba sposoby wyprowadzania informacji kolumnami mają jedną wadę. Wyrównują one wartości do lewego brzegu kolumny. Takie wyrównanie do lewego marginesu jest wygodne dla słów i innych wartości alfabetycznych. Liczby są łatwiejsze do odczytania, jeśli są wyrównane w prawo. **PROGRAM NR 2 zawiera podprogram wyrównania liczb do prawego marginesu**. Podprogram ten wymaga indywidualnego dostępu do każdej cyfry liczby, która ma być wyrównana. BASIC umożliwia taki dostęp tylko w zmiennych tekstowych, zatem podprogram tłumaczy liczbę na ciąg znaków (linia 11000). Następnie wypełnia ciąg wyjściowy spacjami (linia 11010). Gwarantuje to stałą wartość w częściach ciągu, które nie kończą się cyfrą. W końcu wyrównuje liczbę w prawo (linia 11020). Sprawdza długość liczby i określa jak blisko prawego brzegu kolumny musi się rozpocząć, aby mogła być umieszczona właściwie.

Program ten działa prawidłowo dla liczb całkowitych. Kolumny liczb z kropkami dziesiętnymi byłyby łatwiejsze do odczytania, gdyby liczby były wyrównane według kropki dziesiętnej. W tym celu musimy określić, gdzie

```
FK 1 REM ******************
AB 2 REM * Program nr 5 *
FM 3 REM ******************
RN 9 REM USTAWIENIE LEWEGO MARGIN
   ESU
FX 10 POKE 82,0
FB 19 REM USTAWIENIE PRAWEGO MARG
   INESU
GF 20 POKE 83,31
XQ 30 GRAPHICS 0:DS=PEEK(88)+PEEK
   (89)*256
DT 35 REM ZMIANA PROGRAMU WYSWIET
   LANIA
NQ 40 DL=1539
WR 50 FOR I=1536 TO 1538
NE 60 POKE I,112:NEXT I
JI 70 FOR I=1 TO 24
YC 80 POKE DL,66
FX 90 POKE DL+2,INT(DS/256):POKE
   DL+1,DS-PEEK(DL+2)*256:DS=DS+4
   0
SL 100 DL=DL+3
FT 110 NEXT I
FM 115 REM USTRWIENIE WASKIEGO TL
   A
QI 120 POKE 559,33
ER 130 POKE DL+2,6:POKE DL,65:POK
   E DL+1,0
ZK 140 POKE 560,0:POKE 561,6
```

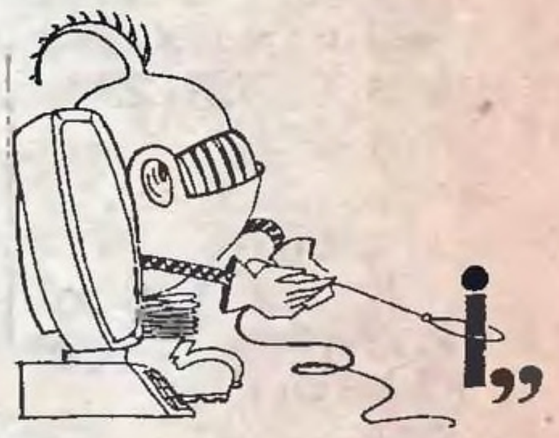
```
FK 1 REM ******************
AV 2 REM * Program nr 6 *
FM 3 REM ******************
HH 10 DIM BL$(40),CMD$(1),R$(1)
OS 50 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
OS 300 AMPSUB=2000
TW 310 PRINT CHR$(125):REM KASOWA
   NIE EKRANU
HT 320 POSITION 2,1
XG 330 PRINT "POLECENIA (ACDEP, H
   =Pomoc)";
UV 340 INPUT CMD$
HL 360 POSITION 2,0
XJ 370 PRINT "Mam wykonac polecen
   ie ";
RY 380 ON ASC(CMD$)-64 GOSUB 600,
   31767,800,1000,1200,31767,3176
   7,1400
XS 390 IF CMD$<>"P" THEN 310
WB 400 PRINT "DRUKUJ":POP :GOSUB
   31766:GOTO 310
FA 600 PRINT "DODAJ ":GOSUB 31766
   :RETURN
VS 800 PRINT "ZMIEN ":GOSUB 31766
   :RETURN
JJ 1000 PRINT "KASUJ ":GOSUB 3176
   6:RETURN
XY 1200 PRINT "KONIEC":POSITION 2
   ,10:END
DL 1400 PRINT "POMOC"
XG 1410 GOSUB 2000
ZT 1420 POP :GOTO 320
YJ 2000 GOSUB 2300
KQ 2010 PRINT "_______Wykaz komen
   d_______"
BX 2030 PRINT " E=dodaj C=zmień
   D=kasuj"
MZ 2040 PRINT " E=koniec P=drukuJ
   H=pomoc";
AP 2050 RETURN
IU 2300 POSITION 2,19
RG 2310 FOR J=20 TO 23
SK 2330 PRINT BL$(1,38);
FT 2340 NEXT J
DU 2350 POSITION 2,20
AY 2360 RETURN
CJ 31766 FOR I=0 TO 400:NEXT I:RE
   TURN
GD 31767 RETURN
```

umieścić kropkę dziesiętną w każdej kolumnie. Następnie trzeba sprawdzić, gdzie jest kropka w każdej liczbie. Nie jest to zadanie trywialne, ponieważ BASIC używa liczb zmiennoprzecinkowych. Kropka dziesiętna może być gdziekolwiek. Po znalezieniu musimy przesunąć ją w prawo lub w lewo. Może to oznaczać gubienie dodatkowych cyfr z prawej strony lub wypełnianie dodatkowymi spacjami.

Nowy podprogram (z PROGRAMU NR 3) **zawiera wszystkie wymagania poprzedniego plus dodatkowe.** Wstępnie zakłada, że nie ma kropki dziesiętnej i za pomocą pętli **FOR-NEXT** poszukuje kropki w ciągu numerycznym. W wyniku działania tego podprogramu otrzymujemy kolumny liczb łatwe do przeglądania. Zauważmy, że BASIC nie drukuje kropek dla liczb całkowitych. Nie drukuje też zer końcowych liczb, które nie zmieniają ich wartości.

Istnieje niebezpieczeństwo, że liczba będzie zbyt duża i nie zmieści się w wydzielonej kolumnie. Możliwe jest wstawienie do programu lub podprogramu instrukcji sprawdzających wielkość liczby i ustawiających odpowiednią szerokość kolumny w celu dostosowania jej do największych możliwych wartości.

Sterowanie kursorem



Atari BASIC umożliwia bezpośrednie sterowanie kursorem dwoma sposobami. Jednym jest programowanie ruchu kursora znakami, przy użyciu funkcji **CHR$** lub ciągów „escape". Drugim sposobem jest używanie instrukcji **POSITION**.

Czasami program wymaga oczyszczenia ekranu. Robi się to przez wyświetlenie znaku ATASCII o kodzie 125, który czyści ekran i ustawia kursor w lewym górnym rogu. W celu wygenerowania niezbędnego znaku można użyć: **CHR$** (125), **ESC/CTRL—<** lub **ESC/SHIFT—<**.

Przesuwanie kursora po ekranie umożliwiają znaki: , , → i . Nie kasują żadnego znaku, przez który przechodzi kursor. Są dokładnie takie same w trybie programowania, jak w trybie bezpośrednim.

PROGRAM NR 4, po obliczeniu wyniku, przesuwa kursor do góry, do miejsca wprowadzenia pierwszej liczby. Pozwala to wprowadzić nową liczbę lub nacisnąć RETURN, aby pozostawić poprzednią wartość.

Instrukcja POSITION działa nieco inaczej. Nie przesuwa kursorem, lecz aktualizuje zawartość pewnych komórek pamięci komputera nowym położeniem kur-

```
FK 1 REM ****************
BP 2 REM * Program nr 7 *
FM 3 REM ****************
AH 10 DIM BL$(40),CMD$(1),R$(1)
OS 50 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
OS 300 AMPSUB=2000
TW 310 PRINT CHR$(125):REM KASOWA
   NIE EKRANU
HT 320 POSITION 2,1
XG 330 PRINT "POLECENIA (ACDEP, H
   =Pomoc)";
UV 340 INPUT CMD$
HL 360 POSITION 2,0
XJ 370 PRINT "Mam wykonac polecen
   ie ";
RY 380 ON ASC(CMD$)-64 GOSUB 600,
   31767,800,1000,1200,31767,3176
   7,1400
XS 390 IF CMD$<>"P" THEN 310
WB 400 PRINT "DRUKUJ":POP :GOSUB
   31766:GOTO 310
FA 600 PRINT "DODAJ ":GOSUB 31766
   :RETURN
VS 800 PRINT "ZMIEN ":GOSUB 31766
   :RETURN
JJ 1000 PRINT "KASUJ ":GOSUB 3176
   6:RETURN
XY 1200 PRINT "KONIEC":POSITION 2
   ,10:END
DL 1400 PRINT "POMOC"
XG 1410 GOSUB 2000
ZT 1420 POP :GOTO 320
VP 1994 REM ****************
XV 1995 REM *  Podprogram  *
NP 1996 REM * wyswietlania *
IV 1997 REM * rozszerzonego *
NU 1998 REM * wykazu komend *
WJ 1999 REM ****************
YJ 2000 GOSUB 2300
PO 2010 PRINT "_________Wykaz kom
   end_____"
ZF 2020 PRINT "Wiecej danych (ACD
   P, N jesli nie)"
BX 2030 PRINT " A=dodaJ C=zmien
   D=kasuj"
MZ 2040 PRINT " E=koniec P=drukuJ
   H=pomoc";
RO 2050 POSITION 36,21
KZ 2060 INPUT R$
IY 2070 IF R$="N" THEN RETURN
HP 2080 ON ASC(R$)-64 GOTO 2150,3
   2767,2210,2210
BN 2090 IF R$<>"P" THEN GOTO 2050
YL 2100 GOSUB 2300
CD 2110 PRINT "_________komenda:
   DRUKUJ_____"
AU 2120 PRINT " Tekst okreslajacy
    dzialanie"
HK 2130 PRINT " komendy drukuj "
AO 2140 RETURN
ZA 2150 GOSUB 2300
VK 2160 PRINT "_________komenda:
   DODAJ _____"
BJ 2170 PRINT " Tekst okreslajacy
    dzialanie"
DW 2180 PRINT " komendy dodaJ"
BD 2190 RETURN
YQ 2210 GOSUB 2300
ZJ 2220 PRINT "__komenda: ZMIEN l
   ub KASUJ___"
AZ 2230 PRINT " Tekst okreslajacy
    dzialanie"
VA 2240 PRINT " komend zmien 1 ka
   suJ"
AT 2250 RETURN
IF 2300 POSITION 1,19
VH 2310 FOR J=19 TO 23
SK 2330 PRINT BL$(1,38);
FT 2340 NEXT J
DU 2350 POSITION 2,20
AY 2360 RETURN
CJ 31766 FOR I=0 TO 400:NEXT I:RE
   TURN
GD 31767 RETURN
```

sora. Nowy numer wiersza jest w komórce o adresie 84, nowy numer kolumny — w komórce 85. W dowolnej chwili można użyć funkcji **PEEK**, aby sprawdzić, gdzie następnie będzie kursor: **PEEK(84)** — dla numeru wiersza, **PEEK (85)** — dla numeru kolumny. W pewnych trybach graficznych komputer Atari używa do przechowania numeru kolumny dwóch komórek. W tym wypadku **PEEK(86)** × **256** + **PEEK(85)** daje numer kolumny.

Po każdym wyświetleniu informacji użycia instrukcji **PRINT** następuje również aktualizacja dwóch innych komórek pamięci przechowujących ostatnią pozycję kursora. Komórka 90 przechowuje numer wiersza, a 91 — numer kolumny.

Dzięki instrukcji **POKE** możemy zmieniać szerokość obrazu przesuwając lewy i prawy margines. W komórce o adresie 82 przechowywana jest wartość lewego marginesu, a w komórce 83 — prawego. Standardowo lewy margines jest w kolumnie 2. Aby go zmienić na 0 używamy instrukcji **POKE 82,0**. Prawy margines standardowo jest w kolumnie 39. Aby go przesunąć używamy: **POKE 83,38.**

Zmianę szerokości obrazu demonstruje PROGRAM NR 5.

Po ponownym ustawieniu marginesów musimy pamiętać, że instrukcja PRINT „obserwuje" ich wartości. Nie ma znaku, który umożliwi przesunięcie kursora poza marginesy. Po osiągnięciu prawego marginesu następuje przejście do nowej linii.

Jedynym sposobem ustawienia kursora poza marginesami jest użycie funkcji POSITION.

Należy również pamiętać, że zwężenie obrazu zmniejsza długości nie tylko linii fizycznej programu, lecz także linii logicznej, która nigdy nie przekracza trzech linii fizycznych, bez względu na ich długość.

Programowanie wejścia

Prawie każdy program wymaga wprowadzenia informacji. Celem sekcji wejścia dowolnego programu powinno być minimalizowanie błędów wprowadzania oraz wygodne wykrywanie i poprawianie popełnionych błędów. Istnieją różne sposoby wprowadzania, które dążą do minimalizacji błędów. Poniżej przedstawione będą następujące:

```
FK 1 REM ****************
FJ 2 REM *                  *
CK 3 REM * Program nr 8 *
FL 4 REM *                  *
FO 5 REM ****************
QN 10 DIM NMSK$(40),SMSK$(40),R$(
   40),CL$(40)
MB 20 NMSK$="#":NMSK$(40)=NMSK$(1
   ):NMSK$(2)=NMSK$(1)
HZ 30 SMSK$="_":SMSK$(40)=SMSK$(1
   ):SMSK$(2)=SMSK$(1)
WU 40 CL$=CHR$(30):CL$(40)=CL$:CL
   $(2)=CL$
JS 90 PRINT CHR$(125);"WPROWADZAN
   IE CENNIKA:":?
OB 100 PRINT "NAZWA TOWARU ";SMSK
   $(1,9);CL$(1,10);
GO 110 INPUT R$
EJ 120 PRINT "CENA JEDNOSTKOWA ";
   NMSK$(1,8);CL$(1,9);
GS 130 INPUT R$
NY 140 END

FK 1 REM ****************
FJ 2 REM *                  *
DE 3 REM * Program nr 9 *
FL 4 REM *                  *
FO 5 REM ****************
VX 10 DIM R$(40)
TD 200 PRINT
TB 210 PRINT "CZY WPROWADZASZ INN
   A POZYCJE ";
GW 220 INPUT R$:R$=R$(1,1)
KZ 230 IF R$="T" OR R$="t" THEN 9
   0
GN 240 IF R$="N" OR R$="n" THEN E
   ND
MM 250 GOTO 210

QD 1 REM ****************
IB 2 REM *                  *
MQ 3 REM * Program nr 10 *
ID 4 REM *                  *
QS 5 REM ****************
WM 10 DIM X$(10)
IV 50 TRAP 8000
UD 200 PRINT "WPROWADZ CIAG ZNAKO
   W ";
JD 210 INPUT X$
EO 220 IF X$="K" THEN 500:REM Kon
   iec programu ?
EG 230 PRINT "WPROWADZ WARTOSC LI
   CZBOWA";
ST 240 INPUT X
CC 249 REM Blad jesli wprowadzono
    0
NW 250 X=X*X/X
JL 499 REM Koniec programu
KR 500 PRINT "OSTATNIE DANE: ";X$
   ;" i ";X
TF 510 TRAP 40000:REM Wylaczenie
   TRAP
NY 520 END
WQ 7998 REM +++++Obsluga bledow++
   +++
HS 7999 REM Pobranie numeru bledu
HN 8000 E=PEEK(195)
CS 8009 REM Pobranie numeru linii
    w ktorej wystapil blad
CD 8010 EL=PEEK(186)+256*PEEK(187
   )
BC 8020 IF E=3 OR E=8 THEN 8100
DJ 8029 REM Blad w programie
NF 8030 PRINT "UWAGA! Blad nr ";E
DD 8040 PRINT "w linii ";EL
FS 8080 END
AB 8100 REM Blad danych wejsciowy
   ch
JH 8110 PRINT CHR$(253);
SV 8120 PRINT "BLAD... WPROWADZ J
   ESZCZE RAZ"
MQ 8130 TRAP 8000:REM Kasowanie T
   RAP
KX 8140 GOTO EL
```

— wyświetlanie pomocniczych komunikatów,
— oczekiwanie naturalnych, intuicyjnych odpowiedzi,
— sprawdzanie wprowadzonych informacji,
— używanie podprogramu obsługi błędów,
— logiczne grupowanie wprowadzanych danych,
— umożliwienie obejrzenia i zmiany pogrupowanych danych,
— ograniczanie odpowiedzi: używanie manipulatorów,
— ograniczanie wyborów:używanie menu.

Jednym ze sposobów ułatwiania współpracy użytkownika z programem, umożliwiającym wyeliminowanie błędów wprowadzania informacji, jest wyświetlanie krótkich komunikatów (podpowiedzi), które opisują oczekiwane dane. W miarę możliwości powinniśmy wstawiać podpowiedzi w jednej linii, a odpowiedzi w następnej. Ponieważ instrukcja **INPUT** zawsze wyświetla znak zapytania, najlepiej jest przedstawiać podpowiedzi w formie zapytań.

Czasami niemożliwe jest sformułowanie zadowalającej podpowiedzi; jest albo zbyt lakoniczna albo zajmuje zbyt dużo miejsca. W takim wypadku możemy wyświetlić rozwiniętą podpowiedź w innym miejscu ekranu. Normalnie program wyświetla krótki komunikat. Umożliwia też użytkownikowi uzyskanie rozwiniętej instrukcji po naciśnięciu klawisza **„H"** (od Help). Wyprowadza wtedy instrukcje w pewnym stałym miejscu ekranu, powiedzmy w czterech dolnych liniach. Po wyświetleniu instrukcji program musi wrócić do miejsca, gdzie nastąpiło wezwanie pomocy (oczekiwanie na wprowadzenie).

PROGRAM NR 6 **oczekuje na jednoliterowe polecenia w drugiej linii ekranu.** Rozszerza polecenia literowe do poleceń słownych i wyświetla słowo z prawej strony górnej linii. Instrukcja może być wywołana przez dowolny znak lub słowo. Literę H wybraliśmy samowolnie. Instrukcje możemy umieszczać w dowolnym miejscu ekranu, lcze zalecane jest wyświetlanie każdego zestawu instrukcji w tym samym obszarze. Oczywiście program musi czekać, aż użytkownik zakończy czytanie każdej części instrukcji, zanim przejdzie do następnej. Realizuje to pojedyncza instrukcja **INPUT**. Jednocześnie program może zezwolić użytkownikowi na przerwanie instrukcji i powrót do normalnego ciągu wprowadzania. W PROGRAMIE NR 7 **podmieniony został podprogram zaczynający się od linii 2000.**

Innym miejscem umieszczania instrukcji jest początek programu. Wyświetlane instrukcje nie zastąpią dobrze napisanej instrukcji drukowanej, lecz są często wystarczającym przypomnieniem dla kogoś, kto jest trochę roztargniony.

```
QO 1 REM ****************
IB 2 REM *                  *
NL 3 REM * Program nr 11 *
ID 4 REM *                  *
Q8 5 REM ****************
EQ 10 DIM PRMT$(40),MSK$(40),BL$(
   40),ERM$(20),R$(20)
OP 20 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
WZ 30 MSK$="_":MSK$(40)=MSK$:MSK$
   (2)=MSK$
GM 40 ERRHDL=8000
IC 90 GOTO 1000
ME 97 REM ********************
UE 98 REM * Kasowanie linii *
MI 99 REM ********************
EI 100 FOR J=L1 TO L2
VX 110 POSITION 0,J
YX 120 PRINT BL$(1,38);
GH 130 NEXT J
ZF 140 RETURN
IB 167 REM ********************
ZX 168 REM * Komunikat bledu *
IH 169 REM ********************
NE 170 FOR J=1 TO 3
MT 180 POSITION 20,0
UD 190 PRINT ERM$;CHR$(253);
LE 200 FOR J2=1 TO 100:NEXT J2:RE
   M Opoznienie
MG 210 POSITION 20,0
ND 220 PRINT BL$(1,19):REM Kasowa
   nie
GI 230 NEXT J
ZG 240 RETURN
NX 246 REM **************
EW 247 REM * Kasowanie  *
LM 248 REM * instrukcji *
OG 249 REM **************
EG 250 L1=20:L2=23:GOSUB 100
OA 260 POSITION 2,20
ZM 270 RETURN
OK 596 REM **************
SR 597 REM * Podprogram *
ZJ 598 REM *   wejscia  *
OT 599 REM **************
IB 600 TRAP ERRHDL
TM 610 GOSUB 250
QV 620 REM
AV 630 POSITION IC+1,IR
LG 640 PRINT MSK$(1,IL);BL$
FE 650 POSITION IC,IR
SC 659 REM
HD 660 INPUT R$
RP 664 REM
MB 665 FOR J1=1 TO LEN(R$):IF R$(
   J1,J1)<>MSK$(1,1) THEN NEXT J1
AG 666 IF J1>1 THEN R$=R$(1,J1-1)
   :GOTO 670
CG 667 R$=""
ZU 670 IF R$="?" THEN GOSUB AMPSU
   B:GOTO 630
VH 680 IF LO>HI THEN GOTO 750:REM
    cias
OD 690 REM Zerowe wejscie
GF 700 IF R$="" THEN R$="0"
YV 705 IF ASC(R$(1,1))<48 OR ASC(
   R$(1,1))>57 THEN 730
UW 710 R=VAL(R$)
PA 720 IF R>LO AND R<HI THEN RETU
   RN
AX 730 ERM$="ZLA WARTOSC !"
FU 740 GOSUB 170:GOTO 610
ZD 749 REM Wprowadzanie ciasu
PG 750 IF LEN(R$)<=IL THEN RETURN
OW 759 REM Zla dlugosc ciasu
YG 760 ERM$="ZLA DLUGOSC CIAGU !"
GA 770 GOSUB 170:GOTO 610
MK 796 REM ****************
VF 797 REM * Wprowadzanie *
MQ 798 REM ****************
SL 799 REM
CU 800 L1=1:L2=2:GOSUB 100
RU 809 REM
HW 810 POSITION 2,1
AA 820 PRINT PRMT$
EZ 829 REM Wartosc wejsciowa
XI 830 IC=2:IR=2:GOSUB 600
ZM 840 RETURN
GY 999 REM ---PROGRAM GLOWNY---
EF 1000 PRINT CHR$(125);:REM Czys
   zczenie ekranu
JW 1009 REM Wprowadzanie ciasu
BE 1010 PRMT$="Podaj imie gracza
   ?"
AN 1020 IL=18:LO=1:HI=0
RR 1030 AMPSUB=31767
YJ 1040 GOSUB 800
LJ 1049 REM Wprowadzanie wartosci
    liczbowej
GB 1050 PRMT$=R$
HX 1060 PRMT$(LEN(PRMT$)+1)="-Jak
   i wynik ?"
IR 1080 IL=7:LO=0:HI=300
CD 1090 AMPSUB=7000
XZ 1100 GOSUB 800
NG 1110 GOTO 1010
SO 6997 REM Instrukcja wprowadzan
   ia wartosci liczbowej
XW 7000 GOSUB 250
YW 7010 PRINT " Wprowadz liczbe d
   odatnia,"
PW 7020 PRINT "      mniejsza od
   300."
AO 7030 RETURN
NT 7996 REM ********************
CH 7997 REM * Obsluga bledow *
OB 7998 REM ********************
HS 7999 REM Pobranie numeru bledu
HN 8000 E=PEEK(195)
OI 8009 REM Pobranie numeru linii
BT 8010 EL=PEEK(187)*256+PEEK(186
   )
BC 8020 IF E=3 OR E=8 THEN 8100
NF 8030 PRINT "UWAGA! Blad nr ";E
KF 8040 PRINT "w linii nr ";EL
FS 8080 END
IF 8100 REM
NA 8110 ERM$="BLAD WE"
YT 8120 GOSUB 170
VQ 8130 TRAP ERRHDL
KX 8140 GOTO EL
GD 31767 RETURN
```

Maskowanie wejścia

Zawsze istnieją pewne ograniczenia długości odpowiedzi. Program może wyświetlać ciąg znaków, które ograniczają tę długość. Taki ciąg nazywamy maską wprowadzania.

Do tego celu nadaje się dowolny znak. Powszechnie stosowane są znaki podkreślenia, gwiazdki, plus i minus. Program może używać jednego rodzaju znaków do maskowania tekstu i innego do maskowania liczb. Dostarcza to dodatkowej informacji o oczekiwanej odpowiedzi. **PROGRAM NR 8 używa znaku podkreślenia dla wprowadzania tekstów i znaku # dla wejścia numerycznego.**

Każda instrukcja **PRINT,** która drukuje maskę, drukuje także ciąg znaków, który przesuwa kursor na początek maski wprowadzania (linie 100 i 120).

Wybieranie odpowiedzi

Możemy zmniejszyć szansę powstania błędu przez ostrożne wybieranie odpowiedzi wejściowych. Nasz program powinien umożliwić użytkownikowi odpowiedzi w sposób naturalny, intuicyjny. Podczas pisania programu wygodnie jest, aby użytkownik kodował wszystkie informacje wejściowe. Zmusza go to jednak do wykonywania mechanicznych czynności podczas każdego używania programu. Ponieważ komputer jest lepszy w czynnościach rutynowych, pozwólmy mu wykonywać kodowanie? Jeśli naturalną odpowiedzią jest słowo lub litera, których program będzie ewentualnie potrzebował przetłumaczyć na liczbę, dajmy mu to zrobić. **Właśnie to wykonują linie 310 do 390 PROGRAMU NR 7.** Użytkownik wprowadza kody mnemoniczne poleceń: A,C,D,E,P lub H. Program bada, który podprogram wywołać (linia 380), w celu wykonania polecenia. Wyobraźmy sobie, o ile łatwiej byłoby napisać program, który wymaga od użytkownika wprowadzania polecenia w postaci numerycznej, lecz o ile trudniejszy byłby on w użyciu.

Sprawdzanie odpowiedzi

Nawet przy najostrożniejszym zaprojektowaniu wprowadzania nie możemy być pewni, w jaki sposób będą odpowiadać użytkownicy. Jeśli wprowadzenie błędnej informacji może spowodować jakąś trudność, program powinien sprawdzić, czy tekst nie jest zbyt długi? czy

```
QO 1 REM ******************
IB 2 REM *                 *
OG 3 REM * Program nr 12 *
ID 4 REM *                 *
QS 5 REM ******************
OS 9 REM Inicjacja zmiennych
GM 10 DIM PRMT$(40),MSK$(40),BL$(
   40),ERM$(23)
OP 20 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
WZ 30 MSK$="_":MSK$(40)=MSK$:MSK$
   (2)=MSK$
GM 40 ERRHDL=8000
YU 50 DIM NA$(20),STT$(20),CI$(20
   ),ST$(2),ZI$(9),R$(25)
GB 89 REM Skok do poczatku progra
   mu
IC 90 GOTO 1000
ME 97 REM ********************
UE 98 REM * Kasowanie linii *
MI 99 REM ********************
EI 100 FOR J=L1 TO L2
VX 110 POSITION 0,J
YX 120 PRINT BL$(1,38);
GH 130 NEXT J
ZF 140 RETURN
LV 165 REM ****************
YT 166 REM * Wyswietlanie *
ZU 167 REM *    bledow    *
ME 168 REM ****************
NE 170 FOR J=1 TO 3
UT 180 POSITION 10,20
UD 190 PRINT ERM$;CHR$(253);
TX 200 FOR J2=1 TO 100:NEXT J2
UG 210 POSITION 10,20
BL 220 PRINT BL$(1,23)
GI 230 NEXT J
ZG 240 RETURN
NU 245 REM **************
ET 246 REM * Kasowanie  *
QK 247 REM *    pola    *
LM 248 REM * instrukcJi *
OG 249 REM **************
EG 250 L1=20:L2=23:GOSUB 100
OA 260 POSITION 2,20
ZM 270 RETURN
MI 596 REM ****************
BK 597 REM *  Podprogram  *
FK 598 REM * wprowadzania *
MR 599 REM ****************
IB 600 TRAP ERRHDL
TO 620 GOSUB 250
AV 630 POSITION IC+1,IR
TR 640 PRINT MSK$(1,IL);BL$(1,23-
   IL)
FE 650 POSITION IC,IR
XO 659 REM Wprowadzenie i sprawdz
   enie odpowiedzi
HD 660 INPUT R$
MB 665 FOR J1=1 TO LEN(R$):IF R$(
   J1,J1)<>MSK$(1,1) THEN NEXT J1
AG 666 IF J1>1 THEN R$=R$(1,J1-1)
   :GOTO 670
CG 667 R$=""
ZU 670 IF R$="?" THEN GOSUB AMPSU
   B:GOTO 630
GE 680 IF LO>HI THEN GOTO 750
OD 690 REM Zerowe wejscie
```

```
GF 700 IF R$="" THEN R$="0"
YV 705 IF ASC(R$(1,1))<48 OR ASC(
   R$(1,1))>57 THEN 730
UW 710 R=VAL(R$)
BS 720 IF R>=LO AND R<=HI THEN RE
   TURN
AX 730 ERM$="ZLA WARTOSC !"
GP 740 GOSUB 170:GOTO 620
SB 749 REM
PG 750 IF LEN(R$)<=IL THEN RETURN
OW 759 REM Zla dlugosc ciagu
ZU 760 ERM$="CIAG ZBYT DLUGI "
GV 770 GOSUB 170:GOTO 620
MH 795 REM **************
VC 796 REM * Wprowadzanie *
GG 797 REM *  wspomagane  *
MQ 798 REM **************
CU 800 L1=1:L2=2:GOSUB 100
QG 809 REM Wyswietlenie komunikat
   u
HW 810 POSITION 2,1
AA 820 PRINT PRMT$
EZ 829 REM Wartosc wejsciowa
XI 830 IC=2:IR=2:GOSUB 600
ZM 840 RETURN
AC 999 REM Kasowanie ekranu i wys
   wietlanie
IZ 1000 PRINT CHR$(125);"PODAJ NA
   ZWISKO I ADRES"
AP 1010 PRINT :PRINT
IC 1020 PRINT "1)Nazwisko:"
RZ 1030 PRINT "2)   Ulica:"
JC 1040 PRINT "3)  Miasto:"
UD 1050 PRINT "4) Panstwo:"
UW 1070 PRINT "5)     Kod:"
WR 1099 REM Wprowadzenie 5 pol
WZ 1100 FOR F=1 TO 5
XA 1110 GOSUB 2000
DQ 1120 NEXT F
CQ 1129 REM Wprowadzanie zmian
IN 1130 PRMT$="Czy chcesz cos zmi
   enic?"
XB 1140 LO=1:HI=0:AMPSUB=31767
VD 1150 IL=1:GOSUB 800
MO 1159 REM Analiza odpowiedzi
EE 1160 IF R$="N" OR R$="n" THEN
   1000
PB 1170 IF R$="T" OR R$="t" THEN
   1200
JO 1180 ERM$="Prosze napisac T lu
   b N":GOSUB 170
PQ 1190 GOTO 1130

XE 1199 REM Pobranie numeru pola
HV 1200 PRMT$="Ktore pole"
AK 1210 LO=1:HI=5:AMPSUB=32767
UW 1220 IL=1:GOSUB 800
WF 1230 F=R:GOSUB 2000:GOTO 1130
FF 1900 END
XM 2000 LO=1:HI=0:AMPSUB=32767
AZ 2010 ON F GOTO 2100,2200,2300,
   2400,2500
KA 2097 REM
IP 2098 REM Wprowadzenie nazwiska
KI 2099 REM
PV 2100 IC=13:IR=3:IL=20:GOSUB 60
   0
KR 2110 NA$=R$:RETURN
KC 2197 REM
OE 2198 REM Wprowadzenie ulicy
KK 2199 REM
QM 2200 IC=13:IR=4:IL=20:GOSUB 60
   0
TV 2210 STT$=R$:RETURN
KE 2297 REM
HZ 2298 REM Wprowadzenie miasta
KM 2299 REM
RD 2300 IC=13:IR=5:IL=20:GOSUB 60
   0
KL 2310 CI$=R$:RETURN
KG 2397 REM
RS 2398 REM Wprowadzenie panstwa
KO 2399 REM
HU 2400 IC=13:IR=6:IL=3:GOSUB 600
RE 2410 ST$=R$:RETURN
KI 2497 REM
ZW 2498 REM Wprowadzenie kodu
KQ 2499 REM
KT 2500 IC=13:IR=7:IL=6:GOSUB 600
PX 2510 ZI$=R$:RETURN
NT 7996 REM ******************
CH 7997 REM * Obsluga bledow *
OB 7998 REM ******************
HS 7999 REM Pobranie numeru bledu
HN 8000 E=PEEK(195)
NF 8009 REM Pobranie numeru bledn
   ej linii
BT 8010 EL=PEEK(187)*256+PEEK(186
   )
BC 8020 IF E=3 OR E=8 THEN 8100
CP 8030 PRINT "ZNALEZIONO BLAD NR
    ";E
AJ 8040 PRINT "  W LINII ";EL
FS 8080 END
IF 8100 REM
HF 8110 ERM$="BLAD WPROWADZANIA"
YT 8120 GOSUB 170
VQ 8130 TRAP ERRHDL
KX 8140 GOTO EL
GD 31767 RETURN
```

dane numeryczne mieszczą się w wybranym zakresie? czy wprowadzona informacja nie spowoduje później błędu w programie?

Jeśli chcemy napisać dobry program, musimy włożyć trochę wysiłku w przeciwdziałanie błędom, które może popełnić ktoś używający naszego programu. Program powinien wykrywać błędy wprowadzania i wymuszać na użytkowniku ponowne wprowadzenie danych, które mogłyby spowodować nienormalne zatrzymanie programu.

Prawdą jest, że BASIC wychwytuje pewne błędy. Jednak jego możliwości są ograniczone. Możliwe jest wprowadzenie wartości poprawnego typu, której wielkość jest nie do przyjęcia. Oznacza to, że wprowadzona wartość może spowodować błąd w dalszej części programu. Poniższy przykład ilustruje ten problem:

```
100 INPUT X
200 PRINT 100/X
300 END
```

Jeśli wprowadzimy 0 (w linii 100), program upadnie podczas próby dzielenia przez zero (linia 200) w instrukcji PRINT. Można tego uniknąć dosyć łatwo. Dodajmy do powyższego programu następujące linie w celu sprawdzenia, czy wprowadzona wartość nie jest zerem i zażądajmy ponownego wprowadzenia, jeśli jest.

```
110 IF X<>0 THEN 200
120 PRINT „Niedozwolone ... wprowadź ponownie"
130 GOTO 100
```

Przez rozwinięcie zilustrowanego sposobu możemy zobaczyć, jak łatwe jest sprawdzenie pod względem wartości. W zależności od okoliczności powinniśmy to robić raczej instrukcją ON—GOTO lub ON—GOSUB niż ciągiem instrukcji IF—THEN.

Czasami kontrola błędów jest kosztowna. Może zająć trochę czasu programowania, obszaru pamięci programu i czasu jego działania. Rozważmy, na przykład, typowe pytanie *tak* lub *nie*. Program powinien umożliwić dowolne poprawne odpowiedzi „naturalne", którymi są: *tak, nie, Tak, Nie, TAK, NIE, t, n, T* lub *N*. Istnieje 10 możliwych odpowiedzi, każdą z nich program musi sprawdzić. **Możemy łatwo zredukować liczbę sprawdzanych wartości, wystarczy sprawdzać pierwszy wprowadzany znak, jak w PROGRAMIE NR 9.** Jeśli odpowiedź nie jest dozwolona, program powtarza pytanie.

Instrukcja TRAP

Atari BASIC ma specjalną instrukcję, która umożliwia śledzenie błędów, które przechwytuje, zanim wyświetli komunikat i wstrzyma realizację programu. Po wykonaniu następującej instrukcji:

```
100 TRAP 20000
```

program przejdzie do podanego numeru linii w razie wykrycia błędu. Umieści także kod numeryczny błędu w komórce o adresie 195, który możemy zbadać funkcją PEEK. BASIC przechowuje także numer linii, w której wystąpił błąd. Możemy go pobrać dzięki wyrażeniu: **PEEK(187)+256*PEEK(186).**

Instrukcja **TRAP** jest deaktywowana po każdym wystąpieniu błędu. Program musi zatem wykonać inną instrukcję TRAP w celu jej reaktywowania. W celu deaktywacji tej instrukcji należy użyć **TRAP 40000.**

Procedura obsługi błędu

Zwykłą reakcją na błędy wykryte instrukcją TRAP jest napisanie procedury obsługi błędów, do której przechodzi program po wykryciu błędu. Na końcu tej procedury program może przejść z powrotem do początku linii, w której wystąpił błąd, lub do dowolnej innej linii programu. Procedura obsługi błędów może podjąć różne akcje w zależności od natury błędu i aktualnego stanu programu.

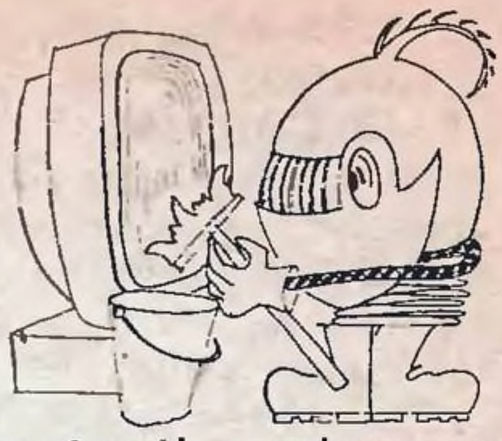
PROGRAM NR 10 demonstruje użycie instrukcji TRAP. Program ten traktuje błędy, które nie są związane z wprowadzaniem danych, jako fatalne. Informuje o numerze błędu i numerze linii oraz zatrzymuje program. Błędy wprowadzenia nie są fatalne. Program sygnalizuje je i żąda ponownego wprowadzenia.

```
QO 1 REM ********************
IB 2 REM *                  *
PB 3 REM * Program nr 13 *
ID 4 REM *                  *
QS 5 REM ********************
EG 10 DIM X(50,14)
RZ 20 DIM BL$(40),X$(4)
OQ 30 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
GN 40 POKE 752,2
SL 49 REM Inicjowanie tablicy
FE 50 PRINT CHR$(125);"Prosze chw
   ile poczekac"
IY 60 FOR K=0 TO 14
IN 70 FOR J=0 TO 50
WQ 80 X(J,K)=(J+1)*100+K+1
JG 90 NEXT J
GL 100 NEXT K
HX 199 REM Program glowny
DS 200 PRINT CHR$(125);"Uzyj mani
   pulatora do przesuniecia okna"
RI 205 R=25:C=7
QL 210 GOSUB 1000
RI 220 FOR K=1 TO 10
MX 230 FOR J=1 TO 3
IU 240 POSITION J*10-1,K+4
PI 250 X$=STR$(X(R+K-1,C+J-1))
ON 260 PRINT BL$(1,10-LEN(X$));X$
   ;
GQ 270 NEXT J
HC 280 NEXT K
IG 290 POSITION 2,1
ZH 299 REM Odczyt manipulatora
DL 300 IF STICK(0)=7 AND C<12 THE
   N C=C+1:GOTO 210
EO 310 IF STICK(0)=11 AND C>0 THE
   N C=C-1:GOTO 210
AT 320 IF STICK(0)=13 AND R<41 TH
   EN R=R+1:GOTO 210
HW 330 IF STICK(0)=14 AND R>0 THE
   N R=R-1:GOTO 210
PV 340 GOTO 290
YG 998 REM +++PODPROGRAM 1000+++
XD 1000 FOR J=1 TO 3
JZ 1010 POSITION 3+J*10,3
GH 1020 PRINT "KOLUMNA";
FJ 1030 NEXT J
WM 1040 FOR J=0 TO 2
EX 1050 POSITION 16+J*10,4
LP 1060 IF C+J+1<10 THEN PRINT "
   ";
JD 1070 PRINT C+J+1;
FY 1080 NEXT J
BQ 1090 FOR J=0 TO 9
AV 1100 POSITION 0,J+5
RG 1110 PRINT "WIERSZ ";
QK 1120 IF R+J+1<10 THEN PRINT "
   ";
PR 1130 PRINT R+J+1;
FO 1140 NEXT J
AQ 1150 RETURN
```

```
QO 1 REM ******************
IB 2 REM *                 *
PW 3 REM * Program nr 14 *
ID 4 REM *                 *
QS 5 REM ******************
FM 1000 R=1:LO=0:HI=100
PB 1010 POKE 752,2
JE 6509 REM Kasowanie starej wart
   osci
MP 6510 PRINT CHR$(125)
XU 6519 REM Wyswietlenie nowej wa
   rtosci
JU 6530 PRINT R;
PK 6539 REM Zakonczenie programu
RX 6550 IF STRIG(0)=0 THEN END
NW 6559 REM Opoznienie
XF 6560 FOR J=1 TO 30:NEXT J
IX 6579 REM Ruch do tylu
FZ 6580 IF STICK(0)=11 AND R<>LO
   THEN R=R-1:GOTO 6510
NE 6589 REM Ruch do przodu
QU 6590 IF STICK(0)=7 AND R<>HI T
   HEN R=R+1:GOTO 6510
BD 6599 REM Bez zmian
UD 6600 GOTO 6550
```

```
QO 1 REM ******************
IB 2 REM *                 *
QR 3 REM * Program nr 15 *
ID 4 REM *                 *
QS 5 REM ******************
ZQ 10 DIM BL$(40)
HC 19 POKE 752,2
OP 20 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
LX 1000 PRINT CHR$(125)
ER 1200 LO=0:HI=200:R=100:INC=0.5
TW 1210 IC=10:IR=11:IL=7
BO 1220 GOSUB 6500
RM 1230 PRINT
WQ 1240 PRINT "Wybrana liczba : "
   ;R
FF 1900 END
BK 6496 REM ***********************
VO 6497 REM * WE z manipulatora*
BS 6498 REM ***********************
AG 6500 POSITION IC,IR
WI 6510 PRINT BL$(1,IL);
AM 6520 POSITION IC,IR
UU 6530 PRINT R;
IP 6539 REM Koniec programu
ZA 6550 IF STRIG(0)=0 THEN RETURN
NW 6559 REM Opoznienie
FZ 6560 FOR J=1 TO SD:NEXT J
SU 6569 REM Opoznienie minimalne
VG 6570 SD=1
YO 6579 REM Ruch w tyl
GW 6580 IF STICK(0)=11 AND R<>LO
   THEN R=R-INC:GOTO 6500
CT 6589 REM Z powrotem
LF 6590 IF STICK(0)=7 AND R<>HI T
   HEN R=R+INC:GOTO 6500
BD 6599 REM Bez zmian
EQ 6600 SD=30:GOTO 6550
```

Użyteczne podprogramy wprowadzania



Obecnie spróbujemy zbudować ogólny podprogram wprowadzania. Będzie on używał wszystkich technik, które omówiliśmy dotychczas: podpowiedzi, rozszerzeń instrukcji, maskowania wejścia, sprawdzania odpowiedzi i procedury obsługi błędu.

Podprogram wprowadzania będzie używał kilku innych podprogramów. Jeden z nich czyści (kasuje) linie na ekranie, inny wyświetla komunikat błędu w prawym, górnym rogu ekranu oraz generuje sygnał dźwiękowy. Potrzebny jest także podprogram czyszczenia obszaru u dołu ekranu, w którym pojawiają się rozszerzone instrukcje. **PROGRAM NR 11 zawiera wszystkie niezbędne podprogramy oraz wzajemne powiązania między nimi.** Podprogramy te są wyróżnione odpowiednimi komentarzami. Używane są dwie zmienne do przechowywania numerów linii: **ERRHDL** i **AMPSUB.** Umożliwia to dostarczenie do programu wywołującego własnych procedur, a zatem różnych sposobów obsługi błędów i rozwinięć instrukcji. **AMPSUB** musi być rzeczywistym numerem linii, która może zawierać samą instrukcję **RETURN,** lecz musi istnieć.

Na początku PROGRAMU NR 11 deklarowane są i inicjowane zmienne używane przez poszczególne podprogramy. Następnie przechodzi on do realizacji głównego ciągu instrukcji (linia 90), gdzie ustawia 20-znakową zmienną tekstową (linie 1010 do 1040) oraz dane numeryczne (linie 1050 do 1070). Zauważmy, że wejście tekstowe nie ma rozwinięć instrukcji — **AMPSUB** ma wartość 31767, adres procedury, która nic nie robi. Istnieją jednak rozwinięcia instrukcji do wprowadzenia wartości numerycznych (linie 7000 do 7040).

Wprowadzanie danych grupowych

Bardzo często program potrzebuje kilku części informacji logicznie powiązanych ze sobą. Może wprowadzać pozycje danych na kilka sposobów. Jednym z nich jest wprowadzanie każdej pozycji w tym samym miejscu ekranu, używając różnych znaków zachęty dla każdej pozycji. **Ten sposób używany jest w procedurze wprowadzania nazw i wartości w PROGRAMIE NR 11.** Program przypomina, którą wartość należy wprowadzić, przez wypisanie odpowiedniego komunikatu.

Lepszym sposobem manipulowania wielokrotnymi pozycjami danych jest wyświetlenie odpowiedniego szablonu na ekranie i wypełnianie go w miarę wprowadzania danych. W tym celu program najpierw wyświetla szablon zawierający opis każdej pozycji i wystarczający obszar dla wprowadzania danych. Tak opisane pozycje nazywane są polami, każde z nich ma numer. Dane wprowadzamy sekwencyjnie, rozpoczynając od pierwszego pola i kończąc na ostatnim.

Do uzyskania takiego efektu potrzebny jest niewielki wysiłek. Załóżmy, że chcemy wprowadzić nazwisko i adres. Mamy zatem do wprowadzenia pięć pozycji: nazwi-

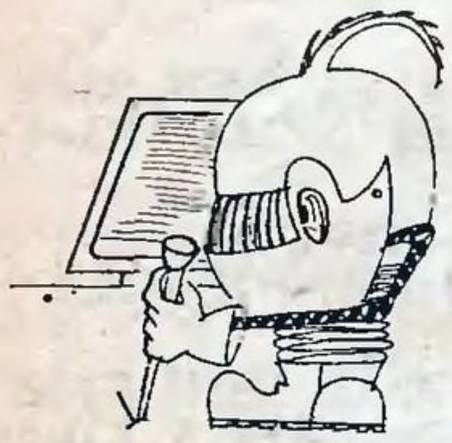
sko, ulica, miasto, państwo i kod pocztowy. Opisany program wprowadzania wykona większość pracy. Wystarczy wprowadzić go do pamięci. Po uruchomieniu program wyświetla maskę wejściową dla każdego z pięciu pól. Informuje także, które pole należy aktualnie wprowadzić.

Przeglądanie i zmienianie wprowadzonych danych

Po zakończeniu wprowadzania program może umożliwić dokonanie zmian poszczególnych pól. Wymaga to znajomości numeru zmienianego pola.

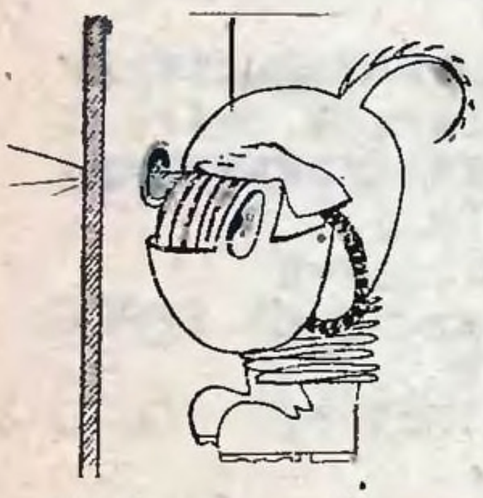
Możemy dodać do tego programu możliwość dokonywania zmian. Po zakończeniu wprowadzania jesteśmy pytani, czy chcemy dokonać zmian. Jeśli tak, to należy podać numer pola, które chcemy zmienić, i użyć odpowiedniej instrukcji do wywołania właściwego podprogramu wprowadzania.

PROGRAM NR 12 realizuje wszystkie opisane wyżej funkcje. Możliwe jest dodanie innych własności, według uznania.



Używanie manipulatorów do ograniczenia odpowiedzi

Jedną z trudności wszystkich sposobów wprowadzania danych jest mnogość wyborów, które ma użytkownik. Każdy dodatkowy wybór to możliwość powstania błędów. Program musi sprawdzać i eliminować niewłaściwe odpowiedzi. Rozwiązaniem tego problemu jest wyeliminowanie klawiatury jako urządzenia wprowadzania i użycie zamiast niej manipulatorów. Najłatwiejszym do adaptacji jest manipulator drążkowy (joystick). Nie zawsze jest możliwe użycie go zamiast klawiatury, lecz wielość sposobów jego użycia jest nieoczekiwana.



Sterowanie obrazem za pomocą manipulatora

Kiedy mamy do czynienia z dużymi ilościami danych, tylko część z nich może być wyświetlona jednocześnie na ekranie. Jednym ze sposobów jest użycie ekranu jako okna nad danymi. W dowolnej chwili pokazuje on tylko część z dostępnych danych. Przeglądanie danych w ten sposób jest łatwe, jeśli są one umieszczone w tablicy numerycznej lub nawet pseudotablicy tekstowej. Wyobraźmy sobie, że tablica danych zapisana jest na dużej

```
QO 1 REM ********************
IB 2 REM *                  *
RM 3 REM * Program nr 16 *
ID 4 REM *                  *
QS 5 REM ********************
FC 999 REM Kasowanie ekranu i wys
   wietlanie podzialki
LX 1000 PRINT CHR$(125)
BY 1010 POSITION 0,11
XL 1020 PRINT "012345678901234567
   8901234567890123456789"
FR 1030 FOR J=0 TO 23
UO 1040 POSITION 21,J
KE 1050 PRINT J-INT(J/10)*10;
FS 1060 NEXT J
ZN 1069 REM Wyswietlanie kursora
NK 1070 POSITION 21,11
JT 1080 PRINT CHR$(253);
KE 1089 REM
AY 1090 SC=21:SR=11
WZ 1099 REM Ograniczenia ruchu
CR 1100 LC=2:RC=39:TR=0:BR=23
VA 1109 REM Ustawienie szybkosci
   zmian
QD 1110 DLY1=20
CW 1119 REM Wywolanie podprogramu
    manipulatora
YZ 1120 GOSUB 6000
FF 1900 END
YF 5996 REM ********************
OF 5997 REM *Podprogram kursora *
YN 5998 REM ********************
TK 5999 REM Nacisnij przycisk w c
   elu zatrzymania kursora
YB 6000 IF STRIG(0)=0 THEN RETURN
NP 6009 REM Wolno w dol
KQ 6010 FOR J=1 TO DLY1:NEXT J
QD 6019 REM Sprawdzenie pozycji m
   anipulatora
UQ 6020 IF STICK(0)=7 AND SC<>RC
   THEN SC=SC+1:PRINT CHR$(31);
MX 6030 IF STICK(0)=11 AND SC<>LC
    THEN SC=SC-1:PRINT CHR$(30);
RO 6040 IF STICK(0)=13 AND SR<>BR
    THEN SR=SR+1:PRINT CHR$(29);
NN 6050 IF STICK(0)=14 AND SR<>TR
    THEN SR=SR-1:PRINT CHR$(28);
PO 6060 GOTO 6000
```

```
QO 1 REM ********************
IB 2 REM *                  *
SH 3 REM * Program nr 17 *
ID 4 REM *                  *
QS 5 REM ********************
KH 939 REM Kasowanie ekranu i wys
   wietlanie menu
LX 1000 PRINT CHR$(125)
VI 1010 PRINT "Wybierz polecenie
   przy pomocy          manipulato
   ra"
EF 1020 PRINT
GS 1030 PRINT "   DODAJ"
DR 1040 PRINT "   ZMIEN "
JB 1050 PRINT "   USUN"
TQ 1060 PRINT "   DRUKUJ"
ZQ 1070 PRINT "   POMOC"
XY 1080 PRINT "   KONIEC"
```

```
KE 1089 REM
YO 1090 POSITION 4,3
WO 1100 PRINT CHR$(29);
VA 1109 REM Ustawienie szybkosci
   zmian
SB 1110 SC=4:SR=3
JL 1119 REM
GZ 1120 LC=4:RC=4:TR=3:BR=8
JO 1129 REM
QU 1130 DLY1=30
ZF 1140 GOSUB 6000
SB 1150 POSITION 15,15
MM 1160 ON SR-2 GOSUB 2000,2500,3
   000,3500,4000:GOTO 1900
DE 1170 POSITION 2,10
FF 1900 END
KG 2000 PRINT "DODAJ    ":RETURN
CN 2500 PRINT "ZMIEN    ":RETURN
LH 3000 PRINT "USUN     ":RETURN
MN 3500 PRINT "DRUKUJ   ":RETURN
ZO 4000 PRINT "POMOC    ":RETURN
YF 5996 REM ********************
OF 5997 REM *Podprogram kursora *
YN 5998 REM ********************
TK 5999 REM Nacisnij przycisk w c
   elu zatrzymania kursora
YB 6000 IF STRIG(0)=0 THEN RETURN
NP 6009 REM Wolno w dol
KQ 6010 FOR J=1 TO DLY1:NEXT J
QD 6019 REM Sprawdzenie pozycji m
   anipulatora
UQ 6020 IF STICK(0)=7 AND SC<>RC
   THEN SC=SC+1:PRINT CHR$(31);
MX 6030 IF STICK(0)=11 AND SC<>LC
   THEN SC=SC-1:PRINT CHR$(30);
RO 6040 IF STICK(0)=13 AND SR<>BR
   THEN SR=SR+1:PRINT CHR$(29);
NN 6050 IF STICK(0)=14 AND SR<>TR
   THEN SR=SR-1:PRINT CHR$(28);
PO 6060 GOTO 6000


QO 1 REM ******************
IB 2 REM *                *
TC 3 REM * Program nr 18 *
ID 4 REM *                *
QS 5 REM ******************
RC 10 DIM R(10),R$(10)
QD 59 REM Otwarcie kanalu wejscio
   wego klawiatury
BO 60 OPEN #1,4,0,"K:"
DT 8230 GOSUB 8400:REM ROK
SV 8270 PRINT CHR$(31);
LH 8280 GOSUB 8400:REM MIESIAC
SO 8340 PRINT CHR$(31);
IZ 8350 GOSUB 8400:REM DZIEN
BK 8380 RETURN
GS 8397 REM ********************
NX 8398 REM * POBIERZ 2 CYFRY *
HA 8399 REM ********************
FM 8400 FOR J1=0 TO 1
BI 8410 GET #1,R:R(J1)=R
PF 8440 PRINT CHR$(R);:REM ECHO
CU 8450 NEXT J1
QZ 8460 R$=CHR$(R(0)):R$(2)=CHR$(
   R(1))
BJ 8470 RETURN
```

```
QO 1 REM ******************
IB 2 REM *                *
TX 3 REM * Program nr 19 *
ID 4 REM *                *
QS 5 REM ******************
ID 10 DIM MSK$(2),R$(2),DAT$(8),R
   (1)
FC 20 MSK$="__"
KQ 59 REM Otwarcie kanalu klawiat
   ury
BO 60 OPEN #1,4,0,"K:"
YU 1000 PRINT CHR$(125);"Podaj ak
   tualna date w postaci:
                rok-miesiac-dzien."
RW 1010 IC=15:IR=4
ZQ 1020 GOSUB 8200:REM Wprowadzen
   ie daty
IM 1030 POSITION 15,6
EW 1040 PRINT DAT$
FC 1050 END
XW 8197 REM ************************
OF 8198 REM * Wprowadzenie daty *
YE 8199 REM ************************
WG 8200 POSITION IC,IR:PRINT CHR$
   (253);
IA 8210 PRINT MSK$;"-";MSK$;"-";M
   SK$;
WM 8220 POSITION IC,IR:PRINT CHR$
   (253);
DT 8230 GOSUB 8400:REM ROK
BO 8240 Y=VAL(R$)
HB 8250 IF Y<0 OR Y>99 THEN 8200
AO 8260 DAT$(1,2)=R$:DAT$(3,3)="-
   "
SV 8270 PRINT CHR$(31);
LH 8280 GOSUB 8400:REM MIESIAC
ZJ 8290 M=VAL(R$)
KU 8300 IF M<1 OR M>12 THEN 8200
IP 8310 DAT$(4,5)=R$:DAT$(6,6)="-
   "
SI 8320 PRINT CHR$(31);
UL 8325 IF D>31 THEN 8200
IT 8330 GOSUB 8400:REM DZIEN
WU 8340 D=VAL(R$)
ZX 8350 IF D<1 THEN 8200
WW 8360 IF M=2 AND D>29 THEN 8200
SS 8370 IF (M=4 OR M=6 OR M=9 OR
   M=11) AND D>30 THEN 8200
VA 8375 IF D>31 THEN 8200
DV 8380 DAT$(7,8)=R$
CH 8395 RETURN
GS 8397 REM ********************
QB 8398 REM * Pobierz 2 cyfry *
HA 8399 REM ********************
FM 8400 FOR J1=0 TO 1
BI 8410 GET #1,R:R(J1)=R
RB 8419 REM Klawisz <BACK SPACE>
   oznacza restart
AO 8420 IF R=126 THEN POP :GOTO 8
   200
OL 8429 REM Ignorowanie klawiszy
   literowych
PB 8430 IF R<48 OR R>57 THEN 8410
PF 8440 PRINT CHR$(R);:REM ECHO
CU 8450 NEXT J1
QZ 8460 R$=CHR$(R(0)):R$(2)=CHR$(
   R(1))
BJ 8470 RETURN
```

planszy i oglądamy ją przez wizjer kamery. Plansza jest wystarczająco duża, aby nie można było jej objąć w całości, lecz możemy obejrzeć dowolną jej część przez przesuwanie wizjera w górę, w dół, w prawo lub w lewo. Ekran może imitować wizjer, a manipulator może sterować jego ruchem nad polem danych.

Pokażemy, jak zastosować opisaną technikę dla tablicy numerycznej dwuwymiarowej. Jako wartość każdego elementu tablicy przydzielimy liczbę czterocyfrową, która określa numer wiersza i kolumny danego elementu. Na ekranie występują nagłówki wierszy i kolumn. Utworzone okno jest mniejsze niż cały ekran, aby lepiej zilustrować koncepcję okna nad danymi. Nie ma przeszkód, aby utworzyć okno zajmujące cały ekran.

Cały PROGRAM (NR 13) jest stosunkowo prymitywny. Ma tylko jedną szybkość. Zużywa około dwóch sekund na ponowne wyświetlenie okna za każdym razem, gdy zmieni się numer wiersza lub kolumny; daje to 20s na przesunięcie okna o dziesięć wierszy. Można poprawić nieco program i skrócić ten czas dwukrotnie, lecz dziesięć sekund to nadal zbyt długo. Zamiast ciągłego wyświetlania okna tak często jak to możliwe, gdy manipulator wychylony jest w jednym kierunku, program mógłby wyświetlać ponownie dopiero wtedy, gdy przyjmie on położenie środkowe. W ten sposób okno wyświetli się tylko raz dla każdego ciągłego ruchu. Możemy w ten sposób zredukować czas przesuwania okna do dwóch sekund plus czas na odchylenie manipulatora od środka. Oczywiście program będzie aktualizował numery kolumn w miarę przesuwania okna poziomo i numery wierszy w czasie ruchu pionowego, zatem zna on położenie okna. Spróbujcie sami dokonać tych zmian w programie.

Można także przesuwać okno po przekątnej. Spróbujcie rozszerzyć program nr 13, między liniami 300 i 330 aby umożliwić taki ruch okna. Po wykryciu ukośnej pozycji manipulatora program musi zmienić jednocześnie wiersz i kolumnę, zmienne R i C.

Wprowadzanie liczb manipulatorem

Możemy napisać program, który używa manipulatora do wprowadzania wartości numerycznych. Rozpoczyna się on od wyświetlania liczby na ekranie. Następnie sprawdza położenie manipulatora. Ruch w lewo powoduje zmniejszenie liczby. Ruch w prawo i liczba zwiększa się. Ustawienie na środku przerywa zmiany. Gdy właściwa liczba jest na ekranie, naciskamy przycisk manipulatora. **PROGRAM NR 14 demonstruje wprowadzanie liczb od 1 do 100.**

Program działa poprawnie, lecz niełatwo jest wybrać określoną liczbę. Jest on zbyt wrażliwy na ruch manipulatora oraz zbyt często sprawdza jego położenie. Należy wprowadzić pętlę opóźniającą (linia 6560).

Lepiej wykonany jest PROGRAM NR 15. Używa tych samych zmiennych co ogólny podprogram wprowadzania dla określenia zakresu wprowadzania wartości (LO i HI), położenia kursora (IR i IC) i wielkości pola (IL). Wprowadzoną wartość zwraca w zmiennej R. Podprogram z tego programu używa dwóch pętli opóźniających do sterowania szybkości zmiany liczby (linia 6560). Startując od małego opóźnienia (linia 6570) dla maksymalnej szybkości zmian. Gdy manipulator jest w położeniu centralnym, podprogram przestawia się na małą szybkość zmian (linia 6600). Jeśli zatem przesuniemy manipulator w prawo lub w lewo i przytrzymamy, liczba zmienia się z dużą szybkością. Szybkie przesunięcie w lewo lub prawo i zmiany następują wolniej.

```
QO 1 REM ********************
IB 2 REM *                  *
NK 3 REM * Program nr 20 *
ID 4 REM *                  *
QS 5 REM ********************
IK 9 REM Inicjowanie zmiennych
DK 10 DIM MSK$(40),R$(20)
WZ 30 MSK$="_":MSK$(40)=MSK$:MSK$
   (2)=MSK$
KP 49 REM Otwarcie kanalu klawiat
   ury
BN 50 OPEN #1,4,0,"K:"
QB 89 REM Skok do programu glowne
   go
IC 90 GOTO 1000
MF 595 REM ******************
BH 596 REM *  Podprogram  *
FH 597 REM * wprowadzania *
SZ 598 REM * ciagu znakow *
MR 599 REM ******************
AZ 600 R$=""
UL 610 POKE 755,0:REM Wylaczenie
   kursora
FA 630 POSITION IC,IR
PH 640 PRINT MSK$(1,IL)
SN 649 REM Poczatek pola
FE 650 POSITION IC,IR
BU 659 REM Wczytaj znak
KE 660 GET #1,R
SZ 670 J=LEN(R$)
AM 679 REM Wlaczenie kursora jesl
   i RETURN
LZ 680 IF R=155 THEN POKE 755,2:R
   ETURN
PR 689 REM Jesli znak dobry to do
   daj do ciagu
AF 690 IF R>=32 AND R<=95 AND J<I
   L THEN R$(J+1,J+1)=CHR$(R):PRI
   NT CHR$(R);:GOTO 660
OJ 699 REM Czy znak kasowania
VO 700 IF R<>126 OR J=0 THEN GOTO
    660
CN 709 REM Kasowanie ostatniego z
   naku
VI 710 POSITION IC+J-1,IR
MV 720 PRINT MSK$(1,1);
VM 730 POSITION IC+J-1,IR
UX 740 IF J>1 THEN R$=R$(1,J-1)
RC 750 IF J=1 THEN R$=""
QK 760 GOTO 660
GI 1000 PRINT CHR$(125):REM CLS
XD 1010 POSITION 2,4
LH 1020 PRINT "Imie: "
AV 1030 IC=8:IR=4:IL=20:GOSUB 600
AC 1040 POSITION 2,8
VV 1050 PRINT "WPROWADZONO: ";R$
NB 1060 CLOSE #1
FI 1070 END
```

Wykorzystanie menu do ograniczenia wybrów

Najłatwiejszym sposobem wyeliminowania błędów jest uważne zaprojektowanie programu w taki sposób, aby użytkownik miał możliwie jak najmniej możliwości wyboru. Istota pytań zadawanych użytkownikowi przez program może uczynić jego pracę łatwą lub trudną. W niektórych programach trzeba wypełnić puste miejsca. Czasami wypełnianie jest jedynym wyborem. Kiedy indziej będzie pytanie o wielu wyborach. Zamiast „Co chcesz zrobić?", program pyta „Którą opcję wybierasz?". Na tym polega istota menu.

Można wrócić do programu nr 6, który wprowadza polecenia. Do wyboru były: A, C, D, E, P lub H. Zamiast tego można przygotować menu realizujące to samo. Korzystając z prezentowanych podprogramów możemy łatwo napisać program do wyświetlenia menu i wprowadzania poleceń.

Wykorzystanie menu jest lepsze dla użytkownika i programisty. Użytkownik nie musi pamiętać lub szukać dostępnych opcji. Programista nie musi pisać skomplikowanych procedur, które wyświetlą rozszerzenia poleceń. Nie ma gwarancji, że użytkownik wprowadzi tylko wyświetlone opcje, zatem program wciąż musi sprawdzać poprawność wprowadzanych informacji.

Prawie wszystkie wejścia mogą być rozłożone na ciąg pytań o wielu wyborach. Każde z nich może być przedstawione jako menu. Użytkownik przechodzi swoją drogą przez menu, aż do udzielenia odpowiedzi na końcowe pytanie.

Używanie manipulatora do wyboru menu

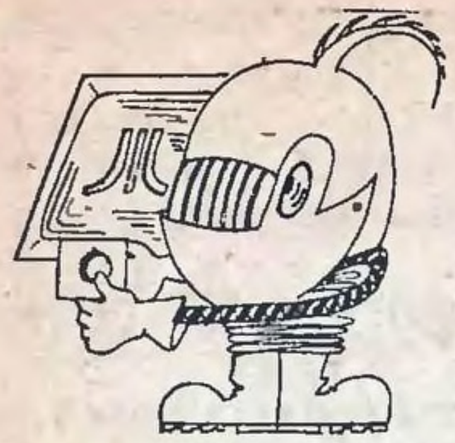
Komputer może być zaprogramowany do przesuwania kursora po ekranie pod kontrolą manipulatora. Jeśli na ekranie wyświetlone jest menu, użytkownik przesuwa kursor, aż osiągnie jeden z wyborów menu. Wtedy naciska przycisk manipulatora dla dokonania wyboru. Program to wykrywa, zna położenie kursora i określa, która pozycja menu odpowiada temu położeniu.

Podprogram kursora z PROGRAMU NR 16 „przywiązuje" manipulator do kursora. Wykrywa położenie manipulatora, wyświetla odpowiednie znaki ruchu kursora i aktualizuje wartości zmiennych przechowujących położenie kursora. **PROGRAM NR 16 demonstruje jego działanie.**

Następny PROGRAM (NR 17) używa opisanej procedury do wprowadzania wyboru z menu. Wyświetla menu i umieszcza kursor obok pierwszego polecenia. Manipulator może przesuwać kursor w górę i dół, nie może jednak przesuwać go w bok. Jak powyżej, naciśnięcie przycisku zatrzymuje kursor; zmienne SC i SR przechowują położenie kursora.

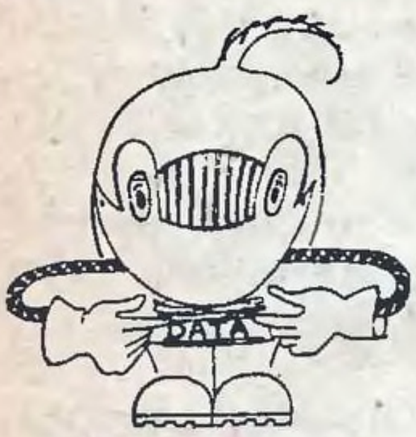
Pozostaje przetłumaczenie pozycji kursora na wybrane polecenie i działanie zgodnie z nim. **W PROGRAMIE NR 17 całe działanie sprowadza się do wyświetlenia nazwy wybranego polecenia.**

```
QO 1 REM **********************
IB 2 REM *                    *
OF 3 REM * Program nr 21      *
ID 4 REM *                    *
QS 5 REM **********************
OA 845 REM ********************
PN 846 REM * Wylaczenie       *
XN 847 REM *  klawisza         *
EN 848 REM *    BREAK          *
OM 849 REM ********************
SS 850 J=PEEK(16)-128
SO 860 IF J<0 THEN RETURN
TZ 870 POKE 16,J
KV 880 POKE 53774,J
ZW 890 RETURN
1 REM **********************
2 REM *                    *
3 REM * Program nr 22      *
4 REM *                    *
5 REM **********************
19 REM Kody ATASCII symboli uz
ywanych w grach
20 DATA 0,16,19,20,96,123
999 REM Otwarcie kanalu we/wy
ekranu
1000 OPEN #6,12,0,"S:"
1010 PRINT CHR$(125)
1019 REM Wyswietlenie zestawu
symboli
1020 PRINT "Wybierz swoj symbo
l"
1030 RESTORE 20
1040 FOR J=3 TO 8
1050 READ TOKEN
1060 POSITION 4,J
1070 PRINT CHR$(TOKEN)
1080 NEXT J
1089 REM Start kursora w 4,3
1090 POSITION 4,2
1100 PRINT CHR$(29);
1110 SC=4:SR=3
1119 REM Okreslenie zakresu ku
rsora
1120 LC=4:RC=4:TR=3:BR=8
1129 REM Szybkosc kursora
1130 DLY1=30
1140 GOSUB 6000:REM Przesuwani
e kursora
1149 REM Pobranie znaku z ekra
nu
1150 LOCATE SC,SR,TOKEN
1159 REM Zamiana symbolu na ne
gatyw
1160 TOKEN=TOKEN-128*SGN(TOKEN
-127)
1170 POSITION 2,12
1180 PRINT "Wybrales  ";CHR$(T
OKEN)
1900 END
```



Wprowadzanie znaków manipulatorem

Używając funkcji **LOCATE** oraz podprogramu kursora możliwe jest wprowadzenie dowolnego znaku bezpośrednio z ekranu za pomocą manipulatora. Program wyświetla najpierw znaki, z których można wybierać. Użytkownik przesuwa kursor od znaku do znaku manipulatorem. Gdy kursor zatrzyma się na żądanym znaku, trzeba nacisnąć przycisk. Program odczytuje wartość z ekranu (za pomocą LOCATE). **Technikę tę ilustruje PROGRAM NR 22.** Wymaga on podprogramu kursora (z programu nr 16).



Wprowadzanie daty

Aby wprowadzić datę musimy podać rok, miesiąc i dzień jako liczby dwucyfrowe. Program dostarcza separatorów rozdzielających poszczególne pozycje. Separatorem może być kreska ułamkowa lub dowolny inny znak.

Użytkownik musi wiedzieć, gdzie wprowadzić następne dane. Dlatego program będzie używał maski wprowadzania. **Instrukcje z PROGRAMU NR 18 akceptują wejście dwucyfrowe.** Program automatycznie kończy wprowadzanie danych po podaniu dwóch cyfr. Użytkownik nie musi naciskać klawisza **RETURN.**

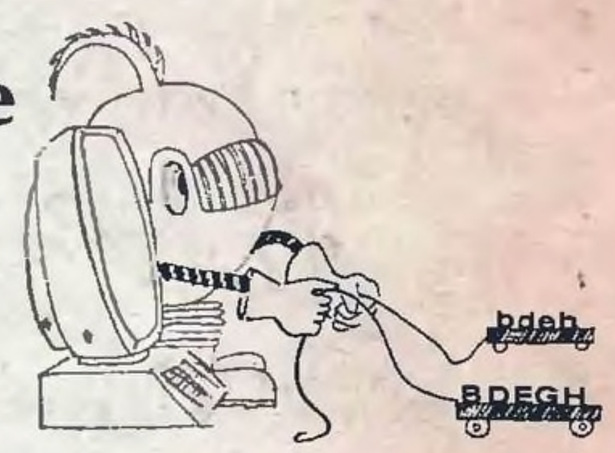
Istnieją różne sposoby wyeliminowania pewnych błędów podczas wprowadzania daty. Należą do nich:

— przyjmowanie tylko znaków numerycznych (cyfr),
— testowanie poprawności roku, miesiąca i dnia,
— dostarczenie środków ponownego wprowadzania daty.

PROGRAM NR 19 zawiera cały podprogram wprowadzania daty obejmujący wymienione ulepszenia. Dozwolone są tylko dane numeryczne (linia 8430). Miesiąc musi mieć wartość między 1 a 12 (linia 8300). Program nie bierze pod uwagę setek lat, lecz sprawdza maksymalną liczbę dni w określonym miesiącu (linie 8350 do 8375). Dozwolony jest dowolny rok od 00 do 99 (linia 8250). Wprowadzenie niepoprawnej daty wznawia cały ciąg wprowadzania. Jeśli użytkownik naciśnie klawisz **BACK SPACE**, również wznawiany jest ciąg wprowadzania daty (linia 8420).

Zauważmy, że data tworzona jest w postaci ciągu ośmioznakowego **DAT$** jako rok, miesiąc i dzień (linia 8260, 8310 i 8380).

Wprowadzanie ciągu znaków



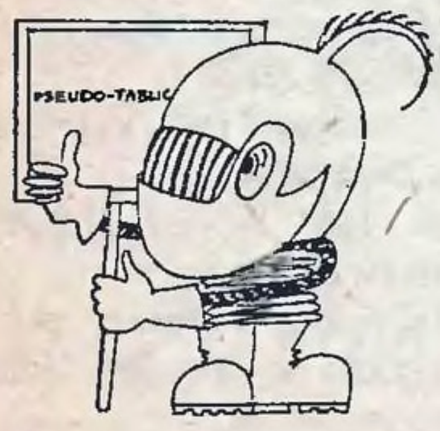
Uniwersalny PODPROGRAM NR 11 wprowadzania miał znaczne uproszczenia. Umożliwiał użytkownikowi wprowadzanie pozycji, które były dłuższe od maski wejścia. Co gorsze, użytkownik mógł przesuwać kursor przez cały ekran klawiszami strzałek. Wszystko to powodowało, że wcześniej czy później użytkownik mógł zniszczyć obraz. Jeśli użyjemy instrukcji GET zamiast INPUT, możemy dokładniej kontrolować wprowadzanie.

PROGRAM NR 20 zawiera podprogram, który wprowadza ciąg znaków. Zauważmy, że naciśnięcie klawisza RETURN powoduje wygenerowanie znaku EOL, który kończy wprowadzanie. Klawisz BACK'S powoduje cofnięcie o jeden znak.

Istnieje kilka sposobów polepszenia tego podprogramu. Poniżej wymienione są pewne pomysły:

— użycie zmiennej do określania numeru kanału wejściowego,
— zezwolenie na wprowadzanie wielkich i małych liter (kody ATASCII 97 do 122),
— dodanie na początku instrukcji TRAP w celu uaktywnienia procedury obsługi błędu od linii ERRHDL,
— wywołanie podprogramu wyświetlania instrukcji rozszerzonych (od linii **AMPSUB**), jeśli wprowadzony jest określony znak.

W ostatnim kroku przed wyjściem z podprogramu powinniśmy sprawdzić, czy wprowadzone informacje są numeryczne. Jeśli tak, to trzeba przetworzyć ciąg znaków na wartość numeryczną i sprawdzić jej zakres. Możemy polegać na instrukcji **TRAP** i podprogramie obsługi błędu w zakresie wykrywania błędów wprowadzania. Nie próbujmy sprawdzać zakresu liczby przed przetłumaczeniem jej na wartość numeryczną.



Działania na zmiennych tekstowych

Dużą wadą języka Atari BASIC jest brak tablic tekstowych. Można ją jednak wyeliminować przez symulowanie takich tablic za pomocą zmiennych tekstowych. Taki twór nazwiemy pseudotablicą. Nasze działanie polega na podzieleniu ciągu znaków na podciągi o stałej długości (warunek podstawowy) i traktowaniu każdego podciągu jako elementu pseudotablicy. W celu wyznaczenia pozycji początkowej elementu pseudotablicy musimy znać numer i długość każdego elementu. Istnieją dwa podstawowe ograniczenia:

1) wszystkie elementy pseudotablicy muszą mieć taką samą długość. Zwykle określa ją najdłuższy element. Jeśli wprowadzany ciąg jest krótszy, jego nieużywaną część musimy wypełnić spacjami,
2) obliczanie adresu początkowego elementu pseudotablicy jest czasochłonne. Można zauważyć opóźnienia w realizacji programu.

Program „SYMULACJA TABLICY TEKSTOWEJ" wykorzystuje pseudotablicę przechowywaną w A$. Zawiera ona dziesięć elementów o długości dziesięciu znaków każdy. Ele-

```
HQ 1 REM ************
ER 2 REM *          *
QC 3 REM * Symulacja *
FT 4 REM *  tablicy  *
SA 5 REM * tekstowej *
EV 6 REM *          *
HW 7 REM ************
HO 10 DIM A$(100),TEMP$(10),BL$(1
   0)
LK 19 REM Dziesiec wartosci ciagu
IL 20 DATA Ryszard,Natalia,Antoni
   ,Marek,,12345678901,+++,A+B=1E
   100,**=,abrakadabra
KK 29 REM zapelnienie BL$ spacjam
   i
MU 30 BL$=" ":BL$(10)=BL$:BL$(2)=
   BL$
QP 39 REM Przydzielenie wartosci
   pseudo-tablicy
HS 40 FOR N=1 TO 10
VW 49 REM Najpierw wstawienie war
   tosci do ciagu przejsciowego
NP 50 READ TEMP$
MZ 59 REM Wyznaczenie dlugosci no
   wej wartosci
SD 60 TL=LEN(TEMP$)
QX 69 REM Uzupelnienie ciagu spac
   jami
VV 70 IF TL<10 THEN TEMP$(TL+1)=B
   L$
IB 79 REM Wyznaczenie adresu elem
   entu tablicy
ZP 80 START=(N-1)*10+1
PK 89 REM Przydzielenie wartosci
   do elementu tablicy
QP 90 A$(START)=TEMP$
HP 100 NEXT N
VV 109 REM Wyswietlenie przydziel
   onych wartosci
SG 110 FOR N=1 TO 10
HI 119 REM Obliczenie poczatku el
   ementu
QN 120 START=(N-1)*10+1
TE 130 PRINT "Element ";N;" to: "
   ;A$(START,START+9)
HX 140 NEXT N
OA 150 END
```

```
NY 1 REM ***************
DR 2 REM *             *
XE 3 REM * Podprogram *
KX 4 REM * sortowania *
OC 5 REM * elementow  *
EV 6 REM *  tablicy   *
DW 7 REM *.            *
OF 8 REM ***************
NO 9 REM
ZS 22050 DATA 104,104,104,133,205
   ,104,104,133,206,104,104,133,2
   03,104,104,133,204
DJ 22060 DATA 104,133,217,104,133
   ,216,104,133,209,104,133,208,1
   69,0,133,218,133,207,162
DS 22070 DATA 1,165,216,133,214,1
   65,217,133,215,24,165,214,133,
   212,101,205,133,214,165,215
GE 22080 DATA 133,213,105,0,133,2
   15,164,203,165,206,240,10,177,
   214,209,212,144,44,240,12
OF 22090 DATA 176,19,177,214,209,
   212,144,13,240,2,176,30,200,19
   6,204,240,227,176,23,144
OI 22100 DATA 223,169,1,133,218,1
   64,205,136,177,214,72,177,212,
   145,214,104,145,212,192,0
ON 22110 DATA 208,241,232,224,0,2
   08,2,230,207,228,208,208,172,1
   65,209,197,207,208,166,165
AM 22120 DATA 218,201,0,208,144,9
   6
YD 22200 W=0:V=1:RESTORE 22050
JX 22210 DIM SO$(142)
GJ 22220 FOR I=1 TO 142:READ A:SO
   $(I,I)=CHR$(A):NEXT I
EP 22225 RETURN
```

```
FK 1 REM ****************
FJ 2 REM *              *
AZ 3 REM *  Podprogram  *
EV 4 REM * wyszukiwania *
JH 5 REM *   elementu   *
AN 6 REM *   tablicy    *
FO 7 REM *              *
FR 8 REM ****************
NO 9 REM
VT 20000 DATA 216,104,104,133,204
   ,104,133,203,104,104,133,205,1
   04,104,133,206,104,133,208,104
   ,133,207,104,104
```

```
W0 20010 DATA 133,209,104,104,133
   ,210,169,0,133,212,133,213,24,
   165,203,101,209,133,203,165,20
   4,105,0,133
VH 20020 DATA 204,164,210,136,177
   ,203,209,207,208,4,136,16,247,
   96,24,165,203,101,205,133,203,
   165,204,105
IP 20030 DATA 0,133,204,24,165,21
   2,105,1,133,212,197,206,208,21
   9,96,-1
NG 20040 RESTORE 20000:DIM SZ$(87
   )
BN 20050 FOR I=1 TO 87
EY 20060 READ A:SZ$(I,I)=CHR$(A)
GM 20070 NEXT I
FX 20080 READ A:IF A=-1 THEN RETU
   RN
JO 20090 ? "BLAD DANYCH":STOP
```

menty tablicy pobierane są z instrukcji **DATA**. Włączono do nich ciąg pusty oraz dwa ciągi zbyt długie, aby zmieścić się w jednym elemencie tablicy. Każdy ciąg z **DATA** wstawiany jest do zmiennej tekstowej **TEMP$**. Jest to konieczne, ponieważ Atari BASIC nie zezwala na używanie zmiennych ze wskaźnikami w instrukcji **READ**. Program dołącza spacje do krótkich ciągów w celu usunięcia pozostałości poprzednich wartości (linia 60 i 70). W końcu wyznacza indeks elementu pseudotablicy i przydziela mu wartość utworzoną w zmiennej **TEMP$**. Obejrzyjmy, co stało się z pustym ciągiem i wartościami, które są zbyt długie.

Elementy tablicy tekstowej lub pseudotablicy są zwykle rekordami składającymi się z kilku pól. Większość działań wykonywanych na nich polega na sortowaniu rekordów według określonego klucza, którym jest zwykle jedno z pól oraz wyszukiwaniu rekordu na podstawie zawartości jednego z pól. Atari BASIC, w odróżnieniu od nowszych, bardziej rozbudowanych wersji tego języka, nie ma niestety wbudowanych funkcji sortowania i wyszukiwania. Podprogramy realizujące te funkcje są bardzo łatwe do napisania i zawierają kilka linii programu. Dlatego nie śmielibyśmy nawet prezentować ich czytelnikom. Mają one jednak podstawową wadę: bardzo długi czas działania, rzędu kilku a nawet kilkunastu minut. Zdarzyło się nam stracić cierpliwość podczas oczekiwania na posortowanie kilkudziesięciu rekordów. **Efektem tego są dwa podprogramy w języku maszynowym: SORTOWANIE oraz WYSZUKIWANIE.** Po ich wykorzystaniu czas oczekiwania na zakończenie tych operacji uległ znacznemu skróceniu.

Oba podprogramy dostarczamy w postaci umożliwiającej włączenie ich do własnych programów. Po wykonaniu skoków do odpowiednich podprogramów, za pomocą instrukcji **GOSUB**, interesujące nas podprogramy zostają umieszczone w ciągach. Są to procedury przemieszczalne (relokowalne) i mogą być umieszczone gdziekolwiek w programie.

Sposoby wywołania obu podprogramów są podobne. I tak wywołanie procedury sortowania ma następującą postać:

A=USR (ADR (SO$), DR, TYP, PP, PK, ADR (X$), IR)

gdzie poszczególne parametry mają następujące znaczenie:

- SO$ — nazwa ciągu zawierającego podprogram sortowania,
- DR — długość rekordu (element pseudotablicy),
- TYP — rodzaj sortowania: 0 — wzrastający, 1 — malejący,
- PP — pozycja początkowa klucza sortowania w rekordzie,
- PK — pozycja końcowa klucza sortowania w rekordzie
- X$ — nazwa pseudotablicy zawierającej sortowane elementy,
- IR — ilość elementów pseudotablicy (rekordów), które należy posortować.

Pozycje początkowe i końcowe klucza sortowania powinny być podane w postaci liczby bajtów od początku rekordu. Na przykład, pierwsza pozycja rekordu ma przesunięcie 0, druga —1, a setna —99.

Podprogram ten sortuje rekordy o stałej długości od 2 do 250 bajtów. Klucz sortowania może być umieszczony gdziekolwiek w rekordzie i może być dowolnej długości. Rekordy muszą być w postaci znaków ATASCII.

Wywołanie podprogramu wyszukiwania rekordu na podstawie zawartości jednego z jego pól ma następującą postać:

K=USR (ADR (SZ$), ADR (D$), DR, IR-1, ADR (P$), KP, DP)

gdzie poszczególne parametry mają następujące znaczenie:

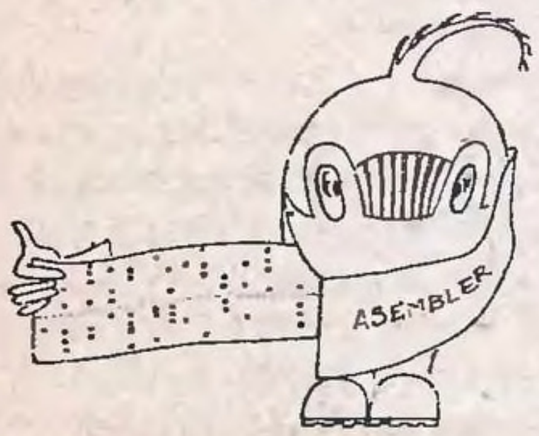
- SZ$ — nazwa ciągu zawierającego podprogram wyszukiwania,
- D$ — nazwa ciągu zawierającego elementy pseudotablicy,
- DR — długość rekordu
- IR — ilość rekordów (elementów pseudotablicy),
- P$ — nazwa ciągu zawierającego poszukiwane pole,
- KP — przesunięcie poszukiwanego pola względem początku rekordu,
- DP — długość poszukiwanego pola.

Po zakończeniu działania zmienna K będzie zawierać numer rekordu zawierającego poszukiwane pole (pierwszego jaki wystąpił).

Unieszkodliwienie klawisza BREAK

Nawet najlepiej zaprojektowany program wciąż ma słaby punkt. Można zatrzymać jego realizację (czasami przypadkowo) poprzez naciśnięcie klawisza **BREAK**. Automatycznie na ekranie pojawia się komunikat, który rujnuje pieczołowicie utworzony obraz. Prawdopodobnie nie będzie możliwe poprawne kontynuowanie programu po takim przerwaniu, ponieważ instrukcja **CONT** wznawia działanie od początku linii programu, w której nastąpiło przerwanie. Jeśli zdarzy się, że jest to linia o wielu instrukcjach, pewne z nich będą wykonane ponownie.

Istnieje sposób unieszkodliwienia klawisza BREAK (PROGRAM NR 21). Niestety jest on wrażliwy na wiele zdarzeń, które przywracają normalne działanie klawisza. Należą do nich: naciśnięcie klawisza RESET, pierwsza instrukcja **PRINT**, dowolna instrukcja **OPEN** dotycząca ekranu (urządzenia o kodach E: lub S:), pierwsza instrukcja **PRINT** po takim **OPEN** oraz **GRAPHICS**. Najprostszym sposobem pokonania tych ograniczeń jest częste wykonywanie podprogramu wyłączenia klawisza **BREAK**. Dobrym miejscem wywoływania go jest podprogram wprowadzania.



Asembler

Wykorzystanie firmowego programu Assembler-Editor przez posiadaczy komputera Atari z pamięcią zewnętrzną w postaci magnetofonu odznacza się wieloma niedogodnościami. Należą do nich;

— długi czas wprowadzania programu (ok. 15 minut) spowodowany tym, że wymaga uprzedniego załadowania Translatora, czyli starej wersji systemu operacyjnego (opracowanej dla komputerów Atari 400 i 800),
— konieczność przestudiowania kilkudziesięciostronicowej instrukcji obsługi programu,
— program wynikowy, przechowany na kasecie magnetycznej i wprowadzony do komputera, nie chce działać.

W celu uniknięcia opisanych trudności można wykorzystać prezentowany program, który udaje asembler (to znaczy symuluje pewne jego funkcje). Umożliwia on przygotowanie krótkich podprogramów w języku maszynowym i przechowanie ich w wybranym obszarze pamięci (najczęściej na stronie 6 RAM). W programie tym nie ma odrębnych faz edycji i translacji, wprowadzane rozkazy wpisywane są (POKE) bezpośrednio do wybranego obszaru pamięci. Rozkazy podawane są w postaci symbolicznej. Wartości argumentów mogą być wprowadzone dziesiętnie lub szesnastkowo, zgodnie z konwencją przyjętą po uruchomieniu programu, która może być zmieniona w każdej chwili. Istnieje możliwość podania skoku do przodu bez znajomości aktualnego adresu przeznaczenia. W tym celu w charakterze argumentu podajemy słowo **WPRZÓD**. Program wstawia wstępnie wartość zerową do bajtu argument. Aktualny adres skoku zostanie wpisany po rozpoznaniu słowa **ETYKIETA**. Wtedy do zaznaczonej komórki wprowadzany jest adres bieżącej komórki (a dokładniej jego odległość od rozkazu skoku).

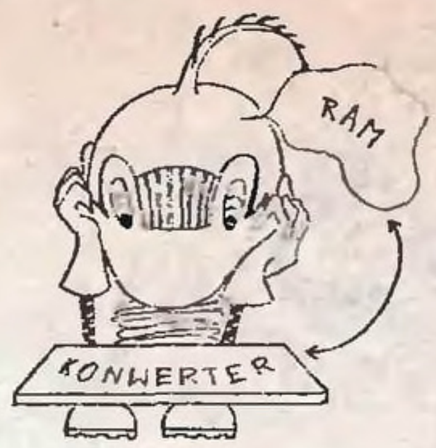
Komenda **LIST** umożliwia wyświetlenie adresów i zawartości poszczególnych komórek. Możliwe jest również umieszczenie dowolnej liczby, z przedziału 0—255, w wybranej komórce pamięci. Służy do tego funkcja **WSTAW,** po której należy podać adres i zawartość komórki, obie wartości koniecznie dziesiętnie.

Za pomocą poleceń **HEX** lub **DZIES** w każdej chwili możemy przejść z jednego systemu liczenia do drugiego. Zakończenie działania programu uzyskamy po napisaniu **KONIEC**. Otrzymamy wówczas adresy początku i końca procedury w języku maszynowym.

Prezentowany program nie pretenduje do roli prawdziwego asemblera, brakuje mu wielu funkcji oraz możliwości podawania nazw symbolicznych komórek. Pomimo to może ułatwić początkową naukę języka asemblera. Zamiast wstawiać kody rozkazów za pomocą **POKE**, możemy pisać je w postaci symbolicznej, jak w prawdziwym asemblerze. Przydaje się on do pisania prostych i krótkich procedur w języku maszynowym.

```
HQ 1 REM **************
ER 2 REM *              *
OB 3 REM * ASSEMBLER *
ET 4 REM *              *
HU 5 REM **************
HM 10 HX=0:REM IF HX=0 TO ASEMBLE
   R DZIALA NA LICZBACH DZIESIETN
   YCH
CT 20 DIM HE$(16),ZO$(3),R$(10),M
   N$(12),ZA$(1),AZ$(1),L$(3),SA$
   (4),H$(4),LR$(1)
HW 30 OPEN #1,12,0,"E:"
DY 50 HE$="0123456789ABCDEF":SZ=1
   :ZO$="000"
EU 100 ? "   KONWENCJA STOSOWANA
   W ASSEMBLERZE"
LA 110 DIM M$(56*3),TY(56),OP(56)
CU 120 FOR I=1 TO 56:READ MN$:M$(
   I*3-2,I*3)=MN$(1,3)
MB 122 TY(I)=VAL(MN$(4,4)):OP(I)=
   VAL(MN$(5))
KI 130 NEXT I:? :? :?
ZH 140 ? "Sposob adresowania":?
XE 150 ? "Bezposredni          LDA
   #15"
SG 155 ? "Absolutny           LDA
   1500"
GE 160 ? "Strony zerowej      LDA
   15"
NQ 165 ? "Akumulator          ASL
   "
QN 170 ? "Posredni X          LDA
   (15X)"
UY 175 ? "Posredni Y          LDA
   (15)Y"
TD 177 ? "Strony zerowej  X   LDA
   15X"
WD 179 ? "Strony zerowej  Y   LDX
   15Y"
ZZ 180 ? "Absolutny  X        LDA
   1500X"
CR 185 ? "Absolutny  Y        LDA
   1500Y"
GN 190 ? "Liczby dziesietne D czy
   szesnastkowe S";:INPUT MN$
KW 192 HX=0:IF MN$(1,1)="S" THEN
   HX=1
JV 194 POSITION 0,13:? CHR$(156);
   CHR$(156);:IF HX=1 THEN ? "SZE
   SNASTKOWO"
XO 195 IF HX=0 THEN ? "DZIESIETNI
   E"
BU 197 ? :? "Adresy: stosuj 1536-
   1791 ($0600-$06FF)":? :?
DA 200 ? CHR$(156);CHR$(156);"Pod
   aj adres poczatku ":? "  twoj
   ego programu";:INPUT SA$
WX 205 IF SA$="" THEN ? CHR$(28);
   CHR$(28);:GOTO 200
VR 210 IF HX=1 THEN H$=SA$:GOSUB
   5000:SA=DE:GOTO 217
NC 215 SA=VAL(SA$)
JQ 217 IF SA<256 OR SA>=40960 THE
   N ? CHR$(28);CHR$(28);CHR$(28)
   ;CHR$(28);"Zly adres !!!":? :G
   OTO 200
GE 220 TA=SA:? CHR$(12):GOTO 230
AI 225 ? :? CHR$(253);"BLAD WEJSC
   IA":? :IF HX=1 THEN ? "(n.p. #
   5 winno byc #05)":?
JY 230 IF HX=1 THEN DE=SA:SZ=3:GO
   SUB 4000:? H$;": ";:GOTO 240
SI 235 ? SA;": ";
CG 240 TRAP 225:INPUT #1;MN$:? CH
   R$(28);:POKE 85,20:IF MN$="" T
   HEN ? CHR$(156);:GOTO 230
OF 241 REM NOWE OPERACJE
IP 242 IF LEN(MN$)>6 THEN IF MN$(
   LEN(MN$)-5)="WPRZOD" THEN FE=S
   A:GOTO 260
SD 243 IF MN$="ETYKIETA" THEN FR=
   SA-FB:POKE FB+1,FR-2:? "  OK":
   GOTO 230
VY 244 IF MN$="WSTAW" THEN ? :? "
   ADRES,LICZBA(DZIS.)";:INPUT AD
   DR,NUM:POKE ADDR,NUM:GOTO 230
FG 246 IF MN$="LIST" THEN GOSUB 9
   000:GOTO 230
VZ 248 IF MN$="HEX" THEN HX=1:?
   GOTO 230
QG 249 IF MN$="DZIES" THEN HX=0:?
   :GOTO 230
NX 250 IF MN$="KONIEC" THEN 8000
JE 260 L=LEN(MN$):L$=MN$(1,3)
MN 270 FOR I=1 TO 56:IF L$=M$(I*3
   -2,I*3) THEN 300
GI 280 NEXT I
QU 290 GOTO 850
TP 300 REM ROZKAZY JEDNOBAJTOWE
ZZ 301 TY=TY(I):OP=OP(I)
```

```
SC 305 IF FB=SA THEN TN=0:GOTO 20
   10
XX 310 IF TY=0 THEN GOTO 1000
VI 320 IF TY=3 THEN TY=1:IF L=3 T
   HEN OP=OP+8:GOTO 1000
IG 330 R$=MN$(5):IF HX=1 THEN GOS
   UB 6000
WH 340 LR$=R$(1,1):LL=LEN(R$):IF
   LR$="#" THEN 480
CY 350 IF LR$="(" THEN 520
EW 360 IF TY=8 THEN 600
EG 370 IF TY=3 THEN OP=OP+8:GOTO
   1000
SN 380 IF R$(LL)="X" OR R$(LL)="Y
   " THEN 630
ZH 390 IF L$(1,1)="J" THEN 820
VD 400 TN=VAL(R$):IF TN>255 THEN
   430
VK 410 IF TY=1 OR TY=3 OR TY=4 OR
    TY=5 THEN OP=OP+4

JX 420 GOTO 2000
HO 430 H=INT(TN/256):L=(TN-256*H)
   :IF TY=2 OR TY=7 THEN OP=OP+8:
   GOTO 470
NU 440 IF TY=1 OR TY=3 OR TY=4 OR
    TY=5 THEN OP=OP+12:GOTO 470
KZ 450 IF TY=6 OR TY=9 THEN 470
QQ 460 GOTO 850
KR 470 GOTO 3000
IT 480 TN=VAL(R$(2))
EU 490 IF TY=1 THEN OP=OP+8:GOTO
   2000
OZ 500 IF TY=4 OR TY=5 THEN GOTO
   2000
QH 510 GOTO 850
NG 520 IF R$(LL-1)=")Y" THEN 540
OA 530 IF R$(LL-1)="X)" THEN 570
QF 540 TN=VAL(R$(2,LL-1))
UY 550 IF TY=1 THEN OP=OP+16:GOTO
    2000
QR 560 GOTO 850
QL 570 TN=VAL(R$(2,LL-1))
ZV 580 IF TY=1 THEN GOTO 2000
QX 590 GOTO 850
GH 600 TN=VAL(R$):TN=TN-SA-2:IF T
   N<-128 OR TN>127 THEN ? "ZA DA
   LEKO";:GOTO 850
IS 610 IF TN<0 THEN TN=TN+256
JZ 620 GOTO 2000
NJ 630 IF R$(LL-1)=")Y" THEN 540
GV 640 IF R$(LL)="X" THEN 720
RE 650 REM *ZERO Y
ZW 660 TN=VAL(R$(1,LL-1)):IF TN>2
   55 THEN 680
FL 670 IF TY=2 OR TY=5 THEN 730
HT 675 IF TY=1 THEN 760
ET 680 GOSUB 770:IF TY=1 THEN OP=
   OP+24:GOTO 710
UQ 690 IF TY=5 THEN OP=OP+28:GOTO
    710
QH 700 GOTO 850
KI 710 GOTO 3000
SL 720 TN=VAL(R$(1,LL-1)):IF TN>2
   55 THEN GOSUB 770:GOTO 780
WL 730 IF TY=2 THEN OP=OP+16:GOTO
    760
PG 740 IF TY=1 OR TY=3 OR TY=5 TH
   EN OP=OP+20:GOTO 760
QR 750 GOTO 850
KI 760 GOTO 2000
NP 770 H=INT(TN/256):L=TN-256*H:R
   ETURN
QS 780 IF TY=2 THEN OP=OP+24:GOTO
    810
UR 790 IF TY=1 OR TY=3 OR TY=5 TH
   EN OP=OP+28:GOTO 810
QI 800 GOTO 850
KJ 810 GOTO 3000
GC 820 TN=VAL(R$)
WT 830 GOSUB 770
OO 840 GOTO 710
WW 850 ? CHR$(253);"BLAD":GOTO 23
   0
QQ 1000 REM ROZKAZY 1 BAJTOWE
PA 1010 POKE SA,OP:SA=SA+1:IF HX=
   1 THEN 1030
ZF 1020 ? OP:GOTO 230
UR 1030 DE=OP:GOSUB 4000:? H$:GOT
   O 230
RJ 2000 REM ROZKAZY 2 BAJTOWE
UO 2005 IF TN>256 THEN ? :? "Erro
   r--";TN;">256 ($100)":GOTO 230
DC 2010 POKE SA,OP:POKE SA+1,TN:S
   A=SA+2:IF HX=1 THEN 2030
GY 2020 ? OP;" ";TN:GOTO 230
DU 2030 DE=OP:GOSUB 4000:? H$;" "
   ;
VU 2040 DE=TN:GOSUB 4000:? H$:GOT
   O 230
SC 3000 REM ROZKAZY 3 BAJTOWE
EF 3010 POKE SA,OP:POKE SA+1,L:PO
   KE SA+2,H:SA=SA+3:IF HX=1 THEN
    3030
JP 3020 ? OP;" ";L;" ";H:GOTO 230
DV 3030 DE=OP:GOSUB 4000:? H$;" "
   ;
FY 3040 DE=L:GOSUB 4000:? H$;" ";
IT 3050 DE=H:GOSUB 4000:? H$:GOTO
    230
ZV 4000 REM DZIESIETNE NA HEX
UK 4010 H$="":A=INT(DE/256):IF A>
   0 THEN AH=INT(A/16):AL=A-AH*16
   :H$=HE$(AH+1,AH+1):H$(2)=HE$(A
   L+1,AL+1)
LS 4020 A=DE-A*256:AH=INT(A/16):A
   L=A-AH*16:H$(LEN(H$)+1)=HE$(AH
   +1,AH+1):H$(LEN(H$)+1)=HE$(AL+
   1,AL+1):SZ=1:RETURN
YJ 5000 REM HEX NA DZIESIETNE
LU 5010 D=0:O=3:FOR M=1 TO 4:W=AS
   C(H$(M))-48:IF W>9 THEN W=W-7
YQ 5030 D=D*16+W:NEXT M:DE=INT(D)
   :RETURN
CP 6000 REM PRZYJECIE SZESNASTKOW
   EGO KODU OPERACJI I TLUMACZENI
   E GO NA DZIESIETNY
YD 6010 IF R$(1,1)="#" THEN H$="0
   0":H$(3)=R$(2):GOSUB 5000:R$="
   #":R$(2)=STR$(DE):RETURN
FR 6020 LS=LEN(R$):AZ$=R$(1,1):ZA
   $=R$(LS):IF AZ$<>"(" THEN 6050
GZ 6030 IF ZA$="Y" THEN H$="00":H
   $(3)=R$(2,4):GOSUB 5000:R$="("
   :R$(2)=STR$(DE):R$(LEN(R$)+1)=
   ")Y":RETURN
BW 6040 IF ZA$=")" THEN H$="00":H
   $(3)=R$(2,4):GOSUB 5000:R$="("
   :R$(2)=STR$(DE):R$(LEN(R$)+1)=
   "X)":RETURN


UY 6050 IF ZA$="X" OR ZA$="Y" THE
   N 6070
MA 6060 H$="":IF LS<4 THEN H$=ZO$
   (1,4-LS)
QY 6065 H$(LEN(H$)+1)=R$:GOSUB 50
   00:R$=STR$(DE):RETURN
TV 6070 IF LS=5 THEN H$=R$(1,4):G
   OTO 6090
FP 6080 H$="00":H$(3)=R$(1,2)
IY 6090 GOSUB 5000:R$=STR$(DE):R$
   (LEN(R$)+1)=ZA$:RETURN
KD 8000 ? :? "POCZATEK  ";TA;:SZ=
   3:DE=TA:GOSUB 4000:? " ($";H$;
   ")"
LE 8010 ? "KONIEC     ";SA-1;:DE=
   SA-1:SZ=3:GOSUB 4000:? " ($";H
   $;")":END
UJ 9000 ? :FOR I=TA TO SA-1
IK 9010 IF HX=1 THEN DE=I:GOSUB 4
   000:? H$,:DE=PEEK(I):GOSUB 400
   0:? H$
QK 9020 IF HX=0 THEN ? I,PEEK(I)
FG 9030 NEXT I
AT 9040 RETURN
AW 20000 DATA ADC1097,AND1033,ASL
   3002,BCC8144,BCS8176,BEQ8240,B
   IT7036,BMI8048
VX 20010 DATA BNE8208,BPL8016,BRK
   0000,BVC80B0,BVS8112,CLC0024,C
   LD0216,CLI0088
FZ 20020 DATA CLV0184,CMP1193,CPX
   4224,CPY4192,DEC2198,DEX0202,D
   EY0136,EOR1065
VI 20030 DATA INC2230,INX0232,INY
   0200,JMP6076,JSR9032,LDA1161,L
   DX5162,LDY5160
TR 20040 DATA LSR3066,NOP0234,ORA
   1001,PHA0072,PHP0008,PLA0104,P
   LP0040,ROL3034
JU 20050 DATA ROR3098,RTI0064,RTS
   0096,SBC1225,SEC0056,SED0248,S
   EI0120,STA1129
WE 20060 DATA STX2134,STY2132,TAX
   0170,TAY0168,TSX0186,TXA0138,T
   XS0154,TYA0152
```



Konwerter języka maszynowego

Podprogram w języku maszynowym, otrzymany w wyniku działania poprzedniego programu i przechowywany w pamięci RAM nie nadaje się do włączenia go do programu w BASIC'u. Należy go przedstawić w postaci zrozumiałej przez BASIC i przechować w pamięci zewnętrznej. Funkcje te realizuje konwerter języka maszynowego, który automatycznie tłumaczy kod maszynowy (przechowywany w pamięci RAM) na pełny podprogram w języku BASIC, który może być łatwo dołączony do naszych programów. Otrzymany podprogram jest kompletny; wymaga tylko spełnienia następujących warunków:

— numery linii programu nie powinny być wyższe niż 31000,
— powinien być on wywołany możliwie jak najwcześniej w programie.

Prezentowany program jest dosyć elastyczny. Można wybrać kilka sposobów przechowywania procedury w języku maszynowym:

— w postaci ciągu (prawdopodobnie najbezpieczniejsza i najbardziej uniwersalna metoda, program będzie automatycznie generował ciąg znaków plus niezbędne instrukcje DIM).
— przechowanie w określonych miejscach pamięci. Adresy mogą być takie same jak dotychczas lub zmienione. Program będzie generował ciąg instrukcji DATA. Dostarczy także krótką procedurę, która pobiera te dane (READ) i wprowadza je (POKE) do pamięci.
— może być używana dowolna kombinacja ciągu i komórek pamięci. Program połączy je w podprogram.

Po zdefiniowaniu tablic i ciągów opisany program zajmuje mniej niż 5KB pamięci.

W celu konwersji języka maszynowego na postać akceptowalną przez BASIC należy wykonać następujące kroki:

1. Zakładamy, że program w języku

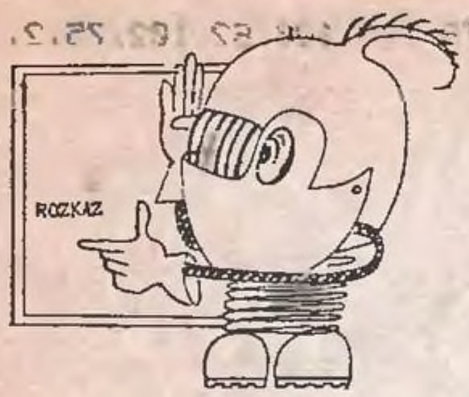
maszynowym umieszczony jest w obszarze pamięci.

2. Wprowadzamy, za pomocą **ENTER**, konwerter języka maszynowego, który wcześniej został przechowany za pomocą **LIST**.
3. Uruchamiamy program. Jesteśmy proszeni o podanie adresów początku i końca procedury maszynowej w RAM. Odpowiedzi należy udzielać w postaci dziesiętnej.
4. Musimy wybrać metodę przechowania procedury. Po wybraniu ciągu należy podać jego nazwę.
5. W wypadku przechowywania w określonym obszarze pamięci musimy podać, czy zmieniamy adresy tego obszaru.
6. Na końcu jesteśmy pytani, czy chcemy dokonać dodatkowych konwersji (na przykład innej procedury). Jeśli tak, to program przejdzie na początek. Jeśli nie, komputer zamyka wszystkie zbiory i kończy działanie.
7. Można teraz wykasować program konwerter (za pomocą NEW), wprowadzić nasz program w BASIC'u, dołączyć procedurę w języku maszynowym za pomocą ENTER „C:" (należy wykasować linię nr 1).
8. Wprowadzamy teraz instrukcję **GOSUB** z podaniem numeru pierwszej linii procedury.

Przechowywanie procedur maszynowych w postaci ciągu wymaga pewnego komentarza. Po pierwsze, aby działały one poprawnie, musimy je pisać w postaci przemieszczalnej. Oznacza to, że nie mogą zawierać żadnych instrukcji **JMP** lub **JSR** do określonych komórek pamięci wewnątrz programu. Ponieważ ciąg może być umieszczony w dowolnym miejscu pamięci, nieprzemieszczalny kod prawie na pewno spowoduje awarię. Po drugie, poważnym problemem jest występowanie znaków o kodach 34 lub 155. Ich obecność powoduje, że edytor ekranowy przedwcześnie obcina nasz ciąg i daje komunikat błędu. Dlatego program wykonuje następujące czynności po napotkaniu jednego z tych znaków:

1) Wstawia znak spacji i oznacza jego pozycję w ciągu.
2) Zapisuje instrukcje podprogramu tak, że wartości te są wstawiane do ciągu bez przejścia przez edytor ekranowy. W tym celu używa on funkcji CHR$.
3) Program może manipulować maksymalnie piętnastoma znakami o kodach 34 i 155. Sprawdza ich ilość i ostrzega, gdy jest zbyt duża.

```
FK 1 REM ****************
WC 2 REM *  KONWERTER   *
PB 3 REM *    JEZYKA     *
VV 4 REM * MASZYNOWEGO   *
FO 5 REM ****************
MX 10 CLR :GRAPHICS 0:POKE 752,1:
   POKE 756,206:? "   KONWERTER
   JEZYKA MASZYNOWEGO ":GOSUB 600
   :POKE 756,224
YC 20 DIM A$(1),D0$(3),QUOTE(14),
   RETRN(14)
KR 30 D0$="    ":TRAP 580:GOSUB 74
   0:V=0:LNO=32050:LN=31000:F=0
JL 35 OPEN #3,8,0,"C:":? #3;"1 DA
   TA ";:FOR I=0 TO 59:? #3;"0,";
   :NEXT I:? #3;"0";:? #3
VU 40 ? :? :? "WPROWADZ ADRES POC
   ZATKOWY KODU":POKE 752,0:GOSUB
    590:INPUT S:SO=S:SF=S
RE 50 ? "WPROWADZ ADRES KONCOWY K
   ODU":GOSUB 590:INPUT S:FO=S:FF
   =S:GOSUB 640
BK 60 ? "SPOSOB PRZECHOWANIA PROC
   EDURY?"
IZ 65 ? :? "   1.W TYCH SAMYCH AD
   RESACH?":? "   2.GDZIEWATEZ CIAG
   U?":? "   3.W NOWYCH ADRESACH?
   "
PE 80 GOSUB 590:? :? "PROSZE WPRO
   WADZIC LICZBE I <RETURN>!":INP
   UT N
NG 90 IF (N<>1 AND N<>2 AND N<>3)
    THEN ? CHR$(253);"ZLA ODPOWIE
   DZ! SPROBUJ JESZCZE RAZ!":GOSU
   B 600:GOTO 60
LU 100 IF N=3 THEN ? :? "NOWY ADR
   ES POCZATKOWY PROCEDURY?":GOSU
   B 590:INPUT S:SF=S
HX 110 IF N=3 THEN ? "NOWY ADRES
   KONCOWY PROCEDURY!":GOSUB 590:
   INPUT S:FF=S
SF 120 IF N=3 THEN IF FF-SF<>FO-S
   O THEN ? CHR$(253);" NIEPOPRAW
   NY ADRES KONCOWY! SPROBUJ JESZ
   CZE RAZ!":? :GOTO 110
RT 130 IF N=1 OR N=3 THEN F=1:V=V
   +1:GOTO 180
CK 140 L=FO-SO+1:GOSUB 680:GOSUB
   610
AI 150 ? "CZY CHCESZ OTRZYMAC WYD
   RUK ":? "CIAGU DANYCH W ASCII
   !":GOSUB 590:INPUT A$
EI 160 IF A$="T" THEN N=4:? "NACI
   SNIJ <RETURN> PO WLACZENIU DRU
   KARKI!":GOSUB 590:INPUT A$:OPE
   N #2,8,0,"P:"
OS 170 GOTO 260
WB 180 ? #3;LNO;" DATA ";SF;",";F
   F:LNO=LNO+10
ST 190 ? #3;LNO;" DATA ";

VY 200 FOR I=0 TO 19
OD 210 IF SO+I=FO+1 THEN POP :IF
   I THEN ? #3;",";-1:? #3:LNO=LN
   O+10:GOTO 490
ZP 215 IF SO+I=FO+1 THEN IF I=0 T
   HEN ? #3;-1:? #3:LNO=LNO+10:GO
   TO 490
TL 220 IF I THEN ? #3;",";
OS 230 ? #3;PEEK(SO+I);
CJ 240 NEXT I:? #3:LNO=LNO+10:SO=
   SO+20:GOTO 190
UL 260 IF N=4 THEN ? #2:? #2;"**D
   ANE DLA ";D0$;"**"
HX 270 LS=1:Z=0:W=0:? #3;LN;" DIM
    ";D0$;"(";L;"):";
II 280 IF N=4 THEN FOR I=0 TO L-1
   :? #2;PEEK(SO+I);:IF I<L-1 THE
   N ? #2;",";
CE 290 IF N=4 THEN NEXT I
WG 300 LR=L-80:IF LR<=0 THEN LF=L
   S+L-1
EL 310 IF LR>0 THEN LF=LS+80-1:L=
   LR
YI 320 ? #3;D0$;"(";LS;",";LF;")=
   ";:? #3;CHR$(34);:FOR I=LS TO
   LF
WG 330 IF PEEK(SO+I-1)=34 THEN ?
   #3;" ";:QUOTE(Z)=I:Z=Z+1:GOTO
   360
MP 340 IF PEEK(SO+I-1)=155 THEN ?
    #3;" ";:RETRN(W)=I:W=W+1:GOTO
    360
KL 350 ? #3;CHR$(PEEK(SO+I-1));
YQ 360 NEXT I:IF LR>0 THEN LS=LS+
   80:? #3;CHR$(34):? #3:LN=LN+10
   :? #3;LN;" ";:GOTO 300
BX 370 ? #3;CHR$(34);? #3:LN=LN+1
   0
DL 380 QT=0:RT=0:FOR X=0 TO 14:IF
    QUOTE(X) THEN QT=QT+1
FU 390 IF RETRN(X) THEN RT=RT+1
UK 400 NEXT X:IF QT=0 AND RT=0 TH
   EN 490
RF 410 ? #3;LN;" RESTORE ";LN+20:
   LN=LN+10
BR 420 IF QT THEN ? #3;LN;" FOR X
   =1 TO ";QT;":READ Z:";D0$;"(Z,
   Z)=CHR$(34):NEXT X":LN=LN+10
SJ 430 IF QT THEN ? #3;LN;" DATA
   ";:FOR Y=0 TO QT-1:? #3;QUOTE(
   Y);:IF Y AND Y<QT-1 THEN ? #3;
   ",";
DP 440 IF QT THEN NEXT Y:? #3:LN=
   LN+10
HQ 450 IF RT THEN ? #3;LN;" FOR X
   =1 TO ";RT;":READ Z:";D0$;"(Z,
   Z)=CHR$(155):NEXT X":LN=LN+10
PF 460 IF RT THEN ? #3;LN;" DATA
   ";:FOR Y=0 TO RT-1:? #3;RETRN(
   Y);:IF Y AND Y<QT-1 THEN ? #3;
   ",";

ED 470 IF RT THEN NEXT Y:? #3:LN=
   LN+10
MZ 490 GOSUB 740:? "GOTOWE":GOSUB
    590:INPUT A$
ZI 500 IF A$="N" THEN D0$="    ":C
   LOSE #2:GOTO 40
YE 510 IF F=0 THEN 570
GE 520 ? #3;"32000 W=0:V=";V;":RE
   STORE 32050"
RO 530 ? #3;"32010 READ X,Y:FOR I
   =X TO Y:READ Z:POKE I,Z:NEXT I
   "
ZS 540 ? #3;"32020 READ Z:IF Z<>-
   1 THEN ?";CHR$(34);"BLAD KODU!
    SPRAWDZ INSTRUKCJE DATA!";CHR
   $(34);":END"
CW 550 ? #3;"32030 W=W+1:IF W<V T
   HEN 32010"
FH 560 ? #3;"32040 RETURN"
MQ 570 CLOSE #2:CLOSE #3:END
DL 580 CLOSE #2:CLOSE #3:TRAP 400
   00:? CHR$(253);"BLAD ";PEEK(19
   5);" W LINII ";PEEK(186)+256*P
   EEK(187);"!":END
AD 590 FOR T=10 TO 6 STEP -1:FOR
   S=8 TO 0 STEP -1:SOUND 0,15-S,
   10,T:NEXT S:NEXT T:SOUND 0,0,0
   ,0:RETURN
JS 600 FOR T=1 TO 400:NEXT T:RETU
   RN
RH 610 ? :? "WEROWADZ DWUZNAKOWA
   NAZWE CIAGU":? "PLUS ZNAK $":G
   OSUB 590:INPUT D0$
RF 615 IF LEN(D0$)<3 THEN GOSUB 7
   50:GOTO 610
ZC 620 IF ASC(D0$(1,1))>90 OR ASC
   (D0$(1,1))<65 OR D0$(2,2)="$"
   OR D0$(3,3)<>"$" THEN GOSUB 75
   0:GOTO 610
ZI 630 RETURN
DS 640 IF SO<1792 THEN RETURN
XV 645 IF FO>(256*PEEK(106)-1000)
    THEN ? CHR$(253);"NIE MAM TYL
   E PAMIECI":POP :GOTO 40
QK 650 BATOP=PEEK(144)+256*PEEK(1
   45):IF BATOP>SO-100 THEN ? CHR
   $(253);"UWAGA! TEN PROGRAM MOZ
   E ZNISZCZYC ":GOTO 670
ZO 660 RETURN
GP 670 ? "TWOJ KOD! SPRAWDZ WYNIK
   I!":GOSUB 600:RETURN
LP 680 QT=0:RT=0:FOR I=0 TO L-1:I
   F PEEK(SO+I)=34 THEN QT=QT+1
KU 690 IF PEEK(SO+I)=155 THEN RT=
   RT+1
NM 700 NEXT I:IF RT<16 AND QT<16
   THEN RETURN
VZ 710 ? CHR$(253);"UWAGA! TWOJ K
   OD ZAWIERA":? "WIECEJ NIZ 15 W
   ARTOSCI ATASCII":? "DLA RETURN
   LUB NAWIASOW!"

JN 720 ? "DLATEGO NIE MOGE GO PRZ
   ETWORZYC!":? "PROSZE DOKONAC I
   NNEGO WYBORU!":GOSUB 600:GOSUB
    600
GN 730 POP :GOTO 60
VX 740 FOR I=0 TO 14:QUOTE(I)=0:R
   ETRN(I)=0:NEXT I:RETURN
JK 750 ? CHR$(253);"ZLA ODPOWIEDZ
   ! SPROBUJ JESZCZE RAZ!":RETURN
```

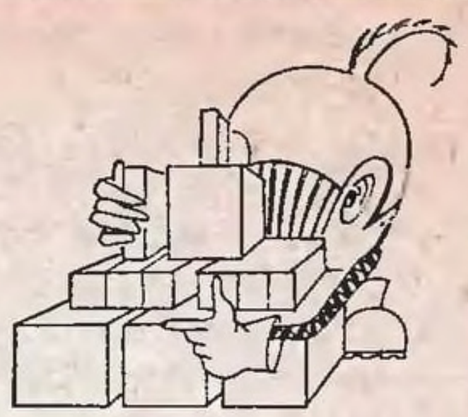


Disasembler

Trzecim programem tej gry jest disasembler realizujący funkcję odwrotną do asemblera: liczby przechowywane w komórkach pamięci przetwarza na postać rozkazową. Prezentowany program umożliwia obejrzenie kodu maszynowego dowolnego programu binarnego, w szczególnym przypadku gry. W aktualnej wersji automatycznie przechodzi do wprowadzenia programu z taśmy magnetycznej. Następnie sam odnajduje adres początku programu i rozpoczyna wyświetlanie rozkazów.

```
VC 98 REM ****************
PS 99 REM *                *
MK 100 REM * DISASSEMBLER *
AB 101 REM *                *
LA 102 REM ****************
WF 105 GRAPHICS 0:POSITION 11,0:?
    "[ DISASSEMBLER Z TM ]":? :? "P
   rosze czekac....."
QI 110 DIM OPCODE$(256*10),LN(255
   ),NB(255),T$(10),D$(5),B$(28)
ET 120 FOR I=0 TO 255
GJ 125 READ T$,NB
DT 130 LN(I)=LEN(T$)
RA 140 OPCODE$(I*10+1,I*10+LN(I))
   =T$
ZA 150 NB(I)=NB
GD 160 NEXT I
NO 170 GRAPHICS 0:POSITION 11,0:?
    "[ DISASSEMBLER Z TM ]"
TE 180 ? :?
UQ 185 GOSUB 800
HV [illegible] TRAP 190:? CHR$(156);CHR$(
   28);"Adres poczatkowy (Dziesie
   tnie)";:ADDR=A:? A:TRAP 40000
UN 195 ADRW=PEEK(A+2)+PEEK(A+3)*2
   56
TY 200 IF ADDR<0 OR ADDR>65535 TH
   EN 190
YD [illegible] OP=PEEK(ADDR):NB=NB(OP)
BR 220 T$=OPCODE$(OP*10+1,OP*10+L
   N(OP))
BV 230 ? ADRW;:POKE 65,10:? OP;:P
   OKE 85,15
BM 240 ON NB+2 GOTO 242,244,250,2
   60,270
TI 242 NB=2:T=PEEK(ADDR+1):IF T>1
   28 THEN T=T-256
EP 243 ? T;:POKE 85,25:? T$;"  ";
   ADRW+2+T:GOTO 300
CR 244 ? "[ BRAK KODU ]":NB=1:GOTO 20
   0
MZ 246 ? T$;" ";ADRW+2+T:GOTO 300
NJ 250 POKE 85,25:? T$:GOTO 300
ZZ 260 ? PEEK(ADDR+1);:POKE 85,25
   :D$=STR$(PEEK(ADDR+1)):GOSUB [illegible]
   GOTO 300
TO 270 ? PEEK(ADDR+1);:POKE 85,20
   :? PEEK(ADDR+2);:POKE 85,25
AU 280 D$=STR$((PEEK(ADDR+1)+256*
   PEEK(ADDR+2))):GOSUB [illegible]
XF 300 ADDR=ADDR+NB:IF ADDR<0 THE
   N ADDR=65536-T
BB 310 IF ADDR>65535 THEN ADDR=T
UM 312 ADRW=ADRW+NB
DB 320 IF PEEK(53279)=7 THEN 210
PJ 330 GOTO 190
```

```
QK [illegible] ? T$(1,4+(LN(OP)>4));D$;T$
   (4+2*(LN(OP)>5)):RETURN
BW 500 DATA BRK,1,ORA (X),2,?,0,?
   ,0,?,0,ORA  ,2,ASL  ,2,?,0,PHP
   ,1,ORA # ,2
CX 510 DATA ASL A,1,?,0,?,0,ORA
   ,3,ASL  ,3,?,0,BPL,-1,ORA ()Y,
   2,?,0,?,0
FH 520 DATA ?,0,ORA  X,2,ASL  X,2
   ,?,0,CLC,1,ORA  Y,3,?,0,?,0,?,
   0,ORA  X,3
WE 530 DATA ASL ,2,?,0,JSR  ,3,AN
   D (X),2,?,0,?,0,BIT ,2,AND ,2,
   ROL ,2,?,0
WH 540 DATA PLP,1,AND # ,2,ROL A,
   1,?,0,BIT ,3,AND ,3,ROL ,3,?,0
   ,BMI,-1,AND ()Y,2
UE 550 DATA ?,0,?,0,?,0,AND  X,2,
   ROL  X,2,?,0,SEC,1,AND  Y,3,CL
   I,1,?,0
PG 560 DATA ?,0,AND  X,3,ROL  X,3
   ,?,0,RTI,1,EOR (X),2,?,0,?,0,?
   ,0,EOR  ,2
OW 570 DATA LSR  ,2,?,0,PHA,1,EOR
    # ,2,LSR  ,3,?,0,JMP  ,3,EOR
   ,3,LSR  ,3,?,0
PU 580 DATA BVC,-1,EOR ()Y,2,?,0,
   ?,0,?,0,EOR  X,2,LSR  X,2,?,0,
   CLI,1,EOR  Y,2
XP 590 DATA ?,0,?,0,?,0,EOR  X,3,
   LSR  X,3,?,0,RTS,1,ADC (X),2,?
   ,0,?,0
KQ 600 DATA ?,0,ADC  ,2,ROR  ,2,?
   ,0,PLA,1,ADC # ,2,ROR A,1,?,0,
   JMP (),108,ADC  ,3
RY 610 DATA ROR  ,3,?,0,BVS,-1,AD
   C ()Y,2,?,0,?,0,?,0,ADC  X,2,R
   OR  X,2,?,0
KJ 620 DATA SEI,1,ADC  Y,3,?,0,?,
   0,?,0,ADC  X,3,ROR  X,3,?,0,?,
   0,STA (X),2
EL 630 DATA ?,0,?,0,STY  ,2,STA
   ,2,STX  ,2,?,0,DEY,1,?,0,TXA,1
   ,?,0
PZ 640 DATA STY  ,3,STA  ,3,STX
   ,3,?,0,BCC,-1,STA ()Y,2,?,0,?,
   0,STY  X,2,STA  X,2
IN 650 DATA STX  Y,2,?,0,TYA,1,ST
   A  Y,3,TXS,1,?,0,?,0,STA  X,3,
   ?,0,?,0
CI 660 DATA LDY # ,2,LDA (X),2,LD
   X # ,2,?,0,LDY  ,2,LDA  ,2,LDX
    ,2,?,0,TAY,1,LDA # ,2
GN 670 DATA TAX,1,?,0,LDY  ,3,LDA
   ,3,LDX  ,3,?,0,BCS,-1,LDA ()
   Y,2,?,0,?,0
TP 680 DATA LDY  X,2,LDA  X,2,LDX
    Y,2,?,0,CLV,1,LDA  Y,3,TSX,1
   ,?,0,LDY  X,3,LDA  X,3
AP 690 DATA LDX  Y,3,?,0,CPY # ,2
   ,CMP (X),2,?,0,?,0,CPY  ,2,CMP
    ,2,DEC  ,2,?,0
BO 700 DATA INY,1,CMP # ,2,DEX,1,
   ?,0,CPY  ,3,CMP  ,3,DEC  ,3,?,
   0,BNE,-1,CMP ()Y,2
NO 710 DATA ?,0,?,0,?,0,CMP  X,2,
   DEC  X,2,?,0,CLD,1,CMP  Y,3,?,
   0,?,0
LQ 720 DATA ?,0,CMP  X,3,DEC  X,3
   ,?,0,CPX # ,2,SBC (X),2,?,0,?,
   0,CPX  ,2,SBC  ,2
SI 730 DATA INC  ,2,?,0,INX,1,SBC
    # ,2,NOP,1,?,0,CPX  ,3,SBC  ,
   3,INC  ,3,?,0
NO 740 DATA BEQ,-1,SBC (Y),2,?,0,
   ?,0,?,0,SBC  X,2,INC  X,2,?,0,
   SED,1,SBC  Y,3
AU 750 DATA ?,0,?,0,?,0,SBC  X,3,
   INC  X,3,?,0
VW 800 FOR II=1 TO 28:READ B:B$(I
   I,II)=CHR$(B):NEXT II
TJ 810 ? "ILE BAJTOW WCZYTAC":INP
   UT ILOSC
SN 820 OPEN #2,4,128,"C:"
IV 830 A=PEEK(144)+PEEK(145)*256+
   256
OZ 840 U=USR(ADR(B$),32,7,A,ILOSC
   )
CW 845 DATA 104,104,104,170,104,1
   04,157,66,3,104,157,69,3,104,1
   57,68,3,104,157,73,3,104,157,7
   2,3,76,86,228
ED 850 POKE 54018,60:ADRW=PEEK(A+
   2)+PEEK(A+3)*256:RETURN
```



Budowa zamku

Ostatnim z prezentowanych programów jest przykład prostej gry edukacyjnej umożliwiającej poznanie kształtów figur geometrycznych lub wzorów na obliczanie pola powierzchni tych figur. Wariant gry może być wybrany bezpośrednio po jej uruchomieniu.

```
FK 1 REM ****************
FJ 2 REM *                  *
WZ 3 REM * Budowa zamku *
FO 5 REM ****************
OL 50 BRK=1:IF PEEK(53279)=5 THEN
    BRK=0
LN 60 GOSUB 3000:GOTO 90
SG 70 POKE 77,0:IF BRK THEN POKE
   16,64:POKE 53774,64
ZY 80 RETURN
TV 90 DIM TP(8),COL(8),ANS$(114),
   INV$(114),FRM$(114),INF$(114)
OS 100 ANS$(1,50)="  PROSTOKAT
   KWADRAT     ROWNOLEGLOBOK    K
   OLO    "
KA 110 ANS$(51,105)="     TRAPEZ
      POLKOLE       TROJKAT
    [ HELP ]    "
CQ 120 FRM$(1,50)="     W * L
      S^2          B * H       PI
   * R^2 "
VI 130 FRM$(51,105)="  1/2 *H*(B+
   T)  1/2 *PI*R^2 1/2 * B * H
    [ HELP ]    "
DV 140 INV$(1,50)="[ PROSTOKAT
   KWADRAT   ROWNOLEGLOBOK    K
   OLO    ]"
SR 150 INV$(51,105)="[     TRAPEZ
      POLKOLE       TROJKAT
    [ HELP ]    ]"
VS 160 INF$(1,50)="[     W * L
      S^2          B * H       PI
   * R^2    ]"
UU 170 INF$(51,105)="[  1/2 *H*(B+
   T)  1/2 *PI*R^2 1/2 * B * H
    [ HELP ]    ]"
RT 175 ANS$(10,10)=CHR$(1):ANS$(2
   8,28)=CHR$(15):ANS$(35,35)=CHR
   $(12):ANS$(46,46)=CHR$(12):ANS
   $(68,68)=CHR$(15)
TM 176 ANS$(82,82)=CHR$(15):ANS$(
   69,69)=CHR$(12):ANS$(85,85)=CH
   R$(1)
EN 177 INV$(10,10)=CHR$(129):INV$
   (28,28)=CHR$(143):INV$(35,35)=
   CHR$(140):INV$(46,46)=CHR$(140
   ):INV$(68,68)=CHR$(143)
ZV 178 INV$(69,69)=CHR$(140):INV$
   (82,82)=CHR$(143):INV$(85,85)=
   CHR$(129)
HP 180 COL(1)=0:COL(2)=13:COL(3)=
   26:COL(4)=0
EX 190 COL(5)=13:COL(6)=26:COL(7)
   =0:COL(8)=13
ZR 200 TRUE=1:FALSE=0:NONE=FALSE:
   GRAPHICS 0:POKE 756,CH/256
BZ 210 GOSUB 720:REM KRESLENIE
YM 220 REM KRESLENIE ZAMKU
SU 230 GRAPHICS 7:POKE 756,CH/256
   :POKE 708,56:POKE 709,186:POKE
    710,130:POKE 712,150:GOSUB 70
   :COLOR 2:RESTORE 970
```

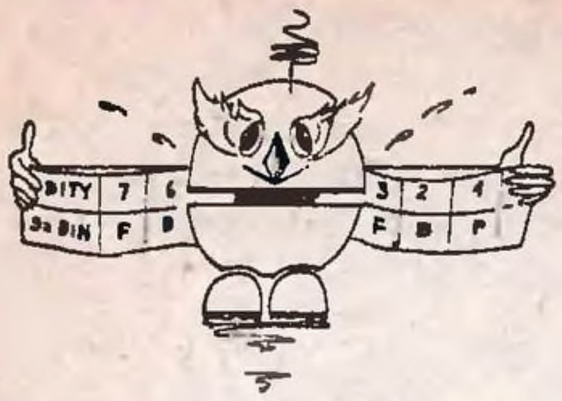
```
CQ 240 FOR I=1 TO 12:READ X1,Y1,X
   2,Y2:GOSUB 530:NEXT I:REM KRES
   LENIE PROSTOKATOW
DH 250 FOR I=1 TO 4:READ X1,Y1,X2
   ,Y2,X3,X4:GOSUB 600:NEXT I:REM
    TRAPEZ
RH 260 FOR I=1 TO 2:READ X1,Y1,X2
   ,Y2,Y3,Y4:GOSUB 550:NEXT I:REM
    ROWNOLEGLOBOK
GK 270 FOR I=1 TO 4:READ X1,Y1,X2
   ,Y2,X3,X4
MW 280 PLOT X1,Y2:DRAWTO X2,Y1:DR
   AWTO X4,Y1
ZD 290 DRAWTO X3,Y2:DRAWTO X1,Y2:
   NEXT I
OY 300 PLOT 41,10:DRAWTO 41,0:DRA
   WTO 49,3:DRAWTO 41,6
OC 310 FOR I=1 TO 2:READ X1,Y1,X2
   ,Y2:GOSUB 620:NEXT I:REM TROJK
   ATY
BG 320 FOR P=1 TO 7:READ X1,Y1,X2
   ,Y2,E,J:GOSUB 570:NEXT P:REM O
   KREGI
WS 330 ON WYBOR GOSUB 640,680
SD 340 LET TEXT=PEEK(660)+256*PEE
   K(661)
JM 350 TP(1)=TEXT:TP(2)=TEXT+13:T
   P(3)=TEXT+26:TP(4)=TEXT+40
JJ 360 TP(5)=TEXT+53:TP(6)=TEXT+6
   6:TP(7)=TEXT+80:TP(8)=TEXT+93
ZD 370 SP=5:OLDSP=SP:GOSUB 1460:R
   EM GLOWNY PROGRAM
UN 380 FOR DR=1 TO NUMSHAPES-1:CO
   RRECT=FALSE
KI 390 READ SHAPE:GOSUB 1700:COLO
   R 2
WN 400 ON SHAPE GOSUB 1750,1820,1
   840,1980,2080,2190,2210,2280
MQ 410 NEXT DR
BX 420 GOSUB 1700:REM KASOWANIE R
   YSUNKU
NF 430 READ SHAPE,COLR,X1,Y1,X2,Y
   2:CORRECT=FALSE:COLOR 2
PJ 440 GOSUB 620:GOSUB 1600:READ
   X1,Y1,X2,Y2
UC 450 COLOR COLR
QT 460 PLOT X2,Y2:DRAWTO X1,Y1
DW 470 Y1=Y1+1:IF Y1<>(Y2+4) THEN
    460
GI 480 ? CHR$(125);:IF NUMWRONG=0
    THEN ? " KA";CHR$(26);"DA TWO
   JA ODPOWIED";CHR$(6);" BY";CHR
   $(12);"A POPRAWNA!!!"
UO 490 IF NUMWRONG>0 THEN ? "OPU"
   ;CHR$(19);CHR$(3);"I";CHR$(12)
   ;"E";CHR$(19);" ";NUMWRONG:?
LD 500 ? "W celu dalszej gry naci
   ";CHR$(23);"nij PRZYCISK...";
RQ 510 IF STRIG(0) THEN 510
NO 520 GRAPHICS 0:POKE 756,CH/256
   :GOSUB 70:GOTO 210
FZ 530 PLOT X1,Y1:DRAWTO X2,Y1:DR
   AWTO X2,Y2:REM KRESLENIE PROST
   OKATA
TZ 540 DRAWTO X1,Y2:DRAWTO X1,Y1:
   RETURN
WG 550 PLOT X1,Y2:DRAWTO X2,Y1:DR
   AWTO X2,Y3:REM KRESLENIE ROWNO
   LEGLOBOKU
WM 560 DRAWTO X1,Y4:DRAWTO X1,Y2:
   RETURN
JB 570 PLOT X1,Y1:FOR I=0 TO 6 ST
   EP J:REM OKREGI & POLOKREGI
AZ 580 X=6*COS(I)+X2:Y=(-6)*SIN(I
   )+Y2
LK 590 DRAWTO X,Y:NEXT I:RETURN
LB 600 PLOT X1,Y2:DRAWTO X2,Y1:DR
   AWTO X3,Y1:REM KRESLENIE TRAPE
   ZU
WK 610 DRAWTO X4,Y2:DRAWTO X1,Y2:
   RETURN
VY 620 PLOT X1,Y1:DRAWTO X2,Y2:DR
   AWTO X1,Y2:REM TROJKAT
YV 630 DRAWTO X1,Y1:RETURN
OX 640 POKE 82,0:? "":POKE 752,1:
   REM KSZTALTY
PY 650 ? ANS$(1,39):? ANS$(40,78)
   :? ANS$(79,104)
WL 660 IF  NOT HELP THEN ? "U";CH
   R$(24);"yj manipulatora do wyb
   rania KSZTA";CHR$(12);"TU.";
ZQ 670 RETURN
UQ 680 POKE 82,0:? "":POKE 752,1:
   REM WYKONAJ WZORY
WJ 690 ? FRM$(1,39):? FRM$(40,78)
   :? FRM$(79,104)
GK 700 IF  NOT HELP THEN ? " U";C
   HR$(24);"yj manipulatora do wy
   brania WZORU";:POKE 752,2
ZF 710 RETURN
RL 720 NUMSHAPES=32:? CHR$(125):P
   OKE 710,146:POKE 752,1:REM INT
   RO EKRAN
ZY 730 POSITION 2,0:? "
   BUDOWA ZAMKU                ";
AS 740 POSITION 2,3:? "  Wzory
          Kszta";CHR$(251);"ty"
SW 750 POSITION 2,5:? "W * L
          = PROSTOK";CHR$(1);"T"
KP 760 POSITION 2,6:? "S^2
          = KWADRAT"
HG 770 POSITION 2,7:? "B * H
          = R";CHR$(15);"WNOLEG";C
   HR$(12);"OBOK"
RG 780 POSITION 2,8:? "PI * R^2
          = KO";CHR$(12);"O "
AN 790 POSITION 2,9:? "1/2 * H *
   (B+T) = TRAPEZ"
PK 800 POSITION 2,10:? "1/2 * PI
   * R^2  = P";CHR$(15);CHR$(12);
   "KOLE "
DS 810 POSITION 2,11:? "1/2 * H *
    B     = TR";CHR$(15);"JK";CHR
   $(1);"T"
ID 820 POSITION 0,13:? "Aby gra";
   CHR$(22);" naci";CHR$(23);"nij
    przycisk manipulatora.":GOSUB
    70:POKE 764,255
WF 830 IF STRIG(0) THEN 830
ZZ 840 S=55:GOSUB 1570:REM BEEP
EV 850 IF  NOT STRIG(0) THEN 850
EF 860 GRAPHICS 2+16:POKE 756,CH/
   256:WYBOR=1:REM START
NS 870 POSITION 0,4:PRINT #6;"CO
   CHCESZ SPR";CHR$(92);"BOWA";CH
   R$(37);":":GOSUB 70
RJ 890 IF  NOT STRIG(0) THEN S=25
   :GOSUB 1570:RETURN
VC 900 IF WYBOR=2 THEN 930
NE 910 POSITION 6,7:PRINT #6;"RYS
   UNKI";
IF 920 POSITION 6,9:PRINT #6;"WZO
   RY";
AR 925 PR=0
NO 930 IF STICK(0)=14 THEN WYBOR=
   1:S=75:GOSUB 1570:GOTO 890
XL 940 IF STICK(0)=13 THEN WYBOR=
   2:GOSUB 1540:S=55:GOSUB 1570
SL 950 GOTO 890
HS 960 REM DANE ZAMKU
QQ 970 DATA 5,37,77,76,39,60,46,7
   6,7,15,17,32,65,15,75,32
PB 980 DATA 10,6,14,10,70,10,80,1
   5,60,10,70,15,68,6,72,10
SM 990 DATA 2,10,12,15,12,10,22,1
   5,41,22,57,37,25,22,41,37
HM 1000 DATA 5,32,7,37,17,20,62,3
   2,65,37,75,77
JN 1010 DATA 46,70,52,76,71,77,5,
   70,11,76,33,39
OC 1020 DATA 10,20,15,17,30,27,67
   ,17,72,20,27,30
CN 1030 DATA 18,6,20,10,22,24,2,6
   ,0,10,7,4
YL 1040 DATA 60,6,57,10,64,62,75,
   6,78,10,80,82
VZ 1050 DATA 41,10,57,22,41,10,25
   ,22
EI 1060 DATA 28,47,22,47,6.6,0.1
VR 1070 DATA 39,30,33,30,6.6,0.1
ZP 1080 DATA 55,30,49,30,6.6,0.1
IZ 1090 DATA 68,47,62,47,6.6,0.1
CY 1100 DATA 16,65,22,65,3.15,0.1
HJ 1110 DATA 56,65,62,65,3.15,0.1
HI 1120 DATA 37,48,42,48,3.15,0.1
IH 1130 REM
EB 1140 DATA 1,3,100,67,120,75,5,
   38,77,75
IG 1150 DATA 4,2,112,65,106,65,6.
   6,0.1,44,96,6.6,0.1
UG 1160 DATA 5,2,100,65,105,75,12
   0,125,48,70,53,75,72,78
TC 1170 DATA 1,1,100,65,120,75,39
   ,60,46,75
WR 1180 DATA 6,1,100,65,106,65,3.
   15,0.1,124,130,3.15,0.1
BQ 1190 DATA 5,2,100,60,110,75,12
   0,130,5,70,10,75,32,38
VI 1200 DATA 4,2,112,65,106,65,6.
   6,0.1,124,96,6.6,0.1
CY 1210 DATA 6,1,105,65,111,65,3.
   15,0.1,44,130,3.15,0.1
CK 1220 DATA 1,1,100,62,110,75,7,
   15,17,32
NH 1230 DATA 3,3,100,65,105,75,55
   ,65,10,20,15,17,30,27
HW 1240 DATA 2,1,100,65,105,70,10
   ,6,15,9
NH 1250 DATA 1,2,100,62,102,75,2,
   10,12,15
TC 1260 DATA 3,1,100,65,105,75,55
   ,65,1,6,0,9,6,7
EP 1270 DATA 5,2,100,65,110,75,12
   0,130,4,32,7,37,18,20
OE 1280 DATA 1,2,100,65,122,75,12
   ,10,22,15
HU 1290 DATA 3,1,100,65,110,75,60
   ,70,18,6,20,9,26,23
IM 1300 DATA 2,2,105,65,115,75,25
   ,22,41,37
PU 1310 DATA 7,1,100,65,120,75,40
   ,10,25,22
IC 1320 DATA 2,2,100,65,110,75,41
   ,22,57,37
AJ 1330 DATA 4,3,112,65,106,65,6.
   6,0.1,66,60,6.6,0.1
KK 1340 DATA 1,1,100,60,105,75,65
   ,16,75,32
NY 1350 DATA 1,2,100,62,102,72,60
   ,10,70,15
UU 1360 DATA 3,1,100,65,110,75,60
   ,70,59,6,56,9,63,68
JL 1370 DATA 5,2,110,75,100,65,13
   0,120,62,32,64,37,75,79
WJ 1380 DATA 1,2,100,67,122,72,70
   ,10,80,15
LV 1390 DATA 7,1,100,62,110,75,41
   ,10,41,22
BI 1400 DATA 3,3,100,65,110,75,60
   ,70,68,17,73,20,28,32
CJ 1410 DATA 2,1,100,70,105,75,68
   ,6,72,9
GN 1420 DATA 3,1,100,65,105,75,55
   ,65,76,6,78,9,84,80
YZ 1430 DATA 4,3,112,65,106,65,6.
   6,0.1,100,60,6.6,0.1
JH 1440 DATA 6,1,100,65,106,65,3.
   15,0.1,84,96,3.15,0.1

FW 1450 DATA 7,1,102,65,115,75,42
   ,0,50,3
RO 1460 ROW=INT((SP-1)/3):OLDROW=
   INT((OLDSP-1)/3):REM NEGATYW T
   EKSTU
SZ 1470 POKE 656,OLDROW:POKE 657,
   COL(OLDSP):POKE 658,0
ZS 1480 IF WYBOR=1 THEN ? ANS$((O
   LDSP-1)*13+1,(OLDSP-1)*13+13);
CW 1490 IF WYBOR=2 THEN ? FRM$((O
   LDSP-1)*13+1,(OLDSP-1)*13+13);
CN 1500 POKE 656,ROW:POKE 657,COL
   (SP):POKE 658,0
FC 1510 IF WYBOR=1 THEN ? INV$((S
   P-1)*13+1,(SP-1)*13+13);
PU 1520 IF WYBOR=2 THEN ? INF$((S
   P-1)*13+1,(SP-1)*13+13);
RS 1530 RETURN
EB 1540 POSITION 6,7:PRINT #6;"RY
   SUNKI":REM NEGATYW WYBOROW
YC 1550 POSITION 6,9:PRINT #6;"WZ
   ORY"
BB 1560 RETURN
OG 1570 SOUND 0,5,10,10:REM BEEP
ON 1580 FOR D=1 TO 20:NEXT D
YL 1590 SOUND 0,0,0,0:RETURN
JN 1600 GOSUB 70:TR=STRIG(0):ST=S
   TICK(0):OLDSP=SP:IF ST=15 AND
   TR=1 THEN 1600:REM SPRAWDZENIE
    PRZYCISKU
UQ 1610 POKE 53279,7:IF TR=0 THEN
    S=35:GOSUB 1570:ANS=SP:GOSUB
   2320
EM 1620 IF ANS=8 THEN GOSUB 2280
PP 1630 IF CORRECT THEN RETURN
JS 1640 IF ST=7 THEN SP=SP+1:IF (
   SP-1)/3=INT((SP-1)/3) THEN SP=
   SP-3
VG 1650 IF SP=9 THEN SP=7
KZ 1660 IF ST=11 THEN SP=SP-1:IF
   SP/3=INT(SP/3) THEN SP=SP+3:IF
    SP=9 THEN SP=8
DL 1670 IF ST=13 THEN SP=SP+3:IF
   SP>8 THEN SP=SP-9:IF SP=0 THEN
    SP=3
VP 1680 IF ST=14 THEN SP=SP-3:IF
   SP<1 THEN SP=9+SP:IF SP=9 THEN
    SP=6
ZP 1690 GOSUB 1460:GOTO 1600
RI 1700 COLOR 0:X1=85:X2=150:REM
   KASOWANIE RYSUNKU
SU 1710 FOR Y=55 TO 80:SOUND 0,25
   5-(2*Y),10,10
YD 1720 PLOT X1,Y:DRAWTO X2,Y
OW 1730 NEXT Y:SOUND 0,0,0,0
AZ 1740 RETURN
SL 1750 READ COLR,X1,Y1,X2,Y2:REM
    PROSTOKAT
```

```
JY 1760 GOSUB 530:GOSUB 1600
WA 1770 READ X1,Y1,X2,Y2:COLOR CO
   LR
QW 1780 FOR Y=Y1 TO Y2
YY 1790 PLOT X1,Y:DRAWTO X2,Y
LZ 1800 NEXT Y
RS 1810 RETURN
EX 1820 GOSUB 1750:REM KWADRAT
RY 1830 RETURN
MV 1840 READ COLR,X1,Y1,X2,Y2,Y3,
   Y4:REM ROWNOLEGLOBOK
SC 1850 PA=PA+1
LR 1860 GOSUB 550:GOSUB 1600
WU 1870 IF PA=1 OR PA=5 THEN 1930
BZ 1880 READ X1,Y1,X2,Y2,X3,X4:CO
   LOR COLR
IK 1890 PLOT X2,Y1:DRAWTO X1,Y2
DU 1900 X1=X1+1:X2=X2+1
YM 1910 IF X2<>X3 THEN 1890
AX 1920 RETURN
DJ 1930 READ X1,Y1,X2,Y2,Y3,Y4:CO
   LOR COLR
HX 1940 PLOT X1,Y2:DRAWTO X2,Y1
FN 1950 Y2=Y2+1:Y1=Y1+1
WW 1960 IF Y2<>Y4 THEN 1940
BM 1970 RETURN
TN 1980 READ COLR,X1,Y1,X2,Y2,E,J
   :REM KOLO
ML 1990 GOSUB 570:GOSUB 1600
GD 2000 READ XC,YC,E,J:COLOR COLR
RQ 2010 XC=XC/2:YC=YC/2
PG 2020 FOR R=0 TO E STEP J
SQ 2030 X=E*COS(R)+XC
NY 2040 Y=(-6)*SIN(R)+YC
XN 2050 PLOT XC,YC:DRAWTO X,Y
JD 2060 NEXT R
AV 2070 RETURN
YH 2080 READ COLR,X1,Y1,X2,Y2,X3,
   X4:REM TRAPEZ
YB 2090 GOSUB 600
ZM 2100 GOSUB 1600
YZ 2110 READ X1,Y1,X2,Y2,X3,X4
EF 2120 COLOR COLR
HF 2130 PLOT X1,Y2:DRAWTO X2,Y1
DR 2140 X1=X1+1:X2=X2+1
NY 2150 IF X2<>X3 THEN 2130
IA 2160 PLOT X3,Y1:DRAWTO X1,Y2
TT 2170 X1=X1+1:IF X1<>X4 THEN 21
   60
BR 2180 RETURN
NV 2190 GOSUB 1980:REM POLKOLE
AE 2200 RETURN
YM 2210 READ COLR,X1,Y1,X2,Y2:REM
    TROJKAT
YK 2220 GOSUB 620
ZX 2230 GOSUB 1600
BF 2240 READ X1,Y1,X2,Y2:X3=X2:CO
   LOR COLR

NT 2250 PLOT X1,Y1 DRAWTO X2,Y2:X
   2=X2+1
RF 2260 IF X2<>X3+17 THEN 2250
AZ 2270 RETURN
MP 2280 HELP=TRUE:? CHR$(125):ON
   WYBOR GOSUB 690,650
FB 2290 FOR D=1 TO 500:NEXT D:HEL
   P=FALSE:ANS=0
XD 2300 ? CHR$(125):ON WYBOR GOSU
   B 650,690
CU 2310 ANS=0:RETURN
GS 2320 IF ANS=SHAPE THEN CORRECT
   =TRUE
EW 2330 IF ANS=8 THEN RETURN
DX 2340 IF ANS<>SHAPE THEN NUMWRO
   NG=NUMWRONG+1:GOSUB 2360
AV 2350 RETURN
YO 2360 S=150:FOR BUZZ=1 TO 3:GOS
   UB 1570
QU 2370 NEXT BUZZ:RETURN
JC 3000 DIM PGM$(32):CH=(PEEK(106
   )-8)*256:POKE 106,PEEK(106)-16
ZV 3010 RESTORE 3035:FOR I=1 TO 3
   2:READ X:PGM$(I,I)=CHR$(X):NEX
   T I:X=USR(ADR(PGM$),224*256,CH
   )
QO 3020 READ X:IF X=-1 THEN POKE
   756,CH/256:RETURN
ED 3030 FOR I=0 TO 7:READ Y:POKE
   CH+X*8+I,Y:NEXT I:GOTO 3020
AY 3035 DATA 104,104,133,213,104,
   133,212,104,133,215,104,133,21
   4,162,4,160,0,177,212,145,214,
   200,208,249,230
GF 3036 DATA 213,230,215,202,208,
   240,96
KN 3040 DATA 65,0,24,60,102,102,1
   26,102,12
OR 3050 DATA 67,12,60,102,96,96,1
   02,60,0
LN 3080 DATA 70,12,24,126,12,24,4
   8,126,0
LP 3100 DATA 76,0,96,120,112,224,
   96,126,0
EB 3130 DATA 79,12,60,102,102,102
   ,102,60,0
GF 3160 DATA 83,12,60,96,60,6,6,6
   0,0
ZL 3170 DATA 86,12,24,0,60,96,96,
   60,0
JX 3180 DATA 87,12,24,62,96,60,6,
   124,0
DP 3190 DATA 88,0,24,0,126,12,48,
   126,0
SI 3200 DATA 90,24,0,126,12,24,48
   ,126,0
EZ 3210 DATA 123,0,56,24,28,56,24
   ,60,0
ZL 3220 DATA 60,12,60,102,102,102
   ,102,60,0

TX 3230 DATA 5,12,60,102,96,96,10
   2,60,0
EW 3240 DATA -1
```

ATARI

```
10 REM *** SPEKTROGRAF MASOWY ***
20 OPEN #1,4,0,"K:":GRAPHICS 2:POKE 710,0:POKE 752,1:POKE 764,255
30 POSITION 0,5:? #6;" SPEKTROGRAF MASOWY":SETCOLOR 0,1,6
35 ? CHR$(127);"PROSZE CHWILE POCZEKAC":GOSUB 460
40 ? CHR$(28);CHR$(127);CHR$(253);"NACISNIJ DOWOLNY KLAWISZ":GET #1,S
50 REM ____________________
60 REM SCHEMAT SPEKTROGRAFU
70 REM ____________________
80 GRAPHICS 7:COLOR 2:DL=PEEK(560)+256*PEEK(561)
90 POKE DL+85,13:POKE 660,54:POKE 661,157:POKE 710,0:GOSUB 330
100 PLOT 110,30:DRAWTO 130,30:DRAWTO 130,1:DRAWTO 110,1
110 PLOT 100,1:DRAWTO 80,1:DRAWTO 80,30:DRAWTO 100,30:DRAWTO 30,30
120 FOR T=-1.55 TO 1.6 STEP 0.05:DRAWTO SIN(T)*50+80,COS(T)*50+30:NEXT T
130 PLOT 140,15:DRAWTO 120,15:DRAWTO 120,3:DRAWTO 120,27
140 PLOT 70,15:DRAWTO 90,15:DRAWTO 90,3:DRAWTO 90,27
150 PLOT 135,13:DRAWTO 140,13
160 PLOT 70,12:DRAWTO 76,12:PLOT 73,10:DRAWTO 73,14
170 FOR T=30 TO 80:PLOT T,20:DRAWTO T,30:NEXT T:COLOR 0:POKE 752,1
180 REM ____________
190 REM RUCH CZASTEK
200 REM ____________
210 COLOR 0:PLOT 34,20:DRAWTO 34,30:PLOT 45,20:DRAWTO 45,30
220 PLOT 73,20:DRAWTO 73,30:PLOT 57,20:DRAWTO 57,30:A=1
230 ? CHR$(127);"  W-WOLNO S-SZYBKO R-RESTART":?
240 U=USR(15000):FOR T=1 TO A:NEXT T:B=PEEK(764)
250 IF B=46 THEN A=A+20:POKE 764,255
260 IF B=62 THEN A=A-20:POKE 764,255
270 IF B=40 THEN GOTO 290
280 GOTO 240
290 U=USR(15256):RUN
300 REM _________________
310 REM GRAFIKA PLAJEROW
320 REM _________________
330 POKE 623,8:POKE 559,62:POKE 54279,130:POKE 53277,3
340 POKE 704,8:POKE 705,10:POKE 706,12:POKE 707,15:U=USR(1545)
350 RESTORE 390:FOR T=33794 TO 33806:READ A:POKE T,A:NEXT T
360 RESTORE 400:FOR T=34050 TO 34062:READ A:POKE T,A:NEXT T
370 RESTORE 410:FOR T=34306 TO 34318:READ A:POKE T,A:NEXT T
380 RESTORE 420:FOR T=34562 TO 34574:READ A:POKE T,A:NEXT T:RETURN
390 DATA 0,0,0,0,24,60,60,24,0,0,0,0
400 DATA 0,0,0,8,28,28,8,0,0,0,0,0
410 DATA 0,0,0,0,8,28,8,0,0,0,0,0
420 DATA 0,0,0,0,0,1,1,0,0,0,0,0
430 REM _________________
440 REM PROGRAM MASZYNOWY
450 REM _________________
460 RESTORE 490:FOR T=1545 TO 1639:READ A:POKE T,A:NEXT T
470 RESTORE 580:FOR T=15000 TO 15508:READ A:POKE T,A:NEXT T
480 RESTORE 860:FOR T=1528 TO 1539:READ A:POKE T,A:NEXT T:RETURN
490 DATA 104,160,0,169,0,133,26,169,127,133,27,169,0,145,26,165,26
500 DATA 105,1,133,26,144,244,24,230,27,165,27,201,143,208,235,96,0
520 DATA 170,160,1,177,26,136,145,26,200,200,192,14,208,245
530 DATA 198,26,202,224,0,208,236,234,234,234,234,234,234,24,96,0,0
550 DATA 170,160,12,177,26,200,145,26,136,136,192,0,208,245
560 DATA 230,26,202,224,0,208,236,234,234,234,234,234,234,24,96,0
580 DATA 104,172,250,5,140,0,208,192,76,208,5,160,151,140,250,5
590 DATA 206,250,5,173,248,5,133,26,173,249,5,133,27,185,96,59
600 DATA 201,0,240,26,74,176,5,32,74,6,144,13,24,201,80,208,5
610 DATA 160,0,140,0,208,32,43,6,165,26,141,248,5,172,253,5
620 DATA 140,1,208,192,86,208,5,160,151,140,253,5,206,253,5,173,251,5
630 DATA 133,26,173,252,5,133,27,185,160,59,201,0,240,26,74,176,5
640 DATA 32,74,6,144,13,24,201,80,208,5,160,0,140,1,208,32,43,6
650 DATA 165,26,141,251,5,172,0,6,140,2,208,192,96,208,5,160,151
660 DATA 140,0,6,206,0,6,173,254,5,133,26,173,255,5,133,27,185,214,59
670 DATA 201,0,240,26,74,176,5,32,74,6,144,13,24,201,80,208,5,160,0
680 DATA 140,2,208,32,43,6,165,26,141,254,5,172,3,6,140,3,208,192,111
690 DATA 208,5,160,151,140,3,6,206,3,6,173,1,6,133,26,173,2,6,133,27
700 DATA 185,254,59,201,0,240,26,74,176,5,32,74,6,144,13,24,201,80,208,5
710 DATA 160,0,140,3,208,32,43,6,165,26,141,1,6,173,252,2,201,255
720 DATA 96,0,0,0,0,0,104,169,0,141,0,208,141,1,208,141,2
730 DATA 208,141,3,208,141,111,2,96,0,0,0
750 DATA 161,17,17,17,7,7,7,7,7,7,5,5,5,5,5,5,3,3,3,3,3,3,3,3,3,3,3,3
760 DATA 3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4
770 DATA 4,4,4,6,6,6,6,6,6,16,16,16,160,0
780 DATA 161,17,17,11,11,7,7,5,5,5,5,5,3,3,3,3,3,3,3,3,3,3,3,3,0,0,0,0,0
790 DATA 0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,6,6,10,10
800 DATA 16,16,160,0
810 DATA 161,15,15,5,5,5,5,5,5,3,3,3,3,3,3,3,3,3,3,3,0,0,0,0,0,0,0
820 DATA 0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,14,14,160,0
830 DATA 161,17,13,5,5,3,3,3,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
840 DATA 2,2,2,2,2,2,4,4,12,16,160
860 DATA 1,132,150,1,133,150,1,134,150,1,135,150
```

JAN GOLLA



SPECTRUM

DYNAMICZNE BAJTY

Wprowadzenie dowolnej procedury w kodzie maszynowym pociąga za sobą konieczność deklarowania obszaru pamięci (np. instrukcją CLEAR), a natomiast przesyłanie pewnych wartości zmusza do POKE-wania. W numerze 5/1986 „IKS-a" pojawiła się propozycja przesyłania parametrów za pomocą funkcji użytkownika (FN).

Obecna propozycja dotyczy wprowadzania kodu maszynowego do... tablicy tekstowej.

FN przy wywołaniu znajduję w pamięci adresy przesyłanych tablic tekstowych, co w łatwy sposób pozwala wywołać USR. Dużym udogodnieniem jest możliwość przesyłania tylko fragmentu tablicy tekstowej zawierającej odpowiednią procedurę. Pozwala to na utworzenie całych zestawów (np. graficznych, edycyjnych, matematycznych) trzymanych w jednej tablicy tekstowej.

Oto niektóre wady i zalety tej metody:

Wady:

1) konieczność stosowania DEF FN — należy pamiętać o zgodności parametrów i ich liczby między DEF FN i FN, co ewentualnie pociąga za sobą definiowanie kilku DEF FN. 2) konieczność rezerwowania określonej tablicy tekstowej na kod maszynowy (przy znacznej liczbie tablic, jakie można zadeklarować, nie jest to zbyt trudne. 3) nie wolno używać RUN i CLEAR — powodują one kasowanie obszaru zmiennych, czyli niszczą również procedury (program można z równym powodzeniem uruchomić rozkazami GO TO i GO SUB). 4) procedury muszą być całkowicie relokowalne i wymagają stosowania skoków względnych (skoki te mają jednak ograniczony zasięg).

Zalety:

1) umieszczenie procedury w tablicy tekstowej pozwala zachować maksymalny obszar pamięci dla Basic'a. Nie trzeba się też martwić o miejsce (tablica tekstowa jest automatycznie umieszczona w obszarze zmiennych). 2) eleganckie i czytelne wprowadzenie parametrów przy użyciu FN (interpreter dodatkowo sprawdza, czy przesyłane dane istnieją i czy struktura danych jest taka sama, jak w DEF FN — w razie stwierdzenia błędu, Spectrum przerywa działanie programu i wyświetla odpowiedni komunikat). 3) zestawianie i porządkowanie procedur odbywa się tak jak składanie i kasowanie ciągów tekstowych (trzeba to jednak wykonywać bardzo ostrożnie, aby nie skasować np. części procedury — uruchomienie jej spowoduje najprawdopodobniej reset systemu). 4) zapis, odczyt, weryfikację procedur przeprowadzamy identycznie, jak dla innych tablic tekstowych. 5) kod maszynowy możemy wprowadzić do dowolnej tablicy tekstowej. 6) DEF FN można umieścić w każdym miejscu programu (ze względu na szybkość odszukania przez interpreter DEF FN powinno być raczej na początku programu). 7) procedury są nieliczne i mogą być łączone w dowolne zestawy. 8) procedury są całkowicie zabezpieczone przed błędami — każdy błąd powoduje awaryjne przerwanie programu i wyświetlanie odpowiedniego komunikatu.

Opisana wyżej metoda i jej własności nasunęły mi pewien pomysł: INDYWIDUALNY BANK PROCEDUR. Osoby umiejące programować w kodzie maszynowym mogłyby tworzyć różnorodne procedury spełniające wymienione wymagania. Użytkownik mógłby wykorzystywać je w dowolnych zestawach, które sam by złożył. Wymiana jednej procedury na inną (np. lepszą i szybszą) polegałaby na wymianie odpowiedniego fragmentu tablicy tekstowej. Proponuję również nadsyłać propozycje procedur, byłoby to źródło pomysłów dla twórców oprogramowania.

Aby przybliżyć omawiany problem zachęcam do przeanalizowania procedury, która spełnia wszystkie wymagane kryteria (jest jednocześnie udoskonaleniem programu zamieszczonego w nr. 5/1986 „IKS-a"). Dotyczy zmiany atrybutów koloru w zadanym obszarze ekranu. Wywołuje się ją przez:

FN f(a$,x,y,dx,dy,,w,p,s)

gdzie: a$ — tablica tekstowa zawierająca procedurę maszynową
x — numer kolumny, jak w AT ($0 \leq x \leq 31$)
y — numer wiersza, jak w AT ($0 \leq y \leq 21$)
dx — długość obszaru ($1 \leq dx \leq 32$)
dy — wysokość obszaru ($1 \leq dy \leq 22$)

Uwaga: ponieważ obszar nie może wykroczyć poza ekran, powinny być spełnione dodatkowe warunki:

$1 \leq x + dx \leq 32$

$1 \leq y + dy \leq 22$

w — atrybut wzorcowy (w = 128 * flash + 64 * bright + 8 * paper + ink) jest on porównywany z atrybutami zadanego obszaru i ich częściowa lub całkowita zgodność (o czym decyduje zmienna „s") może spowodować określane zmiany w atrybucie ekranu.

p — atrybut podstawiany (struktura taka sama jak „w") jest on częściowo lub całkowicie wstawiany w miejsce badanego atrybutu ekranu (o czym decyduje również zmienna „s") wtedy, gdy zostaną spełnione warunki dotyczące zmiennej „w".

s — zmienna sterująca ($0 \leq s \leq 255$)

s = BIN	BITY 7	6	5	4	3	2	1	0
	F	B	P	I	F	B	P	I
	DOTYCZY ZMIENNEJ „W"				DOTYCZY ZMIENNEJ „P"			

gdzie: F — flash, B — bright, P — paper, I — ink } F,B,P,I = {0,1}

dla zmiennej „w": jest kontrolowana zgodność tylko tego fragmentu atrybutu ekranu ze zmienną „w", dla którego odpowiednio F,B,P,I przyjmują wartość 1.

dla zmiennej „p": jest wymieniany tylko ten fragment atrybutu ekranu na odpowiadający mu fragment zmiennej „p", dla którego F,B,P,I są równe 1.

Funkcje dodatkowe:

s = BIN 00000000 — procedura nie działa

s = BIN 00001111 — wymiana bezwzględna

s = BIN 11110000 — FN przyjmuje adres pierwszego atrybutu spełniającego zadane warunki (jeśli taki atrybut istnieje).

Przykład: chcemy na całym obszarze ekranu wymienić ink na czerwony w każdym atrybucie, który ma zie-

lony paper, włączony flash i wyłączony bright:
FN f(a$, 0, 0, 22, 32, BIN 10100XXX, BIN XXXXX010, BIN 11100001), gdzie X może przyjąć wartość „0" albo „1" (przecież zmienna „s" nie narzuca warunku na ink w porównaniu ze zmienną „w", a w czasie wymiany należy wymienić tylko ink).

Ważne wyjaśnienie: kod maszynowy z równym powodzeniem możemy trzymać w zmiennej tekstowej. Można ją w prosty sposób sumować i kasować. Nie można jej niestety zapisywać na taśmie i dlatego przed zapisem tekst ze zmiennej tekstowej powinien być zapisany do tablicy tekstowej.

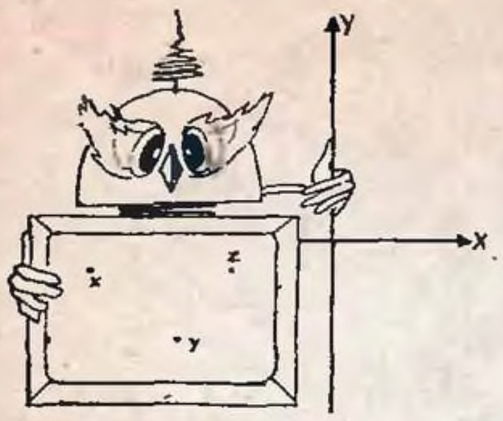
```
  1 REM
**********************************
*                                *
*      Krzysztof Poźniak         *
*                                *
*    Klub Mikrokomputerowy       *
*          HOBBYTE    © 1987     *
*                                *
**********************************

  4 RESTORE : DIM a$(233): LET
l=1: LET d=100
  5 LET s1=0: LET s2=0: LET z=1
: FOR k=1 TO 20: READ x: LET s1=
s1+x: LET s2=s2+z*x: LET a$(l)=C
HR$ x: IF l=233 THEN GO TO 7
  6 LET l=l+1: LET z=-z: NEXT k
  7 READ s01: READ s02: IF s1<>
s01 OR s2<>s02 THEN PRINT FLASH
1;"blad w linii ";d: STOP
  8 LET d=d+10: IF l<233 THEN G
O TO 5
  9 CLS
  10 FOR k=1 TO 21: PRINT TAB 10
;"tekst": NEXT k
  20 DEF FN f(a$,a,b,c,d,e,f,g)=
USR (PEEK (PEEK 23563+256*PEEK 2
3564+4)+256*PEEK (PEEK 23563+256
*PEEK 23564+5))
  30 LET s=2: FOR x=19 TO 1 STEP
 -3
  31 IF FN f(a$,s+4,s,x+INT (2*R
ND),x+INT (2*RND),8*INT (7*RND),
INT (7*RND)*8,INT (256*RND)) THE
N
  33 LET s=s+1: NEXT x
  40 FOR p=6 TO 1 STEP -1: IF FN
 f(a$,3+INT (3*RND),3+INT (3*RND
),15,15,P*7,INT (7*RND)*7,INT (2
55*RND)) THEN
  45 NEXT p
  50 GO TO 30
 100 DATA 221,42,11,92,17,8,0,6,
7,205,126,11,221,182,12,221,182,
14,221,182,1997,39
 110 DATA 15,32,45,221,25,15,238
,221,42,11,92,221,182,29,40,32,1
75,221,182,37,2077,-5
 120 DATA 40,26,221,126,13,221,1
34,29,56,18,254,33,48,14,221,126
,21,221,134,37,1993,291
 130 DATA 56,6,254,23,48,2,24,5,
46,25,195,65,0,175,221,182,61,20
0,175,221,2004,156
 140 DATA 102,61,203,124,40,2,24
6,128,203,116,40,2,246,64,203,10
8,40,2,246,56,2232,506
 150 DATA 203,100,40,2,246,7,87,
175,203,92,40,2,246,126,203,84,4
0,2,246,64,2210,898
 160 DATA 203,76,40,2,246,56,203
,68,40,2,246,7,95,221,166,53,111
,123,47,95,2100,694
 170 DATA 122,221,166,45,103,253
,229,213,17,32,0,253,33,0,88,221
,70,21,175,176,2438,-432
 180 DATA 40,4,253,25,16,252,22,
0,221,94,13,253,25,209,221,78,37
,221,70,25,2083,-247
 190 DATA 253,229,175,178,40,12,
253,126,0,162,168,32,13,62,255,1
87,40,28,253,126,2612,328
 200 DATA 0,163,181,253,119,0,25
3,35,16,228,13,175,177,40,11,253
,225,213,17,32,2404,-360
 210 DATA 0,253,25,209,24,207,19
3,253,229,193,253,225,201,2265,-
415
```

```
00001 ; procedura ustawia atrybuty
00002 ; koloru w/g okreslonych warun-
00003 ; kow przekazanych przez para-
00004 ; metry z Basic'a.
00005 ; program calkowicie relokowalny
00006 defadd equ 23563
00007 odleg equ 8 ;odleglosc miedzy
00008 ;                kolejnymi danymi
00009 ilosc equ 7 ;ilosc zmiennych
00010 dlug  equ 32 ;ilosc bajtow w
00011 ;              wierszu ekranu
00012 atryb equ 22528 ;adres pier-
00013 ;               wszego atry-
00014 ;               butu koloru
00015       ld  ix,(defadd)
00016       ld  de,odleg
00017       ld  b,ilosc
00018 ;kontrola poprawnosci danych
00019 popr  ld  a,(ix+11)
00020       or  (ix+12)
00021       or  (ix+14)
00022       or  (ix+15)
00023       jr  nz,error
00024       add ix,de
00025       djnz popr
00026       ld  ix,(defadd)
00027       or  (ix+29)
00028       jr  z,error
00029       xor a
00030       or  (ix+37)
00031       jr  z,error
00032       ld  a,(ix+13)
00033       add (ix+29)
00034       jr  c,error
00035       cp  33
00036       jr  nc,error
00037       ld  a,(ix+21)
00038       add (ix+37)
00039       jr  c,error
00040       cp  23
00041       jr  nc,error
00042       jr  prog
00043 ; obsluga bledu
00044 ;komunikat: Q Parametr error
00045 error ld  l,25
00046       jp  55h
00047 ; program wlasciwy wymiany
00048 ; tworzenie masek dla porownania
00049 ; i ewentualnej wymiany w/g "s"
00050 prog  xor a
00051       or  (ix+61)
00052       ret z
00053       xor a
00054       ld  h,(ix+61)
00055       bit 7,h
00056       jr  z,skok1
00057       or  a,10000000b
00058 skok1 bit 6,h
00059       jr  z,skok2
00060       or  a,01000000b
00061 skok2 bit 5,h
00062       jr  z,skok3
00063       or  b,00111000b
00064 skok3 bit 4,h
00065       jr  z,skok4
00066       or  a,00000111b
00067 skok4 ld  d,a
00068       xor a
00069       bit 3,h
00070       jr  z,skok5
00071       or  a,10000000b
00072 skok5 bit 2,h
00073       jr  z,skok6
00074       or  a,01000000b
00075 skok6 bit 1,h
00076       jr  z,skok7
00077       or  a,00111000b
00078 skok7 bit 0,h
00079       jr  z,skok8
00080       or  a,00000111b
00081 skok8 ld  e,a
00082       and (ix+53)
00083       ld  l,a
00084       ld  a,e
00085       cpl
00086       ld  e,a
00087       ld  a,d
00088       and (ix+45)
00089       ld  h,a
00090 ; obliczenie w iy adresu poczat-
00091 ; ku okna na ekranie
00092       push iy
00093       push de
00094       ld  de,dlug
00095       ld  iy,atryb
00096       ld  b,(ix+21)
00097       xor a
00098       or  b
00099       jr  z,dod
00100 mnoz   add iy,de
00101       djnz mnoz
00102 dod   ld  d,0
00103       ld  e,(ix+13)
00104       add iy,de
00105       pop de
00106 ; wymiana atrybutow
00107       ld  c,(ix+37)
00108 petla1 ld  b,(ix+29)
00109       push iy
00110 petla2 xor a
00111       or  d
00112       jr  z,podst
00113       ld  a,(iy+0)
00114       and d
00115       cp  h
00116       jr  nz,next
00117       ld  a,255
00118       cp  e
00119       jr  z,pozyc
00120 podst ld  a,(iy+0)
00121       and e
00122       or  l
00123       ld  (iy+0),a
00124 next   inc iy
00125       djnz petla2
00126       dec c
00127       xor a
00128       or  c
00129       jr  z,pozyc
00130       pop iy
00131       push de
00132       ld  de,32
00133       add iy,de
00134       pop de
00135       jr  petla1
00136 pozyc pop bc
00137       push iy
00138       pop bc
00139       pop iy
00140       ret
```

SPECTRUM

```
 10 REM Ruch jednostajnie przyspieszony,autor JAN GOLLA
 20 DIM x(300): DIM y(300)
 30 INPUT "Podaj przyspieszenie a=";a: REM aby wydruk zmiescil sie na ekraniepr
zyjmc a<=5
 40 PRINT AT 1,15;"a=";a: PLOT 110,155: DRAW 50,0
 50 LET t=1
 60 PRINT AT 3+t,0;"v";t;"=";a*t;AT 3+t,8;"s";t;"=";a*t^2/2;AT 3+t,18;"delta s"
;t;"=";a*t^2/2-a*(t-1)^2/2
 70 LET t=t+1
 80 IF t<=17 THEN  GO TO 60
 90 INPUT "Czy sporzadzic wykres v=f(t)";a$
100 IF a$="tak" THEN  GO TO 120
110 IF a$="nie" THEN  STOP
120 CLS
130 PLOT 0,0: DRAW 250,0: PLOT 0,0: DRAW 0,160: PRINT AT 1,0;"v";AT 21,31;"t"
140 FOR t=1 TO 255
145 IF 10*t)=250 OR a*t)=170 THEN  GO TO 165
150 CIRCLE 10*t,a*t,1
160 NEXT t
165 FOR t=1 TO 255
166 IF a*(t/10))=170 THEN  GO TO 170
168 PLOT t,a*(t/10)
169 NEXT t
170 INPUT "czy sporzadzic wykres s=f(t)";b$
180 IF b$="tak" THEN  GO TO 200
190 IF b$="nie" THEN  STOP
200 CLS
210 PLOT 0,0: DRAW 250,0: PLOT 0,0: DRAW 0,165: PRINT AT 1,0;"s";AT 21,31;"t"
220 FOR t=1 TO 255
225 IF a*t^2/2)=170 OR 10*t)=250 THEN  GO TO 250
230 CIRCLE 10*t,a*t^2/2,1
240 NEXT t
250 FOR t=1 TO 255
255 IF a*(t/10)^2/2)=170 THEN  STOP
260 PLOT t,a*(t/10)^2/2
270 NEXT t
```

J. GOLLA



SPECTRUM

PRYWATNY JĘZYK PROGRAMOWANIA

Zapewniam cię, drogi czytelniku, że po przeczytaniu tego artykułu i „wpalcowaniu" kilkuset cyferek kodu maszynowego, będziesz mógł stworzyć swój własny język programowania.

Zasada jest prosta: procedura w kodzie maszynowym porównuje wzorce rozkazów z rozkazem rzeczywistym. Jeżeli nastąpi zgodność, odpowiednie parametry porównania zostaną przekazane do Basicu.

Procedurę wywołuje się za pomocą funkcji użytkownika (FN). DEF FN znajduje się w linii piątej programu. Ma pięć parametrów:

n$ — tablica tekstowa, w której trzymana jest procedura maszynowa. Opis tej metody znajduje się w opisie „Dynamicznych bajtów" (obok),

t$ — tekst rozkazu rzeczywistego, który będzie porównywany ze wzorcami,

o$ — tablica bądź zmienna tekstowa przeznaczona na adresy początku i końca parametrów rozkazu,

r$ — zestaw rozkazów wzorcowych utworzonych z tekstów stałych i symboli zawartych w zmiennej w$,

w$ — zestaw znaków wzorcowych definiujący poszczególne elementy wzorca plus znak-separator, rozdzielający kolejne wzorce.

Proponuję porównać teraz linię 201 programu, gdzie definiuję r$, z programem „Grafika przestrzenna". Oprócz paru dziwnych znaczków teksty stałe wzorców i programu są identyczne. Teraz parę słów o dziwnych znaczkach: opisują one poszczególne elementy wzorca. W nawiasach podaję sposób zapisu. Używam znaków podanych w FN (linia 210). Odpowiadają one pozycji w$ w DEF FN. Uwaga: □ oznacza dowolny znak.

Elementami wzorca są:

— dowolny tekst stały zapisany jako zestaw znaków,

— (£□) tekst wstawiony jako parametr: za tekst uważa się zestaw znaków do napotkania znaku umieszczonego za £,

— (#□) to samo co powyżej, ale łącznie ze znakiem końcowym,

— (@□) pozwala opuścić dowolną ilość zadanego znaku,

— ($) tekst bez narzuconego znaku końcowego,

— (&) dowolna liczba akceptowana przez ZX Spectrum,

— "/" separator poszczególnych wzorców.

Znaki definiujące poszczególne elementy możemy określić sami. Kolejność znaków przekazywanych do w$ jest następująca: £ # @ $ / & (porównaj linię 210).

Najlepiej prześledzić działanie procedury na przykładzie. Niech rozkazem będzie szósta linia programu „Grafika komputerowa":

pozycja:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24		29	30	31	32
znak:				ś	r	o	d	e	k		o	s	i		1	Ø	Ø	,	8	Ø									

Procedura porównuje wzorce po kolei. Zgodność nastąpi dla wzorca:

pozycja:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
znak:	@		ś	r	o	d	e	k		o	s	i		&	,	&	@	

Działanie analizy jest następujące:

— znak @, a po nim spacja oznacza, że w rozkazie może wystąpić dowolnie dużo spacji — analiza zatrzyma się na znaku „s" (czwarta pozycja rozkazu) gdyż jest on różny od spacji),

— następuje porównanie tekstu stałego aż do symbolu & i pozycji 15 rozkazu,

— symbol & oznacza, że musi wystąpić teraz liczba — pozycja liczby zostaje przekazana do zmiennej o$,

— po liczbie musi wystąpić przecinek,

— potem & narzuca kolejną liczbę — jej parametry miejsca zostają przekazane do zmiennej o$,

— może wystąpić dowolna ilość spacji.

Zmienna o$ wygląda następująco:

pozycja:	1	2	3	4	pozostałe znaki
wartość:	15	18	19	21	nie są ustawione

Za koniec parametru jest uważany pierwszy znak nie należący do danego parametru. Odczyt wartości ze zmiennej o$ uzyskujemy za pomocą CODE. Zgodność wzorca i rozkazu powoduje przesłanie do Basica numeru wzorca (w przypadku, gdy żaden wzorzec nie odpowiada rozkazowi, FN przyjmuje wartość zero). W ten sposób za pomocą FN można skakać do odpowiedniej linii obsługi rozkazu (porównaj linie 210). W przedstawionym przykładzie FN=3, stąd linią obsługi jest linia 214.

Oto pozostałe formaty pozycji:

— £□ zostaje podany adres pierwszego znaku i adresu zadanego znaku,

— #□ zostaje podany adres pierwszego znaku tekstu i adres znaku zadanego zwiększony o jeden,

— $ zostaje podany tylko adres pierwszego znaku.

Procedura jest całkowicie zabezpieczona przed błędami. Złe dane wywołują komunikat systemu:

6 Number too big — gdy podana liczba przekracza zakres arytmetyki ZX Spectrum,

C Nonsense in Basic — gdy wzorzec ma niekompletne dane,

E Out of Data — gdy zmienna odpowiedzi nie może pomieścić wszystkich pozycji parametrów lub gdy rozkaz przekracza 255 znaków,

Q Parametr error — gdy nie ma zgodności parametrów lub gdy wzorzec znaków elementów (w$) nie zawiera dokładnie 6 znaków.

Pojawia się kolejny problem: jak wprowadzić program? W liniach 10—113 znajduje się prosty edytor tekstu, pozwalający na wprowadzenie do 22 linii programu.

Ma on następujące opcje:

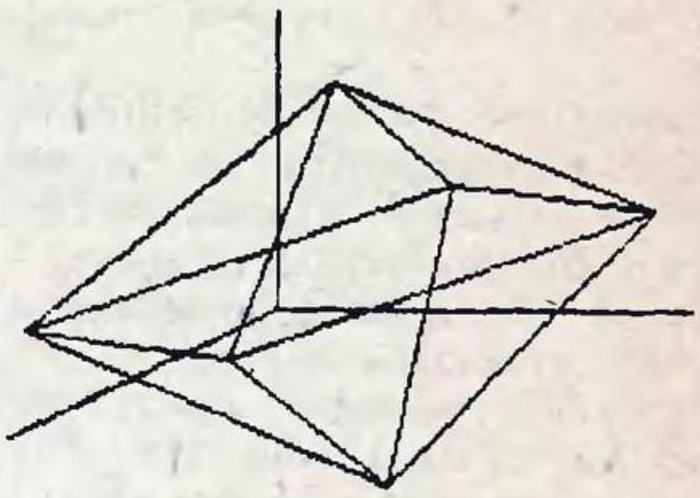
— znacznik można poruszać za pomocą kursorów,

— EDIT ustawia znacznik na początku linii,

— CAPS LOOK ustawia znacznik na początku kolejnej linii.

```
  1 REM
*************************************
*                                   *
*     KRZYSZTOF   POZNIAK           *
*                                   *
*     Klub Mikrokomputerowy         *
*                                   *
*     [redacted]                    *
*                                   *
*                     © 1987        *
*                                   *
*************************************

  5 DEF FN d(n$,t$,o$,r$,w$)=US
R (PEEK (PEEK 23563+256*PEEK 235
64+4)+256*PEEK (PEEK 23563+256*P
EEK 23564+5))
  8 DIM n$(526): GO TO 9100
 10 REM [redacted]
 15 INK 7: PAPER 0: BORDER 0: C
LS
 20 DIM t$(22*32): DIM i$(32):
LET kl=0: LET kk=0
 25 PRINT AT 0,0;t$: LET inv=1:
 LET dk=0: LET dl=0
 30 GO SUB 100
 31 LET i=CODE INKEY$: IF NOT i
 THEN GO TO 31
 35 BEEP .02,30
 40 IF i=8 AND kk THEN LET dk=-
1: LET dl=0: GO TO 30
 41 IF i=9 AND kk<31 THEN LET d
k=1: LET dl=0: GO TO 30
 42 IF i=10 AND kl<21 THEN LET
 dk=0: LET dl=1: GO TO 30
 43 IF i=11 AND kl THEN LET dk=
0: LET dl=-1: GO TO 30
 44 IF i=13 THEN GO TO 200
 45 IF i=12 AND (inv OR kk) THE
N LET i=32*kl+1: LET t$(i TO i+
31)=t$(i TO i+kk-(NOT inv)-1)+t$
(i+kk+1-(NOT inv) TO i+31): PRIN
T AT kl,0;t$(i TO i+31): LET dk=
-(NOT inv)*(NOT NOT kk): LET dl=
0: GO TO 30
 46 IF i=14 THEN LET i=32*kl+1:
 LET t$(i TO i+31)=i$: LET kk=0:
 GO TO 25
 47 IF i=6 THEN LET dl=NOT NOT
(kl<21): LET dk=-kk: GO TO 30
 48 IF i=7 THEN LET kk=0: GO TO
 25
 49 IF i=15 THEN LET inv=NOT in
v: LET dk=0: LET dl=0: GO TO 30
 50 IF i=4 THEN LET i=32*kl: LE
T t$=t$( TO i)+t$(i+33 TO ): GO
TO 25
 51 IF i=5 THEN LET i=32*kl: LE
T t$=t$( TO i)+i$+t$(i+1 TO ): G
O TO 25
 52 IF i>31 AND i<128 AND inv T
HEN LET t$(kl*32+kk+1)=CHR$ i: P
RINT AT kl,kk;CHR$ i: LET dk=(kk
<31): LET dl=0: GO TO 30
 53 IF i>31 AND i<128 AND NOT i
nv THEN LET k=32*kl+1: LET i$(k
TO k+31)=t$(k TO k+kk-1)+CHR$ i+

t$(k+kk TO ): PRINT AT kl,kk;t$(
k+kk TO k+31): LET dk=(kk<31): L
ET dl=0: GO TO 30
 58 LET dk=0: LET dl=0: GO TO 3
0
 100 REM [redacted]
 110 PRINT INVERSE 0; FLASH 8;AT
 kl,kk;SCREEN$ (kl,kk)
 111 LET kk=kk+dk: LET kl=kl+dl
 112 PRINT INVERSE inv; FLASH NO
T inv;AT kl,kk;SCREEN$ (kl,kk)
 113 RETURN
 200 REM [redacted]
 201 LET o$="                    ": LET
r$="0 :$/@ czysc ekran0 /@ srode
k osi &,&0 /@ wymiary osi &,&,&,
&0 /@ rysuj osie0 /@ punkt &,&,&
0 /@ prosta &,&,&0 /@ polacz &,&
,&0 /@ koniec0"
 205 REM parametry wstepne
 208 LET px=0: LET py=0: LET pz=
0: LET sx=120: LET sy=80: LET wx
=15: LET wy=15: LET wz=15: LET w
xz=-10: LET wyz=-11
 210 FOR k=0 TO 21: LET i=32*k:
GO TO 211+FN d(n$,t$(i+1 TO i+32
),o$,r$,"£%0$/&")
 211 BEEP .2,20: PRINT #1;AT 0,0
;"blad w linii :"; INVERSE 1;i$(
k*32+1 TO k*32+31): PAUSE 0: INP
UT "": GO TO 25
 212 GO TO 1000
 213 CLS : GO TO 1000
 214 LET sx=VAL t$(i+CODE o$(1)
TO i+CODE o$(2)-1): LET sy=VAL t
$(i+CODE o$(3) TO i+CODE o$(4)-1
): GO TO 1000
 215 LET wx=VAL t$(i+CODE o$(1)
TO i+CODE o$(2)-1): LET wy=VAL t
$(i+CODE o$(3) TO i+CODE o$(4)-1
): LET wz=VAL t$(i+CODE o$(5) TO
 i+CODE o$(6)-1): LET wyz=SIN (P
I/180)*VAL t$(i+CODE o$(7) TO i+
CODE o$(8)-1): LET wxz=INT (wz*C
OS wyz): LET wyz=-INT (wz*SIN wy
z): GO TO 1000
 216 PLOT sx,sy: DRAW 255-sx,0:
 PLOT sx,sy: DRAW 0,175-sy: PLOT
 sx,sy: LET z1=sx/wxz: LET z2=sy
/wyz: LET z1=z1*(z1>z2)+z2*(z2>z
1): DRAW -z1*wxz,-z1*wyz: GO TO
1000
 217 LET px=VAL t$(i+CODE o$(1)
TO i+CODE o$(2)-1): LET py=VAL t
$(i+CODE o$(3) TO i+CODE o$(4)-1
): LET pz=VAL t$(i+CODE o$(5) TO
 i+CODE o$(6)-1): PLOT sx+px*wx+
wxz*pz,sy+py*wy+wyz*pz: GO TO 10
00
 218 LET z1=VAL t$(i+CODE o$(1)
TO i+CODE o$(2)-1): LET z2=VAL t
$(i+CODE o$(3) TO i+CODE o$(4)-1
): LET z3=VAL t$(i+CODE o$(5) TO
 i+CODE o$(6)-1): DRAW z1*wx+z3*
wxz,z2*wy+z3*wyz: LET px=px+z1:
LET py=py+z2: LET pz=pz+z3: GO T
O 1000
 220 LET x4=PEEK 23677: LET z5=P
EEK 23678: LET z1=VAL i$(i+CODE
o$(1) TO i+CODE o$(2)-1): LET z2
=VAL i$(i+CODE o$(3) TO i+CODE o
$(4)-1): LET z3=VAL i$(i+CODE o$
(5) TO i+CODE o$(6)-1): DRAW (z1
-px)*wx+(z3-pz)*wxz,(z2-py)*wy+(
z3-pz)*wyz: POKE 23677,x4: POKE
23678,z5: GO TO 1000
 1000 GO TO 1000
 1001 GO TO 9000
 1002 NEXT k
 9000 PRINT #1;AT 0,0; INVERSE 1;
"program wykonany": PAUSE 0: INP
UT "": GO TO 25
 9100 RESTORE 9950: LET data=9950
: LET z3=1: LET z1=LEN n$
 9101 LET si=0: LET s2=0: LET r=1
: FOR k=1 TO 20: READ z2: LET n$
(z3)=CHR$ z2: LET s1=s1+z2: LET
s2=s2+r*z2: LET r=-r: IF z3=z1 T
HEN GO TO 9105
 9103 LET z3=z3+1: NEXT k
 9105 READ s1,s2: IF s1<>si OR
 s2<>s2 THEN PRINT "blad w lini
i ";data: STOP
 9106 IF z3=z1 THEN GO TO 10
 9107 LET data=data+1: GO TO 9102
 9950 DATA 221,42,11,92,221,126,4
3,183,32,7,221,126,42,254,6,40,9
 9951 DATA 217,46,25,195,85,8,42,
92,229,217,221,110,31,221,102
 9952 DATA 221,55,34,1,1,0,221,11
7,4,221,115,5,221,115,6,221,114,
7,217,221,201,161
 9953 DATA 78,40,221,70,41,197,22
1,110,13,221,102,14,221,94,15,22
1,86,16,221,225,24,7,11
 9954 DATA 217,122,176,40,93,125,
35,87,221,190,0,40,76,221,190,1,
48,72,221,190,2,40,242
 9955 DATA 2,40,69,221,190,3,40,6
6,217,221,190,4,40,27,221,190,5,
40,64,8,1858,216
 9956 DATA 122,179,40,52,8,190,32
,48,35,27,24,204,33,88,35,217,46
,11,195,85,1675,-527
 9957 DATA 0,122,179,217,40,28,3,

221,229,221,42,11,92,221,117,4,2
21,116,5,221,225,24,16,-161
 9958 DATA 115,5,221,114,7,217,24
,154,24,74,24,24,24,116,24,116,2
4,116,217,122,178,-411
 9959 DATA 179,40,167,122,179,40,
196,128,254,48,32,8,35,27,5,122,
179,40,114,5,1762,-634
 9960 DATA 126,285,87,45,50,107,3
4,93,213,229,217,205,213,197,
 9961 DATA 45,193,209,225,217,209
,42,93,92,183,237,82,235,225,25,
235,3164,-240
 9962 DATA 66,71,24,145,122,179,4
0,140,122,217,5,122,179,40,56,5,
40,48,235,35,2038,48
 9963 DATA 7,24,243,24,137,122,17
9,40,33,128,35,217,5,122,179,40,
48,35,217,217,1739,-767
 9964 DATA 5,122,179,40,109,5,122
,32,167,35,24,243,217,221,122,
 9965 DATA 179,40,35,87,40,10,24
,243,1,80,0,33,88,35,197,217,
 9966 DATA 34,93,92,201,217,221,5
,221,78,4,221,70,5,221,110,4,221,
 9967 DATA 102,5,221,94,7,221,115
,4,221,114,5,217,201,-161
 9968 DATA 221,86,7,221,113,4,221
,94,5,221,78,4,221,70,5,217,201,
 9969 DATA 17,1,0,221,25,217,122,
179,40,87,221,86,1,221,94,2,-87
 9970 DATA 221,86,3,221,94,4,221,
78,5,221,70,6,217,221,190,0,40,1
 9971 DATA 100,1,40,110,221,190,2
,40,113,221,190,3,40,114,35,35,4
,24,200,1816,-26
 9972 DATA 33,88,35,217,46,13,195
,85,0,33,88,35,221,42,11,92,217,
221,78,4,1768,94
 9973 DATA 221,70,5,225,34,93,92,
201,24,180,120,177,40,222,11,123
,217,2,3,217,2277,-743
 9974 DATA 126,254,45,32,5,35,126
,28,40,206,34,93,92,213,229,217,
229,213,197,205,2610,-373
 9975 DATA 155,44,205,213,45,193,
209,225,217,209,42,93,92,183,237
,82,235,225,25,235,3164,-240
 9976 DATA 122,183,32,174,42,93,9
2,120,177,40,165,11,123,217,2,3,
24,186,24,165,1993,-387
 9977 DATA 24,26,24,46,24,71,24,1
40,120,177,40,144,11,123,217,2,3
,126,217,190,1755,-347
 9978 DATA 40,221,35,28,40,130,24
,247,120,177,40,245,11,123,217,2
,3,126,35,27,1994,-764
 9979 DATA 217,28,40,236,190,35,4

0,195,24,247,217,126,35,87,217,1
50,32,204,35,28,2353,-269
 9980 DATA 40,8,24,247,217,122,17
8,40,185,126,221,190,4,40,178,24
,203,120,177,40,2380,478
 9981 DATA 199,123,217,2,24,168,7
33,147
```

```
; grafika przestrzenna
; rysowanie osmioscianu
;
   czysc ekran
; parametry ekranu
   srodek osi 100,80
   wymiary osi 20,20,20,155
   rysuj osie
; osmioscian
   punkt 1,0,2
   polacz 4,-2,2
   polacz 2,4,1
   prosta 0,2,4
   polacz 4,-2,2
   polacz 2,4,1
   prosta 7,2,-1
   polacz 4,-2,2
   polacz 2,4,1
   prosta 0,-2,-4
   polacz 4,-2,2
   polacz 2,4,1
   prosta -7,-2,1
```



— TRUE VIDEO kasuje linię, w której znajduje się znacznik,
— INV VIDEO tworzy nową linię nad linią ze znacznikiem (zniszczeniu ulega linia 22),
— ENTER uruchamia program,
— EXTEND MODE kasuje treść linii, w której znajduje się znacznik i ustawia znacznik na początku tej linii,
— GRAPHICS ustawia tryb wprowadzania i kasowania:

a) znacznik INVERSE
— znak w znaczniku jest wymieniany,
— kasowanie niszczy znak w znaczniku,

b) znacznik FLASH:
— znak jest dopisywany przed znacznikiem (ginie ostatnia pozycja linii),
— kasowanie niszczy znak przed znacznikiem,

— DELETE kasuje znaki zgodnie z trybem GRAPHICS,
— mogą być wprowadzane znaki pojedyncze o kodach od 32 do 127.

Mniej doświadczonym programistom zwracam uwagę na identyfikację opcji za pomocą CODE INKEY$.
Uruchamianie edytora:
od linii 10 — kasowania tekstu programu,
od linii 25 — program nie zostaje zniszczony.

I ostatni problem: dotyczy programu-przykładu „Grafika przestrzenna". Należy przedstawiony program wprowadzić za pomocą edytora i uruchomić przez ENTER. Na ekranie powinien pojawić się rysunek pokazany poniżej. Życzę ciekawych wyni-

ków graficznych dzięki zmianie parametrów w rozkazach. Zwracam jednak uwagę, że ze względu na prostotę programu, nie ma tu zabezpieczeń przed błędnymi danymi. Z tego powodu rysunek może „wyjść" poza ekran i wywołać błąd systemu. Należy wtedy wykonać GO TO 25.

Przedstawiam dostępne rozkazy grafiki przestrzennej:
— ;dowolny tekst komentarza,
— czyść ekran — odpowiada CLS,
— środek osi x, y — ustawia środek osi w punkcie odpowiadającym PLOT x, y,
— rysuj osie — rysuje osie układu przestrzennego,
— punkt x, y, z — stawia na ekranie punkt o podanych współrzędnych,
— polacz x, y, z — łączy ostatnio narysowany punkt z punktem o zadanych współrzędnych,
— prosta dx, dy, dz — rysuje prostą analogicznie do DRAW, ale w przestrzeni,
— koniec — powoduje pominięcie linii poniżej i zakończenie programu.
Uwagi dodatkowe:
— program pomija linię złożoną z samych spacji,
— program wypisuje linię nie odpowiadającą żadnemu wzorcowi,
— wymiary osi wx, wy, wz, kąt — określa liczbę pikseli przypadających na odległość jednostkową oraz ustala kąt rzutu osi z,
— prawidłowe wykonanie programu powoduje komunikat: „program wykonany"; naciśnięcie dowolnego klawisza spowoduje powrót do edytora.

Zamiast przedstawionego programu w liniach 200—9000 można umieścić swój prywatny język programowania.

Krzysztof POŹNIAK

SPECTRUM

KATALOG GIER

Program powstał z myślą o ułatwieniu życia zwolennikom gier, którzy wymieniają je między sobą. Jest on specjalizowaną bazą danych, umożliwiającą zapis do 300 rekordów. Każdy z nich zawiera 4 pola (zmienne N$, R$, V$, T$). Na każdym polu można zapisać do 25 symboli (liter lub cyfr). Program przeznaczony jest głównie do wykonywania wykazów gier w alfabetycznej kolejności tytułów, wzbogaconych dodatkowo o charakterystykę gry wg symboliki zamieszczonej na planszy „CHARAKTERYSTYKA GIER". Program ładuje się normalnie, tzn. LOAD"" i obsługuje zgodnie z instrukcjami wyświetlanymi na ekranie. Menu zawiera 6 opcji:

1) Opcja A — umożliwia, po podłączeniu drukarki (ZX-Printer, Seikosha GP-50S), automatyczne uzyskanie wykazu gier. Poza tytułem i charakterystyką gry drukowany i wyświetlany jest również numer, pod którym gra jest zapisana w „KATALOGU GIER".

2) Opcja S — umożliwia szybkie sprawdzenie, jakie tytuły gier, zaczynające się na daną literę, są zapisane w „KATALOGU GIER". Ułatwia ona właściwe wpisanie tytułu gry po wybraniu opcji P.

3) Opcja P — umożliwia dostęp do informacji dotyczących gry. Wyświetlany jest tytuł gry, jej producent, krótki opis gry oraz rok produkcji i nr taśmy, na której jest nagrana. Opcja P pozwala również na zmodyfikowanie zapisów na poszczególnych polach, bez kasowania wszystkich danych opcją K.

4) Opcja W — umożliwia wpisanie kolejnego rekordu. Jeżeli nie chcesz lub nie możesz zapisać danego pola, to wciśnij ENTER. Dwie pierwsze litery na polu o nazwie „Rodzaj gry" zapisz w trybie C, zgodnie z symboliką wyświetlaną na planszy „CHARAKTERYSTYKA GIER", dostępnej po wybraniu opcji A*. Resztę pola możesz zapisać dowolnie, np. dla tytułu ANT ATTACK: ZP Królewny i mrówki, co charakteryzuje grę jako zręcznościowo-przygodową i informuje, co cię czeka podczas gry. Tytuły gier zawsze wpisuj w trybie C, gdyż klawiatura automatycznie do niego przechodzi po wybraniu opcji P lub S.

5) Opcja K — umożliwia skasowanie dowolnego zapisu. Jeżeli wybrałeś ją przez pomyłkę, to wpisz O i wciśnij ENTER, a program wróci do menu. Jeśli pomyliłeś się przy wpisywaniu numeru zapisu do skasowania, to dopisz kilka cyfr, aby liczba która powstanie, była większa od 300 i wciśnij ENTER, co pozwoli ci ponownie wpisać numer zapisu, który chcesz skasować.

6) Opcja Z — umożliwia zapis wprowadzonych danych na taśmie. Po zakończeniu nagrywania nie jest wyświetlany komunikat O.K., przestaje jedynie migać border.

I jeszcze parę uwag praktycznych. Program wgrywa się około 3 minut. Korzystnie jest 2 egzemplarze programu uaktualniać na zmianę, co zabezpiecza przed utratą danych i programu. Nic nie stoi na przeszkodzie, aby użytkownik „KATALOGU GIER" zmienił nazwy pól w rekordzie, jeżeli chce wprowadzić inne dane niż zaproponowane, lub też rozszerzyć o swoje dane personalne i adres nagłówek drukowany automatycznie w opcji A. Po wykonaniu wykazu gier, program daje możliwość wydrukowania dowolnego tekstu, stosownie do potrzeb.

Inspiracją do opracowania „KATALOGU GIER" był zamieszczony w nr. 811/85 francuskiego miesięcznika „Science et Vie" program pt. „Fichier confidentiel". Usunąłem z niego błędy, zaadaptowałem oraz rozszerzyłem.

* P — przygodowa
S — sportowa
L — losowa
Y — symulacyjna
T — tekstowa
Z — zręcznościowa
W — wojenna
R — strategiczna
K — kosmiczna
U — pr. użytkowy

A. URBANKOWSKI

```
  10 REM KATALOG
 100 DIM N$(300,25)
 110 DIM R$(300,25)
 120 DIM V$(300,25)
 130 DIM T$(300,25)
 140 DIM X$(1,25)
 141 BORDER 7: PAPER 7: INK 0: C
LS
 142 BORDER 1: PAPER 1: INK 7: P
RINT AT 5,10;"KATALOG GIER": INV
ERSE 1: PRINT AT 0,0;" © 1987 A.
Urbankowski": INVERSE 0: BEEP .1
,20: FLASH 1: PRINT AT 19,8;"ZAT
RZYMAJ  TASME": INVERSE 1: PRINT
 AT 21,5;"WCISNIJ DOWOLNY KLAWIS
Z": INVERSE 0: FLASH 0
 144 PRINT AT 5,7;"**************
*****";AT 11,7;"****************
**"
 146 PRINT AT 6,7;"*": PRINT AT
6,24;"*"
 148 PRINT AT 7,7;"*": PRINT AT
7,24;"*"
 150 PRINT AT 8,7;"*": PRINT AT
8,24;"*"
 152 PRINT AT 9,7;"*": PRINT AT
9,24;"*"
 154 PRINT AT 10,7;"*": PRINT AT
10,24;"*"
 156 IF INKEY$="" THEN GO TO 156
 157 CLS : BEEP .1,20
 158 BORDER 0: PAPER 0: INK 6
 160 FLASH 1: INK 6: PRINT AT 1,
13;"UWAGA": FLASH 0
 164 PRINT AT 3,0;"Program ""KAT
ALOG"" umozliwia ska-talogowanie
 300 gier.Program ob-sluguje sie
 wedlug instrukcji  wyswietlany
ch na ekranie.Pod za-dnym pozore
m nie nalezy uzywac  rozkazu RUN
, ktory niszczy dane  zgromadzon
e w pamieci! Program zatrzymany
przez C.SH+SPACE moz-na uruchomi
c rozkazem GO TO 141,GO TO 200 l
ub CONTINUE.Tytul na-lezy wpisyw
ac w trybie C (duze  litery).Pod
laczenie drukarki po-zwala uzysk
ac wykaz posiadanych gier w porz
adku alfabetycznym.  Tytuly gier
 majace na poczatku  takie same
wyrazy zapisz laczac je znakiem"
"-"",np:JET-PAC,JET-MAN"

 166 INVERSE 1: INK 6: PRINT AT
21,5;"WCISNIJ DOWOLNY KLAWISZ":
INVERSE 0: INK 8
 170 IF INKEY$="" THEN GO TO 170
 180 BEEP .1,20: CLS
 200 OUT 254,0: PAPER 0: INK 6:
PRINT AT 9,1;"Poszukiwanie danyc
h-wcisnij: P";AT 13,3;"Wpisanie
danych-wcisnij: W";AT 5,2;"Spraw
dzenie tytulu-wcisnij: S"
 210 PRINT AT 0,7;"Automatyczny
wydruk";AT 1,8;"tytulow-wcisnij:
A",AT 17,3;"Kasowanie zapisu-wci
snij: K";AT 21,0;"Zapis danych n
a tasme-wcisnij: Z"
 215 LET M=0
 220 LET D$=INKEY$
 250 IF D$="p" THEN GO SUB 1000
 260 IF D$="w" THEN GO SUB 2000
 270 IF D$="k" THEN GO SUB 5000
 280 IF D$="z" THEN GO SUB 7000
 282 IF D$="s" THEN GO SUB 8000
 285 IF D$="a" THEN GO SUB 9000
 290 IF M=1 THEN GO SUB 2210
 300 GO TO 200
1000 CLS : BEEP .1,20
1010 PRINT AT 0,7;"POSZUKIWANIE
DANYCH"
1020 PRINT AT 5,3;"Wpisz tytul,w
cisnij ENTER"
1030 FOR B=1 TO 25
1040 LET X$(1,B)=" "
1050 NEXT B
1100 LET C=0
1110 IF INKEY$="" THEN GO TO 111

1120 LET C$=INKEY$
1130 IF INKEY$<>"" THEN GO TO 11
30
1135 IF CODE C$>90 THEN LET C$=C
HR$ (CODE C$-32)
1140 IF CODE C$<>13 THEN LET X$(
1,C+1)=C$
1150 LET C=C+1-2*((CODE C$=8)*(C
<>0))
1160 INK 2: PRINT AT 7,1+C,"-":
BEEP .01,50
1170 INK 4: PRINT AT 7,C;C$: INK
 6
1180 IF C=25 OR CODE C$=13 THEN
GO TO 1200
1190 GO TO 1110
1200 LET R=1
1205 PRINT AT 2,10;"TYTUL Nr:"
1210 LET T=0
1230 FOR I=1 TO 25
1235 IF R=301 THEN GO TO 1400
1240 IF X$(1,I)<>N$(R,I) THEN LE
T T=1
1245 IF N$(R,I)=" " OR T=1 THEN
LET I=25
1250 NEXT I
1255 INK 2: PRINT AT 2,21;R
1275 LET R=R+1
1280 IF T=1 THEN GO TO 1210
1290 BEEP .1,20
1310 INK 7: PRINT AT 7,0;N$(R-1)
;AT 10,0;R$(R-1);AT 13,0;V$(R-1)
,AT 15,0;T$(R-1)
1330 INK 5
1340 PRINT AT 5,3;" Informacje d
otyczace gry: ";AT 19,1;"NASTEPNY
 TYTUL-WCISNIJ SPACE";AT 21,1;"M
ODYFIKACJE ZAPISOW-WCISNIJ: M"

1350 IF INKEY$="m" THEN GO SUB 7
500
1360 IF INKEY$<>" " AND M=0 THEN
 GO TO 1350
1380 CLS
1390 RETURN
1400 FLASH 1: INK 4: PRINT AT 15
,9;"TYTUL NIEZNANY": FLASH 0: IN
K 5
1410 BEEP 1,0: BEEP 1,12: BEEP 1
,24: CLS
1420 RETURN
2000 CLS : BEEP .1,20
2010 PRINT AT 0,6;"WPISANIE DANY
CH"
2020 INK 2: FLASH 1: PRINT AT 2,
6;"  POCZEKAJ  ": FLASH 0
2100 LET R=1
2110 IF N$(R,1)=" " OR R>300 THE
N GO TO 2200
2120 LET R=R+1
2130 IF R<>301 THEN GO TO 2110
2140 PRINT AT 2,6;"  BRAK MIEJSC
A"
2145 RETURN
2200 PRINT AT 2,9;"ZAPIS  Nr:";R

2210 INK 6: PRINT AT 5,3;"Wpisz
tytul,wcisnij ENTER"
2211 IF M=1 THEN PRINT AT 10,0;"
Tytul aktualny:",AT 11,0;N$(R)
2215 LET C=0
2220 GO SUB 5000
2230 IF CODE C$<>13 AND CODE C$<
>8 AND C<>0 THEN LET N$(R,C)=C$
2240 IF CODE C$=13 OR C=25 THEN
GO TO 2300
2250 GO TO 2220
2300 CLS
2310 INK 6: PRINT AT 5,1;"Wpisz
Producenta,wcisnij ENTER"
2311 IF M=1 THEN PRINT AT 10,0;"
Producent aktualny:";AT 11,0;R$(
R)
2315 LET C=0
2320 GO SUB 5000
2330 IF CODE C$<>13 AND CODE C$<
>8 AND C<>0 THEN LET R$(R,C)=C$
2340 IF CODE C$=13 OR C=25 THEN
GO TO 2400
2350 GO TO 2320
2400 CLS
2410 INK 6: PRINT AT 5,1;"Wpisz
rodzaj gry,wcisnij ENTER"
2411 IF M=1 THEN PRINT AT 10,0;"
Rodzaj gry aktualny:";AT 11,0;V$
(R)
2415 LET C=0
2420 GO SUB 5000
2430 IF CODE C$<>13 AND CODE C$<
>8 AND C<>0 THEN LET V$(R,C)=C$
2440 IF CODE C$=13 OR C=25 THEN
GO TO 2500
2450 GO TO 2420
2500 CLS
2510 INK 6: PRINT AT 3,1;"Wpisz
rok produkcji i nr tasmy.";AT 5,
9;"wcisnij ENTER"
2511 IF M=1 THEN PRINT AT 10,0;"
Rok produkcji,nr tasmy aktualny:
";AT 11,0;T$(R)
2515 LET C=0
2520 GO SUB 5000
2530 IF CODE C$<>13 AND CODE C$<
>8 AND C<>0 THEN LET T$(R,C)=C$
2540 IF CODE C$=13 OR C=25 THEN
GO TO 2600
2550 GO TO 2520
2600 CLS
2610 GO TO 200
5000 IF M=1 THEN PRINT AT 19,2;"
Wcisnij ENTER po modyfikacji";AT
 20,2;"danych,lub aby je pozosta
wic";AT 21,11;"bez zmian"
5005 IF INKEY$="" THEN GO TO 500
5
5010 LET C$=INKEY$
5020 IF INKEY$<>"" THEN GO TO 50
20
5030 LET C=C+1-2*((CODE C$=8)*(C
<>0))
5040 INK 2: PRINT AT 7,1+C;"-":
BEEP .01,50
5050 INK 4: PRINT AT 7,C;C$: INK
 6
5060 RETURN
6000 INK 2: PRINT AT 18,4;"NR ZA
PISU DO SKASOWANIA?";AT 20,0;"
[illegible]
: INK 6: BEEP 1,-24
6010 INPUT F
6020 IF F=0 THEN GO TO 6100
6030 IF F<1 OR F>300 THEN GO TO
6010
6040 FOR I=1 TO 25
6050 LET N$(F,I)=""
6060 LET R$(F,I)=""
6070 LET V$(F,I)=""
6080 LET T$(F,I)=""
6090 NEXT I
6095 BEEP .1,20
6100 CLS
6110 RETURN
7000 SAVE "KATALOG" LINE 141
7100 RETURN
7500 LET H=1
7510 LET R=R-1
7520 RETURN
8000 CLS : PRINT AT 0,2;"PODAJ P
IERWSZA LITERE TYTULU";AT 1,11;"
I POCZEKAJ": PRINT : PRINT : BEE
P .1,20
8005 IF INKEY$<>"" THEN GO TO 80
05
8010 IF INKEY$="" THEN GO TO 801
0
8020 LET K$=INKEY$
8030 IF CODE K$>90 THEN LET K$=C
HR$ (CODE K$-32)
8100 FOR K=1 TO 300
8110 IF N$(K,1)=K$ THEN PRINT K;
TAB 4;N$(K);TAB 30;V$(K,1);TAB 3
1;V$(K,2)
8120 NEXT K
8130 BEEP .1,20
8140 FLASH 1: INVERSE 1: PRINT A
T 21,0;"POWROT DO MENU:WCISNIJ C
OKOLWIEK": INVERSE 0: FLASH 0
8150 IF INKEY$="" THEN GO TO 815
0
8160 CLS : BEEP .1,20: RETURN
9000 CLS : PAUSE 25: BEEP .1,20
9001 PRINT AT 0,6;"CHARAKTERYSTY
KA GIER";AT 1,5;"..............
.......": PRINT : PRINT : PRINT
"P-przygodowa","Z-zrecznosciowa"
: PRINT : PRINT "S-sportowa","W-
wojenna": PRINT : PRINT "L-losow
a","R-strategiczna": PRINT : PRI
NT "Y-symulacyjna","K-kosmiczna"
: PRINT : PRINT "T-tekstowa","U-
pr. uzytkowy": FLASH 1: INVERSE
1: PRINT AT 21,5;"WCISNIJ DOWOLN
Y KLAWISZ": INVERSE 0: FLASH 0
9002 IF INKEY$="" THEN GO TO 900
2
9003 BEEP .1,20: CLS
9004 PRINT " WYKAZ GIER    ZX-SP
ECTRUM 48kb ": PRINT "==========
==========================": PRINT :
 PRINT
9005 LPRINT " WYKAZ GIER    ZX-S
PECTRUM 48kb ": LPRINT "========
==========================": LPRIN
T : LPRINT : LPRINT
9006 POKE 23692,255
9008 FOR L=65 TO 90
9010 LET K$=CHR$ L: GO SUB 9100
9012 NEXT L
9065 LPRINT : LPRINT : LPRINT "
    CHARAKTERYSTYKA GIER    "
: LPRINT "          ..............
.....     ": LPRINT : LPRINT : L
PRINT "P-przygodowa","Z-zrecznos
ciowa": LPRINT : LPRINT "S-sport
owa","W-wojenna": LPRINT : LPRIN
T "L-losowa","R-strategiczna": L
PRINT : LPRINT "Y-symulacyjna","
K-kosmiczna": LPRINT : LPRINT "T
-tekstowa","U-pr. uzytkowy": LPR
INT
9070 IF K$=CHR$ 90 THEN INPUT "W
prowadz tekst do wydruku i nacis
nij ENTER";Y$: LPRINT : LPRINT :
 LPRINT : LPRINT Y$
9080 CLS : FLASH 1: INVERSE 1: P
RINT AT 10,10;"WYKAZ GOTOWY": BE
EP 1,0: BEEP 1,12: BEEP 1,24: IN
VERSE 0: FLASH 0: CLS : RETURN
9100 FOR K=1 TO 300
9110 IF N$(K,1)=K$ THEN PRINT K;
TAB 4;N$(K);TAB 30;V$(K,1);TAB 3
1;V$(K,2)
9120 IF N$(K,1)=K$ THEN LPRINT K
;TAB 4;N$(K);TAB 30;V$(K,1);TAB
31;V$(K,2)
9130 NEXT K
9140 BEEP .1,20: BEEP .1,20: BEE
P .1,20: BEEP .1,20: PAUSE 50: P
RINT : LPRINT : RETURN
```

PROSTOPADŁOŚCIAN

```
 210 LET wskrys=1
 220 GO SUB rysowanie
 230 LET wskrys=0
 240 LET dlbloku=dlbloku+DELTA
 250 GO SUB rysowanie
 260 GO TO wczytaniezn
 900 REM

1000 REM rysowanie
       prostopadlosciany
1010 PLOT INVERSE wskrys;
     110-dlbloku,64+dlbloku
1020 DRAW INVERSE wskrys;
     2*dlbloku,0
1030 DRAW INVERSE wskrys;
     0,-2*dlbloku
1040 DRAW INVERSE wskrys;
     dlbloku/2,dlbloku
1050 DRAW INVERSE wskrys;
     0,1.8*dlbloku
1060 DRAW INVERSE wskrys;
     -dlbloku/2,-0.8*dlbloku
1070 PLOT INVERSE wskrys;
     110+1.5*dlbloku,
     64+1.8*dlbloku
1080 DRAW INVERSE wskrys;
     -1.8*dlbloku,0
1090 DRAW INVERSE wskrys;
     -0.7*dlbloku,-0.8*dlbloku
1100 DRAW INVERSE wskrys;
     0,-2*dlbloku
1110 DRAW INVERSE wskrys;
     2*dlbloku,0
1120 RETURN
```

```
  20 LET rysowanie=1000:
     LET wczytaniezn=160
  40 BORDER 0: PAPER 6:
     INK 9: BRIGHT 1: CLS
  50 PRINT PAPER 1;AT 2,0;   "
Powiekszanie i zmniejszanie
     prostopadloscianu       "
  60 PRINT AT 10,3;"Nacisniecie"
  70 PRINT AT 12,1;   "P lub p -
powoduje powiekszenie"
  80 PRINT AT 14,1;   "Z lub z -
powoduje zmniejszenie"
  90 PRINT FLASH 1;AT 19,4;
     "nacisnij dowolny klawisz"
 100 PAUSE 0
 110 REM
 120 CLS
 130 LET dlbloku=25
 140 LET wskrys=0
 150 GO SUB rysowanie
 155 REM

 160 REM wczytanie znaku
 170 LET DELTA=(CODE INKEY$=80)
                +(CODE INKEY$=112)
                -(CODE INKEY$=90)
                -(CODE INKEY$=122)
 180 LET DELTA=DELTA*3
 190 IF dlbloku+DELTA>63 OR
        dlbloku+DELTA<4 THEN
        LET DELTA=0
 200 IF DELTA=0 THEN
        GO TO wczytaniezn
```

„Życie maklera" jest przetłumaczonym programem z Commodore C-64, opublikowanym w „IKS-e" nr 2/86, na komputer Spectrum, Timex lub ELWRO 800 Junior. Tak więc już nie tylko comodorowcy będą mogli dzięki operacjom giełdowym zostać milionerami lub... zbankrutować.

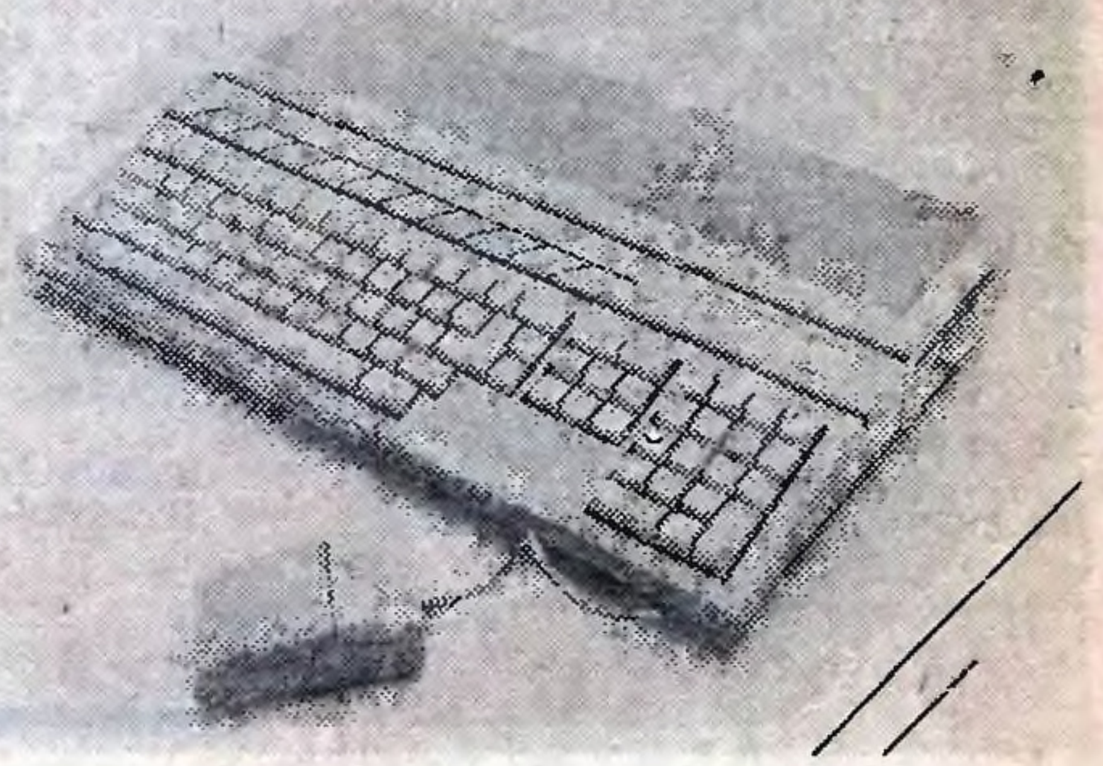
```
 10 REM     PRZETLUMACZYL
            P. WASKIEWICZ
 15 PAPER 6: BORDER 6: INK 2: C
LS : PRINT AT 2,3; FLASH 1;" Z Y
 C I E  M A K L E R A "
 20 PRINT AT 4,4;"JESTES WLASCI
CIELEM KOPALNI.WYKORZYSTUJAC  OP
ERACJE NA GIEL-DZIE SPROBUJ ZARZ
ADZAC  JAK NAJ-DLUZEJ POSIADANYM
 MAJATKIEM."
 25 PRINT AT 10,5; INVERSE 1;"
U W A G A ! ! "
 30 PRINT AT 13,4;"NIE MOZE PRA
COWAC MNIEJ NIZ 10 GORNIKOW W JE
DNEJ KOPALNI."
 40 PRINT AT 15,0;"POZIOM  ZYCI
A  GORNIKOW OKRESLA WYDAJNOSC IC
H PRACY I WIELKOSC  ZATRUDNIENIA
."
 50 PAUSE 700
 100 LET LK=INT (RND*4+5)
 110 LET LG=INT (RND*30+60)
 120 LET SK=INT (RND*50+10)*LG
 130 LET RW=INT (RND*40+80)
 140 LET Z=0: LET PZG=1: LET R=1
: LET ZN=0
 200 PAPER 1: INK 5: BORDER 1: C
S : PRINT AT 2,3; INVERSE 1;" Z
Y C I E  M A K L E R A "
 203 PLOT 0,175: DRAW 255,0: DRA
 0,-35: DRAW -255,0: DRAW 0,35
 205 PRINT AT 4,14;"ROK ";R
 207 PLOT 0,125: DRAW 255,0: DRA
 0,-55: DRAW -255,0: DRAW 0,55
 210 PRINT AT 6,7; INVERSE 1;" O
BECNIE POSIADASZ "
 215 PRINT AT 7,1;"STAN KONTA...
..... ";SK;" $"
 220 PRINT AT 8,1;"LICZBA GORNIK
OW..... ";LG
 225 PRINT AT 9,1;"LICZBA KOPALN
I..... ";LK
 230 PRINT AT 10,1;"ROCZNE WYDOB
YCIE..... ";RW;" TON"
 235 LET Z=Z+RW*LK
 240 PRINT AT 11,1;"ZAPASY......
..... ";Z;" TON"
 245 PRINT AT 12,1;"POZIOM ZYCIA
 GORNIKOW ";PZG
 247 PLOT 0,59: DRAW 255,0: DRAW
 0,-29: DRAW -255,0: DRAW 0,29
 250 PRINT AT 14,12; INVERSE 1;"
 GIELDA "
 255 LET CR=INT (RND*12+7)
 260 LET CK=INT (RND*2000+2000)
 265 LET CZ=INT (RND*40+80)
 270 PRINT AT 15,1;"CENA TONY RU
DY..... ";CR;" $"
 275 PRINT AT 16,1;"CENA KOPALNI
..... ";CK;" $"
 280 PRINT AT 17,1;"CENA ZYWNOSC
 270 PRINT AT 15,1;"CENA TONY RU
DY..... ";CR;" $"
 275 PRINT AT 16,1;"CENA KOPALNI
..... ";CK;" $"
 280 PRINT AT 17,1;"CENA ZYWNOSC
I GORNIKA ";CZ;" $"
 283 PLOT 0,4: DRAW 255,0
 285 PRINT AT 21,8; INVERSE 1;"
KUPNO/SPRZEDAZ "
 288 PRINT AT 19,1; INVERSE 1;"S
YTUACJE:"
 290 GO SUB 1000
 295 INPUT "ILE TON RUDY SPRZEDA
JESZ ? ";SR
 300 IF SR<0 OR SR>Z THEN  GO TO
 295
 305 IF SR<>0 THEN  LET Z=Z-SR:
LET SK=SK+SR*CR: GO SUB 1010: GO
 SUB 1030
 310 GO SUB 1000
 315 INPUT "ILE KOPALNI SPRZEDAJ
ESZ ? ";KS
 320 IF KS<0 OR KS>LK THEN  GO T
O 315
 325 IF KS<>0 THEN  LET LK=LK-KS
: LET SK=SK+KS*CK: GO SUB 1010:
GO SUB 1020
 330 IF LK=0 THEN  GO TO 2000
 335 GO SUB 1000
 337 IF KS<>0 THEN  GO TO 360
 340 INPUT "ILE KOPALNI KUPUJESZ
 ? ";KK
 345 IF KK<0 OR KK*CK>SK THEN  G
O TO 340
 350 IF KK<>0 THEN  LET LK=LK+KK
: LET SK=SK-KK*CK: GO SUB 1010:
GO SUB 1020
 355 GO SUB 1000
 360 INPUT "ILE $ WYPLACASZ GORN
IKOM "|WG
 365 IF WG<0 OR WG>SK THEN  GO T
O 360
 370 IF WG<>0 THEN  LET SK=SK-WG
: GO SUB 1010
 400 IF WG/LG>CZ+20 THEN  LET PZ
G=PZG+.1: GO SUB 1040
 405 IF WG/LG<CZ-20 THEN  LET PZ
G=PZG-.1: GO SUB 1040
 410 IF PZG<.6 THEN  GO TO 2010
 415 IF PZG>1.1 THEN  LET RW=RW+
INT (RND*20+5)
 420 IF PZG>=1 THEN  GO TO 435
 425 IF RW>25 THEN  LET RW=RW-IN
T (RND*20+5): IF RW<0 THEN  LET
RW=0
 430 GO SUB 1050
 435 IF LG/LK<10 THEN  GO TO 202
0
 440 IF PZG>1.1 THEN  LET LG=LG+
 440>IF PZG>1.1THEN L
ET LG=LG+INT (RND*10+1):GO SUB 1
060
 445 IF PZG<.9 THEN  LET LG=LG-I
NT (RND*10+1): GO SUB 1060
 450 IF RND*1>.2 THEN  GO TO 465
 455 PRINT AT 19,10; FLASH 1;"RA
DIOAKTYWNE MINERALY....WIELU ZGI
NELO"
 460 LET LG=INT (LG/2): GO SUB 1
060: LET ZN=1
 465 IF RW<150 THEN  GO TO 500
 470 IF ZN=1 THEN  LET ZN=0
 475 PRINT AT 19,10; FLASH 1;"KR
YZYS   NADPRODUKCJI ....WYDOBYCI
E SPADA"
 480 LET RW=INT (RW/2): GO SUB 1
050
 500 LET R=R+1
 505 IF R=100 THEN  GO TO 3000
 510 PAUSE 200
 520 GO TO 200
1000 PAUSE 100
1001 RETURN
1010 PAUSE 50: BEEP .2,20: PRINT
 AT 7,23; INVERSE 1;SK;" $ ": PA
USE 20: PRINT AT 7,23; INVERSE 0
;SK;" $ ": RETURN
1020 PAUSE 50: BEEP .2,20: PRINT
 AT 9,23; INVERSE 1;LK;" ": PAUS
E 20: PRINT AT 9,23; INVERSE 0;L
K;" ": RETURN
1030 PAUSE 50: BEEP .2,20: PRINT
 AT 11,23; INVERSE 1;Z;" TON ":
PAUSE 20: PRINT AT 11,23; INVERS
E 0;Z;" TON ": RETURN
1040 PAUSE 50: BEEP .2,20: PRINT
 AT 12,23; INVERSE 1;PZG;"  ": P
AUSE 20: PRINT AT 12,23; INVERSE
 0;PZG;"  ": RETURN
1050 PAUSE 50: BEEP .2,20: PRINT
 AT 10,23; INVERSE 1;RW;"TON ":
PAUSE 20: PRINT AT 10,23; INVERS
E 0;RW;" TON ": RETURN
1060 PAUSE 50: BEEP .2,20: PRINT
 AT 8,23; INVERSE 1;LG;"  ": PAU
SE 20: PRINT AT 8,23; INVERSE 0;
LG;"  ": RETURN
2000 PRINT AT 19,10; FLASH 1;"NI
E MASZ  KOPALNI.....KONIEC GRY":
 GO TO 3000
2010 PRINT AT 19,10; FLASH 1;"ST
RAJKI  GORNIKOW.....KONIEC GRY":
 GO TO 3000
2020 PRINT AT 19,10; FLASH 1;"ZA
  MALO GOR-NIKOW....KONIEC GRY":
 GO TO 3000
3000 PAUSE 300: INK 2: PAPER 6:
BORDER 6: CLS
3010 PRINT AT 9,10; FLASH 1;" KO
NIEC GRY!! "
3020 PRINT AT 11,1;"BYLES MAKLER
EM ";R;" LAT. "
3025 PRINT AT 12,0;"SKONCZYLES Z
 KASA ";SK;" $"
3030 PRINT AT 13,0;"CHCESZ GRAC
DALEJ ? (T/N) "
3040 INPUT R$
3050 IF R$="N" OR R$="n" THEN  S
TOP
3060 IF R$="T" OR R$="t" THEN  R
UN 100
3070 GO TO 3040
```

```
Ż Y C I E  M A K L E R A

JESTES WLASCICIELEM KOPALNI.
WYKORZYSTUJAC  OPERACJE NA GIEL-
DZIE SPROBUJ ZRRZADZAC  JAK NAJ-
DLUZEJ POSIADANYM MAJATKIEM.

NIE MOZE PRACOWAC MNIEJ NIZ
10 GORNIKOW W JEDNEJ KOPALNI.
POZIOM  ZYCIA  GORNIKOW OKRESLA
WYDAJNOSC ICH PRACY I WIELKOSC
ZATRUDNIENIA.
```

```
ROK 1

STAN KONTA.....  1922 $
LICZBA GORNIKOW.....  62
LICZBA KOPALNI.....  5
ROCZNE WYDOBYCIE.....  111 TON
ZAPASY.....  555 TON
POZIOM ZYCIA GORNIKOW  1

CENA TONY RUDY.....  10 $
CENA KOPALNI.....  2378 $
CENA ZYWNOSCI GORNIKA  88 $
```

KAREN — to właśnie tu można naprawić każdy typ Atari. Tu trafiają uszkodzone komputery z całej Polski, bowiem właśnie KAREN prowadzi naprawy gwarancyjne tych mikrokomputerów (ale najczęściej psują się wrażliwe magnetofony). Ludzie z tej firmy dbają jednak nie tylko o zepsuty sprzęt, ale zabiegają, aby był przede wszystkim dobrze eksploatowany.

1040STF Personal Computer



To jeden z nowszych modeli Atari. Podstawą tej konstrukcji jest mikroprocesor Motorola 68 000. Pamięć RAM ma 1024 Kb, ROM 192 Kb. Wśród wielu zalet Atari wymienia się znakomitą grafikę. Atari 1040 STF dysponuje 512 kolorami. Duża rozdzielczość to 640×400 punktów na monochromatycznym monitorze; średnia rozdzielczość to punktów w czterech kolorach.