P. 1877/85

mikroKLAN 19

7 1985

informatyka

PRACOWNIA KLASOWA LOGO

Język LOGO

Kwalifikacja prawna programu komputerowego

Informatyka w szkołach średnich

Nowości w brytyjskiej mikroinformatyce

WYDAWNICTWO NAUKOWE PWN

Cena 100 zł

informatyka

7 1985

Nr 7

Miesięcznik Rok XX

Lipiec 1985

P.1877/85

POLITECHNIKA ŚLĄSKA Biblioteka Główna w Gliwicach

Organ Komitetu Informatyki MNSZWiT oraz Komitetu Naukowo-Technicznego NOT ds. Informatyki

KOLEGIUM REDAKCYJNE:

Dr inż. Wacław ISZKOWSKI, mgr Teresa JABŁOŃSKA (sekretarz redakcji), Władysław KLEPACZ (zastępca redaktora naczelnego), prof. dr hab. Leon ŁUKASZEWICZ (redaktor naczelny), mgr inż. Andrzej J. PIOTROWSKI, dr inż. Janusz ZALEWSKI

STALE WSPÓŁPRACUJĄ:

Mgr inż. Witold ABRAMOWICZ (Szwajcaria), mgr inż. Ryszard K. KOTT (Wielka Brytania), mgr inż. Teresa WILCZEK

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Tadeusz PECHE

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pok. 243 i 244, tel. 27-71-40 lub 26-82-61 w. 184

Zakł. Graf. „Tamka". Zam. 0573-1300/85. Obj. 4,0 ark. druk. Nakład 6300 egz. N-9.

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza 100 zł
Prenumerata roczna 1200 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA CZASOPISM I KSIĄŻEK TECHNICZNYCH

SIGMA

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

W NASTĘPNYCH NUMERACH:

- Wacław Iszkowski i Paweł Grzegorzewicz o systemie generowania systemów operacyjnych LOGO
- Peter Broczko o produkcji mikrokomputerów w krajach RWPG
- Cezary Stępień i Jarosław Szyrkowiec o wielopoziomowym systemie oprogramowania pamięci kasetowych PK-1
- Eugeniusz Bilski, Zbigniew Fryźlewicz, Zbigniew Huzar, Andrzej Kaliś o rozległych sieciach komputerowych
- Bolesław Mikołajczak i Janusz Stokłosa o złożoności algorytmów
- Adam Gacek, Stanisław Kozielski, Piotr Stróżyna o elastycznie mikroprogramowanym komputerze dydaktycznym EW

STANISŁAW WALIGÓRSKI
Instytut Informatyki
Uniwersytet Warszawski

LOGO (1)

Grafika żółwia

Biblioteka Główna Politechniki Śląskiej

W związku z naciskami na przedstawienie języka LOGO, odstępujemy od planowanej prezentacji UNIXA, którą przesuwamy do następnego rocznika INFORMATYKI lub do mikroKLANU.

W najbliższych numerach Czytelnicy otrzymują więc trzyodcinkowy artykuł na temat języka zdobywającego w Polsce zadziwiającą popularność. Może dlatego, że jest lansowany przez specjalistów, którzy proponują nawet wprowadzenie go do szkolnego programu informatyki. Na rynku mikrokomputerowym LOGO jest głównym przedstawicielem języków nie wymuszających na użytkownikach dostosowania sposobu myślenia do struktur programowych. Czy znaczy to, że zajmie miejsce BASICA? Na pewno nie, ale może nauczyć przyszłych adeptów trudnej sztuki programowania — racjonalnego podejścia do tych zagadnień, uchronić profesjonalistów przed zbyt wczesnym zawodowym „skrzywieniem", a programistów-amatorów — przed bezproduktywnym zmaganiem się z językami nieprzystosowanymi do programowania! Zapraszamy do spotkania z żółwiem Papperta. (Red.)

Pierwsza wersja języka LOGO powstała w końcu lat sześćdziesiątych w ramach eksperymentów, mających na celu sprawdzenie użyteczności programowania w uczeniu dzieci. Jej autorami byli W. Feurzeig, D. Bobrow, S. Papert i C. Solomon. Była ona później udoskonalona w Artificial Intelligence Laboratory w MIT (Massachussetts Institute of Technology) w grupie kierowanej przez S. Paperta, którego zasługą było także sformułowanie i spopularyzowanie dydaktycznych zasad uczenia dzieci przy użyciu LOGO. Jego idee zostały wyłożone m.in. w programowej książce [1].

Do sukcesu LOGO jako języka programowania dla początkowego nauczania przyczyniło się dobre przystosowanie języka do psychologii dzieci oraz bardzo dobrze pomyślana grafika. Eksperymenty dydaktyczne w szkołach prowadzono od początku istnienia LOGO, ale dopóki był to język dostępny na dużych komputerach, jego zasięg był ograniczony. Rzeczywiście powszechne stosowanie LOGO stało się możliwe dzięki mikrokomputerom — ich taniość i możliwość stosowania w szkołach na skalę masową usuwa techniczne przeszkody na drodze do stworzenia dzieciom możliwości aktywnego uczenia się za pomocą komputera.

Większość wersji LOGO, rozpowszechnianych obecnie przez firmy amerykańskie, pochodzi od wersji MIT i w wielu przypadkach aktywny udział w ich opracowaniu brali członkowie grupy LOGO z MIT. Są to m.in. trzy wersje LOGO na APPLE II i APPLE II plus, rozpowszechniane przez firmy APPLE COMPUTER INC., TERRAPIN INC. i KRELL SOFTWARE CO., oraz znacznie różniące się od nich TI-LOGO — na TI-99/4 i TI-99/4a z TEXAS INSTRUMENTS, a także TRS-80 COLOR LOGO, rozpowszechniane przez RADIO SHACK EDUCATION DIVISION z TANDY CORP. W Europie pierwszym ośrodkiem, zajmującym się aktywnie tym językiem był Department of Artificial Intelligence w University of Edinburgh, w którym działała grupa pod kierunkiem J. Howe. Od wersji LOGO uniwersytetu w Edynburgu pochodzą niektóre brytyjskie wersje handlowe tego języka, jak na przykład RESEARCH MACHINES LOGO, dostępne na mikrokomputerach tej firmy.

Szczegóły o różnych wersjach LOGO można znaleźć w [2] i [3]. W podanych niżej przykładach została użyta wersja TERRAPIN LOGO, chociaż większość występujących w nich komend [1]) jest wspólna dla prawie wszystkich wariantów tego języka.

WŁAŚCIWOŚCI JĘZYKA

LOGO kojarzy się przede wszystkim z tak zwaną „żółwią grafiką" (ang. turtle graphics). Być może dzieje się tak dlatego, że zwykle każdy elementarny wykład LOGO zaczyna się właśnie od niej i rzeczywiście przy jej użyciu można łatwo pokazać różne charakterystyczne cechy i możliwości języka, mimo że nie ograniczają się one tylko do grafiki.

Język LOGO został stworzony przede wszystkim dla dzieci i tak opracowany, aby ich uwaga mogła koncentrować się cały czas na rozwiązywaniu problemów, na tym, co ma zrobić komputer, a nie — na technicznych problemach programowania; aby wynik czynności wykonywanych przez dziecko przy komputerze był widoczny możliwie natychmiast, a w każdym razie jak najszybciej; aby wykonanie raz zaprogramowanych złożonych akcji komputera było równie łatwe, jak wykonywanie akcji najbardziej elementarnych; żeby można było z opracowanych już procedur tworzyć inne, w każdym momencie, kiedy zechce się to zrobić, łatwo i bez absorbowania technicznymi szczegółami tego procesu; żeby tak stworzone nowe procedury nadawały się do użycia praktycznie natychmiast, niezależnie od tego, jak bardzo są skomplikowane; żeby można łatwo tworzyć efektowne rysunki możliwie najprostszymi środkami.

Do uzyskania takich właściwości języka przyczyniły się zasadniczo dwa czynniki: niezależne definiowanie i przechowywanie procedur oraz żółwia grafika.

LOGO jest językiem proceduralnym, w którym programista może definiować procedury w każdym momencie pracy. Procedury mogą być wywoływane przy użyciu komend języka, przez inne procedury, albo rekurencyjnie. Nie jest potrzebny żaden program główny ani uprzednie deklarowanie procedur lub zmiennych. Nowym definicjom nie musi towarzyszyć żadna translacja ani ponowne łączenie lub kojarzenie rzeczy już dawniej zdefiniowanych.

Istotą żółwiej grafiki jest użycie pełzającego po ekranie wskaźnika, zwanego tu żółwiem (ze względu na dzieci, pierwotnych adresatów tej metody), którym można sterować z klawiatury przy użyciu prostych komend i w ten sposób rysować obrazy na ekranie.

Żółw ma w każdym momencie określone dwie współrzędne położenia na ekranie oraz kąt, określający kierunek, w jakim jest ustawiony i może się poruszać. Zwykle przedstawia się go na ekranie w formie małego trójkąta równobocznego, tak że zarówno jego położenie jak i kierunek są dobrze widoczne. Odpowiednikiem żółwia może być pisak urządzenia rysującego (plotera) albo sprzężony z komputerem żółw-robot, czyli małe urządzenie na kółkach (którego kształt może być tak dobrany, aby rzeczywiście przypominał żółwia), wyposażone w pisak, który może się pod-

Dr hab. inż. STANISŁAW WALIGÓRSKI jest docentem w Instytucie Informatyki Uniwersytetu Warszawskiego. Z informatyką jest związany od 1959 roku, początkowo jako projektant układów cyfrowych i architektury komputerów. Od 1964 r. zajmuje się metodami i językami programowania do przetwarzania danych tekstowych oraz metodami programowania konwersacyjnej pracy z komputerem.

[1]) W LOGO mamy do czynienia z dwoma rodzajami opisu czynności: operacja (ang. operation), której celem jest wykonanie czynności dla niej właściwej, zakończone przekazaniem pewnej wartości, oraz komenda (rozkaz, ang. command), która jest związana wyłącznie z wykonaniem pewnej czynności. Przykładem operacji jest opisana niżej XCOR, albo operacja arytmetyczna +, —, *, czy /. Słowo „komenda" wybrałem dla tego opisu dlatego, że bardziej niż „rozkaz" kojarzy się z natychmiastowym wykonaniem, a LOGO jest językiem konwersacyjnym. Poza tym „rozkaz" bywał używany jako odpowiednik angielskiego „instruction" używanego przede wszystkim w językach o innej strukturze, a więc budzącego inne skojarzenia. Nie jest to jednak wzgląd tak ważny, by „rozkaz" całkowicie odrzucić. Natomiast „zdanie" jako odpowiednik angielskiego „sentence" jest użyte w LOGO w innym znaczeniu i nie można go już użyć jako terminu określającego tylko formę opisu czynności.

nosić lub opuszczać. Pisak może rysować na dużym arkuszu papieru, powtarzając ruchy i czynności żółwia-wskaźnika z ekranu i przenosząc obraz na papier w powiększonej skali.

CZYNNOŚCI ELEMENTARNE

Początek układu współrzędnych jest w środku ekranu, przy czym współrzędne poziome x zwiększają się od strony lewej do prawej, a współrzędne pionowe y — z dołu do góry. Zakresy współrzędnych zależą od rozdzielczości ekranu. Przykładowo, dla APPLE II ekran ma szerokość 280, a wysokość 240 jednostek (kroków żółwia). Kąt określający kierunek ustawienia żółwia jest odmierzany w stopniach, od kierunku pionowo w górę, w prawo, to znawięc żółw skierowany pionowo w górę wyznacza kąt 0°, czy zgodnie z kierunkiem obrotu wskazówek zegara. Tak skierowany poziomo w prawo — 90° itd.

Przy użyciu komend języka LOGO można przesuwać żółwia do przodu lub do tyłu na zadaną odległość (liczbę kroków żółwia), albo obracać go w miejscu w prawo lub w lewo o zadaną liczbę stopni. Można przesuwać żółwia do dowolnego miejsca ekranu o zadanych współrzędnych, nie zmieniając jego kierunku, albo skierować go pod dowolnym zadanym kątem, nie zmieniając jego położenia. Można wykonywać dowolne sekwencje takich operacji.

Żółw trzyma pióro, które może podnieść, aby przesuwanie nie pozostawiało śladu, albo — opuścić i wtedy w czasie ruchu żółwia pióro wlecze się po ekranie (lub papierze), pozostawiając ślad. Jeśli jest to ekran kolorowy, można określić kolor rysowanej kreski, do czego służy specjalna komenda. Zmieniając kolor pióra w czasie rysowania, można otrzymać obraz o dowolnie wybranych barwach. Żółw może również zacierać już narysowane kreski, rysując na nich następne w kolorze tła bądź też — przez ponowne przejście po tych kreskach po ustawieniu pióra w stan zacierania specjalną komendą. Przydaje się to do usuwania pomyłek w rysunkach oraz przy tworzeniu rysunków animowanych.

Obraz żółwia można w razie potrzeby ukryć, usuwając go z ekranu, albo uczynić go znowu widocznym. Operacje te nie wpływają na położenie ani na kierunek ruchu żółwia. Skutki komend powodujących przesuwanie lub obracanie żółwia są w obu przypadkach takie same, niezależnie od tego, czy wizerunek żółwia jest widoczny na ekranie czy ukryty.

Ta metoda tworzenia elementarnych rysunków okazała się tak dobra, że obecnie wiele języków, zwłaszcza dla mikrokomputerów, ma już wbudowaną żółwią grafikę jako element języka. Takie wersje żółwiej grafiki mogą mieć jednak mniejsze możliwości niż w LOGO, ze względu na ograniczenia narzucane przez strukturę tych języków.

Metody grafiki komputerowej wykraczają oczywiście poza ramy tego artykułu, wobec czego przedstawię tylko przykłady najprostsze, które umożliwiają pokazanie w sposób elementarny charakterystycznych cech języka LOGO, idei posługiwania się tym językiem i sposobów korzystania z jego możliwości, zwłaszcza przy tworzeniu prostych rysunków za pomocą żółwia. Oczywiście możliwości języka nie są ograniczone tylko do takich elementarnych i błahych zastosowań.

KOMENDY GRAFICZNE

Poniżej przedstawiono wykaz komend umożliwiających rysowanie, ograniczony do najważniejszych i najczęściej stosowanych, które jednak wystarczają do pokazania zasad posługiwania się językiem.

DRAW — oczyść ekran, ustaw żółwia w środku ekranu, skieruj go pionowo w górę i uczyń widocznym.

NODRAW — skasuj rysunki na ekranie, przeznaczając go wyłącznie do pokazywania tekstów.

Wszystkie komendy graficzne mogą być wydawane dopiero po właściwym użyciu komendy DRAW, która przełącza ekran na graficzny tryb pracy z żółwiem.

SPLITSCREEN — pozostaw na dole ekranu wydzielone wiersze, w których będzie ukazywać się komenda, wprowadzana z klawiatury.

FULLSCREEN — przeznacz cały ekran na rysunek, bez możliwości powtarzania na nim komend z klawiatury.

CLEARSCREEN — usuń rysunek z ekranu, nie zmieniając położenia ani kierunku żółwia.

HIDETURTLE — uczyń żółwia niewidocznym, nie zmieniając jego położenia ani kierunku.

SHOWTURTLE — pokaż żółwia, czyniąc go odtąd widocznym na ekranie.

PENDOWN — opuść pióro, umożliwiając rysowanie.

PENUP — podnieś pióro, umożliwiając ruch żółwia bez rysowania.

PENCOLOR — ustal podany kolor jako kolor kreski, rysowanej w położeniu PENDOWN.

BACKGROUND — ustal podany kolor jako kolor tła.

Często do oznaczania kolorów używa się liczb naturalnych z pewnego zakresu, począwszy od 0. Do kasowania kresek może służyć specjalny „kolor" o określonym numerze, np. najwyższym możliwym, lub komenda PENERASE, ustawiająca pióro w stan kasowania. W tym drugim przypadku użycie komendy PENCOLOR sprowadza pióro ponownie do stanu rysowania.

HOME — przesuń żółwia do położenia w środku ekranu.

FORWARD — przesuń żółwia do przodu o podaną odległość.

BACK — przesuń żółwia do tyłu o podaną odległość, nie zmieniając jego kierunku.

RIGHT — obróć żółwia w miejscu w prawo o podaną liczbę stopni.

LEFT — obróć żółwia w miejscu w lewo o podaną liczbę stopni.

SETHEADING — obróć żółwia w miejscu tak, aby ustawił się pod podanym kątem.

SETXY — ustaw żółwia w punkcie o podanych współrzędnych x i y, nie zmieniając jego kierunku.

SETX — zmień współrzędną x żółwia na podaną, przesuwając go poziomo do tego miejsca i nie zmieniając jego współrzędnej y ani kierunku.

SETY — zmień współrzędną y żółwia na podaną, przesuwając go pionowo do tego miejsca i nie zmieniając jego współrzędnej x ani kierunku.

Wszystkie wymienione komendy powodują wykonanie pewnych czynności, jak zmiana stanu lub położenia żółwia. Poniżej przejdziemy do operacji, określających elementy położenia żółwia. Wartość wyznaczanego parametru położenia, podana w opisie, staje się wartością opisywanej procedury.

XCOR — podaj współrzędną poziomą x żółwia.

YCOR — podaj pionową współrzędną y żółwia.

HEADING — podaj kąt określający kierunek żółwia.

TOWARDS — podaj kąt, pod jakim byłby ustawiony żółw, gdyby był skierowany na punkt o podanych współrzędnych x i y.

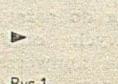
Reakcje żółwia na dojście do brzegu ekranu są określone przez komendy wyboru trybu pracy:

FENCE — każda próba wyjścia poza brzeg ekranu jest nieskuteczna i sygnalizowana jako błąd odpowiednim komunikatem, pojawiającym się na ekranie

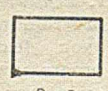
WRAP — przejście żółwia poza brzeg ekranu powoduje pojawienie się go po przeciwnej stronie, tak jak gdyby przeciwlegle krawędzie ekranu — lewa i prawa oraz górna i dolna — były parami sklejone ze sobą.

PIERWSZE KROKI

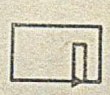
Załóżmy, że żółw jest w środku czystego ekranu, skierowany pionowo w górę i widoczny, a pióro jest opuszczone. Jeśli jest to ekran kolorowy, to kolor rysowanej kreski i kolor tła są określone i różne. Szerokość ekranu jest równa 280 jednostek.



Rys.1



Rys.2



Rys.3

Pisząc

```
RIGHT 90
```

powodujemy obrócenie żółwia w prawo bez zmiany jego położenia (rys. 1).
Komenda

```
FORWARD 50
```

spowoduje narysowanie poziomego odcinka o podanej długości.
Pisząc dalej:

```
LEFT 90
FORWARD 25
LEFT 90
FORWARD 50
LEFT 90
FORWARD 25
LEFT 90
```

poprowadzimy żółwia tak, aby narysował prostokąt i wrócił do położenia początkowego. Ostatni obrót w lewo nastąpił po to, by żółw znów był skierowany poziomo w prawo (rys. 2).

Teraz możemy podnieść pióro i przesunąć żółwia nieco do przodu:

```
PENUP
FORWARD 35
PENDOWN
```

po czym można znów narysować prostokąt o nieco innych wymiarach. Zauważmy jednak, że w sekwencji rysującej poprzedni prostokąt ostatnie cztery komendy były powtórzeniem poprzednich. Aby zapisać to krócej, użyjemy komendy REPEAT, powodującej powtórzenie zadaną liczbę razy sekwencji komend ujętej w nawiasy kwadratowe:

```
REPEAT 2 [FORWARD 7 LEFT 90 FORWARD 17 LEFT 90]
```

Po wykonaniu tej komendy na ekranie pojawia się nowy prostokąt, a żółw jest w jego dolnym rogu, skierowany poziomo w prawo (rys. 3).

PROCEDURY

Jeśli rysowanie prostokątów miałoby się powtarzać wielokrotnie, to można stworzyć odpowiednią procedurę, dodając na początku wykonującej to sekwencji komend zwrot TO z nazwą, jaką chcemy nadać tej procedurze, a na końcu pisząc END. Jeśli ponadto ten prostokąt ma mieć wymiary dobierane stosownie do potrzeb, to można wprowadzić zamiast konkretnych liczb odpowiednie parametry. W celu odróżnienia parametru od nazwy procedury umieszcza się przed nim dwukropek. Piszemy więc

```
TO PROSTOKĄT :SZEROKOŚĆ :WYSOKOŚĆ
REPEAT 2 [FORWARD :SZEROKOŚĆ LEFT 90 FORWARD :WYSOKOŚĆ LEFT 90]
END
```

Po wprowadzeniu słowa END nazwa procedury i jej znaczenie zostają zapamiętane i odtąd można posługiwać się tą nazwą tak samo, jak komendami języka. Można więc użyć jej jako komendy z odpowiednimi parametrami, aby spowodować wykonanie określonego przez nią działania, albo — wykorzystać w definicji nowej procedury.

Aby ułatwić rysowanie prostokątów w różnych miejscach ekranu, utworzymy jeszcze dodatkową procedurę

```
TO PRZESUŃ :POZIOMO :PIONOWO
PENUP
SETXY XCOR + :POZIOMO YCOR + :PIONOWO
PENDOWN
END
```

W czasie wprowadzania tych definicji rysunek pozostaje niezmieniony. Teraz można przesunąć żółwia i narysować nowy prostokąt:

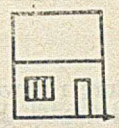
```
PRZESUŃ -35 25
PROSTOKĄT 50 22
```

Poniżej przedstawiono przykład użycia procedury już zdefiniowanej w definicji nowej procedury

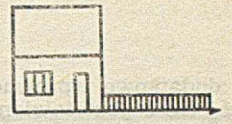
```
TO OKNO
REPEAT 3 [PROSTOKĄT 5 10 FORWARD 5]
END
```

Umieszczamy okno we właściwym miejscu i doprowadzamy żółwia do prawego dolnego rogu budynku:

```
PRZESUŃ 7 -17
OKNO
PRZESUŃ 27 -7
```



Rys.4



Rys.5

PROCEDURY REKURENCYJNE

Płot możemy potraktować jako ciąg określonej liczby sztachet.

```
TO SZTACHETA
FORWARD 1
PROSTOKĄT 2 7
FORWARD 2
END
```

Rysowanie płotu wykorzystamy jako okazję do wprowadzania przykładu procedury rekurencyjnej.

```
TO PLOT :ILESZTACHET
IF :ILESZTACHET=0 THEN STOP
SZTACHETA
PŁOT :ILESZTACHET — 1
END
```

Komenda STOP powoduje zakończenie wykonywania procedury. Jeśli więc liczba sztachet jest dodatnia, to procedura rysuje jedną sztachetę i obok niej płot krótszy o tę sztachetę. Gdy liczba sztachet dojdzie do zera, kończy się działanie procedury.

W czasie pisania tych definicji żółw pozostaje na miejscu. Teraz można dorysować obok domu fragment płotu (rys. 5)

```
PLOT 20
```

Wreszcie można zbudować blok o zadanej liczbie pięter oraz okien na każdym piętrze. Jeżeli liczba pięter jest równa zeru, będzie to budynek parterowy, ale liczba okien musi być zawsze dodatnia. Wyrysowywanie okien w fasadzie można też zaprogramować rekurencyjnie, ale tym razem użyjemy komendy REPEAT. Pełna procedura będzie okazją do użycia komendy MAKE, nadającej wartości zmiennym (w tym przypadku oznaczającym wysokość i szerokość rysowanego budynku), oraz komendy warunkowej o konstrukcji IF...THEN...ELSE, która w tym przypadku uzależni kształt dachu od wysokości domu.

KOMENDA MAKE I DZIAŁANIA ARYTMETYCZNE

Komenda MAKE nadaje zmiennej, która jest jej pierwszym parametrem, wartość równą wartości jej drugiego parametru. Cudzysłów poprzedzający nazwę parametru oznacza, że w tym przypadku chodzi o nazwę parametru jako taką, a nie o jego wartość. Rzeczywiście, jeżeli zmiennej nadaje się nową wartość, to trzeba wiedzieć, jak ta zmienna się nazywa, natomiast jej poprzednia wartość nie jest potrzebna. Drugi parametr komendy MAKE może mieć dwukropek, gdyż określa wartość nadawaną.

```
TO DOM :ILEPIĘTER :ILEOKIEN
MAKE "WYSOKOŚĆ (:ILEPIĘTER + 1) * 22 + 12
MAKE "SZEROKOŚĆ :ILEOKIEN * 25 + 10
PROSTOKĄT :SZEROKOŚĆ :WYSOKOŚĆ
REPEAT (:ILEPIĘTER + 1) [PIĘTRO :ILEOKIEN :SZEROKOŚĆ]
PRZESUŃ 0 12
IF :WYSOKOŚĆ<75 THEN DACH :SZEROKOŚĆ ELSE
    PRZESUŃ :SZEROKOŚĆ 0
PRZESUŃ 0 —:WYSOKOŚĆ
END

TO PIĘTRO :ILEOKIEN :SZEROKOŚĆ
PRZESUŃ 10 12
```

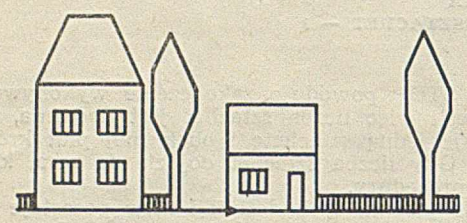
```
OKNO
REPEAT (:ILEOKIEN — 1) [PRZESUŃ 10 0 OKNO]
PRZESUŃ (10 — :SZEROKOŚĆ) 10
END

TO DACH :SZEROKOŚĆ
LEFT 60
FORWARD 35
RIGHT 60
FORWARD :SZEROKOŚĆ — 35
RIGHT 60
FORWARD 35
LEFT 60
END
```

Można dodatkowo wprowadzić procedurę rysowania drzewa, której treść pozostawiamy inwencji Czytelnika. Trzeba przy tym pamiętać, aby po zakończeniu procedury żółw był gotów do rysowania następnego elementu obrazu, bez straty czasu na specjalne ustawianie go we właściwej pozycji.

```
TO DRZEWO
...
END
```

W trakcie wprowadzania powyższych definicji rysunek się nie zmienia, a żółw pozostaje pod płotem, ostatnio narysowanym elementem obrazu, skierowany poziomo w prawo, gotowy do rozpoczęcia rysowania następnego elementu rysunku. Może to być na przykład dom, drzewo lub płot, ale ponieważ żółw jest już blisko prawej krawędzi ekranu, trzeba uważać, aby jej nie przekroczyć, albo zdecydować z góry, co w takim przypadku ma nastąpić. Wybierzmy tryb pracy, w którym żółw przechodzi cyklicznie z prawego brzegu ekranu na lewy i rysujmy dalej (rys. 6).



Rys.6

```
WRAP
DRZEWO
PŁOT 12
DOM 1 2
```

```
PŁOT 5
DRZEWO
SETX 0
```

Ten nieskomplikowany obrazek z domami, drzewami, płotami i ulicą jest prostym przykładem tego, co można uzyskać przy użyciu środków języka LOGO, przedstawionych zresztą tylko częściowo i w dużym uproszczeniu. Zainteresowany Czytelnik sięgnie zapewne po odpowiednie podręczniki lub poszuka komputera z translatorem tego języka.

* * *

Przeglądowy charakter tego artykułu nie pozwala zagłębiać się zbytnio w szczegóły, ale już takie proste przykłady pozwalają poznać sposoby posługiwania się żółwią grafiką, swobodnym definiowaniem procedur w dowolnym momencie rozmowy z komputerem, swobodnym składaniem procedur i rekurencją. Wszystko to jest w LOGO wprowadzone w sposób prosty i łatwo zrozumiały dla każdego, można powiedzieć — naturalny. Na to, aby wszystko mogło w ten sposób działać, trzeba jednak użyć dość skomplikowanego aparatu, który, choć ukryty przed okiem początkującego lub nawet przeciętnego programisty LOGO, jest tak zbudowany, by uwolnić go od troski o techniczne szczegóły tego procesu. W skład tego aparatu wchodzą między innymi: dynamiczna gospodarka pamięcią bez stałych rezerwacji miejsc, tworzenie, przechowywanie i likwidowanie struktur danych bez potrzeby angażowania programisty, analizowanie zadań języka i interpretowania ich z prawidłowym wykorzystaniem niezbędnych i zapamiętanych wcześniej definicji, właściwe zapamiętywanie, przechowywanie i przekazywanie wartości parametrów procedur wywoływanych wprost przez inne procedury lub rekurencyjnie, kontrola poprawności użytych w rozmowie zwrotów języka LOGO i sygnalizacja błędów. Część używanych do tego środków i narzędzi, jak na przykład działania na strukturach listowych i słowach, jest jawnie dostępna dla programisty LOGO dzięki istnieniu odpowiednich komend języka.

LITERATURA

[1] Papert S.: Mindstorms: children, computers, and powerful ideas. Basic Books, 1980

[2] Ross P.: Introducing Logo for the Apple II computer, Texas Instruments 99/4A and Tandy Color Computer. Addison Wesley, 1983

[3] Williams G.: Logo for the Apple II, the TI-99/4A, and the TRS-80 Color Computer. Byte, vol. 7, No. 8, August, 1982, pp. 230—290.

Centrum Szkolenia Informatycznego ZETO w Łodzi organizuje w terminie 9—11.12.1985 r. wspólnie z Sekcją Systemów Mikroprocesorowych Polskiego Towarzystwa Informatycznego spotkanie nt.:

II SZKOŁA: „Mikrokomputery 16-bitowe"

Szkoła ta jest kontynuacją Pierwszej Szkoły, która była również poświęcona architekturze systemów 16-bitowych.

Głównym celem Szkoły jest zapoznanie uczestników ze specyficzną problematyką programowania systemów mikroprocesorowych 16-bitowych na rozszerzonym poziomie języków asemblerowych rodziny INTEL MCS 86.

Tematyka Szkoły obejmuje:

- tryby adresowania w mikroprocesorze 8086
- listę rozkazów 8086/8087/8089
- programowanie w asemblerze 8086
- asemblery 8087 oraz 8089
- programowanie w językach PL/M-86 oraz C.

Szkoła adresowana jest przede wszystkim do osób, które zajmować się będą projektowaniem oprogramowania dla mikrokomputerów 16-bitowych, głównie w zakresie oprogramowania podstawowego oraz oprogramowania użytkowego dla systemów sterowania. Szkoła ma charakter informacyjny.

W sprawach dotyczących programu szkoły informacji udziela sekretarz naukowy Szkoły, doc. dr hab. inż. Wojciech Cellary (przewodniczący Sekcji Systemów Mikroprocesorowych PTI) lub mgr inż. Jerzy Kręglewski (przewodniczący Klubu Mikrokomputerów 16-bitowych Koła PTI w Poznaniu). Wspólny adres:

Instytut Automatyki Politechniki Poznańskiej,
ul. Piotrowo 3a, 60-965 Poznań, tel. 78-23-70.

Wszelkich informacji udziela również **sekretariat CSI ZETO Łódź — telefony: 36-47-70, 32-50-70 lub telex 88-52-08.**

Warunkiem uczestnictwa w sympozjum jest przesłanie pod adresem: „CSI ZETO-Łódź, ul. Hutora 69, 90-558 Łódź", imiennego zgłoszenia zawierającego oświadczenie o wpłaceniu na konto: „ZETO-Łódź, ul. Narutowicza 136 w NBP I OM Łódź nr 47018-2219", kwoty 4 tys. zł od osoby (bez noclegów i wyżywienia). Uczestnikom, którzy zgłoszą potrzebę rezerwacji noclegów, zapewniamy zakwaterowanie w hotelach miejskich. Ostateczny termin przyjmowania zgłoszeń upływa 15 listopada 1985 r.

JACEK IRLIK
Centrum Techniki Obliczeniowej
Uniwersytet Śląski
Katowice

Kwalifikacja prawna programu komputerowego

Spośród kwestii prawnych związanych z informatyką bodaj najbardziej poruszającą jej środowisko zawodowe jest kwestia ochrony, której obowiązujące prawo autorskie udziela twórcom programów komputerowych. W obowiązującym w Polsce systemie prawa, jedynym właściwie punktem odniesienia w rozważaniach dotyczących tej kwestii jest, niejednolite zresztą, stanowisko doktryny, ponieważ jak dotąd, brak w praktyce sądowej rozstrzygnięć dotyczących tej kwestii (por. [3] s. 60, [7], [10]). W większości prac poświęconych temu zagadnieniu autorzy zwracają uwagę na trudności, które napotyka próba zakwalifikowania programu komputerowego w układzie pojęć występujących w przepisach z zakresu ochrony praw na dobrach niematerialnych, czyli tzw. własności intelektualnej. Należy jednak zaznaczyć, że wyrażając swoje poglądy w tej kwestii autorzy wspomnianych prac nie bardzo mogli oprzeć się o jakąkolwiek bardziej gruntowną analizę pojęcia programu, w odróżnieniu na przykład od autorów zajmujących stanowisko w kwestiach ochrony dzieł literackich, muzycznych, plastycznych itp., mogących odwołać się do ustaleń teorii dzieła literackiego lub teorii sztuki.

W literaturze informatycznej poświęconej teorii programowania można co prawda znaleźć wiele rozmaicie formułowanych definicji programu, algorytmu i innych pojęć związanych z programowaniem. Są to jednak przede wszystkim definicje dość hermetyczne, adresowane do informatyków i związane z rozwojem ich wiedzy zawodowej, którę są mało czytelne dla niefachowca, a więc, mało przydatne przy dokonywaniu kwalifikacji prawnej. Dokonanie bowiem takiej kwalifikacji wymaga takiego zrozumienia treści wymienionego pojęcia, które pozwoliłoby na zaliczenie programu do którejś z szerszych kategorii obejmujących niematerialne wytwory ludzkiego umysłu.

Niniejszy tekst zawiera próbę zajęcia stanowiska w kwestii istoty pojęcia „program komputerowy" (a także — pojęć z nim związanych), pozwalającego na dokonanie takiej kwalifikacji. Wstępem do rozważań prowadzących do tego celu będzie ogólna charakterystyka procesu opracowywania systemu oprogramowania. Charakterystyka ta pozwoli na wskazanie konstruowanych w toku tego procesu wytworów intelektualnych oraz na określenie ich wzajemnych uwarunkowań. Pozwoli ona więc na ustalenie zakresu pojęć używanych w dalszych dociekaniach.

Charakteryzując proces tworzenia systemu oprogramowania i przedstawiając konstrukcje wieńczące poszczególne fazy tego procesu należy z góry zastrzec, że nie w każdym przypadku wszystkie one dają się wydzielić w sposób wyraźny. Zależy to od stopnia złożoności struktury problemu, którego rozwiązanie (lub rozwiązywanie) jest zadaniem oprogramowywanego komputera.

Dr JACEK IRLIK ukończył Wydział Fizyki na Uniwersytecie Jagiellońskim w 1963 r. Zajmował się początkowo zastosowaniem maszyn cyfrowych w obliczeniach naukowych, później również zagadnieniami teorii programowania. W roku 1975 obronił w Centrum Obliczeniowym PAN (obecnie IPIPAN) pracę doktorską dotyczącą teorii programowania. Ukończył także studia prawnicze na Uniwersytecie Śląskim. Zajmuje się zagadnieniami prawnymi, związanymi z rozwojem i zastosowaniem informatyki. Jest dyrektorem Centrum Techniki Obliczeniowej Uniwersytetu Śląskiego.

PROCES TWORZENIA OPROGRAMOWANIA

Pierwszą w kolejności konstrukcją intelektualną, wydzielaną w procesie tworzenia systemu oprogramowania jest **model zadania** [9], którego przedmiotem jest problem przewidziany do rozwiązania. Skonstruowanie takiego modelu polega na przyjęciu:

— rodzaju obiektów abstrakcyjnych oraz konkretnego zestawu zmiennych reprezentujących te obiekty, jako matematyczno-logicznej reprezentacji problemu
— formuł wyrażających wzajemne zależności zachodzące pomiędzy tymi obiektami.

Przyjęty model jest abstrakcyjną reprezentacją problemu przewidzianego do rozwiązania przez komputer. Przyjęte w jego ramach rodzaje obiektów oraz konkretny zestaw zmiennych mogą w lepszy lub gorszy sposób odzwierciedlać istotne cechy problemu lub stawiane w jego ramach pytania, a w związku z tym **przyjęty model zadania może podlegać subiektywnej ocenie** zainteresowanego (zainteresowanych) jego rozwiązaniem. Przyjęte w ramach modelu **formuły podlegają obiektywnej weryfikacji empirycznej lub analitycznej** — w zależności od rodzaju problemu.

Przykładowo, w modelu zadania polegającego na obliczeniach statycznych dla projektu budowlanego, przyjmowane obiekty abstrakcyjne powinny reprezentować strukturę budowli, przewidywane obciążenia lub rozkłady sił w poszczególnych punktach struktury. Formuły wyrażające wzajemne zależności pomiędzy tymi obiektami wynikają z empirycznie weryfikowalnej wiedzy w zakresie statyki.

Natomiast w modelu zadania polegającego na wyszukiwaniu informacji przyjęte obiekty powinny reprezentować cały zbiór informacji oraz kryteria, według których dokonywany ma być wybór poszczególnych jego fragmentów. Przyjęte rodzaje obiektów oraz konkretny zestaw zmiennych jest określoną, założoną reprezentacją zbioru informacji, a formuły wyrażające wzajemne zależności pomiędzy obiektami będą podlegać weryfikacji analitycznej ze względu na przyjęte założenia dotyczące reprezentacji.

Można posługiwać się też modelem zadania w przypadkach, w których przedmiotem zadania jest umożliwienie programowania w nowym języku (w stosunku do dotychczas dostępnych dla danego komputera), umożliwienie innej organizacji pracy maszyny itp. Przyjęcie modelu zadania może być więc traktowane jako punkt wyjścia w procesie tworzenia systemu oprogramowania zarówno użytkowego jak i podstawowego.

Opierając się na przyjętym modelu zadania opracowuje się projekt systemu oprogramowania. Istota takiego projektu polega na przyjęciu określonej struktury realizacji zadania przez komputer, a więc na rozłożeniu go na proponowane do bezpośredniej realizacji podzadania i ustaleniu wzajemnych zależności pomiędzy nimi. Skonstruowanie takiego projektu wymaga w szczególności:

- określenia typów danych, reprezentujących obiekty przyjętego modelu, oraz konkretnego zestawu zmiennych reprezentujących wartości tych danych

- ustalenia, dla każdego z programów przewidzianych w celu realizowania poszczególnych podzadań, danych wejściowych i wyjściowych oraz oczekiwanej funkcji
- ustalenia przepływu danych pomiędzy programami oraz pomiędzy systemem a otoczeniem.

Wybrana struktura realizacji zadania może zapewnić większą lub mniejszą wygodę w przyszłym użytkowaniu systemu i w tym aspekcie **skonstruowany projekt może podlegać subiektywnej ocenie** zainteresowanego (zainteresowanych) przyszłym użytkowaniem systemu. Projekt systemu oprogramowania podlega **obiektywnej weryfikacji analitycznej** ze względu na następujące kwestie:

- czy przyjęta w projekcie struktura danych stanowi zupełną i wzajemnie jednoznaczną reprezentację zbioru obiektów przyjętego modelu zadania
- czy sformułowania wyrażające oczekiwane funkcje każdego z programów są właściwą reprezentacją wzajemnych zależności zachodzących pomiędzy reprezentowanymi (przez projektowane dane) obiektami przyjętego modelu
- czy realizacja podzadań wyczerpuje rozwiązanie problemu zgodnie z przyjętym modelem.

ISTOTA PROGRAMU KOMPUTEROWEGO

Mając dla każdego z projektowanych programów określoną funkcję, którą jest oczekiwane przyporządkowanie określonych danych wyjściowych określonym danym wejściowym, zapisuje się **program komputerowy** będący odbiciem ustalonego w tym celu **algorytmu.** Program komputerowy podlega **obiektywnej weryfikacji analitycznej,** odpowiadającej na pytanie, czy zapewnia realizację oczekiwanej funkcji.

Rola algorytmu

Rozważania dotyczące istoty programu komputerowego należy rozpocząć od ustalenia związku, jaki zachodzi pomiędzy programem i algorytmem. Mówiąc najogólniej [4, 6], **algorytmem jest sposób realizacji określonego zadania przez wykonanie skończonej liczby działań prostych.** Kwestią, czy dane **działanie jest proste** należy rozważać ze względu na osobę, która zadanie to ma wykonywać. Przez stwierdzenie, że działanie jest dla danej osoby proste należy rozumieć uznanie, że osoba ta umie i może to działanie wykonać. Można więc stwierdzić, że dany sposób wykonania określonego zadania można uznać lub nie za **algorytm dla określonej osoby,** w zależności od zestawu działań, które można uznać za działania dla niej proste.

Przykładowo, znanego ze szkoły sposobu rozwiązywania równania kwadratowego nie można uznać za algorytm dla osoby, która nie umie wykonać obliczenia pierwiastka kwadratowego liczby rzeczywistej. Sposób wymagający wykonania działań, które dla danej osoby nie są proste (w podanym wyżej sensie) może stać się dla niej algorytmem wskutek **rozwinięcia,** polegającego na określeniu takiego sposobu wykonania każdego z wspomnianych działań, który będzie dla tej osoby algorytmem. Sposób wymagający wykonania działań, które nie są dla danej osoby proste, może zostać uznany za algorytm dla niej również wtedy, gdy w jakiś inny sposób działania takie staną się prostymi dla tej osoby. Szkolny sposób rozwiązywania równania kwadratowego może dla osoby nie umiejącej obliczać pierwiastka stać się algorytmem wskutek rozwinięcia polegającego na określeniu dla niej algorytmu pierwiastkowania. Może on również zostać uznany za algorytm dla osoby, która może obliczyć pierwiastek wskutek naciśnięcia odpowiedniego klawisza we wręczonym jej minikalkulatorze.

Algorytm staje się zobiektywizowanym elementem wiedzy funkcjonując społecznie w postaci **zapisu algorytmu.** Zapis taki musi być oczywiście sformułowany w języku zrozumiałym dla kręgu osób zainteresowanych jego wykorzystaniem.

Rola języka programowania

Przedstawione uwagi mają szczególne znaczenie dla przypadków, w których określone zadanie ma być zrealizowane przez komputer.

Po pierwsze, przekazany komputerowi sposób realizacji zadania musi być dla niej algorytmem, co oznacza, że przewidziane zgodnie z nim do wykonania działania muszą być dla tego komputera proste, tj. muszą być działaniami należącymi do repertuaru operacji tego komputera lub działaniami, które komputer może wykonać opierając się na przekazanych mu wcześniej algorytmach.

Po drugie, sposób realizacji zadania musi być przekazany komputerowi w języku, który jest przezeń akceptowany.

Spełnienie obu powyższych wymagań musi ponadto nastąpić przy uwzględnieniu wybranego języka programowania, czyli języka przewidzianego w celu przedstawienia komputerowi algorytmu realizacji żądanego zadania. Algorytmem dla komputera jest więc sposób realizacji takiego zadania, odwołujący się do operacji elementarnych, zdefiniowanych w ramach wybranego języka programowania lub do działań, które maszyna będzie mogła wykonać w oparciu o wcześniej ustalone algorytmy na podstawie wezwań przewidzianych w ramach tego języka. Ustalenie zatem, czy określony sposób realizacji zadania może być uznany za algorytm dla danego komputera może być dokonane jedynie przy uwzględnieniu wybranego do wykorzystania, przewidzianego dla tego komputera języka programowania. Można więc mówić o **algorytmie dla komputera, widzianego na poziomie wybranego języka programowania.**

Wynika więc stąd, że działalność osoby zapisującej w danym języku program, na podstawie którego dany komputer ma zrealizować określone zadanie, **polega głównie na ustaleniu odpowiedniego algorytmu,** tj. algorytmu dla tego komputera, widzianego na poziomie tego języka. Jeżeli ustalony jest algorytm realizacji jakiegoś zadania dla innego komputera lub nawet tego samego, ale widzianego na poziomie innego języka programowania, to problem zaprogramowania czy przeprogramowania jest nadal problemem ustalenia **odpowiedniego** algorytmu. To samo stwierdzenie dotyczy sytuacji, w której zapis sposobu realizacji zadania ma postać schematu blokowego i odwołuje się do działań wykraczających poza repertuar operacji przewidzianych w języku, w którym program ma być zapisany. Natomiast czynności, które pozostają do wykonania po ustaleniu określonego wyżej, odpowiedniego algorytmu, składają się już tylko na **rutynową działalność zawodowego programisty.**

Rozumowanie powyższe prowadzi więc do wniosku, że poza zdefiniowaniem danych, **program przeznaczony dla danego komputera, zapisany w określonym języku programowania tego komputera jest niczym innym jak tylko postacią ustalenia odpowiedniego algorytmu.** Taki wniosek można porównać z syntetycznym ujęciem przedstawionym przez S. Alagić'a i M. Arbiba [1], którzy piszą, że aby zapisać program trzeba „sformułować ten algorytm tak dokładnie, aby według jego instrukcji komputer mógł działać automatycznie. Innymi słowy algorytm ten trzeba zapisać jako program w języku programowania".

Program komputerowy jako wypowiedź naukowa

Jeżeli dostępna dla jakiegoś eksperymentatora **X** aparatura, za pomocą której chce on dokonać pomiaru interesującej wielkości fizycznej **W** nie umożliwia dokonania tego pomiaru przez jej proste zadziałanie i odczyt wyniku, to eksperymentator próbuje znaleźć sposób realizacji tego zadania w drodze skończonego ciągu działań, z których każde jest albo prostym zadziałaniem tej aparatury i odczytem wyniku, albo możliwym do wykonania krokiem obliczeniowym. Ustalony przez niego sposób realizacji pomiaru podlega uzasadnieniu (obiektywnej weryfikacji) w oparciu o przyjęty i empirycznie zweryfikowany model zjawisk zachodzących w trakcie tak zaplanowanego postępowania.

Ustalony sposób funkcjonuje społecznie po zobiektywizowaniu go w postaci odpowiedniej wypowiedzi językowej, powoływanej zwykle pod nazwą „Metoda **X'a** wyznaczania wielkości **W**". Wypowiedź ta jest formą ustalenia faktu, że zgodnie ze zweryfikowanym empirycznie modelem rzeczywistości obiektywnej określone postępowanie doprowadzi do określonego wyniku. Wypowiedź taką nazywa się zwykle wypowiedzią naukową.

Sytuacja osoby, która zamierza zaprogramować dostępny komputer w celu realizacji określonego zadania, jest podobna do sytuacji opisanej powyżej. Osoba ta chce uzyskać określone wyniki i nie może ich uzyskać przez proste zadziałanie komputera (wykonanie operacji elementarnej lub dostępnego jej podprogramu). Osoba ta dąży do ustalenia sposobu realizacji zadania w drodze skończonego ciągu działań, z których każde jest prostym zadziałaniem tego

komputera, widzianego na poziomie wybranego języka programowania. Ustalony sposób realizacji zadania podlega obiektywnej weryfikacji analitycznej w oparciu o przyjęty model zadania i wynikającą z niego, określoną w projekcie, funkcję programu. Ustalony sposób zostaje zobiektywizowany w postaci programu komputerowego, zapisanego w wybranym języku programowania. Zapisanie zweryfikowanego sposobu jest formą ustalania faktu, że zgodnie z przyjętym modelem zadania, zdefiniowane programem działanie określonego komputera doprowadzi do otrzymania określonych wyników.

Opisane podobieństwo sytuacji pozwala na zaproponowanie tezy, że **program komputerowy można uznać za wypowiedź naukową**, tj. formę ustalenia faktu dotyczącego rzeczywistości obiektywnej.

SKUTKI UZNANIA PROGRAMU ZA WYPOWIEDŹ NAUKOWĄ

Przyznanie jakiejś wypowiedzi cechy naukowości uzależnia się zwykle od spełnienia przez nią szeregu rozmaicie formułowanych warunków [5]. Nie wchodząc głębiej w rozważanie tej kwestii, można zająć stanowisko typologizujące i w przypadku każdej konkretnej wypowiedzi badać czy:

a) wzbogaca ona istotnie społecznie nieobojętną wiedzę o rzeczywistości obiektywnej?

b) jest sformułowana dostatecznie ściśle?

c) jest dostatecznie uzasadniona?

i uzależniać przyznanie stopnia naukowości takiej wypowiedzi od stopnia, w jakim można uznać powyższe kwestie za rozstrzygnięte pozytywnie.

Jak należałoby rozstrzygać powyższe kwestie w przypadku programu komputerowego?

Najwięcej wątpliwości w konkretnych przypadkach może budzić rozstrzygnięcie kwestii a). Wydaje się, że mogłaby ona być rozstrzygana pozytywnie na przykład wtedy, jeśli problem przewidziany do rozwiązania przy użyciu komputera jest istotnym problemem w jakiejś dyscyplinie społecznej aktywności (nauka, jej zastosowania, technika, organizacja), a wyrażony programem sposób jego rozwiązania jest albo pierwszym znanym sposobem, w ogóle, albo oryginalnym sposobem, innym niż znane dotychczas. Przykładowo, może to być zapis oryginalnego algorytmu rozwiązania istotnego problemu naukowego wyrażony w jakimkolwiek języku programowania.

Trudno natomiast uznać możliwość pozytywnej odpowiedzi w omawianej kwestii w przypadkach, w których wyrażony — ocenianym programem — sposób rozwiązania jest taką odmianą znanego wcześniej algorytmu, jaka może być **rutynowo** uzyskana przez osobę zawodowo zajmującą się programowaniem. Przykładowo, może to być program w języku ASSEMBLER uzyskany jako tłumaczenie programu zapisanego wcześniej w języku FORTRAN lub — uzyskany na podstawie algorytmu zapisanego uprzednio w formie schematu blokowego czy w formie przepisu odczytanego, na przykład, z podręcznika metod numerycznych. Pomiędzy opisanymi przypadkami skrajnymi można jednak wyobrazić sobie wiele pośrednich, w których rozstrzygnięcie omawianej kwestii będzie nastręczać wiele wątpliwości. Może być tak na przykład wówczas, gdy zaprogramowanie wymaga takiego rozwinięcia algorytmu, którego nie można uznać za rutynowe działanie zawodowego programisty. Przy rozstrzyganiu omawianej kwestii należy rozważać też zasięg społecznego zainteresowania rozwiązaniem problemu. Rozwiązywany przy użyciu komputera problem może być w niektórych przypadkach problemem o znaczeniu, choćby potencjalnie, uniwersalnym. W innych przypadkach, może wynikać z jednorazowej, lokalnej potrzeby niewielkiej instytucji.

Rozstrzygnięcie pozytywne w kwestii b) nie powinno budzić wątpliwości, jeśli oceniana wypowiedzią jest program. Wynika to w sposób oczywisty z charakteru języka, w którym wypowiedź taka jest sformułowana.

Jednak wiele wątpliwości może budzić w konkretnych przypadkach rozstrzyganie kwestii c), a więc rozstrzyganie, czy wypowiedź-program jest dostatecznie uzasadniona. Nie budzi wątpliwości odpowiedź w tej kwestii w przypadkach, w których program jest analitycznie zweryfikowany ze względu na funkcję, którą ma wyrażać zgodnie z przyjętym modelem zadania. Nie są to jednak przypadki częste. W większości przypadków programy, jako wypowiedzi dotyczące obiektywnej rzeczywistości, nie są naprawdę uzasadniane, lecz jedynie **popierane subiektywnym przekonaniem** o ich poprawności, wynikającym z intuicji i fragmentarycznych testów. Ponieważ wykorzystanie programu ma sens jedynie wówczas, gdy jest on poprawny, przeto użytkowanie programu świadczy o określonej zbiorowej zgodzie na uznanie go za uzasadniony. Istotne jest jednak to, w jakim stopniu wspomniane przekonanie ma przynajmniej częściowo podłoże racjonalne.

ZAGADNIENIA ROZPOWSZECHNIANIA

Sytuacja, w której następuje rozpowszechnienie wypowiedzi naukowej, nie jest prawie obojętna. Wypowiedź taka bowiem zostaje rozpowszechniona zwykle za pośrednictwem utworu (dzieła) naukowego, a więc artykułu w czasopiśmie naukowym, monografii itp. Utwory tego rodzaju są przedmiotem praw autorskich, co wynika z art. 1 Ustawy o prawie autorskim (Dz. U. z 1952 r., nr 34, poz. 234).

Jak dokładnie przedstawia się zakres takiej ochrony?

Kwestia ta była przedmiotem kontrowersji w literaturze prawniczej [8]. Niezależnie od stanowisk zajmowanych w toku polemik, nie ulega jednak wątpliwości konieczność odróżnienia samego dzieła naukowego od tego, z czym ono zapoznaje [2]. To mianowicie, z czym dzieło takie zapoznaje, a więc prezentowane w nim **ustalenie naukowe, nie podlega ochronie prawem autorskim**, jest bowiem zobiektywizowaną postacią faktów dotyczących rzeczywistości obiektywnej.

W przedstawionej sytuacji, w której znajduje się eksperymentator **X**, ochronie podlegają więc wszelkie dzieła naukowe, które zapoznają z „Metodą **X'a** wyznaczania wartości **W**", a więc artykuły, monografie itp. Ochrona taka polega na przyznaniu autorowi każdego takiego dzieła wyłącznych uprawnień, na przykład, do decydowania o publikacji lub do wyrażania zgody na dokonywanie w nim zmian (na podstawie art. 15, p. 1 Ustawy o prawie autorskim), do wyrażania zgody na tłumaczenie dzieła (na podstawie art. 32 Ustawy o prawie autorskim lub do wynagrodzenia za jego wykorzystanie (na podstawie art. 15, p. 3 ustawy). Nie podlega natomiast ochronie prawem autorskim prezentowana w opublikowanym utworze, sama „Metoda **X'a...**". **Jej zapis może być wykorzystany przez kogokolwiek** do przeprowadzenia własnych pomiarów, wykorzystany w celu ulepszenia metody lub opracowania podobnej do niej itd. Podobna swoboda powszechnego wykorzystania dotyczy wypowiedzi formułujących inne ustalenia naukowe, jak na przykład — formuł wyrażających prawa odkryte w ramach nauk przyrodniczych, przedstawiających rozwiązania określonych równań matematycznych czy formuł obliczeniowych dla potrzeb techniki.

Nie należy jednak stąd wnosić, że autor ustalenia naukowego nie ma do niego żadnych praw. Fakt dokonania ustalenia przez określoną osobę jest jej dobrem osobistym i na podstawie przepisów o ochronie tych dóbr (art. 23 i 24 kodeksu cywilnego) „autor ustalenia może domagać się, aby w sposób zwyczajowo przyjęty powoływano jego autorstwo i by go sobie nie przywłaszczano" ([2], s. 55). W omawianej sytuacji oznacza to, że fakt korzystania z „Metody **X'a...**" należy zaznaczyć — na przykład — w dziełach dotyczących badań, w których metodę tę wykorzystano. Nie można też, na przykład, przy żadnej okazji nazywać tej metody, powiedzmy — „Metodą **Y'a...**".

Uznanie programu za rodzaj wypowiedzi naukowej, będącej ustaleniem faktu, rodzi określone konsekwencje. Wszelkie utwory zapoznające z programem, w szczególności, na przykład **wszelkie rodzaje dokumentacji określonego programu podlegają ochronie prawem autorskim. Ochronie takiej nie podlega jednak sam program**, z którym zapoznają opublikowane utwory tego rodzaju. Jest on bowiem postacią ustalenia algorytmu, który jest odbiciem rzeczywistości obiektywnej.

W podobny sposób, w jaki dotychczasowe rozważania pozwalają na uznanie programu za wypowiedź o określonym stopniu naukowości i na odpowiednie zajęcie stanowiska w kwestii jego ochrony prawnej, pozwalają one na zaproponowanie kwalifikacji również innych konstrukcji intelektualnych, powstających w procesie tworzenia systemu oprogramowania, takich jak model zadania czy projekt takiego systemu. Jak stwierdzono poprzednio, przyjęty model zadania można uznać za abstrakcyjną reprezentację problemu przeznaczonego do rozwiązywania przez komputer.

Konstrukcję taką można więc uznać za wypowiedź formułującą problem. Wypowiedzi tego rodzaju mogą również być włączane w obszar nauki ([5], s. 73), przy czym stopień naukowości powinien być oceniany według tych samych kryteriów, według których oceniany powinien być stopień naukowości wypowiedzi-programów. Tak więc problem, którego dotyczą, powinien być istotny dla jakiejś dziedziny społecznej aktywności, a poszukiwanie nowego sposobu rozwiązania powinno być uzasadnione nieefektywnością sposobów znanych dotychczas (jeśli są znane w ogóle).

Projekt systemu oprogramowania został natomiast uznany za propozycje określonej struktury realizacji zadania, a więc za propozycję określonej strategii rozwiązywania problemu. Na podobnych jak wyżej zasadach, wypowiedź definiująca taką propozycję może być również uznana za wypowiedź o możliwym do oceny stopniu naukowości. Utwory zapoznające z takimi wypowiedziami jak wymienione powyżej, a więc opracowania przedstawiające model zadania czy dokumentacja projektu, są utworami chronionymi prawem autorskim. Nie podlegają jednak takiej ochronie ustalenia, z którymi zapoznają opublikowane utwory tego rodzaju. Ustalenia te mogą być swobodnie wykorzystywane przez kogokolwiek, z poszanowaniem jednak dóbr osobistych ich autorów.

Ograniczenie, w przypadku twórczości naukowej, ochrony udzielanej przez prawo autorskie do samych utworów oraz odmowa tej ochrony w stosunku do ustaleń, z którymi one zapoznają, ma swoje podłoże w humanistycznym postulacie, aby nie monopolizować ani sposobów formułowania problemów dotyczących obiektywnej rzeczywistości, ani sposobów ich rozwiązywania. Ustalenia takie, jeśli mają charakter rozwiązań technicznych, mogą jedynie w niektórych przypadkach być chronione patentem.

Szybki wzrost nakładów niezbędnych do prowadzenia działalności naukowo-badawczej powoduje jednak, że wymieniony postulat staje się coraz trudniejszy do pogodzenia z interesem majątkowym osób fizycznych lub prawnych finansujących taką działalność. Ze względu więc na interes tych osób, istniejący dotychczas model ochrony dóbr niematerialnych — jedynie prawa autorskie lub prawa wynikające z patentu jako prawa o charakterze bezwzględnym — okazuje się zbyt wąski. Stan ten powoduje coraz powszechniejsze w ostatnich latach zjawisko chronienia przez te osoby rozwiązań, które nie mogą być przedmiotem patentu (i nie są, jak była o tym mowa, chronione prawem autorskim), w drodze utrzymywania tych rozwiązań w tajemnicy (tzw. know-how). Brak ustawowo przyznanych wyłącznych praw majątkowych zastępowany jest więc stanem faktycznej wyłączności. Zjawisko takie występuje również w odniesieniu do rozwiązań zawartych w oprogramowaniu komputerów.

Praktyka taka nie jest sprzeczna z przepisami prawa autorskiego. Pomimo braku podstawy prawnej dla ograniczania swobody powszechnego wykorzystania ustaleń, z którymi zapoznają wszelkie utwory — w tym utwory dotyczące modeli zadań, projektów systemów oprogramowania czy samych programów — wykorzystanie takie może nastąpić z oczywistych powodów jedynie w sytuacji, w której utwory te zostały opublikowane. Należy rozumieć, że publikacja utworu następuje, jeśli został on zwielokrotniony, choćby w znikomej liczbie egzemplarzy, o ile zostały one udostępnione publicznie ([2], s. 105). Przez publiczne udostępnienie należy rozumieć takie udostępnienie, wskutek którego teoretycznie dowolna osoba może zapoznać się z utworem. Nie stanowi takiej publikacji udostępnienie egzemplarzy utworu wybranym osobom, uczynione z wyraźnym zastrzeżeniem, że ich treść jest przeznaczona do wyłącznej wiadomości tych osób.

Uprawnienie do decydowania o pozostawieniu utworu nieopublikowanym jest składnikiem treści autorskich praw osobistych (art. 15, p. 1 Ustawy o prawie autorskim). Udostępnienie więc egzemplarzy utworu w sposób niezgodny z zastrzeżeniem wyrażonym przez autora jest naruszeniem jego praw autorskich (art. 52, p. 5 Ustawy o prawie autorskim), za które odpowiada osoba dokonująca takich naruszeń — na ogólnych zasadach prawa cywilnego.

Należy jednak zauważyć, że wymieniony sposób poszukiwania ochrony interesu majątkowego osób finansujących działalność naukowo-badawczą, przez utrzymywanie rozwiązań w tajemnicy, nie jest w pełni dogodny dla producentów oprogramowania, ze względu na coraz bardziej przemysłowy charakter opracowywania oprogramowania oraz — ze względu na szczególną łatwość rozpowszechniania rozwiązań (kopiowanie nośników z programami).

* * *

Przedstawiona w niniejszym artykule kwalifikacja konstrukcji powstających w procesie tworzenia systemu oprogramowania i wynikające stąd wnioski mogą stanowić podstawę do dokonania oceny sytuacji prawnej podmiotów uczestniczących w tym procesie (twórców, pracodawców, zleceniodawców, zamawiających). Dokonanie takiej oceny jest jednak odrębnym, czysto prawnym zagadnieniem, wykraczającym poza zamierzony zakres niniejszego artykułu.

LITERATURA

[1] Alagić S., Arbib M.: Projektowanie programów poprawnych i dobrze zbudowanych. WNT, 1982, s. 9
[2] Błeszyński J., Staszków M.: Prawo autorskie i wynalazcze. PWN, 1983, s. 60
[3] Czachórska B.: Umowy w zakresie informatyki i ochrona programów komputerowych. Ossolineum, 1980
[4] Encyklopedia Powszechna. PWN, 1973, s. 58
[5] Kamiński S.: Pojęcie nauki i klasyfikacja nauk. Wyd. KUL, 1961, s. 71
[6] Słownik języka polskiego. t. 1, PWN, 1982, s. 33
[7] Sołtysiński S.: Ochrona prawna programów komputerowych. Zeszyty naukowe UJ. Prace z zakresu prawa cywilnego i praw na dobrach niematerialnych, 1973, z. 1, s. 393
[8] Staszków M.: W jakim zakresie prawo autorskie chroni dzieła naukowe. Studia Juridica Silesiana, 1979, t. 5, s. 226
[9] Turski W. M.: Propedeutyka informatyki. PWN, 1978, s. 168
[10] Waluszewski J.: Sytuacja prawna twórców programów dla maszyn cyfrowych. Państwo i Prawo, 1978, nr 6, s. 74.

Książki nadesłane

WYDAWNICTWA NAUKOWO-TECHNICZNE

Misiurewicz P.: Podstawy techniki cyfrowej. Wyd. 2, 272 str., nakład 10 000 egz., cena 210 zł, Warszawa, 1985

Książka jest podręcznikiem dla studentów wyższych szkół technicznych oraz dla inżynierów — projektantów układów cyfrowych. Zawiera podstawowe wiadomości o współczesnej technice cyfrowej. Omówiono w nim elementy logiczne SSI, bloki funkcjonalne MSI, metodykę projektowania złożonych układów cyfrowych z bloków i zespołów funkcjonalnych oraz architekturę i projektowanie systemów mikroprocesorowych. Prezentacji mikroprocesorów jest poświęcona znaczna część podręcznika — po wprowadzeniu przykładowego mikroprocesora o przejrzystej strukturze i liście rozkazów — omówiono typowe, współczesne mikroprocesory 8- i 16-bitowe oraz mikrokomputery jednoukładowe, a następnie problemy projektowania systemów mikroprocesorowych ze szczególnym uwzględnieniem specyfiki pracy mikroprocesorów w układach sterowania. Przedstawiono także przykłady realizacji typowych cyfrowych układów automatyki i pomiarów.

Podstawową zaletą podręcznika jest zaprezentowanie jednolitego podejścia do projektowania układów cyfrowych zbudowanych bądź to z bloków funkcjonalnych MSI, bądź to przy użyciu mikroprocesora. W obu wypadkach punktem wyjścia jest algorytm działania układu, zapisany w postaci sieci działań, od którego przechodzi się do struktury sprzętowej układu zbudowanego z bloków funkcjonalnych lub do programu dla mikroprocesora.

Sobczak W., Malina W.: Metody selekcji i redukcji informacji. Wyd. 2, 252 str., nakład 4000 egz., cena 146 zł, Warszawa, 1985

Książka pomocnicza dla studentów wydziałów elektroniki: kierunków telekomunikacja oraz informatyka. Może być przydatna dla pracowników naukowo-technicznych ośrodków telekomunikacyjnych i informatycznych.

W książce omówiono metody selekcji informacji na podstawie teorii decyzji statystycznych, teorii informacji, porównywanie rozkładów prawdopodobieństwa oraz metodę programowania matematycznego. Przedstawiono również zagadnienia związane z selekcją danych binarnych, grupowaniem danych, transformacjami liniowymi oraz dyskretyzacją zmiennych. Na zakończenie omówiono niektóre problemy redukcji danych powstających przy formowaniu zbiorów uczących i ocenie jakości klasyfikacji.

JAN BIELECKI
Warszawa

FORTH — definiowanie kompilatorów i instrukcji strukturalnych

W artykule przedstawiono przykłady definiowania kompilatorów i instrukcji strukturalnych języka FORTH w wersji fig-FORTH.

PROCESOR JĘZYKA FORTH

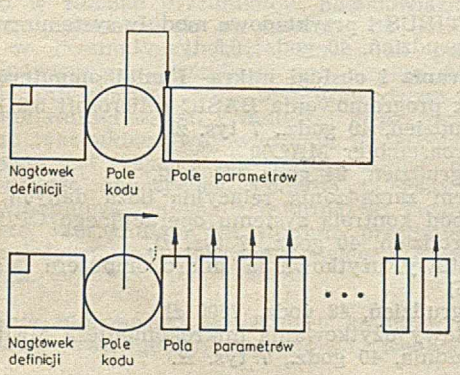
Procesorem języka FORTH, albo krótko **procesorem**, nazywa się poniżej dowolny czynnik umożliwiający interpretowanie operatorów języka. Czynnikiem tym może być wyspecjalizowane urządzenie obliczeniowe, odpowiednio zaprogramowany komputer lub symulator każdego z takich obiektów.

W celu opisania sposobu działania procesora przyjmiemy, że dysponuje on dwoma stosami roboczymi: **stosem parametrów** i **stosem powrotów.** Stos parametrów jest pomocniczą pamięcią stosową, której dotyczy większość operacji, a stos powrotów służy do przechowywania śladu złożonego wywoływania operacji. Może on także, choć w ograniczonym zakresie, być wykorzystywany jako pamięć robocza podczas wykonywania operacji.

Poszczególne operacje rozpoznawane przez procesor są zapamiętywane w **słowniku.** Słownik jest strukturą drzewiastą dzielącą się na podsłowniki, w których zapisane są definicje poszczególnych operacji języka.

W każdym momencie w słowniku wyróżnione są dwa podsłowniki: podsłownik kontekstu i podsłownik bieżący. Pierwszy z nich określa sposób poszukiwania definicji operatora, a drugi — miejsce umieszczania w słowniku nowych definicji. Przeszukiwanie słownika rozpoczyna się od podsłownika kontekstu i jest kontynuowane wzdłuż gałęzi drzewa, aż do jego pnia. Po znalezieniu definicji następuje jej zinterpretowanie. Ponieważ interpretowaniu podlega pierwsza napotkana definicja o danej nazwie, w słowniku może występować wiele definicji operatorów o tych samych nazwach.

Definicja operatora składa się z trzech części: nagłówka, pola kodu i pola parametrów. Nagłówek zawiera nazwę operatora, połączenie jego definicji z innymi definicjami zawartymi w słowniku, a także określenie, czy operator jest czynny czy bierny. Operatory czynne są wykonywane w każdym stanie procesora i z tego względu nazywa się je także operatorami natychmiastowymi. Operatory bierne nie są wykonywane w czasie, gdy procesor znajduje się w stanie kompilacji. W takim stanie napotkanie operatora biernego powoduje jedynie umieszczenie w słowniku wskazania na ten operator, bez interpretowania go.



Rys. 1. Struktura definicji słownikowych

Struktura pola kodu i pola parametrów operatorów zależy od sposobu ich zdefiniowania. W typowych przypadkach pole kodu składa się z dwóch bajtów zawierających adres podprogramu zapisanego w kodzie maszynowym, a pole parametrów składa się z podprogramu albo z dwubajtowych wskazań na słownikowe definicje operatorów (rys. 1).

Działanie procesora polega na sukcesywnym wykonywaniu operacji określonych przez interakcyjnie przekazywane mu operatory. W większości implementacji sekwencje operatorów są grupowane w wiersze, a podjęcie interpretowania operatorów zawartych w wierszu następuje dopiero po jego skompletowaniu.

Operatorami procesora są dowolne spójne ciągi znaków graficznych, którym przypisano definicje słownikowe. Operatory służące do definiowania innych operatorów nazywa się kompilatorami. Zinterpretowanie takich operatorów powoduje m.in. potraktowanie pewnego spójnego ciągu znaków graficznych jako nazwy nowego, aktualnie definiowanego operatora. W ogólnym przypadku zinterpretowanie operatora może powodować dowolne potraktowanie następujących po nim znaków. Za przykład takiego operatora może służyć nawias otwierający, którego zinterpretowanie powoduje zignorowanie następujących po nim znaków, aż do nawiasu zamykającego włącznie.

DEFINIOWANIE KOMPILATORÓW

Zdefiniowanymi pierwotnie kompilatorami języka fig-FORTH są kompilatory CONSTANT i VARIABLE oraz dwukropek. Inne kompilatory można łatwo definiować posługując się sekwencją

```
: nazwa
   <BUILDS
    kompilacja
   DOES>
    wykonanie ;
```

w której „nazwa" identyfikuje kompilator, „kompilacja" określa czynności towarzyszące kompilowaniu nowego operatora, a „wykonanie" określa czynności realizowane podczas wykonywania skompilowanej operacji.

W szczególności wykonanie sekwencji

```
: ARRAY
   <BUILDS
    OVER , * ALLOT
   DOES>
    DUP @ ROT * + + 2+ ;
```

powoduje zdefiniowanie kompilatora ARRAY, który może być wykorzystany do kompilowania definicji operatorów organizujących dostęp do tablic dwuwymiarowych.

Kompilator ARRAY powinien być wywołany w kontekście

```
wiersze kolumny ARRAY tablica
```

w którym „wiersze" i „kolumny" określają liczbę elementów operatora „tablica". W najprostszym przypadku liczba wierszy i kolumn tablicy może być określona za pomocą literałów, np.

```
2 3 ARRAY MATRIX
```

wskutek czego wykonanie kompilacji słowa ARRAY spowoduje utworzenie 6-elementowej tablicy w obrębie pola parametrów definicji operatora MATRIX.

Każdorazowe wykonanie operacji MATRIX wywołanej w kontekście

```
wiersz kolumna MATRIX
```

w którym „wiersz" i „kolumna" określają położenie elementu tablicy, spowoduje umieszczenie na stosie parametrów adresu elementu

```
MATRIX (wiersz, kolumna)
```

w szczególności wykonanie sekwencji

```
5 1 2 MATRIX C!
```

spowoduje nadanie elementowi MATRIX (1, 2) wartości 5.

Nieco ciekawszym przykładem definiowania kompilatorów może być posłużenie się sekwencją

```
: SELECT
   <BUILDS
     ]
   DOES>
     SWAP 2 * + @EXECUTE ;
```

definiującą kompilator operatorów wyboru SELECT. Użycie operatora SELECT w kontekście

```
SELECT nazwa op0 op1 ... opk;
```

powoduje skompilowanie takiego operatora wyboru „nazwa", że sekwencja

```
numer nazwa
```

w której „numer" powoduje umieszczeniu na stosie parametrów danej o wartości d z przedziału [0, k], jest równoważna wykonaniu operacji op_d.

W szczególności, posłużenie się definicją

```
SELECT OBEY DUP SWAP DROP ;
```

powoduje zdefiniowanie takiego operatora OBEY, że

```
0 OBEY
```

jest równoważne operacji DUP, a

```
2 OBEY
```

jest równoważne operacji DROP.

DEFINIOWANIE INSTRUKCJI STRUKTURALNYCH

Kwintesencję możliwości rozbudowania języka FORTH stanowi definiowanie nowych instrukcji strukturalnych. Zilustrowano je na przykładzie instrukcji wyboru CASE, która w ogólnym przypadku będzie wywoływana w kontekście

```
wyróżnik CASE
  cecha1 WHEN sekwencja1
  cecha2 WHEN sekwencja2
  ...
  cechak WHEN sekwencjak
         OTHER sekwencja0
```

Wykonanie instrukcji wyboru CASE polega na wyznaczeniu wartości wyróżnika, a następnie — porównywaniu go z kolejnymi cechami. Jeśli stwierdzi się równość wyróżnika i pewnej „$cecha_d$", to nastąpi wykonanie ciągu operacji „$sekwencja_d$" i zakończenie wykonywania instrukcji wyboru. Jeśli nie stwierdzi się równości, nastąpi wykonanie ciągu „$sekwencja_0$", chyba że w instrukcji nie zostanie użyta fraza

```
OTHER (sekwencja0)
```

kiedy to nie wykona się żadnych czynności.

Za pomocą opisanej tu instrukcji można zdefiniować przedstawiony uprzednio operator wyboru OBEY. Jak łatwo się przekonać, po zinterpretowaniu sekwencji

```
: OBEY CASE
    0 WHEN (DUP)
    1 WHEN (SWAP)
    2 WHEN (DROP)
      OTHER (."ERROR")
  END ;
```

następuje zdefiniowanie takiego operatora OBEY, że

```
0 OBEY
```

jest równoważne operacji DUP,

```
2 OBEY
```

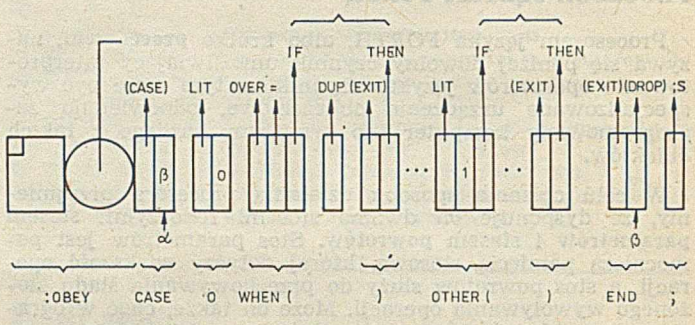
jest równoważne operacji DROP, a

```
n OBEY
```

dla n różnego od 0, 1 i 2 jest równoważne wykonaniu operacji powodującej wyprowadzenie tekstu ERROR.

```
( JANB - CASE STATEMENT )
: (CASE) R> DUP @ >R 2+ >R ;
: (EXIT) R> DROP ;
: CASE COMPILE (CASE) HERE 0 , ; IMMEDIATE
: WHEN( COMPILE OVER COMPILE = [COMPILE] IF  ; IMMEDIATE
: OTHER( COMPILE 1 [COMPILE] IF  ; IMMEDIATE
: ) COMPILE (EXIT) [COMPILE] THEN  ; IMMEDIATE
: END COMPILE (EXIT) HERE SWAP ! COMPILE DROP ; IMMEDIATE
```

Rys. 2. Implementacja instrukcji strukturalnej CASE



Rys. 3. Struktura definicji operatora OBEY

Sposób implementowania instrukcji wyboru przedstawiono na rys. 2. Aby ułatwić jego analizę, na rys. 3 przedstawiono istotne fragmenty słownikowej definicji operatora OBEY bezpośrednio po wykonaniu jego kompilacji.

Centrum Szkolenia Informatycznego ZETO

ul. Hutora 69, 90-558 Łódź

organizuje w grudniu 1985 r.

kursy dotyczące:

Komputerów Jednolitego Systemu RIAD

- Programowanie w języku ASSEMBLER pod kontrolą systemu operacyjnego OS
2—13 grudzień, 80 godz., koszt uczestnictwa 12.800 zł.
- Programowanie w języku FORTRAN — opis języka
2—6 grudnia, 40 godz., 7 tys. zł.
- System operacyjny OS dla operatorów
9—18 grudzień, 60 godz., 10 tys. zł.

Budowy i projektowania systemów mikroprocesorowych

- MULTIBUS i przykładowe moduły systemu MCS-80
16—20 grudzień, 40 godz., 6,5 tys. zł.

Użytkowania i obsługi mikro- i minikomputerów

- Język programowania BASIC (Microsoft 5.21)
2—6 grudzień, 40 godz., 7 tys. zł.
- Makroasembler MAC80
16—18 grudzień, 24 godz., 4200 zł.
- System zarządzania relacyjną bazą danych (działający pod kontrolą systemu operacyjnego CP/M)
9—13 grudzień, 40 godz., 7 tys. zł.
- Podstawy użytkowania mikrokomputera MK-4501 (IMP-85)
16—20 grudzień, 48 godz., 7800 zł.
- Podstawy użytkowania mikrokomputera MERITUM
2—6 grudnia, 40 godz., 7 tys. zł.

ZBIGNIEW SURAJ
Wyższa Szkoła Pedagogiczna
Instytut Matematyki
Rzeszów

Problemy nauczania informatyki w szkołach średnich

W niniejszym artykule przedstawiono ogólne informacje o powszechnym nauczaniu informatyki oraz zamieszczono wyniki badań ankietowych przeprowadzonych w kraju wśród nauczycieli informatyki oraz kandydatów na studia wyższe.

Zamierzeniem badań było uzyskanie ogólnych informacji dotyczących problemów dydaktycznych związanych z nauczaniem informatyki w szkołach średnich województwa rzeszowskiego oraz — konfrontacja uzyskanych wypowiedzi z wynikami badań ankietowych wśród kandydatów na studia. Wyniki tych badań zostały wykorzystane do opracowania propozycji dotyczących doskonalenia procesu nauczania informatyki w szkołach średnich oraz do opracowania programu współpracy Pracowni Obliczeniowej Wyższej Szkoły Pedagogicznej w Rzeszowie ze szkołami średnimi województwa rzeszowskiego, w zakresie nauczania tego przedmiotu. Propozycje te oraz program współpracy wraz z częściową oceną jego realizacji przedstawiono w końcowej części artykułu.

STAN POWSZECHNEGO NAUCZANIA INFORMATYKI W POLSCE

Nauczanie informatyki w Polsce w szkołach średnich rozpoczęło się w roku szkolnym 1974—1975. Odtąd nauczanie tego przedmiotu odbywa się w ramach przedmiotów nadobowiązkowych i kół zainteresowań. Dzięki zaleceniom władz oświatowych nastąpił znaczny rozwój nauczania informatyki w liceach ogólnokształcących w latach 1974—1981. W roku szkolnym 1974—1975, tj. w pierwszym roku wprowadzenia tego przedmiotu, informatyka była nauczana w 22 szkołach średnich (w tym, w 2 szkołach województwa rzeszowskiego), zaś w roku 1980—1981 w 445 liceach oraz 70 technikach [1]) (w tym, w 9 liceach oraz 2 technikach woj. rzeszowskiego). W związku z przejściem szkolnictwa na pięciodniowy system pracy, od roku szkolnego 1981—1982 zmieniły się plany nauczania (są one znacznie ograniczone co do liczby godzin). Przedmiot informatyka pozostał w dalszym ciągu w grupie przedmiotów nadobowiązkowych.

Obecnie warunkiem wprowadzenia w szkole zajęć z informatyki w ramach przedmiotów nadobowiązkowych jest dobrowolne zgłoszenie się co najmniej 25 uczniów z klas trzecich w liceum ogólnokształcącym lub przedostatniej klasy odpowiedniego technikum lub liceum zawodowego. Nauczanie informatyki może być prowadzone również w ramach kół zainteresowań z mniejszą grupą uczniów. Podstawowym warunkiem wprowadzenia do szkoły nauczania informatyki jest zapewnienie przez szkołę odpowiednio przygotowanego nauczyciela. Ponadto zalecany bywa dostęp do komputera. Zajęcia z informatyki mogą prowadzić nauczyciele, którzy ukończyli studium przedmiotowo-metodyczne z informatyki oraz wykładowcy z ośrodków obliczeniowych.

Nauczanie informatyki w szkołach średnich odbywa się w wymiarze dwu godzin tygodniowo, według programu „Informatyka — program nauczania dla kl. III liceum ogólnokształcącego", Warszawa 1980. Program ten zatwierdzony decyzją ministra oświaty i wychowania z dnia 14 kwietnia 1980 r. (nr TP2-700-39/80) obowiązuje od roku szkolnego 1980—1981. W wymienionym programie zakłada się, że nauczanie informatyki w szkole powinno zagwarantować uczniom:

- zdobycie podstawowej wiedzy o informatyce
- zrozumienie specyfiki metod rozwiązywania problemów uwarunkowanych istniejącymi środkami informatyki
- wyrobienie umiejętności uporządkowanego działania przy planowaniu i wykonywaniu zadań
- umiejętność dostrzegania w życiu codziennym problemów, których rozwiązanie z zastosowaniem metod i środków informatyki pozwala osiągnąć istotne korzyści lub zgoła umożliwia ich rozwiązanie
- zrozumienie znaczenia metod i środków informatyki we współczesnym życiu.

Ponadto, program sugeruje, że byłoby pożądane, aby prowadzone zajęcia rozbudziły u uczniów zainteresowanie informatyką oraz zachęciły ich do samodzielnego pogłębienia i rozszerzania zdobytych wiadomości. Program nie zakłada dostępu do sprzętu informatycznego w trakcie zajęć dydaktycznych.

Kształceniem nauczycieli informatyki w Polsce zajmuje się Instytut Kształcenia Nauczycieli w Warszawie. Prowadzi on roczne Studium Informatyki dla nauczycieli matematyki z wyższym wykształceniem. Absolwenci Studium otrzymują prawo do nauczania informatyki w szkole średniej. Studium Informatyki funkcjonuje od lipca 1974 r. W niektórych wyższych uczelniach są organizowane studia podyplomowe z zakresu informatyki lub programowania, w których mogą uczestniczyć również nauczyciele matematyki. Ukończenie studiów nie upoważnia wprawdzie do nauczania informatyki w szkole, ale jest niekiedy bardzo pomocne w nauczaniu.

Obecnie brak jest podręcznika do nauczania informatyki w szkole średniej, przystosowanego do obowiązującego programu. Natomiast dla nauczycieli pomoc w przygotowywaniu się do zajęć z informatyki może stanowić pozycja [3].

Na ogół szkoły średnie nie są wyposażone w sprzęt informatyczny. Niektóre z nich organizują swoim uczniom pewne formy współpracy z blisko położonymi ośrodkami obliczeniowymi. Uczniowie zwiedzają ośrodek obliczeniowy, zapoznają się z jego organizacją, wyposażeniem i zakresem działania, czasami mają możliwość wykonywania pewnych obliczeń. Efekty współpracy oceniane są na ogół pozytywnie. Należy jednak zwrócić uwagę na dwa dość istotne ograniczenia możliwych do osiągnięcia korzyści.

Ośrodki obliczeniowe nie dopuszczają użytkownika komputera bezpośrednio do sprzętu; napisany program należy złożyć w dziale przygotowawczym — i kilka dni czekać na wyniki obliczeń. Program z reguły zawiera błędy; każda poprawka powoduje stratę następnych kilku dni. Pełne rozwiązanie problemu trwa około miesiąca; w chwili otrzymania wyników uczniowie już nie pamiętają o co chodziło w programie.

[1]) Obecnie liczby te zmalały ponad dwukrotnie (przyp. red.)

Językiem programowania jest prawie wyłącznie FORTRAN, język mało elastyczny i technicznie trudny. Proces uruchamiania programu w tym języku jest tak skomplikowany, że aktywność użytkownika pomija sprawę realizacji problemu, a skupia się na zagadnieniu formalnego zapisu tekstu programu.

Na zakończenie uwag ogólnych, dotyczących powszechnego nauczania informatyki, warto dodać, że 5 listopada 1984 r. Prezydium Rządu PRL podjęło postanowienie w sprawie elektronizacji gospodarki narodowej. Postanowiono m.in., że ministrowie i kierownicy urzędów centralnych, nadzorujący szkoły średnie i wyższe, opracują do końca września 1985 r. program powszechnej edukacji w zakresie wiedzy informatycznej oraz program wdrażania i zastosowania techniki komputerowej w procesach kształcenia na lata 1986—1990.

BADANIA ANKIETOWE WŚRÓD KANDYDATÓW NA STUDIA

W lipcu 1984 r. przeprowadzono badania ankietowe wśród kandydatów na studia na kierunki: matematyka, fizyka oraz wychowanie techniczne w Wyższej Szkole Pedagogicznej w Rzeszowie. Ankietowani kandydaci uczyli się informatyki w szkole średniej. Ankietę wypełniło łącznie 51 kandydatów (z liceów ogólnokształcących — 45, z techników — 6). Reprezentowali oni 18 szkół średnich, w tym — 15 liceów ogólnokształcących.

Wśród odpowiedzi o charakterze informacyjnym na uwagę zasługuje fakt, że przed rozpoczęciem nauczania informatyki 65% ankietowanych zetknęło się z pojęciem informatyka poprzez radio, telewizję i prasę, a 16% poprzez czytanie literatury fachowej. O wiele mniej ankietowanych kandydatów uzyskało wiadomości na temat informatyki w czasie rozmów fachowych w domu — zaledwie 8%. Dziedzinę tę chciało poznać 67% ankietowanych i uważają wprowadzenie jej w szkole średniej za potrzebne. Dla części inspiracją poznania była też chęć imponowania — 22%. Nie ulega wątpliwości, że zainteresowanie przedmiotem było znacznie uzależnione od atrakcyjności zajęć lekcyjnych. W tym zakresie na uwagę zasługuje cztery spośród 18 szkół, gdzie zajęcia były prowadzone najbardziej atrakcyjnie.

Przeciętna liczba uczniów w klasie, w której uczono informatyki wynosiła 30. Do zajęć z informatyki przygotowywano się na podstawie notatek sporządzonych w czasie lekcji, najczęściej pod dyktando nauczyciela. Taka forma zajęć była mocno krytykowana, ponieważ znaczną część lekcji nauczyciel poświęcał na dyktowanie nowego materiału, nie mając dostatecznego czasu na jego wyjaśnienie. Miało to duży wpływ na zrozumienie materiału. W konsekwencji zainteresowanie przedmiotem malało, stąd tylko kilku spośród ankietowanych przygotowywało się systematycznie do zajęć z informatyki. Znaczna większość uczniów miała podręcznik z informatyki. Na podstawie ankiet stwierdzono, że tylko w dwóch szkołach wyposażenie biblioteki w literaturę informatyczną, było zadowalające. W żadnej szkole (reprezentowanej przez kandydatów) nie zorganizowano spotkania ze specjalistami z dziedziny informatyki, tylko w jednej były wyświetlane filmy o komputerach.

Ankietowani kandydaci w formie opisowej próbowali też określić sposoby uatrakcyjnienia zajęć z informatyki. Do głównych form w tym zakresie zaliczono między innymi:

- zapewnienie przystępnie napisanego podręcznika z informatyki, dostosowanego do programu nauczania
- zorganizowanie pracowni informatycznych wyposażonych w pomoce dydaktyczne, takie jak: elementy związane z programowaniem, obsługą operatorską komputera, elementy elektroniki i pamięci komputera, zdjęcia modułów komputera itp.
- wyświetlanie filmów, przeźroczy
- zwiększenie częstotliwości kontaktu z komputerem, przez wycieczki do ośrodków obliczeniowych lub bezpośrednią pracę przy komputerze
- dobre zaopatrzenie bibliotek szkolnych w literaturę z zakresu informatyki
- zmniejszenie treści teoretycznych na korzyść przykładów z praktyki i ćwiczeń z zakresu budowy algorytmów, schematów blokowych
- informowanie o nowościach światowych i krajowych z dziedziny informatyki oraz zapraszanie na odczyty specjalistów z tego zakresu.

W trzech szkołach nauczyciele zgromadzili i prezentowali pomoce dydaktyczne, w dziewięciu zaś zorganizowali wycieczkę do ośrodków obliczeniowych, w opinii ankietowanych kandydatów pomogło im to w lepszym zrozumieniu przedmiotu. W piętnastu szkołach nauczyciele nie zaprezentowali zdjęć modułów komputera, a liczba pomocy dydaktycznych była znikoma — w konsekwencji ankietowani ocenili zajęcia z informatyki w tych szkołach jako nudne. Jak widać odpowiednia liczba i jakość pomocy dydaktycznych może w znacznym stopniu przyczynić się do uatrakcyjnienia zajęć.

W ankiecie zapytano też czy ankietowany jest zadowolony, że uczył się tej nowej dziedziny wiedzy — ponad połowę stwierdziło, że tak, jednocześnie motywując, że przedmiot ten wyrabia umiejętność logicznego myślenia oraz może pomóc w zrozumieniu innych przedmiotów, szczególnie matematyki. Fakt ten świadczy wyraźnie o celowości nauczania tego przedmiotu.

Blisko połowa ankietowanych kandydatów wypowiedziała się, że przedmiot ten jest interesujący a część z tej grupy stwierdziła wręcz, że jest on ciekawszy nawet od innych przedmiotów. Tyluż samo stwierdziło, że czyta obecnie literaturę informatyczną.

BADANIA ANKIETOWE WŚRÓD NAUCZYCIELI

Badania ankietowe przeprowadzono także wśród nauczycieli matematyki, posiadających uprawnienia do nauczania informatyki w szkole średniej. Celem ankiety było uzyskanie ogólnych informacji dotyczących problemów dydaktycznych związanych z nauczaniem informatyki w szkołach średnich, na terenie województwa rzeszowskiego, oraz — konfrontacja uzyskanych wypowiedzi z wynikami badań ankietowych wśród kandydatów na studia.

Łącznie ankietę wypełniło 17 nauczycieli (z liceów ogólnokształcących — 13, a z techników — 4. W ogólnej liczbie ankietowanych 7 nauczycieli uczyło tego przedmiotu w poprzednich latach, a także uczy go obecnie.

Tylko jeden spośród nauczycieli nie uważał za celowe nauczanie informatyki w szkołach średnich. Z kolei 14 nauczycieli było zdania, że przedmiotu tego należy uczyć w liceach ogólnokształcących, a 12, że należy go uczyć także w technikach lub liceach zawodowych. Zdecydowana większość, uważała, że informatyka winna być odrębnym przedmiotem, rozpoczynającym się w przedostatnich klasach szkół średnich. Większość była za zwiększeniem wymiaru godzin nauczania informatyki — przez 2 lata po 2 godziny tygodniowo, lub w rok — przez 3 godziny tygodniowo.

Prawie wszyscy ankietowani nauczyciele twierdzili, że nauczanie informatyki winno obejmować zajęcia praktyczne oparte na kontaktach ze sprzętem informatycznym oraz że pokazywanie zdjęć komputerów i wyświetlanie filmów o sprzęcie informatycznym, w szczególności dotyczących ich zastosowań jest celowe.

Wszyscy ankietowani uważali, że zwiedzanie ośrodka obliczeniowego jest potrzebne dla wszystkich uczniów bez względu na to czy uczą się tego przedmiotu czy nie. Ponad połowa nauczycieli stwierdziła, że informatyki w szkole średniej mogą uczyć nauczyciele matematyki po odpowiednim przeszkoleniu w tym zakresie, pozostali zaś, że przedmiotu tego winni uczyć specjaliści — informatycy.

Trzynastu nauczycieli zauważyło, że zdolność przyswajania wiedzy z informatyki koreluje z wynikami nauczania matematyki oraz, że informatyka pomaga uczniom w innych przedmiotach. Ankietowani nauczyciele twierdzili, że zdolni uczniowie są zainteresowani nauczaniem tego przedmiotu, natomiast uczniowie słabi lekceważą go, po-

dokończenie na str. 21

Z zaskoczeniem możemy odnotować fakt, że panująca do niedawna wśród profesjonalistów moda na stosowanie mikroprocesora 8080 lub co najwyżej Z80 jakby nieco osłabła. Konstruktorzy przymierzają się i do innych mikroprocesorów — chociaż w próbach z „16-bitowcami" monopol firmy INTEL utrzymuje się niezmiennie.

Tym, co koniecznie potrzebują podparcia w fakcie, że dany układ produkowany jest w którymś z krajów RWPG spieszymy donieść, że znana do niedawna z wypasu owiec Bułgaria podjęła produkcję bogatej rodziny układów z serii 6800.

Opisany w artykule moduł został zaprojektowany tak, aby mógł współpracować z zasobami (pamięć, układy we-wy) systemu przystosowanego do sterowania przez 8080. Pozwala to na stosunkowo „bezkrwawe" udoskonalenie posiadanego systemu. Możliwa jest też do realizacji konstrukcja wykorzystująca dwa procesory np. Z80 i właśnie 6800 (przełączane), podobnie jak spotyka się to w niektórych rozwiązaniach komputerów kompatybilnych z APPLE II. Tym, którzy chcieliby uzyskać szczegółowe informacje podajemy kontakt telefoniczny z p. Jerzym Orkiszewskim: tel. domowy 21-36-20, tel. do pracy 49-98-71 w. 538. Autor dysponuje również pewnymi modułami programowymi dla systemów wykorzystujących mikroprocesor 6800.

Moduł procesora MOTOROLA 6800

Mikroprocesor MC6800 (U1 — rys. 1) ma 16-bitową szynę adresową, 8-bitową synchroniczną szynę danych oraz trzy podstawowe sygnały sterujące: E (ang. Enable) — zegar systemu, który synchronizuje przesłania na szynie danych, R-W/ — (ang. Read-Write) — wskazujący kierunek przesyłania danych oraz VMA (ang. Valid Memory Address) sygnalizujący poziomem wysokim H obecność na szynie adresowej stabilnego adresu pamięci. Mikroprocesor 6800, podobnie jak 8080, wymaga zewnętrznego układu zegarowego. Rolę tę spełnia układ MC6875 (U2), który dostarcza sygnały zegarowe Φ1 i Φ2 o poziomach MOS, przyjmuje sygnał wstrzymania MRDY (odpowiednik WAIT/) oraz żądania DMA lub cyklu odświeżenia pamięci dynamicznej DMA/-REF REQ/. W przypadku braku aktywnego sygnału MRDY (poziom L) układ 6875 zatrzymuje przebiegi zegarowe w stanie Φ1=„0", Φ2=„1" co odpowiada drugiej połowie cyklu zegara (rys. 2). Reakcją układu na sygnał DMA/-REF REQ/ jest zatrzymanie zegara w pierwszej połowie cyklu oraz ustawienie linii REF GRNT w stan „1". Oscylator wymaga dołączenia kwarcu o częstotliwości czterokrotnie większej niż częstotliwość zegara systemu, która dla standardowej wersji elementów rodziny M6800 wynosi 1 MHz. Warto dodać, że w związku dynamiczną realizacją wewnętrznych rejestrów mikroprocesora częstotliwość ta nie może być niższa niż 100 kHz.

Układ 6875 wytwarza ponadto następujące sygnały: E, który z uwagi na dużą obciążalność wyjścia nie wymaga buforowania (N=30 TTL), sygnały pochodzące z oscylatora 4f i 2f oraz sygnał MCLK (ang. Memory Clock). MCLK daje identyczny przebieg jak E — jednak nie reaguje na stan wejść wstrzymujących, a zatem może być pomocny przy obsłudze pamięci dynamicznej.

Buforowanie sygnałów systemowych

Wydajność prądowa wyjść mikroprocesora pozwala na bezpośrednie wysterowanie ok. 10 układów należących do rodziny M6800, co odpowiada jednemu standardowemu obciążeniu TTL. W wypadku dołączania do systemu typowych bloków pamięci niezbędne są wzmacniacze. Szyna adresowa buforowana jest przez układy U3 i U4 (74LS244) a szyna danych — układ U5 (74LS645). Pozostałe sygnały buforowane są bramkami 74125. Układ 74LS244 jest jednokierunkowym, trójstanowym nadajnikiem szyny (ang. octal bus driver). Układy te mogą być zastąpione przez inne o podobnej funkcji; należy jednak uważać, aby nie przekroczyć obciążalności wyjść mikroprocesora A10...A15, które sterują dwoma układami: U3 i U6. Nadajnik-odbiornik szyny 74LS645 (ang. octal bus transceiver) może być zastąpiony dwoma krajowymi układami UCY 74S416.

Bufory szyny adresowej, danych i linii R-W/ wprowadzane są w stan dużej impedancji przez buforowany sygnał BA (ang. Bus Available). Mikroprocesor wymusza na linii BA poziom aktywny (H) w odpowiedzi na wysterowanie poziomem aktywnym (L) wejścia HALT/ lub wykonanie rozkazu WAI — „czekaj na przerwanie". Kierunek transmisji przez układ U5 wyznacza poziom linii R-W/. Bufory U11 i U12 (74125) są otwarte na stałe (z wyjątkiem bufora linii R-W/).

Zerowanie systemu

Procesor zostaje wyzerowany przez podanie poziomu niskiego (trwającego przynajmniej osiem taktów zegara) na wejście RESET!. Źródłem tego sygnału może być albo układ 6875, realizujący funkcję „power-on reset" (ang. zerowanie przy włączeniu zasilania), lub inny układ zewnętrzny wymuszający logiczne zero na linii RESIN/. W celu zachowania kompatybilności z układami rodziny MCS80 wyprowadzony jest dodatkowo sygnał RESET, aktywny przy poziomie wysokim. Układ zerowania zbudowany został na bramkach układu U13 (7400).

Sygnały sterujące typu INTEL

Do układu jednostki centralnej dołączony został moduł wytwarzający sygnały sterujące, zgodne z szyną rodziny INTELA. Pozwala to na wykorzystanie wcześniej skonstruowanych bloków np. pamięci, we-wy lub specjalizowanych sterowników, np. napędów dysków elastycznych lub monitora telewizyjnego. Moduł zrealizowany został z układów U6 (74LS138), 1/2 U7 (74S139), 1/4 U13 (7400) oraz U8 (74S416).

Mikroprocesor 6800 nie ma rozkazów we-wy takich, jak IN i OUT dla mikroprocesora 8080. Układy we-wy traktowane są jako adresy pewnego bloku w obszarze adresowym. W celu wygenerowania sygnałów I-OR/ i I-OW/ przyjęto, że układy we-wy umieszczone będą w obszarze pamięci o rozmiarach 1 KB. W celu zasygnalizowania, że mikroprocesor generuje adresy z tego obszaru, wytwarzany jest sygnał PA/. Powstaje on w dekoderze U6 (74LS138) sterowanym liniami adresowymi A10...A15 w taki sposób, że w zależności od położenia przełącznika P sygnał PA/ generowany jest dla adresów z zakresu 8000_H-$83FF_H$ lub 9000_H-$93FF_H$,...,FOOO-F3FF. Sygnał PA/ doprowadzony jest do dekodera U7, do którego przyłączony jest również sygnał R-W/. W zależności od stanu tych linii aktywne są wyjścia I-OR/, I-OW/ bądź MEMR/, MEMW/. Pierwsze przyjmują stan aktywny dla PA/=L, drugie dla PA/=H. Dekoder uaktywniany jest tylko w drugiej połowie okresu zegara (E przyjmuje poziom H), o ile procesor generuje odwołanie do pamięci (VMA przyjmuje poziom H/).

Należy zwrócić uwagę, że sygnał PA/ użyty do wydzielenia obszaru adresowego dla układów peryferyjnych rodziny 6800 musi być ziloczynowany z sygnałem VMA. Można to zrealizować w sposób, w jaki zaprojektowano dekoder adresowy dla układu MC6850. Tworzą go U9 i 1/2 U7.

Intelowskie sygnały sterujące buforowane są układem 74S416, wprowadzanym w stan dużej impedancji sygnałem BA.

Sygnały przerwań

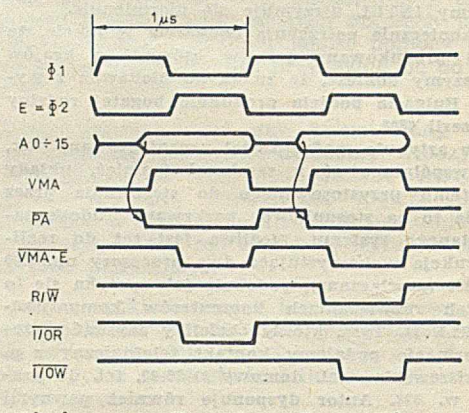
Mikroprocesor 6800 przyjmuje dwa przerwania zewnętrzne — maskowalne IRQ/ i niemaskowalne NMI/. Na oba te wejścia, jeżeli nie są używane, powinien być podawany poziom wysoki.

Układ we-wy szeregowego MC6850

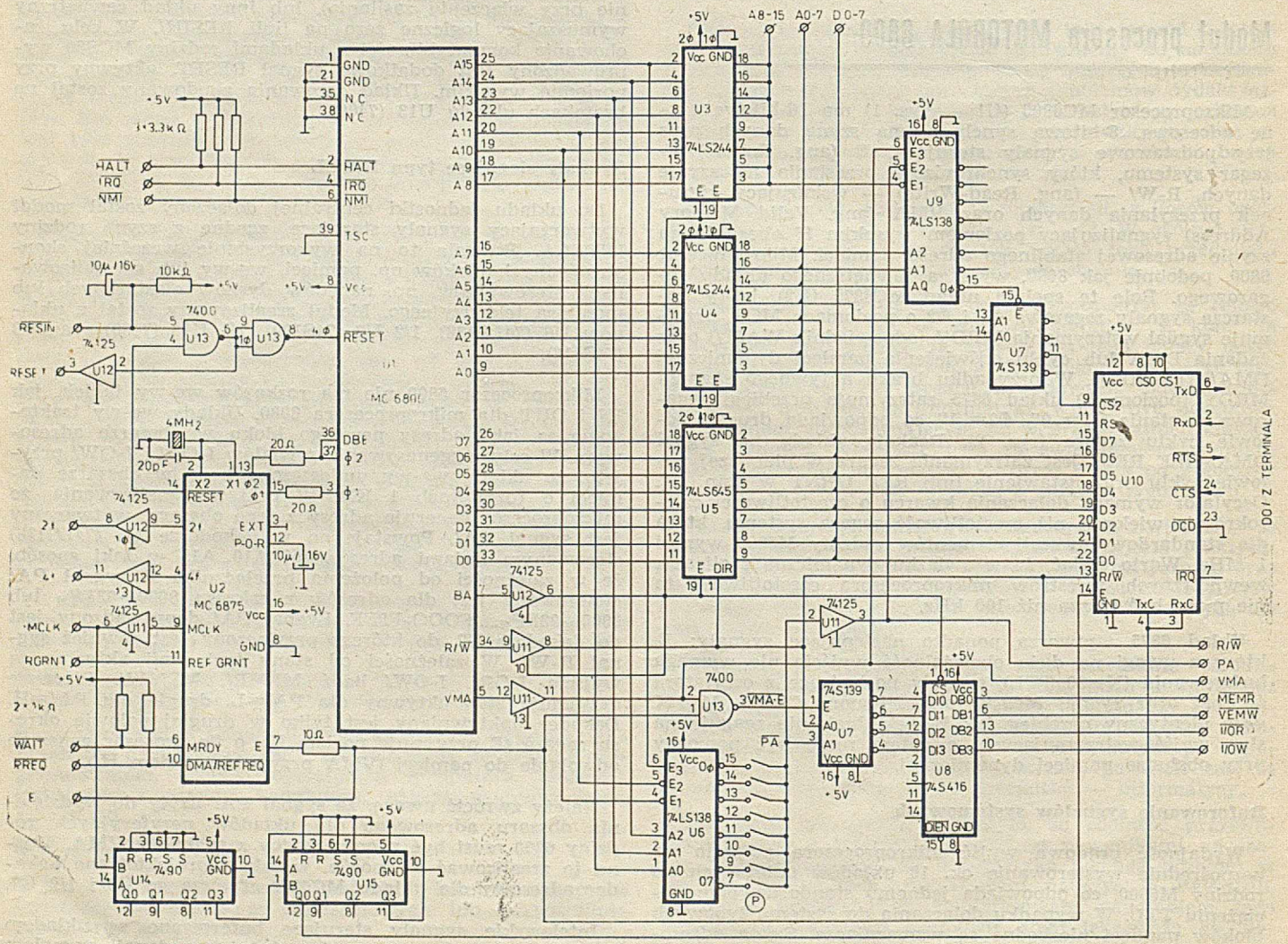
Do układu CPU dołączony został układ 6850, znany pod nazwą Asynchronous Communications Interface Adapter — ACIA (ang. adapter asynchronicznego interfejsu komunikacyjnego). Z pewnym przybliżeniem można powiedzieć, że spełnia on rolę analogiczną do 8251. Obecność tego układu w zaprojektowanym systemie wynika z faktu, że jest on często wykorzystywany przez procedury we-wy bardzo bogatego oprogramowania firmowego dla rodziny 6800. Układ może zostać wykorzystany do współpracy z terminalem lub innym komputerem.

Oznaczenia wyprowadzeń układu od strony systemu są zgodne z opisanymi liniami mikroprocesora, a od strony terminala odpowiadają oznaczeniom linii wg standardu RS-232C. Warto przypomnieć, że linia CTS/ oznacza gotowość terminala do przyjęcia znaku i jej niski poziom na wejściu 6850 odblokowuje sekcję nadajnika. Linia RTS/ sterowana jest programowo. Wyjście sygnału przerwania IRQ/ nie jest wykorzystywane. Może jednak zostać przyłączone do mikroprocesora (konstrukcja typu „otwarty dren" pozwala na galwaniczne sumowanie z innymi sygnałami przerwań).

Częstotliwość sygnału na wejściach RxC i TxC ustalona została na ok. 20 kHz, co w zależności od zaprogramowania wewnętrznego dzielnika odpowiada transmisji z szybkością ok. 19 200 bodów (podział przez 1), 1200 bd (podział przez 16) lub 300 bd (podział przez 64). Sygnał RxC i TxC wy-



Rys. 2. Przebiegi czasowe dla wybranych sygnałów



Rys. 1. Schemat ideowy układu z mikroprocesorem 6800

twarzany jest z sygnału MCLK o częstotliwości 1 MHz w dzielnikach U14 (7490), dzielącym przez 5 i U15 (7490) dzielącym przez 10 ze współczynnikiem wypełnienia 1/2.

W zależności od typu używanego terminala linie RxD i TxD oraz ewentualnie RTS/ i CTS/ należy zaopatrzyć w odpowiednie nadajniki-odbiorniki linii. Dla standardu RS 232C będą to pary MC1488/1489 firmy MOTOROLA bądź SN75150/SN75154 firmy TEXAS INSTRUMENTS.

Dekoder adresowy, zbudowany z U9 (74LS138) i połówki U7 (74S139) wybiera układ ACIA pod adresami 8 i 9 strony układów we-wy.

W bardzo obszernej literaturze omawiającej rodzinę M6800 pozycje krajowe są niestety nieliczne i nieaktualne. Pełną informację techniczną dotyczącą MC6800 i MC6850 można znaleźć w skrypcie Politechniki Warszawskiej: C. Stępień i in.: „Układy mikroprocesorowe serii INTEL 8080, MOTOROLA 6800 i Am 2900". Podstawowe pozycje firmowe to: „M6800 Microprocessor Applications Manual", „M6800 Microprocessor Programming Manual" oraz oczywiście katalog [1]).

JERZY ORKISZEWSKI
Warszawa

[1]) Obszernym wyborem katalogów firmy MOTOROLA dysponuje czytelnia PIE-BOINTE, Warszawa, ul. Marynarska 10 (przyp. AJP)

Wykorzystanie mikrokomputera jako notatnika ma swoje wady i zalety. O ile notes można zawsze mieć pod ręką, to ZX 81 wraz z telewizorem wymaga wcale niemałej torby. Jeżeli jednak korzystamy z często zmieniającego się zbioru informacji, to notes wkrótce stanie się mało czytelny. Polecamy więc program osobom prowadzącym „burzliwą" działalność ale... w jednym miejscu.

Przytoczona przez autora artykułu wersja programu dedykowana jest... pośrednikom w handlu nieruchomościami. Oczywiście bez trudu można ją przystosować do informacji dowolnej natury. Sądzimy, że pokazanie pewnego konkretnego zastosowania ułatwi czytelnikom zrozumienie sposobu funkcjonowania programu.

SPREADSHEET dla ZX 81

Nazwa „SPREADSHEET" odnosi się do pewnego rodzaju programów, umożliwiających przeglądanie większej przestrzeni niż tylko tej ograniczonej do powierzchni ekranu. W tym przypadku ekran jest jak gdyby okienkiem, przez które oglądamy fragment większej całości.

Taka technika jest stosunkowo wygodna przy tworzeniu wieloformatowych tabel, zawierających, jak w tym przypadku, 100 kolumn i 60 wierszy. Dla tego rodzaju programów w Polsce stosuje się czasem nazwę „tablplan".

```
  1 REM 0.....
.........1.....
.........2.....
.........3.....
.........4.....
.........5.....
.........6.....
.........7.....
.........8.....
.........9.....
.........0.....
.........1.....
.........2.....
```

Wydruk 1. Linia „1" programu

Pierwszym krokiem w tworzeniu przedstawianego programu jest umieszczenie linii: 1 REM i 395 dowolnych znaków — w naszym przypadku są to kropki. Poszczególne wiersze tego wydruku zostały ponumerowane dla ułatwienia liczenia (wydruk 1). Zanim zaczniemy wystukiwać owe rzędy kropek, przełączmy komputer na tryb pracy FAST, co zaoszczędzi nam sporo czasu. Po wystukaniu wymaganej liczby kropek (może ich być więcej — nigdy mniej) należy komputer przełączyć na SLOW i dla uchronienia się przed skutkami ewentualnych późniejszych błędów — przepisać na kasetę efekt dotychczasowej pracy. Nie kasując linii 1 REM, wprowadzamy program przedstawiony na wydruku 2 i od tej chwili jesteśmy gotowi do zapełnienia poszczególnych komórek pamięci liczbami przedstawionymi na wydruku 3, poczynając od adresu 16 514. Przedtem jednak kilka słów o wprowadzonym przed chwilą do pamięci „assemblerze" [1]). Jest on tak skonstruowany, że liczby wprowadzane do pamięci ukazują się na ekranie monitora po sześć w wierszu, co bardzo ułatwia kontrolę prawidłowości prowadzonej pracy. Należy tu podkreślić, że wprowadzanie do pamięci poszczególnych liczb jest najbardziej odpowiedzialnym etapem tworzenia programów maszynowych, gdyż właśnie w tej fazie powstaje najwięcej błędów.

```
9800 LET m=16514
9810 LET n=0
9820 INPUT a
9830 IF a>300 THEN GO TO 9920
9840 POKE m,a
9850 PRINT AT 20,m;a
9860 LET m=m+1
9870 LET n=n+5
9880 IF n<26 THEN GO TO 9820
9890 LET n=0
9900 REM SCROLL
9910 GO TO 9820
9920 IF a>555 THEN GO TO 9999
9930 IF a=555 THEN GO TO 9980
9940 LET N=N-1
9950 IF n<26 THEN LET n=n-5
9960 PRINT AT 20,n;" "
9970 GO TO 9820
9980 SAVE "ASSEMBLER"
9990 GO TO 9820
9999 STOP
```

Wydruk 2. Program umożliwiający wprowadzenie kodu maszynowego

Proponowany tu „assembler" zawiera pewne ułatwienia w wykonywaniu tej niewdzięcznej pracy. I tak: wprowadzenie liczby 333 [2]) umożliwia nam poprawienie błędnie wprowadzonej ostatniej liczby. Wybrano liczbę 333, gdyż są to trzy uderzenia w ten sam klawisz. Liczba 555 jest równoważna instrukcji SAVE, która powoduje zapisanie na taśmie wszystkich dotychczas wprowadzonych liczb. Można jedynie radzić by pomocniczą liczbę 555 wprowadzać stosunkowo często, zapisując wynik naszej pracy nawet co trzy, cztery wiersze. Tak częste zapisywanie na kasecie programów cząstkowych wcale nie jest przesadą — będzie się można o tym przekonać przy wprowadzaniu programu.

Dla łatwiejszej orientacji długie kolumny liczb podzielono na trzy grupy. Po wprowadzeniu ostatniej liczby należy nacisnąć trzy dziewiątki czyli 999, co jest równoznaczne instrukcji STOP zatrzymującej pracę „assemblera".

Zapewne nurtować nas będzie teraz pytanie: czy udało się nam wprowadzić wszystkie liczby bez pomyłek? Odpo-

---

[1]) Program przedstawiony na wydruku 2 został przez autora żartobliwie nazwany „assembler". Chociaż kompletuje on kod maszynowy programu, to jednak jest daleką namiastką programów typu asembler wykorzystywanych do translacji źródłowej (w postaci symbolicznej) wersji programu na binarną. Ponieważ autor użył tej nazwy również na wydruku — zachowaliśmy ją w tekście artykułu. (przypis AJP)

[2]) Największa liczba będąca kodem programu wynosi 255 (przypis AJP)

wiedź uzyskamy przez zsumowanie zawartości poszczególnych komórek, co najlepiej można zrobić przy użyciu następującego programiku:

```
9995 LET M=0
9996 FOR N=16514 TO 16870
9997 LET M=M+PEEK N
9998 NEXT N
9999 PRINT M
```

```
0     0     0     0     0     0
0     0     0     0     62    8
14    0     33    130   64    113
35    61    254   0     32    249
42    16    64    126   254   198
40    3     35    24    248   35
126   254   117   40    3     35
24    239   62    7     35    61
254   0     32    250   34    132
64    58    134   64    42    132
64    1     100   0     254   0
40    5     237   74    61    24
247   34    132   64    58    135
64    42    132   64    254   0
40    4     35    61    24    248
34    132   64    42    12    64
1     166   0     237   74    34
130   64    205   187   2     124
198   2     56    9     68    77
205   189   7     6     0     78

121   50    137   64    1     0
12    11    120   177   32    251
58    137   64    254   35    32
14    58    134   64    254   50
40    7     58    134   64    60
50    134   64    58    137   64
254   34    32    14    58    134
64    254   0     40    7     58
134   64    61    50    134   64
58    135   64    254   0     40
14    58    137   64    254   36
32    7     58    135   64    61
50    135   64    58    135   64
254   68    40    14    58    137
64    254   33    32    7     58
135   64    60    50    135   64
62    10    50    136   64    42
16    64    126   254   217   40
3     35    24    248   35    126
254   103   40    3     35    24

239   35    35    35    35    35
58    135   64    254   0     40
4     35    61    24    248   34
138   64    42    12    64    1
100   0     237   74    237   91
138   64    14    33    26    119
13    121   61    35    19    254
0     40    2     24    243   42
130   64    237   91    132   64
14    33    58    136   64    254
10    32    5     26    198   128
24    1     26    119   13    121
61    35    19    254   0     40
2     24    231   35    62    68
19    61    254   0     32    250
58    136   64    61    50    136
64    254   0     194   168   65
58    137   64    254   50    40
7     254   40    40    3     195
154   64    201   27    27    27
```

Wydruk 3. Kod maszynowy programu

Program ten uruchamiamy instrukcją RUN 9995. Jeżeli otrzymana suma wyniesie 29 192 — możemy przystąpić do następnego etapu. Jeżeli wynik nie jest zgodny z podaną

```
  S LET =1
  10 DIM T$(100)
  20 DIM P$(50,10)
  30 LET T$="TYPE PRICE 1ST MOR
TGAGE RATE LOAN          ADDRESS
          AGENCY PHONE"
  40 FOR N=1 TO 60
  50 FAST
  60 LET A$(N)="    :$      :$
        :    :   :
             :   :     :
               :"
  70 LET A$(N,2 TO 3)=STR$ N
  80 NEXT N
  90 SLOW
 105 CLS
 110 PRINT AT 5,11;"M E N U";AT
7,8;"(ENTER CHOICE)";AT 10,5;"1.
 ENTR RECORD.";AT 12,5;"2. VIEW
SPREADSHEET."
 120 IF INKEY$="1" THEN GOTO 200
 130 IF INKEY$="2" THEN GOTO 150
 140 GOTO 120
 150 CLS
 160 PRINT "  REAL ESTATE SPREAD
SHEET";AT 16,0;"
                ";TAB 6;"PRESS ""
M"" FOR MENU."
 170 RAND USR 16524
 180 IF PEEK 16521=50 THEN GOTO
100
 182 IF PEEK 16521=40 THEN GOTO
186
 184 GOTO 100
 186 PRINT AT 18,4;"ENTER CORREC
TED ADDRESS"
 188 INPUT A$(PEEK 16518+1,37 TO
 67)
 190 RAND USR 16507
 192 GOTO 180
 200 CLS
 210 PRINT "ENTER RECORD INFORMA
TION"
 220 PRINT ,,"1. TYPE? (4 LETTER
S)"
 230 INPUT A$(A,1 TO 4)
 240 PRINT "2. PRICE?"
 250 INPUT X
 260 LET A$(A,13-LEN STR$ X TO 1
2)=STR$ X
 270 PRINT "3. FIRST MORTGAGE PR
ICE?"
 280 INPUT X
 290 LET A$(A,21-LEN STR$ X TO 2
0)=STR$ X
 300 PRINT "4. 1ST MTG. LOAN RAT
E?"
 310 INPUT X
 320 LET A$(A,31-LEN STR$ X TO 3
0)=STR$ X
 330 PRINT "5. TYPE OF LOAN (VA-
FHA-CONV.)?"
 340 INPUT A$(A,32 TO 35)
 350 PRINT "6. PROPERTY ADDRESS?
"
 360 INPUT A$(A,37 TO 67)
 370 PRINT "7. CITY?"
 380 INPUT A$(A,69 TO 83)
 390 PRINT "8. AGENCY OFFERING P
ROPERTY?"
 400 INPUT A$(A,85 TO 91)
 410 PRINT "9. AGENCIES PHONE NU
MBER?"
 420 INPUT A$(A,93 TO 100)
 430 LET A=A+1
 440 GOTO 100
 450 SAVE "SPREADSHEET"
```

Wydruk 4. Program „Spreadsheet" (część napisana w języku BASIC)

liczbą, nie należy podejrzewać pomyłki zecera, gdyż wydruk 3 jest fotokopią wydruku działającego programu. Teraz może nam się przydać zapis na taśmie wprowadzanych liczb. Gdy uda się nam uzyskać poprawną sumę 29 192 możemy wprowadzić z klawiatury treść wydruku 4, tworzącego „szatę graficzną" naszego programu. Układ, rozmiar i treść można oczywiście dowolnie zmieniać w celu dostosowania programu do potrzeb i gustów użytkownika. Przed uruchomieniem programu należy wykasować wszystkie linie od 9000 do 9999, gdyż nie będą już nam potrzebne.

Po uruchomieniu, program zgłasza się planszą tytułową, proponując nam dwupozycyjne menu:
klawisz 1 — wprowadzenie danych
klawisz 2 — wyświetlenie tabeli.

Po naciśnięciu 1 ukazuje się treść rubryk, które kolejno zapełniamy, udzielając odpowiedzi na stawiane pytania. Korzystanie z zawartości tabeli możliwe jest po naciśnięciu klawisza 2. Naciskanie odpowiednich strzałek (bez SHIFT) umożliwia wybranie interesującego nas fragmentu tabeli.

W okienku ukazuje się 10 wierszy. Stosownie do potrzeb możemy okienko zwiększyć lub zmniejszyć (od 1 do 17 wierszy). Gdy zechcemy oglądać np. 15 wierszy, musimy dokonać następujących zmian: POKE 16371,15 oraz POKE 16657,60 — 15. Na tym jednak nie koniec, gdyż naruszona została struktura tablicy. Musimy także wprowadzić liczbę wierszy (tj. 15) do komórki o adresie 16814 — a więc POKE 16814,15.

Program ten zawiera też możliwość dokonywania zmian w rubryce „ADDRESS", dzięki istnieniu w programie wierszy od 180 do 192. Po naciśnięciu „C" ukazuje się napis „ENTER CORRECTED ADDRESS". Po wprowadzeniu nowego adresu zostanie on wpisany do pierwszego z aktualnie wyświetlanych wierszy. Dla wyróżnienia, pierwszy wiersz jest wyświetlany w postaci negatywowej.

W komórce o adresie 16640 umieszczona jest liczba 12, regulująca prędkość poruszania się „okienka" po tablicy. Liczba większa niż 12 spowolni przesuwanie się okienka, zaś liczba mniejsza przyśpieszy ruch kolumn i wierszy na ekranie. Umożliwia to dostosowanie prędkości przesuwania się okienka do własnych wymagań.

W przestawionej wersji programu tablica składa się z 60 wierszy i 100 kolumn. Poniższa tabelka podaje niezbędne zmiany jakie należy dokonać w celu uzyskania tablic o wymiarach: 50×115, 80×100 i 100×100.

Zmiany jakie należy wykonać w programie dla uzyskania tabeli o innych wymiarach, niż podane w przykładzie

Operacja	50×115	80×100	100×100
A$ DIM	(50, 115)	(80, 100)	(100, 100)
T$ DIM	(115)	(100)	100)
POKE 16552,	123	69	21
POKE 16576,	115	100	100
POKE 16657,	40	70	90
POKE 16713,	83	68	68
POKE 16749,	118	103	103
POKE 16837,	83	68	68

Po dokonaniu wskazanych w tabeli zmian w liniach 10 i 20 programu oraz po zmianie zawartości wymienionych komórek pamięci, musimy pamiętać jeszcze o przystosowaniu wierszy 40 i 60 do nowych rozmiarów tabeli.
Program przeznaczony jest dla ZX 81 z pamięcią 16 KB.

WITOLD MAJEWSKI
Wrocław

Niektórzy ze współpracowników mikroKLANU przeglądając jeszcze w maszynopisie tekst artykułu o sprzętowej realizacji poszerzenia możliwości graficznych ZX 81 mocno kręcili głowami: po co „wpuszczać" Czytelników w krajanie komputera, jeżeli istnieje gotowy program spełniający podobną funkcję w sposób „bezkrwawy".

Przeważył jednak argument, że skalpel i lutownica znajdują się praktycznie w zasięgu każdego posiadacza ZX 81, natomiast wspomniany program — niekoniecznie.

Okazuje się jednak, że wykonana przeróbka pozwala również... rozszerzyć możliwości programu „High Resolution Graphic" (ang. grafika z Dużą Rozdzielczością).

Jeszcze o możliwościach graficznych ZX 81 16 KB

Opisana w 14 mikroKLANIE przeróbka komputera ZX 81 16 KB umożliwiła wykonanie pierwszego kroku w kierunku poprawy jego możliwości graficznych. Dzięki stosunkowo prostej zmianie konfiguracji połączeń wewnętrznych udało się stworzyć możliwość przechowywania tablicy znaków w pamięci RAM, co pozwoliło na deklarację dowolnego ich zestawu. Tym, którzy zdecydowali się wykonać te zmiany, proponuję kolejne — tym razem już tylko programowe — rozwiązanie, pozwalające na dalsze rozszerzenie możliwości graficznych komputera. Rozwiązanie to wykorzystuje program „High Resolution Graphic" (dostępny w kraju dzięki rozwijającemu się prywatnemu importowi oprogramowania i sprzętu komputerowego).

Program HRG umożliwia uzyskanie rozdzielczości 192×256 z wyłączeniem niektórych punktów. Uzyskanie tak dużej poprawy rozdzielczości stało się możliwe dzięki przejęciu sterowania wyświetlaniem na ekranie i zmianie wymiaru każdego ze znaków z 8×8 na 1×8. Dodatkowo, program zmienia położenie tablicy znaków w ROM i z nowej tablicy wykorzystuje co ósmy bajt, jako znak graficzny. Pamięć obrazu zajmuje 6,2 KB pamięci i przechowywane są w niej kody znaków umożliwiające programowi odnalezienie położenia znaku w tablicy.

Istnieją jednak pewne ograniczenia programu. Brakuje niestety znaków symbolizujących niektóre pojedyncze punkty. Nie są też możliwe wszystkie kombinacje położenia punktów w znaku. Ograniczenia te można usunąć wprowadzając do programu HRG zmiany, wykorzystujące możliwości przerobionego komputera. Zasadą tych zmian jest przesunięcie tablicy znaków do RAM, zadeklarowanie odstępu i ośmiu znaków oznaczających kolejne punkty (tzn. wpisanie do pamięci co ósmego bajtu wartości: 0, 1, 2, 4, 8, 16, 32, 64, 128) oraz deklarowanie nowych znaków w miarę potrzeb wykreślonego rysunku. Oczywiście deklarowaniem tym zajmuje się specjalny podprogram, który najpierw sprawdza czy wymagany znak znajduje się już w pamięci i ewentualnie deklaruje go. Jak wykazała praktyka na 64 możliwe znaki, wykorzystanych zostaje:
17 znaków — przy wykreślaniu kwadratów i prostokątów
23 znaki — przy wykreślaniu funkcji sinus i cosinus
35 znaków — przy wykreślaniu okręgu.
Czas wykonywania rysunków (pod warunkiem napisania podprogramu deklarującego w kodzie maszynowym) jest w przybliżeniu taki sam, jak w przypadku oryginalnego programu.

Efekty? Sprawdźcie sami.

JACEK PSZONA
Warszawa

W poprzednim mikroKLANIE opisaliśmy algorytm definiowania dodatkowych komend rozszerzających BASIC w ZX SPECTRUM. W tej części artykułu publikujemy przykład konkretnej realizacji tego algorytmu. Wydruk procedury napisanej w języku ASSEMBLER Z80 można z łatwością wykorzystać jako „szkielet" własnych, być może o wiele ciekawszych niż prezentowana w przykładzie, instrukcji. Zachęcamy do prezentacji ich na naszych łamach.

Jak rozszerzyć BASIC ZX SPECTRUM (2)

Jak wspomniano w poprzedniej części tekstu, dla przejęcia sterowania od INTERPRETERA języka BASIC należy w programie umieścić pewien błędny zapis. W przykładzie wykorzystano klucz **SCREEN$**, który normalnie wykorzystywany jest w instrukcjach **LOAD** i **SAVE** dla odczytu i zapisu na taśmie obrazu z ekranu oraz dla odczytania znaku w określonym miejscu ekranu. Inne użycie tego klucza powoduje generację komunikatu błędu i przerwania realizacji programu.

Nowa instrukcja ma postać:

SCREEN$ p,k

gdzie p — to numer odpowiadający nowo definiowanemu kolorowi tła, natomiast k — to numer odpowiadający numerowi koloru nowo definiowanej kreski. Wykonanie instrukcji powoduje zmianę kolorów na całym ekranie. Za wyborem klucza SCREEN$ przemawia prosty sposób sprawdzania syntaktyki (poprawnego zapisu) nowej instrukcji, gdyż daje on pojedynczy kod przechowywany w pamięci obok argumentów. Analogiczną rolę mogą spełniać klucze od RND (kod: 165) do STEP (kod: 205).

Program przedstawiony na wydruku 1 może bez żadnych modyfikacji być wykorzystany zarówno dla wersji 16K jak i 48K. W instrukcjach DATA zawarto kod maszynowy. Uruchomienie programu powoduje umieszczenie kodu za pozycją wskazywaną przez adres przechowywany w zmiennej **RAMTOP.** Dla wersji 16K jest to lokacja o adresie 32401, natomiast dla wersji 48K jest to pozycja o adresie 65101. W trakcie przepisywania kodu sprawdzana jest suma wszystkich danych (powinna ona wynosić 16918). Jeżeli wprowadzony kod nie jest poprawny, przerywana jest realizacja programu. Wskazane jest jednak zarejestrowanie programu na taśmie przed pierwszą próbą uruchomienia, gdyż złośliwy błąd może dać prawidłową sumę kontrolną i równocześnie spowodować blokadę komputera. Jeżeli stracimy kontrolę nad programem, pozostanie jedynie wyłączenie i włączenie komputera — a wtedy mozolnie wprowadzone dane przepadną.

W drugiej części programu demonstrowany jest przykład ilustrujący sposób działania nowej instrukcji **SCREEN$.**

```
  1 REM ********************************
  2 REM ** "SCREEN$" - jako komenda **
  3 REM ********************************
 10 CLEAR 32400+32700*(PEEK 23733=255)
 15 REM
 16 REM ustawianie RAMTOP zaleznie od wersji (16K lub 48K)
 17 REM
 20 LET start=1+PEEK 23730+256*PEEK 23731
 25 REM
 26 REM Kod bedzie wprowadzony za RAMTOP
 27 REM
 30 RESTORE : LET sum=0
 40 FOR j=start TO start+157
 50 READ n: POKE j,n
 60 LET sum=sum+n
 70 NEXT j
 80 IF sum<>16918 THEN PRINT "Blad sumy kontrolnej": STOP
 90 REM ------------------------------------------------
410 DATA 58,58,92,254,11,40,35,253,203,1,126,32,8,42,178,92,35
411 DATA 229,195,183,18,205,3,19
412 DATA 253,54,0,255,42,89,92,205,167,17,42,178,92,35,229,195
413 DATA 180,18,42,93,92,43,126,254
414 DATA 170,32,212,253,54,0,255,253,54,38,0,205,122,28,253
415 DATA 203,0,126,40,195,254,13,40,8
416 DATA 253,54,0,11,254,58,32,183,253,54,0,255,253,203,1,126
417 DATA 32,12,42,178,92,35,229,33
418 DATA 183,18,229,195,118,27,42,141,92,34,143,92,205,148,30
419 DATA 254,8,40,5,87,55,205,53,34
420 DATA 205,148,30,254,8,40,5,87,167,205,53,34,205,173,28,58
421 DATA 141,92,33,0,88,17,1,88
422 DATA 1,255,2,119,237,176,42,178,92,35,229,195,118,27
430 REM ------------------------------------------------
495 REM
496 REM Ustawianie adresu "nowej" obslugi bledu
497 REM
500 LET errsp=PEEK 23613+256*PEEK 23614
510 POKE errsp,start-256*INT (start/256)
520 POKE errsp+1,INT (start/256)
601 REM ************************************************
602 REM ************************************************
603 REM * Nalezy wykonac dotychczas wprowadzone linie  *
604 REM * dla sprawdzenia poprawnosci wprowadzonego kodu *
605 REM * przed wprowadzeniem pozostalych              *
606 REM ************************************************
607 REM ************************************************
608 REM
700 CLS : LIST 740:SCREEN$ 5,1
710 FOR j=1 TO 3
720 PRINT '"CAPSSHIFT & BREAK zatrzymuja"'"          Program"
730 NEXT j
740 FOR j=0 TO 7
750 FOR k=0 TO 7
760 PAUSE 20
770 IF j=k THEN NEXT k
900 SCREEN$ j,k
910 NEXT k
920 NEXT j
930 GO TO 740
1000 STOP
```

Wydruk 1.

```
          ORG   RAMTOP+1     ;Procedura jest relokowalna, musi jednak
                             ;byc ladowana zaraz ponad RAMTOP
START     LD    A,(26610)    ;Czy kod bledu = 11 (Nonsense in BASIC)
          CP    11
          JR    NZ,NONSENSE
ERROR     BIT   7,(IY+1)     ;Bit 7 zmiennej FLAGS jest ustawiony
          JR    NZ,RUNERROR  ;tylko w trybie realizacji programu
SYNTXER   LD    HL,(23730)   ;Blad syntaktyczny - START = RAMTOP+1
          INC   HL           ;START odkladany na stos - gotowy na
          PUSH  HL           ;nastepny blad
          JP    12B4H        ;Powrot do ROM
RUNERR    CALL  1303H        ;Blad realizacji - wypisz komunikat
          LD    (IY+0),255   ;Zerowanie numeru bledu
          LD    HL,(23641)   ;Usuniecie zapisow zmiennoprzecinkowych
          CALL  11A7H        ;z obszaru edytowanego przed
          LD    HL,(23730)   ;sprawdzeniem syntaktyki
          INC   HL           ;START odkladany na stos
          PUSH  HL
          JP    12B4H        ;Powrot do ROM
NONSENS   LD    HL,(23645)   ;W CH ADD znajduje sie adres do ktorego
          DEC   HL           ;dotarl interpreter. Pobranie znaku,
          LD    A,(HL)       ;ktory byl przyczyna bledu
          CP    170          ;Czy to byl SCREEN$ ?
          JR    NZ,ERROR     ;Skok pod ERROR jezeli nie
          LD    (IY+0),255   ;Zerowanie ERR NR i X PTR
          LD    (IY+38),0
          CALL  1C7AH        ;Sprawdzenie czy nastepne w linii sa
          BIT   7,(IY+0)     ;dwie liczby oddzielone przecinkiem;
          JR    Z,ERROR      ;jezeli nie to ERR NR wskazuje blad.. CH-
          CP    13           ;ADD jest przesuwany a do A wprowadzany
          JR    Z,OK         ;jest nastepny znak (ENTER lub dwukropek
          LD    (IY+0),11    ;gdyz innaczej blad: "Nonsense in
          CP    58           ;BASIC")
          JR    NZ,ERROR
OK        LD    (IY+0),255   ;Jezeli syntaktyka jest poprawna . to
                             ;zerowanie ERR NR
          BIT   7,(IY+1)     ;Jezeli tryb realizacji to "nowa"
          JR    NZ,DO-IT     ;instrukcja moze teraz byc wykonana
          LD    HL,(23730)   ;W innym przypadku adresy START i 12B7H
          INC   HL           ;sa odkladane na stos i ma miejsce
          PUSH  HL           ;powrot do ROM-u
          LD    HL,12B7H
          PUSH  HL
          JP    1B76H
DO-IT     LD    HL,(23693)   ;Realizacja. Na wstepie kolory "stale"
          LD    (23695),HL   ;kopiowane sa do "tymczasowych". Kolor
          CALL  1E94H        ;kreski pobierany jest ze stosu. Jesli
          CP    8            ;jest = 8 to bez zmian, jesli nie to
          JR    Z,PAPER      ;procedura z ROM-u wykorzystywana jest
          LD    D,A          ;do zmiany ATTR T kreski
          SCF
          CALL  2235H
PAPER     CALL  1E94H        ;Kolor papieru pobierany jest ze stosu i
          CP    8            ;jezeli nie jest = 8 to powtornie
          JR    Z,OUT        ;wywolywana jest procedura z ROM-u
          LD    D,A
          AND   A
          CALL  2235H
OUT       CALL  1CADH        ;"Tymczasowe" kolory przechodza na
          LD    A,(23693)    ;stale. Rozkaz LDIR wykorzystywany jest
          LD    HL,5800H     ;do ustawienia atrybutow koloru takich
          LD    DE,5801H     ;samych jak ATTR P.
          LD    BC,2FFH
          LD    (HL),A
          LDIR
          LD    HL,(23730)   ;START odkladany jest na stos
          INC   HL           ;przygotowujac obsluge nastepnego bledu.
          PUSH  HL
          JP    1B76H        ;Powrot do ROM-u.
```

Wydruk 2.

Jeżeli program funkcjonuje prawidłowo, powinny mieć miejsce sekwencyjne zmiany kolorów tła i kreski. Realizację programu można przerwać w dowolnym momencie, naciskając równocześnie klawisze **BREAK** i **CAPS SHIFT.**

Użyteczne procedury w ROM

Adres procedury (szesnastkowo)	Wykonywane działania
18H	RST 18H ładuje kolejny znak wskazywany przez **CH ADD** do akumulatora. Znaki niedrukowalne (np. kody kolorów) są ignorowane, a wskaźnik w **CH ADD** jest przesuwany aż do odczytania „normalnego" znaku.
20H	RST 20H powoduje inkrementację wskaźnika w **CH ADD** i wprowadzenie kolejnego znaku do akumulatora.
1C82H	Procedura wyznacza wartość wyrażenia numerycznego i jeżeli komputer pracuje w trybie realizacji wpisuje ją na stos kalkulatora. Przed wywołaniem procedury zmienna **CH ADD** powinna wskazywać adres pierwszego znaku wyrażenia. Po wyjściu z procedury **CH ADD** wskazuje pierwszy znak za wyrażeniem. Jeżeli wyrażenie było numeryczne, to bit 6 w zmiennej **FLAGS** jest ustawiony na 1.
1C7AH	Procedura wyznacza wartości dwóch wyrażeń numerycznych, rozdzielonych przecinkiem i wpisuje je na stos kalkulatora. **CH ADD** i **FLAGS** — identycznie jak dla 1C82H.
1E94H	Procedura pobiera liczbę ze stosu kalkulatora i wpisuje ją do akumulatora. Liczba musi być dodatnia i mniejsza od 256.
1E99H	Pobranie liczby ze stosu kalkulatora i wprowadzenie jej do pary rejestrów BC. Liczba musi być dodatnia i mniejsza od 65536.

Na wydruku 2 przedstawiono postać symboliczną procedury zapisanej w języku ASSEMBLER Z80. Realizuje ona algorytm przedstawiony w pierwszej części artykułu.

Zamiast klucza produkującego pojedynczy kod, nowa instrukcja może być też zapisywana w sposób bezpośredni (tzn. litera po literze). Aby możliwe było wpisanie takiej instrukcji należy ją poprzedzić znacznikiem powodującym zmianę trybu wprowadzania przez komputer (z kursora K na L). Znacznikiem takim może być np. *. Syntaktyka (poprawność) takiej instrukcji również musi być sprawdzana litera po literze. Wykrycie niezgodności zapisu w programie ze wzorcem powoduje powrót do standardowej procedury sygnalizującej błąd. Przeglądanie zapisu instrukcji najprościej można zrealizować wykorzystując wskaźnik **CH ADD** (adres: 23645) i zawarte w ROM procedury RST 18H i RST 20H. Adresy użytecznych procedur z firmowego ROM przedstawiono w tabeli. Procedura sprawdzająca syntaktykę powinna również kontrolować czy po parametrach instrukcji podano znacznik końca (ENTER lub dwukropek). Po zakończeniu sprawdzania adres zapisany w **CH ADD** powinien wskazywać znacznik końca — w innym przypadku zostanie zakłócona rutynowa praca INTERPRETERA języka BASIC.

Po każdym wywołaniu procedura pozostawia na stosie wartości pozwalające na ponowne odwołanie się do nowej instrukcji. Uruchomienie programu przez dyrektywę RUN zeruje jednak stos i wprowadza „normalny" adres obsługi błędu. Dlatego też przed pierwszym odwołaniem do nowej instrukcji należy wykonać instrukcję POKE odpowiednie zmiany, tak jak to pokazano w przykładowym programie zamieszczonym na wydruku 1.

MAREK GÓRECKI
ANDRZEJ J. PIOTROWSKI
Warszawa

W tym numerze mikroKLANU prezentujemy ostatni z regularnych wykładów Akademii, publikując informatyczną odmianę praw Murphy'ego. Są one niejako kwintesencją omawianych do tej pory reguł. Chociaż wiele z nich brzmi humorystycznie, to jednak poważne ich traktowanie może oszczędzić wielu rozczarowań i kłopotów.

Ponawiamy propozycję publikowania nadesłanych nam przez Czytelników krótkich procedur w języku BASIC. Liczymy naturalnie, że będą one nie tylko „chodzące", ale też zapisane w zgodzie z propagowanym przez nas eleganckim stylem programowania. I jeszcze jedno: nie planujemy publikowania procedur typu „Metoda obliczania pogłowia trzody chlewnej z wykorzystaniem testu chi-kwadrat" — choćby autor zarzekał się, że spowodujemy tym zniesienie reglamentacji mięsa. Krąg czytelników mikroKLANU stale się poszerza (niezależnie od nakładu limitowanego przez znacznie ważniejsze „czynniki" niż popyt) i musimy dobierać tematy, które zainteresują rozleglejsze grono (a nie tylko autora procedury i jego kolegę z pokoju). Jeżeli jednak ktoś nie jest pewien czy jego program jest interesujący — to mimo wszystko radzimy zaryzykować kilkanaście złotych na znaczek pocztowy.

Akademia mikroKLANU (8)

Kończąc cykl „akademicki" mamy nadzieję, że pozwolił on Czytelnikom mikroKLANU nieco udoskonalić technikę programowania. By Akademii nie zamienić w zbyt szacowny zakład naukowy kończymy kolekcją różnych zasad i przysłów, popularnych wśród programistów. Przyjemnego programowania.

1. Podstawowe prawa arytmetyki komputerowej:
   a) A=A±coś na ostatnim miejscu
   b) A+B≠B+A
   c) A*B≠B*A
   d) A*(B+C)≠(A*B)+(B*C)
   e) $A^n \neq A*A*...*A$ (n razy)
   f) odwrócenie operacji nie daje poprzedniej liczby
2. Komputery robią to, co się im każe robić, a nie to, co chcielibyśmy by robiły
3. Nikt nie będzie miał tak dobrej opinii o Twoim programie jak Ty sam.
4. Każdy program zawiera błędy.
5. Pierwsza osoba, która użyje Twego programu, znajdzie w nim błąd.
6. Żaden program nie jest dostatecznie zabezpieczony przed użytkownikami.
7. Żaden program nie działa za pierwszym razem. Bądź ostrożny, jeżeli już przy pierwszej realizacji dostałeś poprawną odpowiedź.
8. Kiedy wszystko inne zawodzi przy poprawianiu programu, wytłumacz problem siedmioletniemu dziecku.
9. Im dłużej trwało poprawianie programu tym pewniejsze, że nie został on bezbłędnie zarejestrowany w pamięci masowej.
10. Każde pół godziny analizy oszczędza dwie godziny programowania.
11. Użyteczność programu jest odwrotnie proporcjonalna do czasu poświęconego na jego przygotowanie.
12. Łatwiej napisać nowy program niż zmodyfikować stary.
13. Każdy problem ma więcej niż jedno rozwiązanie. Wniosek: żadne dwa programy nie są jednakowe.
14. 95 procent czasu poświęca się na rozwiązanie pięciu procent problemu.
15. Większość problemów z komputerem wynika z niedokładnego przestudiowania instrukcji obsługi.
16. Instrukcje są bezużyteczne, jeśli masz praktyczny problem.
17. Zmienne są stałe, stałe są zmienne.
18. Liczby całkowite nie są całkowite.

19. Zmienne zawsze są znane z większą dokładnością niż stałe.
20. Użycie większej liczby cyfr znaczących uczyni Twoje wyniki bardziej wiarygodnymi.
21. Żaden komputer nie będzie miał nigdy wszystkich zalet o jakich marzysz.
22. Zwiększenie pamięci o połowę podwaja możliwości komputera.
23. Zawsze są co najmniej dwa sposoby skrócenia każdego programu.
24. Nigdy nie będziesz miał wystarczającej pojemności pamięci.
25. Każdy program można poprawić.
26. Długo wykonujący się program zawsze „pada" na samym końcu.
27. Wewnątrz każdego dużego programu jest mały program, który chce zniszczyć efekty dużego.
28. Podstawowe prawa programowania:
    a) każdy program gdy już działa, jest niepotrzebny
    b) każdy program kosztuje więcej i przygotowywany jest dłużej niż zaplanowano
    c) jeżeli program jest pożyteczny, trzeba będzie go zmodyfikować
    d) jeżeli program jest bezużyteczny, trzeba go udokumentować.
29. Wszystko powinno być tak proste jak to jest tylko możliwe — ale nie bardziej.
30. Użycie skomplikowanego komputera nie tłumaczy trywialności problemu.

JAKUB TATARKIEWICZ

prowadzi
ANDRZEJ J. PIOTROWSKI
tel. dom.: 48-22-85

Jak grzyby po deszczu powstają w kraju firmy oferujące komputery kompatybilne z IBM PC (najprostsza konfiguracja od 4 mln. zł). Typowa, nowo powstająca firma zbiera zamówienia przez kilkanaście tygodni, tzn. aż... nie wyczerpie się limit danej grupy podatkowej. Realizacja dalszych zamówień wiązałaby się z koniecznością podwyższenia i tak niemałej ceny. Spóźnieni otrzymują terminy na następny rok.

Dysponenci dewiz mogą nabyć taki komputer od firmy ENSCH z Luxemburga (reprezentowana przez MERCOMP) już od 1500 dol. Natomiast zamożniejsi hobbyści składają IBM-y (płytki z Hong Kongu) inwestując poniżej tysiąca dolarów.

Natomiast firma CROMEMCO oferuje na polski rynek mikrokomputery zbudowane na M68000 wraz z systemem UNIX, sztywnym dyskiem (od 50 do 150 MB) i pamięcią RAM od 2 MB za ok. 10 tys. dolarów.

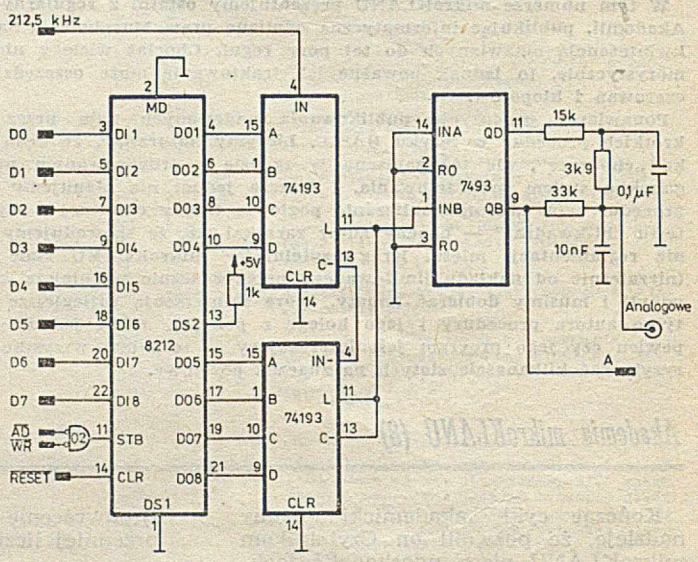
W poprzednim numerze mikroKLANU opisywaliśmy układ scalony, który można traktować niemal jak procesor dźwięków. Układ ten dostępny jest jednak głównie w strefie dolarowej (czasami można go dostać na „perskim") i dlatego też publikujemy obecnie rozwiązanie, które wprawdzie daje znacznie mniejsze możliwości, ale za to można je zrealizować na łatwych do zdobycia elementach.

Prezentowany pomysł jest na tyle prosty, że z nieznacznymi modyfikacjami można go wykorzystać niemal w dowolnej konstrukcji. Zamiast układu 8212 można wykorzystać „wolny" port 8255 a nawet interfejs równoległy do drukarki (jak drukujemy, to nie gramy, i odwrotnie).

Układ syntezy dźwięków

Niemal wszystkie mikrokomputery sprzedawane na rynkach zachodnich zawierają rozbudowane układy dźwiękowe. Służą one do sygnalizacji pracy systemu oraz do „muzycznej oprawy" programów, a czasem nawet do syntezy mowy. Konstrukcje te zawierają specjalizowane układy, procesory dźwiękowe (np. AY-3-8910). Poniżej proponujemy prosty układ syntezy dźwięków, który choć nie ma tak wielu zalet jak układy specjalizowane, jest łatwy do zrealizowania (krajowe elementy) i ma tę właściwość, że otrzymany sygnał jest zbliżony kształtem do sinusoidy.

Układ programowany jest przez wpisanie odpowiedniej wartości do układu 8212 (74S412). Zapis danej z linii D0...D7 następuje po wymuszeniu stanów aktywnych na wejściach WR/ i AD/. Wejście AD/ sterowane jest z dekodera adresów przestrzeni we-wy. Wpisany do portu bajt, określa wartość początkową programowanego dzielnika częstotliwości. Dzielnik zbudowany jest z liczników 74193, liczących „w tył". Liczniki sterowane są sygnałem otrzymywanym z zegara systemowego (2,125 MHz), przez dodatkowy dzielnik 1 : 10. Sygnał „pożyczki" z dzielnika programowanego (c—) wpisuje do liczników informację z bufora 8212, a także steruje licznikiem 7493. Licznik ten generuje przebieg o wypełnieniu 50%, którym sterowany jest układ formowania sygnału zbudowany z elementów RC. Na nich wytwarzany jest sygnał wyjściowy, którego częstotliwość podstawową wyznacza wyjście QD licznika. Wyjście QB wzbogaca go w częstotliwości harmoniczne. W efekcie otrzymujemy sygnał analogowy przypominający sinusoidę. Przy częstotliwości 212,5 kHz sterującej układ programowanego dzielnika, na wyjściu uzyskuje się częstotliwość od ok. 880 Hz. W zakresie słyszalnym umożliwia to uzyskanie, z pewnym przybliżeniem, dwóch oktaw.



Układ ten został zastosowany w komputerze gdzie z wyjścia A pobierany jest sygnał sterujący głośnik kontrolny klawiatury, natomiast sygnał analogowy jest podawany na gniazdo typu „jack" i wzmacniacz zewnętrzny.

TOMASZ PIETRZAK

LITERATURA

[1] Pienkos J., Turczyński J.: Układy scalone TTL w systemach cyfrowych, WKŁ 1980 r.
[2] Informacje techniczne ITE: Układ 74S412.

dokończenie ze str. 12

nieważ ocena negatywna z informatyki jako przedmiotu fakultatywnego nie jest przeszkodą w uzyskaniu promocji do klasy następnej.

Pełniejsze omówienie wyników ankiety przeprowadzonej wśród kandydatów na studia, jak również wśród nauczycieli przedstawiono w pracy [2].

WNIOSKI I PROPOZYCJE

Przeprowadzone badania pozwalają na wyciągnięcie wielu wniosków, dotyczących doskonalenia procesu nauczania informatyki w szkołach średnich. W pierwszej kolejności należy stwierdzić, że zarówno kandydaci jak i nauczyciele pozytywnie wypowiedzieli się o wprowadzeniu do nauczania w szkołach średnich tego przedmiotu. Jednocześnie padły następujące postulaty:

1. Przedmiot informatyka winien być nauczany w dwóch ostatnich klasach szkół średnich jako zajęcie obowiązkowe w wymiarze 2 godzin tygodniowo.

2. Do obecnego programu nauczania informatyki należy opracować podręcznik oraz przewodnik metodyczny dla nauczyciela.

3. Należy — w miarę możliwości — zapewnić testowanie programów pisanych przez uczniów na komputerze.

4. W szkole należy zapewnić niezbędne pomoce dydaktyczne ze szczególnym uwzględnieniem:
— filmów, przeźroczy, modeli komputerów
— zdjęć, rysunkowych schematów działania komputerów
— wydruków, taśm oraz kart perforowanych
— elementów komputera, zwłaszcza pamięci.

5. Wskazane byłoby nawiązanie przez szkołę współpracy z ośrodkami obliczeniowymi, a w miarę możliwości z ośrodkami obliczeniowymi szkół wyższych, w celu zapoznania uczniów z organizacją ośrodków, wyposażeniem i zakresem działania.

6. Należy zapewnić nauczycielom informatyki fachową opiekę metodyczną.

Oprócz wyżej wymienionych wniosków można sugerować następujące propozycje:
- Celowe byłoby systematyczne wyposażanie szkół w mikrokomputery, np. typu ZX SPECTRUM, przeznaczone do celów dydaktycznych.
- Nauczaniem informatyki w szkole średniej powinni zajmować się wyłącznie nauczyciele odpowiednio przygotowani. W okresie przejściowym nauczyciele informatyki mogą być kształceni na studiach podyplomowych z informatyki, bądź w ramach dodatkowej specjalności — na kierunku matematycznym w uczelniach kształcących nauczycieli. Należy jednak zmierzać do powołania w szkołach wyższych nauczycielskich kierunków informatycznych. W trakcie studiów kandydaci na nauczycieli informatyki winni mieć dostęp do komputerów przeznaczonych i odpowiednio przystosowanych do celów dydaktycznych.
- Niezbędne jest podjęcie prac nad dydaktyką informatyki, a w szczególności nad zagadnieniami dydaktycznymi z zakresu nauczania informatyki w szkołach średnich, a może nawet i podstawowych oraz w szkołach kształcących nauczycieli.

PROGRAM WSPÓŁPRACY ZE SZKOŁAMI ŚREDNIMI

Uwzględniając wnioski wypływające z przeprowadzonych badań ankietowych oraz apel Rady Ośrodków Informatyki Szkół Wyższych podległych MNSzWiT, Pracownia Obliczeniowa Wyższej Szkoły Pedagogicznej w Rzeszowie w porozumieniu z Kuratorium Oświaty i Wychowania opracowała program współpracy w roku 1984—1985 ze szkołami średnimi województwa rzeszowskiego w zakresie nauczania informatyki. W ramach tego programu przewidziano:

1. Zorganizowanie kursu programowania dla nauczycieli zainteresowanych nauczaniem informatyki w szkole średniej.

2. Zorganizowanie spotkań z uczniami w celu:
- pokazu Pracowni obliczeniowej i zgromadzonego tam sprzętu, zwłaszcza urządzeń peryferyjnych komputera oraz urządzeń do perforacji taśm
- zapoznanie z organizacją pracy Pracowni obliczeniowej oraz z technologią przetwarzania danych, na przykładach rozwiązywanych w Pracowni problemów
- uruchamiania w trybie bezpośrednim napisanych przez uczniów programów do starannie dobranych, zaproponowanych w zasadzie przez nauczyciela, zadań.

3. Wypożyczenie lub przekazanie do wykorzystania pomocy dydaktycznych w postaci:
- materiałów pomocniczych, dotyczących organizacji i zasad działania znajdującego się w Pracowni sprzętu informatycznego, opisu języka programowania wraz ze zbiorem zadań oraz specjalnie opracowanego dla potrzeb dydaktyki wykazu filmów dydaktycznych z zakresu informatyki i dziedzin pokrewnych, rozpowszechnianych przez Okręgowe Przedsiębiorstwo Rozpowszechniania Filmów w Rzeszowie
- zdjęć modułów minikomputera
- wydruków programów oraz taśm perferowanych.

4. Zorganizowanie seminarium dla nauczycieli szkół średnich nt. problemów dydaktyki informatyki w szkołach średnich.

Należy podkreślić, że wyżej omówiony program współpracy jest pierwszą próbą takiej współpracy. Będzie on stanowił nie tylko pomoc nauczycielom, ale również pozwoli na zebranie doświadczeń, które zostaną wykorzystane dla doskonalenia prorgamu kształcenia nauczycieli oraz rozszerzenia tej współpracy.

Program kursu dla nauczycieli, trwający w semestrze zimowym 1984—1985 obejmował 35 godzin zajęć (wykłady, ćwiczenia) oraz pracę własną słuchaczy. W ramach zajęć własnych każdy słuchacz miał zaprojektować, uruchomić i udokumentować dwa programy napisane w wykładanym języku programowania. Pozytywny wynik testu kontrolnego, aktywny udział we wszystkich zajęciach oraz oddanie zleconych do zaprojektowania, uruchomienia i udokumentowania programów stanowiły podstawę do zaliczenia słuchaczowi kursu i wydania mu odpowiedniego zaświadczenia. Każdy słuchacz otrzymał też odpowiednie materiały pomocnicze, dotyczące opisu języka programowania wraz ze zbiorem zadań. Szczegółowe informacje związane z programem kursu zostały zamieszczone w „Informatorze kursu programowania", opracowanym specjalnie dla uczestników tego kursu.

Kurs był dla słuchaczy bezpłatny, prowadzący zajęcia wykonywali je w ramach pracy społecznej. Kurs został zorganizowany za zgodą rektora Wyższej Szkoły Pedagogicznej w Rzeszowie oraz w porozumieniu z Kuratorium Oświaty i Wychowania w Rzeszowie.

Do lutego 1985 r., odbyły się też dwa spotkania z uczniami szkół średnich. Zaprezentowano uczniom Pracownię obliczeniową, realizując zaplanowany program.

W chwili obecnej trudno jest dokonywać oceny współpracy Pracowni obliczeniowej ze szkołami średnimi z uwagi na krótki czas jej trwania. Niemniej daje się zauważyć wzrost zainteresowania nauczaniem informatyki zarówno u nauczycieli, jak też uczniów. W wielu szkołach województwa rzeszowskiego nauczyciele już zorganizowali lub zorganizują jeszcze w br. szkolnym kółka zainteresowań o tematyce informatycznej. Na zajęciach w tego rodzaju kółkach nauczyciele wykorzystują oraz planują wykorzystać materiały udostępnione przez Pracownię obliczeniową.

Planuje się przeprowadzenie sandażu opinii zainteresowanych nauczycieli oraz uczniów w celu uzyskania pełnej oceny wyników realizowanego obecnie programu współpracy. Przewiduje się także rozszerzenie programu współpracy o nowe formy.

LITERATURA

[1] Stachura B.: Założenia eksperymentu, wyniki badań ankietowych. INFORMATYKA, 1975, nr 10, s. 16—19
[2] Suraj Z.: On education of informatics in secondary schools in Rzeszów voivodship. Proc. of the International Seminar on Education in Application of Computer Science Means and Methods, 1984, September 19—22, Toruń
[3] Zeszyty naukowe IKN, Matematyka, cz. 2, Informatyka, WSiP, Warszawa 1981.

Komputerowy system układania rozkładu zajęć w szkole wyższej

Zastosowanie komputera do układania rozkładu zajęć jest tematem szeregu publikacji i eksperymentów prowadzonych w wielu krajach, w różnych ośrodkach naukowych i obliczeniowych [1]).

W Instytucie Informatyki Uniwersytetu Jagiellońskiego został zrealizowany w 1979 roku komputerowy system układania rozkładu zajęć w szkole wyższej.

Wykorzystany w systemie algorytm jest algorytmem heurystycznym, którego działanie można przedstawić bardzo ogólnie w następujący sposób. Na podstawie danych wejściowych są tworzone dwa zbiory: zbiór zajęć, dla których rozkład jest dany a'priori oraz zbiór zajęć do zaplanowania. Zbiory te są uporządkowane w kolejności malejących wartości priorytetów elementów. Priorytety zajęć do zaplanowania są wyliczane na podstawie odpowiednich współczynników przyjętych dla przedmiotów, grup i wykładowców.

Początkowo tworzony jest rozkład zawierający zajęcia, a'priori zaplanowane. Następnie algorytm próbuje zaplanować kolejno wszystkie zajęcia ze zbioru zajęć do zaplanowania, spełniając jednocześnie ograniczenia podane

[1]) Schmidt G., Ströhlein T.: Timetable construction — an annotated bibliography. The Computer Journal, vol. 23, 1980, s. 307—316

System Instytutu Informatyki UJ stwarza następujące możliwości:

— dowolny dobór czasu trwania jednostek dydaktycznych (godzin) i przerw, momentów ich rozpoczynania i kończenia

— uwzględnienie dowolnego podziału organizacyjnego studentów (podziału na lata, kierunki, specjalności, specjalizacje, grupy wykładowe, ćwiczeniowe, laboratoryjne itd.)

— uwzględnienie indywidualnych postulatów wykładowców dotyczących wyboru godzin i dni wolnych od zajęć oraz godzin, dni i sal, w których chcieliby prowadzić zajęcia

— uwzględnienie postulatów studentów (grup studenckich) dotyczących wyboru godzin i dni wolnych od zajęć oraz godzin, dni i sal, w których chcieliby odbywać zajęcia

— zablokowanie wybranych sal w wybranych dniach i godzinach

— planowanie wybranych zajęć w wybranych dniach, godzinach i salach, z wybranym wykładowcą

— przypisanie wybranych sal do wybranych zajęć

— określenie minimalnej i maksymalnej liczby godzin, jaka może być zrealizowana w ciągu dnia, w kolejnych godzinach, dla każdych zajęć

— zapewnienie przynajmniej dnia przerwy między kolejnymi zajęciami z tego samego przedmiotu

— zapewnienie równomiernego obciążenia dziennego studentów.

Dane wejściowe systemu są podzielone na zbiory: wykładowców, przedmiotów, podziału organizacyjnego studentów, zajęć oraz zawierające informacje sterujące wykonaniem programu i wyprowadzaniem wydruków, i informacje dotyczące podziału organizacyjnego dnia na jednostki dydaktyczne (godziny).

Program analizuje poprawność danych wejściowych, dostarczając bogatej, wszechstronnej diagnostyki błędów. Po przeanalizowaniu poprawności wszystkich danych, w przypadku wykrycia błędów, program kończy pracę.

PTI w sprawie wprowadzania informatyki do szkół

Bezpośrednio przed zamknięciem numeru redakcja otrzymała tekst zawierający stanowisko Prezydium Zarządu Głównego Polskiego Towarzystwa Informatycznego, w sprawie wprowadzania elementów informatyki do szkolnictwa średniego i podstawowego. Tekst ten publikujemy w dosłownym brzmieniu.

1. Przed przystąpieniem do działań praktycznych należy określić cele dydaktyczno-wychowawcze wprowadzenia elementów informatyki do szkolnictwa średniego i podstawowego.

Nie wydaje się, aby celem takim mogło być np. nauczenie programowania, zwłaszcza w prymitywnym języku, ani nauczenie jakiejkolwiek „wewnątrz-infomatycznej" umiejętności. Może natomiast nim być ukazanie przydatności stosowania środków i metod informatyki w pracy umysłowej człowieka.

Dlatego niepomiernie ważniejsze jest doprowadzenie do praktycznego stosowania elementów informatyki w nauczaniu tradycyjnych przedmiotów szkolnych, niż podjęcie nauczania przedmiotu „informatyka". W żadnym kraju, łącznie ze znacznie bardziej rozwiniętymi gospodarczo niż Polska, nie naucza się takiego przedmiotu w szkołach średnich ani podstawowych (poza niektórymi szkołami technicznymi).

2. Należy zdecydowanie przeciwstawić się werbalnym metodom wprowadzania elementów informatyki do szkół. Próby nauczania elementów informatyki bez dostatecznego wyposażenia komputerowego, zapewniającego każdemu uczniowi objętemu nauczaniem możliwość indywidualnej pracy z

Dane wyjściowe systemu zawierają następujące informacje:

— wybrane informacje statystyczne dotyczące zbioru wykładowców, zbioru sal, zbioru przedmiotów, podziału organizacyjnego studentów i zbioru zajęć do zaplanowania

— wartość wskaźnika jakości dla każdego rozkładu zajęć, otrzymanego w kolejnej iteracji algorytmu

— rozkład zajęć dla studentów (grup studenckich)

— rozkład zajęć dla wykładowców

— rozkład zajęć w salach

— zbiór zajęć nie zaplanowanych w końcowym rozkładzie, produkowany w przypadku nie zaplanowania wszystkich przewidzianych zajęć.

Parametry techniczne systemu

System układania rozkładu zajęć w szkole wyższej został zrealizowany w języku FORTRAN w systemie cyfrowym CYBER 72. Tekst programu składa się z około 1800 wierszy. Podstawowymi strukturami danych w systemie są pliki danych o dostępie bezpośrednim, tworzone w pamięci dyskowej. W tym przypadku liczba danych wejściowych jest ograniczona jedynie wielkością pamięci dyskowej. Umożliwia to wykonywanie praktycznie dowolnie dużych zadań. Doświadczenia uzyskane w warunkach Instytutu Informatyki Uniwersytetu Jagiellońskiego (12 jednostek dydaktycznych w ciągu dnia, 80 wykładowców, 25 grup studenckich, 70 wykładanych przedmiotów, 10 sal) wykazują, że średni czas potrzebny na uzyskanie rozkładu zajęć w danej iteracji algorytmu nie przekracza 2 minut. Średnia liczba zajęć nie zaplanowanych w danej iteracji algorytmu wynosi w tych warunkach około 2%, przy praktycznie maksymalnym wykorzystaniu sal. Prezentowany system układania rozkładu zajęć w szkole wyższej może być zrealizowany również w innych systemach cyfrowych, mających możliwość tworzenia plików o dostępie bezpośrednim.

w danych wejściowych. Zajęcia zaplanowane podczas układania rozkładu są usuwane ze zbioru zajęć do zaplanowania. Zajęcia nie zaplanowane podczas układania rozkładu są usuwane ze zbioru zajęć do zaplanowania i umieszczane w zbiorze zajęć nie zaplanowanych. Zbiór zajęć nie zaplanowanych jest uporządkowany według malejących wartości priorytetów elementów. Postępowanie takie jest kontynuowane dopóty, dopóki zbiór zajęć do zaplanowania nie jest zbiorem pustym. Algorytm umożliwia uzyskanie, w kolejnych itercjach, dowolnej liczby rozwiązań (rozkładów zajęć) i wybranie spośród nich rozwiązania najlepszego. Jako wskaźnik jakości rozwiązania, przyjęto liczbę godzin nie zaplanowanych w rozkładzie zajęć. Poszukiwanie kolejnych rozwiązań jest kontynuowane do momentu uzyskania rozwiązania końcowego, to znaczy rozkładu w którym zaplanowano wszystkie zajęcia, bądź też najlepszego rozkładu uzyskanego spośród zadanej liczby rozwiązań w danych wejściowych. Podczas układania kolejnych rozkładów zmieniana jest kolejność w zbiorze zajęć do zaplanowania, poprzez zmianę priorytetów elementów w każdej iteracji. W dowolnym momencie obliczeń są pamiętane najwyżej trzy rozkłady: początkowy, tworzony w bieżącej iteracji algorytmu i najlepszy spośród rozkładów uzyskanych w poprzednich iteracjach. Pamiętany jest również zbiór zajęć nie zaplanowanych w bieżącej iteracji algorytmu oraz zbiór zajęć nie zaplanowanych w najlepszym rozkładzie, spośród rozkładów uzyskanych w poprzednich iteracjach. Rozkład uzyskany w bieżącej iteracji algorytmu jest porównywany z najlepszym rozkładem spośród rozkładów uzyskanych w poprzednich iteracjach. Z dwóch porównywanych rozkładów zapamiętywany jest ten, w którym pozostało mniej nie zaplanowanych godzin. Po uzyskaniu rozwiązania końcowego drukowane są rozkłady zajęć dla grup, wykładowców i sal. W przypadku nie zaplanowania wszystkich przewidzianych zajęć, drukowany również jest zbiór zajęć nie zaplanowanych w końcowym rozkładzie.

MAREK SKOMOROWSKI
RYSZARD WOŹNIAK

Ogłoszenia • Ogłoszenia

Zakupimy wycofane elementy jc. ODRA 1204/1304 lub zamienimy na mikrokomputer „Spectrum". Amstrad. NAPRAWA MULTIMETRÓW V527/V640. Warszawa, tel. 47-22-57, 20-90-61 w. 93.

EO/891/K/85

Oprogramowanie do COMMODORE C-64 piszemy na zlecenie instytucji, osób prywatnych. 27-400 Ostrowiec, skrytka 40

EO/1103/K/85

komputerem, są skazane na niepowodzenie i nie powinny być podejmowane. Członkowie PTI pod żadnym pozorem nie powinni partycypować w tego rodzaju działalności sprzecznej z Uchwałą Zjazdu.

3. Uważamy za niezbędne przestrzec władze oświatowe PRL przed fatalnymi skutkami wprowadzania do szkół — jako urządzeń zalecanych do użytku szkolnego — urządzeń komputerowych nie akceptujących polskiego alfabetu i wymagających obcojęzycznych instrukcji. Wywoływanie przeświadczenia, że nowoczesność wymaga lekceważenia języka ojczystego jest pedagogicznym przestępstwem. Podobnie nie powinno się zezwalać na stosowanie w polskich szkołach komputerów o wyraźnie niższym standardzie technicznym (jakościowym i konfiguracyjnym) niż komputery stosowane w szkołach zagranicznych. Nie należy bowiem wpajać młodzieży przekonania o nieuchronnej lichocie krajowych rozwiązań technicznych, ani o tym, że byle zagraniczna tandeta jest dość dobra dla polskich szkół.

Rynek oświatowy w Polsce jest dostatecznie chłonny, by z czysto ekonomicznego punktu widzenia stać nas było na sformułowanie wyraźnych kryteriów dopuszczalności urządzeń informatycznych przeznaczonych dla szkół, kryteriów zgodnych z interesem polskiej oświaty i kultury.

Przejściowo, do czasu udostępnienia komputerów zgodnych z takimi kryteriami, można przyzwalać na użytkowanie sprzętu nie odpowiadającego warunkom jakościowym, ale tylko w ramach zajęć fakultatywnych i w kółkach zainteresowań, podkreślając amatorski charakter zajęć. Dopuszczanie substandardowego sprzętu do użytku szkolnego powinno zawsze odbywać się na ustalony okres (rok, najwyżej dwa); wydawanie zezwoleń powinno być surowo kontrolowane, a używanie nielicencjonowanego sprzętu — niedozwolone.

4. Należy rozpocząć intensywne przygotowania metodyczne i organizacyjne w dziedzinie wprowadzania elementów informatyki do nauczania przedmiotów szkolnych. Przygotowanie i dystrybucja odpowiedniego oprogramowania i dokumentacji użytkowej, a także wskazówek metodycznych jest przedsięwzięciem o ogromnej wadze i — niestety — koszcie. Jednakże, bez odpowiedniej podbudowy metodycznej i bez przygotowania odpowiednich programów komputerowych, wprowadzanie elementów informatyki do szkół po prostu nie ma sensu.

5. Rozsądek nakazuje poprzedzić powszechne wprowadzanie elementów informatyki okresem doświadczalnym, w którym gromadzone byłyby obserwacje z kilku czy kilkunastu szkół eksperymentalnych. Przedwczesna powszechność i obligatoryjność programów nauczania i zaleceń metodycznych zawiera ogromne ryzyko ośmieszających wręcz pomyłek.

Prezydium ZG PTI

Zastosowania informatyki w WRL

Z elektronicznej techniki obliczeniowej w końcu 1982 r. korzystało 4500, tzn. ok. 65% węgierskich jednostek gospodarczych. W porównaniu do roku poprzedniego liczba ta zwiększyła się o 28%. W przemyśle węgierskim komputery wykorzystuje się głównie do planowania zużycia surowców, środków produkcji i pracochłonności oraz do automatyzacji procesów technologicznych, m.in. w przemysłach włókienniczym i spożywczym. 70% obrabiarek do metali oddanych do eksploatacji w 1983 r. miało sterowanie numeryczne. Systemy informatyczne w budownictwie węgierskim stosowane są przede wszystkim w zarządzaniu procesami inwestycyjnymi. Przykładowymi zastosowaniami w rolnictwie były: przetwarzanie danych genetycznych, dotyczących trzody chlewnej i na tej podstawie — optymalizacja jej hodowli oraz intensyfikacja uprawy kukurydzy metodami przemysłowymi. Innym obszarem zastosowań jest dziedzina transportu. Systemy informatyczne pozwalają tu na optymalne rozwiązywanie zadań transportowych na szlakach o największym natężeniu ruchu. W jednostkach handlu zagranicznego wspomaganie komputerowe obejmuje procesy realizacji umów oraz ewidencję i kontrolę towarzyszących im przepływów pieniężnych i towarowych. Podobnie rozwiązuje się również ewidencję i kontrolę przepływów towarowych w handlu wewnętrznym.

Około 20% pracowników węgierskich jednostek gospodarczych ma stały kontakt z informatyką — dostarczając dane dla systemów informatycznych lub wykorzystując wyniki przetwarzania. Szybko rozwija się zastosowanie systemów automatyzacji projektowania technicznego (CAD), wykorzystywane w ok. 80 jednostkach organizacyjnych, z których połowę stanowią jednostki badawcze, a połowę przedsiębiorstwa przemysłowe. Rozwój tych zastosowań napotyka jednak na trudności wynikające przede wszystkim z braku niektórych urządzeń peryferyjnych komputerów — zwłaszcza ploterów i monitorów graficznych.

Charakterystycznym trendem zastosowania informatyki w zarządzaniu jest szybki wzrost liczby mikrokomputerów, które dzięki wyposażeniu w typowe oprogramowanie ułatwiają koordynację prac różnych jednostek i przyspieszają obieg informacji, a także sprzyjają ujednoliceniu form rachunku gospodarczego. Jednym z ważnych zastosowań jest centralny system ewidencji ludności, w ramach którego w 1984 r. zautomatyzowano m.in. obsługę meldunków mieszkaniowych.

Łączne nakłady inwestycyjne związane z wprowadzeniem w 1982 r. ETO w WRL wyniosły 3100 mln forintów, z czego 2400 mln wydatkowały samodzielne przedsiębiorstwa. W ten sposób wartość brutto środków technicznych ETO w końcu 1982 r. wzrosła do kwoty 22 600 mln forintów, czyli o 15% w stosunku do roku 1981. W roku 1982 wdrożono do eksploatacji ponad 700 nowych systemów informatycznych, a w pierwszej połowie 1983 r. — 240 systemów o wartości 1200 mln forintów.

W końcu 1982 r. park sprzętu komputerowego w WRL składał się z 920 dużych komputerów oraz 457 mini- i 1288 mikrokomputerów. Według danych posiadanych przez władze celne i finansowe, w marcu 1983 r. przywieziono z zagranicy ponad 2000 mikrokomputerów. Większość z nich stanowiły mało wydajne komputery domowe, natomiast ok. 200 było mikrokomputerami profesjonalnymi. Znaczna część tego sprzętu trafiła ostatecznie za pośrednictwem handlu państwowego i komisowego do przedsiębiorstw po cenach wprawdzie niższych od cen sprzętu produkcji węgierskiej, lecz znacznie przewyższających ceny światowe. W pierwszej połowie 1983 r. import sprzętu informatycznego tą drogą osiągnął wartość ok. 500 mln forintów.

Szybkie rozprzestrzenianie się mikrokomputerów we wszystkich obszarach gospodarki narodowej stwarza warunki do wykorzystywania ich bezpośrednio na stanowiskach pracy, a zarazem stymuluje dalszy wzrost zapotrzebowania na ten typ sprzętu. Przy zaspokajaniu tego zapotrzebowania wzrosło znaczenie nowo powstałych małych firm usług informatycznych.

Przeważająca część sprzętu stosowanego w WRL — z wyjątkiem mikrokomputerów — pochodzi z importu z krajów socjalistycznych oraz z produkcji krajowej. Import z krajów kapitalistycznych traktowany jest jako uzupełnienie dostaw z krajów socjalistycznych.

Charakterystyczną tendencją rozwoju parku komputerowego w WRL jest malejący udział dużych komputerów. Wśród 920 dużych komputerów 53% (490 szt.) zakupiono ponad 5 lat temu, 25% (235 szt.) jest już całkowicie zamortyzowana, a 8,7% (80 szt.) liczy sobie ponad 10 lat.

Odmienną strukturę wieku ma park mini- i mikrokomputerów. Tylko 25% (1700 szt.) ma więcej niż 5 lat, 8,5% (150 szt.) jest zamortyzowane, a zaledwie 3,2% (ok. 60 szt.) zakupiono ponad 10 lat temu.

Parametry techniczne oraz wielkość węgierskiego parku komputerowego nie umożliwiają jeszcze szerokiego stosowania nowoczesnych technologii przetwarzania danych, opartych na teleprzetwarzaniu w systemach przetwarzania rozproszonego.

Ceny sprzętu informatycznego węgierskiego i importowanego z krajów RWPG są 2—2,5 raza większe od aktualnych cen na rynku światowym. Z inicjatywy Węgierskiego Głównego Urzędu Statystycznego oraz Komitetu Postępu Technicznego, rozpoczęto w drugiej połowie 1983 r. pod nadzorem Urzędu Cen badanie możliwości obniżenia poziomu cen komputerów krajowych.

Dostawy oprogramowania, zwłaszcza dla komputerów Jednolitego Systemu i Systemu Minikomputerów, mają w WRL oparcie instytucjonalne w postaci Funduszu Rozwoju Zastosowań ETO. W dniu 31.12.1983 r. upłynęło pięć lat funkcjonowania tego funduszu. Uzyskane doświadczenia wskazują, że celowe jest przedłużenie działania funduszu, gdyż zakupy lub tworzenie powielarnego oprogramowania dla głównych grup użytkowników są zadaniami wymagającymi koordynacji merytorycznej oraz finansowej.

W roku 1982 ożywił się nieco handel oprogramowaniem między krajami socjalistycznymi. Wzrosło też znaczenie powielarnego oprogramowania oferowanego przez różnych dostawców węgierskich. W roku tym kwota obrotów oprogramowaniem wyniosła 530 mln forintów, z czego ok. 38% dotyczyło wymiany handlowej z krajami RWPG. W ciągu ostatnich dwóch lat rozpowszechnia się w WRL dzierżawa oprogramowania. Opłaty poniesione w 1982 r. z tytułu korzystania z tej formy udostępniania programów wyniosły 30 mln f. Równocześnie nastąpiła zauważalna poprawa poziomu usług w zakresie konserwacji oprogramowania, co stworzyło lepsze warunki dla stosowania oprogramowania powielarnego. Przychody z tytułu tego rodzaju usług w 1982 r. wyniosły 103 mln forintów, co stanowi ok. 25% łącznej kwoty obrotów oprogramowaniem na rynku krajowym.

Na ogólny poziom zastosowań informatyki w WRL silnie wpływa jakość pracy 80 przedsiębiorstw specjalizujących się w zagadnieniach organizacji i informatyki. Działalnością w tym zakresie zajmuje się łącznie ponad 800 jednostek organizacyjnych, w różnych gałęziach gospodarki narodowej. 130 spośród nich realizuje przetwarzanie danych i związane z tym usługi. Liczba tych jednostek w ostatnich latach rosła bardzo szybko dzięki polityce państwa, sprzyjającej tworzeniu małych firm i tzw. zespołów gospodarczych. O ile w 1981 r. na rynku usług informatycznych było ich zaledwie 37, to w 1983 r. było już 8 spółdzielni, 4 przedsiębiorstwa państwowe, 187 zespołów gospodarczych w takich przedsiębiorstwach oraz 80 spółek i 366 zespołów gospodarczych, działających na zasadach stworzonych przez nowe ustawodawstwo węgierskie. Usługi organizacyjne i informatyczne osiągnęły w 1982 r. łączną wartość obrotów 4100 mln forintów. W tym wartość usług wykonywanych przez małe firmy 260 mln forintów, tj. ok. 6,3%). Stanowi to wzrost o 33,5% w stosunku do 1981 r. Jeśli doliczyć do tego usługi serwisowe oraz towarzyszące usługi poligraficzne, to łączna wartość usług wyniosła w 1982 r. 7100 mln forintów.

Realizacja średnioterminowego państwowego planu prac rozwojowych za-

stosowań informatyki napotkała w okresie lat 1981—1982 duże trudności wynikające przede wszystkim z embarga na sprzęt i oprogramowanie pochodzące z krajów kapitalistycznych. Zahamowanie we wspomnianym okresie zastosowań w roku 1983 nie pogłębiało się. Obecnie trwają prace nad likwidacją powstałego opóźnienia.

Aby zapewnić szybszy rozwój zastosowań mikrokomputerów, Główny Urząd Statystyczny i Komitet Postępu Technicznego postanowiły ująć — w średnioterminowym państwowym planie prac rozwojowych — opracowanie typowych systemów informatycznych opartych na mikrokomputerach. Działając w tym kierunku, już w czerwcu 1983 r. rozstrzygnięto konkurs na systemy mikrokomputerowe, nadające się do powszechnego stosowania, a w ramach perspektywicznego państwowego planu prac badawczych opracowuje się środki techniczne oraz rozwiązania systemowe dla sieci komputerowych.

W Akademii Nauk WRL powstaje wiele rozwiązań z zakresu zastosowań komputerów w medycynie, biologii, automatycznym rozpoznawaniu obrazów, sterowaniu eksperymentami w fizyce jądrowej. Aby stworzyć warunki do rozwoju teleprzetwarzania, Główny Urząd Statystyczny zakupił — z funduszu zastosowań ETO — pewną liczbę terminali i procesorów telekomunikacyjnych wraz z odpowiednim oprogramowaniem, umożliwiając rozpoczęcie prac nad oprogramowaniem użytkowym dla różnych zastosowań. Przeszkodą dalszego rozwoju teletransmisji jest istniejące przeciążenie węgierskiej sieci telefonicznej.

W latach 1981—1983, corocznie 6000 uczniów szkół średnich przyswoiło sobie podstawy informatyki, a prawie 1700 otrzymało profesjonalne przygotowanie do wykonywania zawodu informatyka. Wyższe uczelnie węgierskie kończyło co roku ok. 2600 specjalistów informatyki. Około 1600 osób otrzymało informatyczne przeszkolenie uzupełniające, w tym część na specjalistycznych kursach zastosowań informatyki. W roku szkolnym 1982—1983, w celu wytworzenia powszechnego nawyku stosowania komputerów, w szkołach średnich rozpoczęto realizację programu pod nazwą „Szkolne komputery". W tym celu każda szkoła średnia otrzymała co najmniej jeden komputer. Ogłoszono także konkurs na opracowanie oprogramowania dostosowanego do potrzeb szkół średnich (do października 1983 r. na konkurs ten wpłynęło ponad 100 prac). Koszty realizacji programu wyniosły dotychczas 52,5 mln forintów.

Wartość produkcji krajowego sprzętu komputerowego w WRL wyniosła w 1982 r. 6153 mln forintów, a w pierwszej połowie 1983 r. — 3459 mln forintów. Produkcją tą zajmuje się ok. 30 przedsiębiorstw państwowych i spółdzielni. Od 1979 r. coraz wyraźniej widoczna jest dominacja zainteresowania sprzętem mikrokomputerowym. W połowie 1983 r. w WRL wytwarzano ok. 60 modeli i odmian mikrokomputerów, co stanowi ok. 30% produkcji tego typu urządzeń w krajach RWPG. Czynnikiem utrudniającym rozwój produkcji mikrokomputerów jest małoseryjność (20—100 sztuk), a także brak odpowiedniej liczby i niezadowalająca jakość takich urządzeń peryferyjnych, jak pamięci zewnętrzne i drukarki. Wartość eksportu sprzętu mikrokomputerowego na rynek krajów socjalistycznych wynosiła w 1982 r. ponad 3 mln forintów, podczas gdy import w tej klasie sprzętu wyniósł prawie 1,5 mln forintów. Eksport na rynki krajów kapitalistycznych wyniósł w tym samym roku 400 mln forintów, przy imporcie z tego obszaru około 600 mln forintów. Eksport oprogramowania na te rynki wyniósł 130 mln forintów (wzrost o 15% w stosunku do 1981 r.).

Węgierski przemysł komputerowy, współpracując głównie z krajami socjalistycznymi, w ramach Komisji Międzyrządowej ds. ETO, zainteresowany jest także współpracą z krajami zachodnimi. Konsekwencją tego jest rozwój porozumień z firmami francuskimi, japońskimi i zachodnioniemieckimi.

Dużo uwagi poświęca się w WRL tworzeniu warunków ekonomicznych, prawnych i organizacyjnych oraz kształtowaniu świadomości społecznej, sprzyjających rozwojowi zastosowań informatyki. Po dokonaniu w 1983 r. analizy zasad tworzenia cen w zakresie usług informatycznych, dokonano uproszczenia metod kalkulacji cen oprogramowania i usług towarzyszących oraz obniżono o około 10—15% ceny maszynowego przetwarzania danych. Wydano również rozporządzenie w sprawie ochrony prawnej oprogramowania, które nie tylko zwiększa zainteresowanie krajowych twórców oprogramowaniem powielarnym, ale również sprzyja importowi najnowszych produktów programowych z krajów zachodnich.

Rozwój zastosowań informatyki stymuluje zainteresowanie tą dziedziną przez szerokie kręgi społeczeństwa. Ponad 10 tys. osób uczestniczyło w 1983 r. w seminariach, organizowanych przez Towarzystwo im. J. von Neumanna. Główny Urząd Statystyczny zorganizował wielką konferencję pod nazwą „Stan zastosowań informatyki i ich rozwój w WRL". Większą niż dotychczas rolę w informowaniu społeczeństwa o informatyce powinny w przyszłości odegrać stowarzyszenia naukowe i inżynierskie, Towarzystwo Popularyzacji Nauk, domy kultury oraz oczywiście, środki masowego przekazu.

Opracował M. SIEDLECKI
na podstawie materiałów Węgierskiej Części Rady ds. Zastosowań Środków Techniki Obliczeniowej

Aktualne problemy ochrony oprogramowania

Gwałtowny rozwój technik mikroprocesowych i szczególnie, szybko rosnący w krajach zachodnich rynek komputerów osobistych i domowych dały nowy impuls wysiłkom zmierzającym do ochrony oprogramowania. Chodzi tu nie tylko o „włamania" do systemów komputerowych i związane z tym problemy bezpieczeństwa państwowego i szpiegostwa przemysłowego, zwłaszcza w kontekście zaostrzających się stosunków wschód-zachód, ale przede wszystkim jest to rozgrywka o olbrzymie kwoty pieniędzy. Na oprogramowaniu mini- i mikrokomputerów łatwo jest zrobić fortunę, łatwo też o wielkie, ale nielegalne zyski. O ile trudno sobie wyobrazić kradzież systemu operacyjnego dla komputerów CDC, albo oprogramowania dla superkomputera CRAY, o tyle w przypadku mini- i mikrokomputerów kradzieże oprogramowania są na porządku dziennym. Straty jednostkowe są co prawda mniejsze, ale występując bardzo często, kumulują się w wielomiliardowe sumy.

Producenci oprogramowania najczęściej określają liczbę kopii jaką może wykonać nabywca na potrzeby własne, natomiast powielanie oprogramowania ponad tę liczbę, a szczególnie odstępowanie go osobom trzecim, jest już przestępstwem. Problem stał się drastyczny od czasu, gdy oprogramowanie sprzedawane jest na znormalizowanych dyskach elastycznych, albo, co gorsze, na typowych kasetach magnetofonowych, gdyż w tym przypadku program może skopiować dosłownie każdy. Nic dziwnego, że pojawiło się wielu, którzy próbują to z zyskiem robić na skalę przemysłową. Jak wielki to problem i jakie straty ponoszą producenci oprogramowania łatwo sobie wyobrazić nawet w warunkach polskich, jeśli weźmiemy pod uwagę tylko takie popularne w naszym kraju (a całkowicie bezdewizowe...) produkty programowe jak system CP/M, czy oprogramowanie do mikrokomputerów

ZX 81 i ZX SPECTRUM firmy SINCLAIR.

W większości krajów zachodnich prowadzone są obecnie prace nad propozycjami całkowicie nowych aktów prawnych lub modyfikacją prawa autorskiego, zmierzające do zapewnienia skutecznej ochrony prawnej oprogramowania. W sytuacji, gdy straty są szacowane w miliardach dolarów, 25 mln dol., jakie EWG zamierza wydać w najbliższym czasie na ujednolicenie i modyfikację niezbędnych aktów prawnych, jest sumą niewielką. Jak bardzo złożone są te zagadnienia świadczy fakt, że nabycie praw autorskich do dzieła literackiego bynajmniej nie upoważnia do wprowadzania w nim jakichkolwiek zmian. Natomiast producenci oprogramowania, nie mówiąc już o nabywcach, uważają że modyfikowanie zakupionego produktu przez klienta jest rzeczą normalną, która ponadto umacnia wzajemne więzi i zaufanie. Nieco mniej problemów prawnych nastręcza oprogramowanie sprzedawane w kostkach ROM.

Podobnie jak w krajach EWG, można zaobserwować dużą aktywność w tej dziedzinie w USA i Japonii. Oprócz modyfikacji prawa autorskiego i uzgodnień w skali międzynarodowej, przygotowuje się akty prawne dotyczące regulacji całokształtu zagadnień związanych z komputerami. Na przykład, projekt ustawy o przestępstwach komputerowych (Computer Crime Act), która ma być rozpatrywana przez Kongres USA w br., zawiera definicje wielu nowych rodzajów przestępstw. Po raz pierwszy usiłuje się sprecyzować pojęcia takie, jak: przestępcze uszkodzenie lub załamanie systemu komputerowego, włamanie do systemu komputerowego, fałszerstwo i oszustwo komputerowe, kradzież komputera albo usług, przestępcze nabycie albo zbycie praw dostępu do urządzeń i systemów itd.

Warto też wspomnieć o propozycjach legislacyjnych specyficznych dla niektórych tylko krajów. Np. w Holandii rozważa się utworzenie centralnego rejestru zakupów oprogramowania i odpowiedniego biura o funkcjach nadzorczo-śledczych. Początkowym działaniem jest tworzenie centralnego rejestru wszelkich dostępnych na rynku produktów programowych. Japończycy zamierzają odróżnić prawo używania zakupionego oprogramowania od prawa jego wypożyczania (dzierżawienia). W USA grupa producentów doprowadziła do sformułowania projektu ustawy kongresowej, zmierzającej do całkowitego zakazu wydzierżawiania oprogramowania, gdyż taka formę udostępniania uważa się za furtkę, przez którą przestępcy mają łatwy dostęp do źródeł oprogramowania.

Z kolei w Wielkiej Brytanii powstała specjalna organizacja FAST (Federation Against Software Theft) mająca na celu zwalczanie kradzieży oprogramowania. Organizacja ta finansuje kosztowne precedensowe procesy przeciwko złodziejom programów oraz prace legislacyjne z tej dziedziny. FAST ma duże poparcie producentów, którzy dotują tę organizację proporcjonalnie do osiągniętych obrotów oraz wiążą z nią duże nadzieje. Faktem jest, zapadły już pierwsze korzystne dla nich wyroki sądowe, a parlament brytyjski ma wkrótce rozważać propozycję modyfikacji prawa autorskiego, zmierzającą do wykluczenia nielegalnego kopiowania oprogramowania. FAST idzie już przetartą ścieżką — tą samą, co utworzona kilkanaście miesięcy wcześniej federacja do zwalczania kradzieży praw autorskich FACT (Federation Against Copyright Theft). Organizacja ta na początku 1983 r. wyznaczyła sobie zadanie rozprawienia się z „piratami" kopiującymi na skalę przemysłową w Wielkiej Brytanii kasety video. Sytuacja na tym rynku była wówczas bardzo podobna do tej jaka obecnie panuje na rynku oprogramowania mikrokomputerów. Według przybliżonych szacunków ok. 30% wpływów przejmowały wtedy nielegalne firmy kopiujące i sprzedające kasety bez licencji oraz nie płacące należności za korzystanie z praw autorskich. Szacuje się, że straty te nie przekraczają obecnie kilku procent, chociaż ogólne obroty w tej branży znacznie wzrosły. Te sukcesy FACT sprawiają, że producenci i dystrybutorzy oprogramowania, zwłaszcza dla komputerów osobistych i domowych, wierzą w szybkie efekty działania FAST.

W związku z nieustaloną jeszcze sytuacją prawną oraz zaostrzającą się konkurencją, poszukuje się również rozwiązań pozaprawnych. Najprostsze, ale dość skuteczne podejście, to tworzenie przez producentów specjalnych służb nadzoru nad dystrybucją oprogramowania.

Ostatnio np. Narendra Karmarkar, matematyk z BELL LABORATORIES, wymyślił całkowicie nową, lepszą metodę rozwiązywania układów równań z programowania liniowego, dziedziny ważnej szczególnie w automatyce. Ponieważ algorytmów nie można jak dotąd opatentować, firma zamierza sprzedawać je w formie programu wchodzącego w skład pakietu rozprowadzanego w sposób ściśle kontrolowany.

Opracowuje się również nowe konstrukcje sprzętowe, które mają na celu fizyczną ochronę oprogramowania przed niepowołanym użyciem. Np. M. Knight, z British Technology Groop, zaprojektował na zlecenie brytyjskiego Ministerstwa Obrony nowy, już opatentowany, rodzaj układu scalonego. Jest nim mikrokomputer jednoukładowy, którego zasadniczą cechą jest niedostępna dla użytkownika pamięć ROM albo EPROM o pojemności 4 K bajtów. Mikrokomputer może wykonywać zapisany tam program (na danych pochodzących z pamięci RAM), ale użytkownik nie może w żaden sposób odczytać zawartości chronionego obszaru pamięci. W przypadku mikrokomputera z wersją pamięci EPROM, użytkownik może jedynie zapisać w niej własną treść, poprzez służący do tego celu jednokierunkowy port wejściowy. Obszar chroniony nie jest dostępny z żadnego innego portu, a jego zawartość nie może, nawet przypadkowo, pojawić się na szynach dostępnych z zewnątrz (układ zawiera dodatkowe szyny wewnętrzne). Stan wszystkich portów jest stale kontrolowany i na szyny zewnętrzne przesyłane są tylko wybrane dane. Przewiduje się, że oprogramowanie sprzedawane obecnie na dyskach czy taśmach w przyszłości będzie oferowane w dwu częściach: procedury najważniejsze (lub decydujące o rozwiązaniu) dostępne byłyby tylko w postaci układu scalonego, natomiast pozostała część — w postaci konwencjonalnej. W związku z tym proponuje się stosowanie „standardowych" wstawek, umożliwiających łatwą wymianę „utajnionych elementów oprogramowania. Warto dodać, że układ zaprojektowano tak, że nie można go odczytać za pomocą żadnej ze znanych technik, jakimi dysponują producenci układów scalonych. Wewnątrz tego układu wbudowano bowiem specjalną osłonę chroniącą właściwą strukturę wielowarstwową, zawodzą więc np. elektroniczne techniki próbkowania warstw czy też odczytu napięcia kontrastowego.

Opracował RYSZARD K. KOTT
na podstawie profesjonalnej prasy brytyjskiej z IV kwartału 1984 r.

Jubileusz czasopisma

Odpowiednik branżowy i tematyczny INFORMATYKI, jakim jest wydawane w NRD czasopismo „rd- rechentechnik-datenverarbeitung" obchodziło we wrześniu ub.r. dwudziestolecie swego istnienia. Czasopismo to powstało w konsekwencji uchwalenia w dniu 3 lipca 1964 r. przez Prezydium Rady Ministrów NRD programu rozwoju środków i zastosowań elektronicznej techniki obliczeniowej. Czasopismo, powołane w ramach wydawnictwa „Die Wirtschaft" („Gospodarka"), nosiło początkowo tytuł „Rechentechnik-Rationalisierung — Datenverarbeitung" („Technika obliczeniowa-racjonalizacja--przetwarzanie danych"). Było ono adresowane w pierwszym rzędzie do kręgu osób zatrudnionych w organach i placówkach służb finansowych, gdzie stosowano już szeroko mechanizację prac obrachunkowych (automaty księgujące i fakturujące, maszyny licząco--analityczne).

W miarę szybkiego rozwoju w NRD produkcji sprzętu komputerowego oraz jego zastosowań, czasopismo przekształciło się w publikację o charakterze wyłącznie informatycznym. Reprezentuje wysoki poziom fachowy, stanowiący odbicie bardzo szerokiego frontu zastosowań informatyki w NRD. Jest niezwykle cennym źródłem materiałów dotyczących praktyki eksploatacji sprzętu JS EMC, a ostatnio również sprzętu mini- i mikrokomputerowego produkcji NRD. Znakomicie redagowane i wydawane w bardzo starannej szacie graficznej może być niewątpliwie wzorem dla czasopism informatycznych wydawanych w krajach RWPG (WK).

Nowości w brytyjskiej mikroinformatyce

W ciągu ostatnich 2—3 lat można stwierdzić znaczne zbliżenie społeczeństwa brytyjskiego do komputerów. Ich użyteczność w biurach, fabrykach, szkołach i domach staje się coraz w większym stopniu widoczna. Ostatnie ankiety magazynu konsumentów „Which?" wykazały, że w W. Brytanii jest najwięcej komputerów domowych w przeliczeniu na jednego mieszkańca. Taki stopień nasycenia powstał m.in. dzięki wielu pismom poświęconym komputerom domowym, jakie są do nabycia w brytyjskich kioskach. Niektóre z nich to tygodniki.

Najbardziej znane z brytyjskich mikrokomputerów to ZX 81 i ZX SPECTRUM firmy SINCLAIR RESEARCH oraz BBC MICRO firmy ACRON COMPUTERS. Stwarzają one użytkownikom znaczne możliwości zastosowań, za stosunkowo niską cenę. Są one też powszechnie wprowadzane w szkołach. W kwietniu 1981 r. rząd brytyjski rozpoczął realizację programu „Mikrokomputery w szkolnictwie", który początkowo obejmował tylko szkoły średnie. Rząd pokrywał połowę kosztu sprzętu, który obejmował — oprócz wspomnianych mikrokomputerów BBC MICRO — również mikrokomputery RML380Z firmy RESEARCH MACHINES. Ocenia się, że w szkołach znajduje się już ok. 18 tys. mikrokomputerów, a znacznie większa ich liczba została zamówiona.

W lipcu 1982 r. program został rozszerzony na szkoły podstawowe, a wśród zalecanych komputerów znalazł się ZX SPECTRUM. Inny program rządowy nazwany „Mikroelektronika w programie nauczania" obejmował opracowywanie oprogramowania, kształcenie nauczycieli i nowe programy nauczania.

W przemyśle komputery były wprawdzie używane od wielu lat, lecz ze względu na ich koszt stosowano je tylko w większych przedsiębiorstwach. Wraz z postępem miniaturyzacji, a zwłaszcza pojawieniem się układów o bardzo dużym stopniu scalenia (VLSJ) komputery stały się nie tylko mniejsze, ale wzrosła ich moc obliczeniowa przy jednoczesnej obniżce ceny. Pozwoliło to wprowadzić komputery do większości przedsiębiorstw. Do niedawna komputery stosowano głównie do rozwiązywania problemów finansowych i naukowych. Obecnie szczególnie szybko rozwija się wspomaganie projektowania, wytwarzania i testowania, określane nazwą techniki wspomagania komputerowego (CAE — computer aided engineering). Tu również rząd brytyjski podjął wiele inicjatyw. Można wymienić dwa projekty: CADCAM — obejmujący wspomagane komputerowo projektowanie i wytwarzanie w przemyśle maszynowym i elektrotechnicznym, oraz CADMAT — dotyczący projektowania, wytwarzania i testowania w przemyśle elektronicznym. W ramach tych programów przedsiębiorstwa mogą występować o dotacje, sięgające jednej trzeciej wartości projektu, a także uzyskać konsultacje, szkolenie oraz możliwości zapoznania się z działaniem istniejących systemów CAE.

Do najbardziej popularnych i najszybszych systemów mikrokomputerowych, oferowanych obecnie na rynku brytyjskim, należy 16-bitowy ORION wytwarzany przez firmę FUTURE TECHNOLOGY SYSTEMS dla firmy OFFICE and ELECTRONIC MACHINES. Oparty jest on na mikroprocesorze INTEL 8086, i ma pamięć operacyjną o pojemności 128 K bajtów, która może być powiększona do 896 K bajtów. Istnieje możliwość przyłączenia dysków typu Winchester średnicy 133 mm i pojemności 6 lub 12 M bajtów, a także dysków elastycznych o pojemności 500 K lub 1 M bajtów i tej samej średnicy. Monitor o przekątnej ekranu 30 cm może wyświetlać teksty (zielonymi znakami na szarym tle) o pojemności 25 wierszy 40- lub 80-znakowych. Monitor ten jest wyposażony w pamięć wizyjną o pojemności 72 K bajtów, z czego 64 K stanowi pamięć układu graficznego, przy czym generowanie znaków oparte jest na matrycy 8×14, w ramach obszaru 10×16 elementów, co daje wysoką rozdzielczość. Systemy operacyjne EOS/C i EOS/M są rozszerzoną wersją CP/M-86 i MP/M-86. Kopiowanie zawartości dyskietki o pojemności 1 MB trwa poniżej 70 s. 8-znakowe hasło zapewnia skutecznie zabezpieczenie systemu przed niepowołanymi dostępem.

System może zaspokoić wszelkie potrzeby obliczeniowe przedsiębiorstwa i stwarza możliwość użycia podstawowych języków programowania, takich jak M BASIC, C BASIC CIS COBOL, RM COBOL, MS PASCAL i MS FORTRAN. Koszt konfiguracji podstawowej, obejmującej monitor, klawiaturę, pojedynczą jednostkę dyskową, oprogramowanie oraz porty typu RS232C, wynosi 2350 funtów.

Seria mikrokomputerów SPIRIT firmy ALMERC DATA SYSTEMS może być wykorzystana do działań na słowach 8- lub 16-bitowych i określana jest jako niezależna od procesora. Może ona wykorzystywać mikroprocesory Z80A lub 8086 oraz dodatkowo 8087. Mikrokomputer SPIRIT 1 wykorzystuje układ Z80A. Konstrukcja pięciopakietowa typu S-100 obejmuje dwa porty szeregowe RS232, dwa 8-bitowe porty równoległe i pamięć operacyjną o pojemności 256 K bajtów, którą można dwukrotnie powiększyć. Dwie jednostki dyskowe (dla dysków o średnicy 133 mm) mogą tworzyć albo konfigurację złożoną z dwóch dysków elastycznych o pojemności po 800 K bajtów, bądź jedną dla dyskietek, a drugą dla dysków typu Winchester o pojemności 5—40 M bajtów. Systemem operacyjnym jest w tym przypadku CP/M80, natomiast w maszynach opartych na procesorze 8086 (jak SPIRIT 1—16) system CP/M86.

Oba mikrokomputery wykorzystują ekran o przekątnej 30 cm, z zielonymi znakami na bursztynowym tle i pojemnością 25 wierszy po 80 znaków (640×300 punktów). Monitor ma pamięć wizyjną o pojemności 64 K bajtów dostępną przez port szeregowy.

Inne modele tej serii to SPIRIT 2 oraz SPIRIT 2—16, które mają 15-pakietową konstrukcję umożliwiającą dodatkowe opcje. Natomiast SPIRIT 3 obsługuje czterech użytkowników, którzy mogą podzielić między sobą zasoby procesora centralnego. Wreszcie SPIRIT 4 pozwala na przetwarzanie rozłożone przy wykorzystywaniu do 10 terminali.

SPIRIT 1 z monitorem, klawiaturą i podwójną stacją dysków elastycznych (1,6 MB) kosztuje 2695 funtów. Podstawowa konfiguracja SPIRIT 1—16 kosztuje 2995 funtów, a SPIRIT 4 z pięcioma terminalami i jednostką dyskową typu Winchester o pojemności 20 M bajtów kosztuje 12015 funtów.

Brytyjska filia firmy BURROUGHS zaprojektowała i opracowała komputer B95 stanowiący ogniwo pośrednie między typowymi mikrokomputerami biurowymi a minikomputerami. Jest on wyposażony w pamięć operacyjną o pojemności 64 K bajty, które może rozszerzyć do 512 K bajtów.

Jako pamięć masowa służą dwie jednostki dyskowe, przy czym dyski elastyczne mają pojemność 700 K bajtów, a dyski typu Winchester 96 lub 14,4 M bajta. Łączna pojemność pamieci dyskowej może wynosić 30,2 M bajta. B95 można stosować jako mały system biurowy lub tani procesor komunikacyjny (do 5 linii). W tym ostatnim przypadku może on być stosowany jako sterownik terminali inteligentnych lub nieinteligentnych, a także jako sterownik w systemach zbierania danych przemysłowych lub mały węzeł sieci przetwarzania rozproszonego.

B95 jest składnikiem systemu CMS firmy BURROUGHS na poziomie wprowadzania danych. Kosztuje poniżej 10 tys. funtów. Bez zmian oprogramowania można rozszerzyć go do poziomu największych konfiguracji systemu B1900, który kosztuje ponad 250 tys. funtów.

Opracował JAN RYŻKO

na podstawie „London Press Service"

TG 0725/3

Komputer roku 2000

„Do roku 2000 liczba elementów pojedynczego układu scalonego będzie większa niż liczba komórek ludzkiego mózgu. Producenci układów scalonych odwrócą się od krzemu i innych konwencjonalnych materiałów i będą projektować generację komputerów biologicznych, zbudowanych częściowo lub całkowicie z cząstek złożonych białek, takich jak te, które znajdują się w żywych komórkach". To niektóre z przewidywań Briana Oakley, jednego z luminarzy brytyjskiego świata informatyki.

Brian Oakley jest dyrektorem programu ALVEY, założonego wspólnym wysiłkiem rządu Wielkiej Brytanii, świata nauki i prywatnego przemysłu w celu podtrzymania obecności Wielkiej Brytanii w czołówce coraz szybciej rozwijającej się dziedziny układów scalonych i technologii komputerowej.

Brian Oakley przewiduje, że w ciągu najbliższych 10 lat prędkość komputerów wzrośnie dziesięciokrotnie przy dalszym utrzymywaniu się tego trendu. Każdego roku gęstość układów krzemowych będzie podwajać się. Ponieważ będą one jednak mniejsze, mniej energochłonne, szybsze i tańsze w produkcji, koszt ich będzie malał o połowę co sześć lat. Oznacza to, że komputery będą coraz mniejsze a równocześnie coraz szybsze, pojemniejsze i inteligentniejsze, drożejąc nieznacznie lub wcale.

Trend ten jak twierdzi Oakley ma jednak swoją granicę. Projektowane obecnie w laboratoriach układy wymagać będą tak gęstego upakowania elementów, że odległość pomiędzy sąsiadującymi elementami zmniejszy się do jednej czwartej długości fali światła. Wprawdzie światło już teraz zaczyna zastępować sygnały elektryczne w komputerowych środkach manipulacji, ale tak małe układy scalone nie będą mogły używać światła, lecz promieni X, które mają o wiele mniejszą długość fali. Technologia budowy takich komputerów staje się — zdaniem Oakleya — już całkowicie realna. Planuje się zasilanie ich przez lasery promieni X. Będą one jednak już bliskie granicy redukcji rozmiarów, a w konsekwencji — zwiększania prędkości.

Do roku 1995, utrzymuje Oakley, maszynistki i operatorzy urządzeń wprowadzania danych muszą zatroszczyć się o swoją przyszłość, ponieważ ich zawód będzie zanikał dzięki rozwojowi urządzeń rozpoznających mowę. Do roku 2000 komputery będą w stanie wnioskować na podstawie niewystarczających danych, nadawać sens („rozumieć") nawet paplaninę oraz rozpoznawać zamazane lub uszkodzone obrazy, tak jak czyni to człowiek. Do tego potrzebna jednak będzie moc obliczeniowa 1000 razy większa od osiąganej dziś. Ale dzięki technice przetwarzania równoległego komputery roku 2000 będą miały moc obliczeniową milion razy większą niż dzisiejsze. Umożliwi to im uczenie się na podstawie własnego doświadczenia, tworzenie własnych zbiorów zadań i reguł oraz — z upływem czasu — ich modyfikację, a więc działanie identyczne jak u ludzi, lecz wielokrotnie szybsze.

Już przed rokiem 2000 komputery powinny osiągnąć poziom rozwoju zbliżony do ludzkiej inteligencji, a nawet pewnej osobowości opartej na własnym „doświadczeniu życiowym". Brian Oakley sądzi jednak, że jeszcze wiele czasu upłynie, zanim komputery staną się „nadludźmi". Ale oczywiście niektórzy eksperci mają inne zdanie.

Tymczasem już w tym roku komputer wyzwie na pojedynek arcymistrza szachowego i, zdaniem jego konstruktorów, pokona go.

Opracował RYSZARD KURZYJAMSKI
na podstawie „Computers in the year 2000", London Press Service

Czechosłowacki program elektronizacji

Podobnie jak u nas w 1983 (uchwała nr 77), również rząd czechosłowacki podjął w kwietniu ub.r. uchwałę nr 253 w sprawie kompleksowego programu elektronizacji gospodarki narodowej do 1995 r. Program ten obejmuje wszystkie gałęzie gospodarki i zakłada, że jego realizacja przyczyni się do wzrostu efektywności gospodarowania oraz wydajności pracy. W wystąpieniu w dniu 20 listopada ub.r. premier Strougal komentując uchwałę stwierdził, że wprawdzie przemysł czechosłowacki utrzymuje nadal tempo rozwoju równe przeciętnej światowej, lecz w zakresie nasycenia gospodarki elektroniką kraj zajmuje odległe miejsce wśród państw uprzemysłowionych.

W ramach wspomnianego programu znaczącą pozycję zajmują środki automatyzacji zarządzania. Zakłada się, że udział środków techniki obliczeniowej i automatyzacji w całości produkcji przemysłu elektrotechnicznego wzrośnie w 1990 r. do 14,5%, przy czym w okresie 1985—1990 nastąpi ponad 70% wzrost produkcji tej grupy wyrobów.

Program elektronizacji zakłada istotną modernizację technologii przetwarzania danych, zgodnie z aktualnymi trendami światowymi, oraz jej powiązanie z kierunkami doskonalenia metod zarządzania gospodarką narodową, określonymi we wcześniejszej uchwale (nr 243 z 1984 r.).

W programie określono również ilościowy rozwój zautomatyzowanych systemów zarządzania (ASŘ). Do 1990 r. ma ich powstać 70 na szczeblu centralnym, 253 — na szczeblu średnim oraz 1710 na szczeblu przedsiębiorstw. Do 1995 r. liczby te mają odpowiednio wzrosnąć do 75, 368 i 2490 systemów zarządzenia.

W uchwale określono również zadania w zakresie wzrostu ilościowego i jakościowego usług telekomunikacyjnych, a zwłaszcza wprowadzenia transmisji cyfrowej, która warunkuje rozwój nowoczesnych systemów informatycznych. Zakłada się wprowadzenie nowych typów abonenckich urządzeń końcowych, umożliwiających modernizację sieci teleksowej oraz wprowadzenie usług typu VIDEOTEXT, a także ogólnodostępnej sieci transmisji danych, z której w 1995 r. powinno korzystać ok. 10 tys. abonentów.

Zakłada się preferowanie takich kierunków rozwoju zastosowań, które prowadzą do wzrostu wydajności pracy oraz oszczędności materiałów, energii i siły roboczej, a także wpływają na wzrost efektywności eksportu, zwłaszcza wyrobów przemysłu maszynowego. Dlatego informatyzacja dotyczyć będzie w pierwszym rzędzie automatyzacji zarządzania procesami produkcyjnymi i technologicznymi. Wśród innych preferowanych dziedzin komentarz premiera Strougala wymienia transport, handel i finanse, a także służbę zdrowia i automatyzację prac projektowych i konstrukcyjnych. (WK)

Mikrokomputery bez cła

Czytelników INFORMATYKI zapewne zainteresuje wiadomość, że od dnia 14 sierpnia br. nie pobiera się cła za przywiezione do kraju:

- mikrokomputery, ich moduły oraz części zamienne
- urządzenia zewnętrzne — monitory ekranowe, drukarki itp.
- oprogramowanie dla systemów komputerowych
- nośniki programów.

Zniesiono również obowiązek uzyskania pozwolenia na ich przywóz.

Można również bez cła przywieźć do 10 sztuk taśm magnetofonowych (bez zapisu i z zapisem) w postaci stosowanej w sprzęcie elektronicznym powszechnego użytku.

Podstawa prawna: rozporządzenie ministra handlu zagranicznego z 12 lipca br. zmieniające rozporządzenie w sprawie ceł i pozwoleń na towary przywożone i na towary wywożone w niehandlowym obrocie towarowym z zagranicą (Dz. Ustaw nr 33, poz. 150).

Japońska elastyczność

Nie znane jeszcze nikomu dziesięć lat temu (jako producenci komputerów), takie firmy japońskie jak HITACHI, FUJITSU, NEC, TOSHIBA i MITSUBISHI są dziś prawdziwą potęgą. Japończycy sprzedali w roku 1983 komputery o wartości 2,7 mld dolarów, ustępując jedynie USA. Przez wiele lat koncentrowali się na produkowaniu coraz szybszego i tańszego sprzętu, nie poświęcając zbyt wiele uwagi zagadnieniom oprogramowania. Sytuacja ta uległa niedawno radykalnej zmianie. Japończycy musieli zareagować na gwałtowny wzrost zapotrzebowania na oprogramowanie. Trzy lata temu HITACHI przeznaczyła na prace związane z oprogramowaniem 10% swego funduszu rozwoju. W ubiegłym roku firma postanowiła wykorzystać w tym celu już 30% środków przeznaczonych na badania i rozwój przedsiębiorstwa, chociaż oprogramowanie przyniesie tylko 10% ogólnego zysku ze sprzedaży.

Oprogramowanie stanowiło dotąd największą przeszkodę w eksporcie japońskich produktów, szczególnie komputerów biurowych i osobistych. W 1982 roku firma IBM oskarżyła HITACHI, jednego z największych japońskich eksporterów, o kopiowanie oprogramowania. Na podstawie zawartego porozumienia HITACHI płaci obecnie IBM od dwóch do czterech mln dol. miesięcznie, w charakterze opłat licencyjnych, oraz przedstawia amerykańskim ekspertom każdy nowy produkt programowy przed jego wprowadzeniem na rynek celem zaopiniowania pod kątem naruszenia praw IBM. Również firma FUJITSU musiała zapłacić IBM wiele milionów dolarów i zobowiązać się do niekopiowania jej produktów. Nie ujawniono jednakże szczegółów zawartego porozumienia.

Różnice języków i kultur powodują, że oprogramowanie pisane przez Japończyków nie zawsze może być przedmiotem eksportu. Trzeba nie tylko tłumaczyć instrukcje obsługi, ale często pisać od nowa całe programy. Na przykład, japoński system rachunkowości jest na tyle różny od amerykańskiego, że wiele programów jest wręcz bezużytecznych w USA, co np. poważnie ogranicza możliwość eksportu komputerów osobistych. Japońscy nabywcy niechętnie odnoszą się do programów standardowych, tak chętnie kupowanych przez Amerykanów. Powoduje to, że firmy japońskie muszą przeznaczać ogromne kwoty na ulepszenie metod wprowadzania i wyprowadzania informacji przy użyciu znaków pisma japońskiego.

Byłoby jednak błędem niedocenianie możliwości Japończyków jako potencjalnego konkurenta na rynku oprogramowania. Już teraz odnoszą oni niemałe sukcesy eksportując proste oprogramowanie przeznaczone dla gier video. Podobnie dzieje się z niektórymi programami specjalistycznymi, np. do obliczeń inżynierskich. Kultura i język nie mają bowiem znaczenia, gdy manipuluje się nie tekstami, lecz rysunkami i cyframi. Takie rodzaje oprogramowania, po niewielkich modyfikacjach, mogą być z powodzeniem sprzedawane w każdym kraju. Japończycy zaprezentowali również swoje wielkie możliwości w dziedzinie oprogramowania systemów rezerwacji i sprzedaży biletów lotniczych na superkomputerach.

Potencjalna przewaga firm japońskich nad amerykańskimi wynika nie tylko z ich sumienności, dzięki której — ich produkty są lepsze, bardziej niezawodne i prostsze w użytkowaniu. Również koszty produkcji japońskiego oprogramowania są niższe niż w USA. Dobrze wyszkolony, samodzielny i bardzo wydajny programista japoński zarabia średnio 10 tys. dolarów rocznie, natomiast jego amerykański odpowiednik otrzymuje jako płacę początkową dwukrotnie więcej, będąc 10—15% mniej wydajny od japońskiego kolegi. Firma TOSHIBA zorganizowała niedawno „wytwórnię oprogramowania", i zamierza otworzyć następną, w której będzie pracować o dwa tysiące programistów więcej.

Firma NEC postanowiła podbić rynek amerykański, przeznaczając na ten cel znaczną część 400 mln dol. wykorzystywanych co roku na prace nad rozwojem oprogramowania. Zaangażowano w tym celu amerykańskich specjalistów, by przeanalizowali potrzeby amerykańskich użytkowników komputerów i przekazali „wytwórni oprogramowania" szczegółowe wymagania tych użytkowników.

O znaczeniu, jakie Japończycy przywiązują do rozwoju oprogramowania najlepiej świadczą posunięcia ich Ministerstwa Handlu i Przemysłu. Ministerstwo to utworzyło sieć laboratoriów badawczych do prac nad oprogramowaniem. Laboratoria te otrzymały na rok 1984 dotację w wysokości 23 mln dol., a producenci pracujący nad ulepszaniem oprogramowania mogą korzystać z nisko oprocentowanych pożyczek i ulg podatkowych.

Amerykanie twierdzą jednak, że nawet wspomniana pomoc rządowa nie zapewni Japończykom możliwości opanowania w najbliższym czasie rynku oprogramowania. W roku 1982 (brak nowszych danych) Japończycy uzyskali 1,4 mld dol. ze sprzedaży oprogramowania — cztery razy mniej niż Amerykanie. Jednakże firmy japońskie są przekonane, że przyszłość należy do nich. „Nie mamy zamiaru kopiować oprogramowania IBM — powiedział przedstawiciel SOFTWARE INDUSTRY ASSOCIATION — chcemy pokonać IBM".

Oprac. K. I. na podstawie
BUSINESS WEEK z 27 lutego 1984

■ Kompania produkcyjna LUCAS FILM, znana ze specjalnych efektów animacyjnych w filmie „The Last Starfighter", zamierza wejść na rynek grafiki komputerowej. Proponowane oprogramowanie graficzne ma być bardzo oryginalne. Firma przygotowuje bowiem procesor graficzny przystosowany do generowania różnorodnych efektów wizualnych. Produkt ten, nazwany PIXAR, opiera się na grafice wysokiej rozdzielczości, o szybkości przetwarzania obrazu rzędu 40 mln pixeli/s (pixel — jednostka rozdzielczości obrazu). System efektów graficznych SUN MICROSYSTEM ma znaleźć zastosowanie nie tylko w tzw. przemyśle animacyjnym, lecz również ma być pomocny w badaniach medycznych oraz sejsmologii. (TW)

*

■ Coraz większą popularnością cieszą się systemy oprogramowania wspomagające projektowanie układów scalonych. Ostatnią, rewelacją w tej dziedzinie jest system CHIPMASTER firmy DAISY. Jest on reklamowany jako najefektywniejszy specjalizowany system graficzny wysokiej rozdzielczości. Procesorem centralnym systemu jest INTEL 80286, zapewniający sprawne projektowanie układów VLSI. Projektant dysponuje zbiorem „okienek" wyświetlanych na ekranie, używając odpowiednich rozkazów — w trybie niemalże konwersacyjnym — uszczegóławia on budowę projektowanego układu. Fragmenty projektu ukazywane są w różnych skalach wielkości, w odpowiednich okienkach. Wszelkie modyfikacje są od razu uwidoczniane. CHIPMASTER realizuje cały cykl projektowy — od schematu płytek i połączeń, przez optymalizację, do wygenerowania danych wyjściowych zgodnych ze standardem przemysłowym układów scalonych. Użytkownicy starszych systemów tego typu, jak LOGICAN i MEGALOGICAN (1981 r.), mogą pracować w systemie CHIPMASTER dołączając sprzęt graficzny i odpowiednie oprogramowanie. (TW)

*

■ Jedna z ostatnich międzynarodowych konferencji dotyczących urządzeń półprzewodnikowych przyniosła nadzieje na trójwymiarowe, a więc niezwykle upakowane struktury układów scalonych. W Japonii, kraju jeszcze niedawno nie liczącym się w sprawach technologii, opracowano nową metodę miniaturyzacji kryształów w procesie produkcji układów scalonych metodą krzem—izolator (ang. silicon—on—isolator — SOI). Dodatek fosforu w trakcie domieszkowania zmniejsza maksymalne rozmiary powstających kryształów jednorodnych z 24 do 6 μm. Opracowany w tokijskim Instytucie Technologii proces produkcji półprzewodników jest ponadto tani i nie wymaga wysokich temperatur — jak było to dotychczas w technologii SOI. Jeżeli tylko zostanie rozwiązany problem realizacji połączeń pomiędzy tak wytwarzanymi płytkami, to struktury trójwymiarowe staną się rzeczywistością. (TW)

Rozwój terminologii języka ADA

Od czasu opublikowania propozycji polskiej terminologii języka ADA (Biuletyn PTI, nr 9—10, 1983) upłynęły już dwa lata. Jest to okres wystarczający do tego, aby poddać ją częściowej choćby weryfikacji. Duża grupa terminów okazała się dobrana trafnie i utrwaliła się, część terminów nie została przyjęta, a inne — wymagają korekty. Warto więc na powrót zastanowić się, w jakim stopniu proponowany zestaw terminów odpowiada używanym.

Proces weryfikacji terminologii rozpoczął się bezpośrednio po jej opublikowaniu. Na moje ręce wpłynęło szereg uwag dotyczących zakresu słowniczka i trafności proponowanych terminów. Wraz z opracowywaniem kompilatorów ADY i stopniowym wprowadzaniem nauczania tego języka na uczelniach, terminologia również ulegała pewnym modyfikacjom. Powolna, choć systematyczna działalność publikacyjna, w której poza artykułami i skryptami warto zwrócić uwagę an przetłumaczenie i bliskie wydanie pierwszej książki o ADZIE (I.C. Pyle, The Ada Programming Language, Prentice-Hall, 1980, 1985), jest także nieodzownym warunkiem udoskonalania terminologii.

Ponieważ złożenie do druku książki I.C. Pyle'a zamknęło pewien etap rozwoju terminologii ADY, postanowiliśmy odświeżyć nieco tę tematykę, poruszając ją z różnych punktów widzenia (por. artykuły J. Bieleckiego i W. Klepacza w rubryce terminologicznej dwóch poprzednich numerów INFORMATYKI). Moje doświadczenia są oparte głównie na uwagach dotyczących słowniczka, działalności publikacyjnej i pracy nad tłumaczeniem książki I.C. Pyle'a.

Nazewnictwo typów stosowanych w ADZIE wydaje się być dobrze ustalone. Jedna z podstawowych zasad terminologicznych, stanowiąca, aby w terminach złożonych unikać używania imiesłowów (szczególnie czynnych), spowodowała wyparcie nazwy typ wskazujący przez nazwę typ wskaźnikowy (ang. access type). Nawiasem mówiąc, przy ustalaniu tej nazwy skorzystano z obecności odpowiedniego pojęcia w polskim języku informatycznym (odpowiednik typów: pointer, reference) i, aby nie wprowadzać synonimu, przyjęto uprzednio proponowaną nazwę wywodzącą się z PASCALA, choć źródłosłów angielski w ADZIE jest zupełnie inny (access znaczy dostęp).

Typ charakteryzuje zbiór wartości danych i zbiór operacji na tych danych, tak więc polska nazwa innego typu — typ całkowity (ang. integer type) — jest utworzona nielogicznie, gdyż przymiotnik całkowity może odnosić się do liczb a nie do operacji na tych liczbach. Dlatego uważam, że lepsze jest określenie typ całkowitoliczbowy.

Mniej istotne uwagi dotyczą częstszego stosowania i preferowania terminu typ logiczny (ang. boolean type) zamiast typ boolowski, co ma uzasadnienie w tradycji języków programowania, i — wprowadzenia terminu rekord wariantowy zamiast rekord z wariantami, co nadaje nazwie złożonej większą zwartość.

Nazwy operacji i operatorów w ADZIE są uświęcone wieloletnią tradycją matematyczną, jako że w większości są to tradycyjne operacje matematyczne. Jednakże, warto zwrócić uwagę na polskie nazwy operatorów **and then** i **or else**. W przeciwieństwie do tego, co proponowano w „Słowniczku ADY" (forma sterująca — control form, short-circuit form), należy je nazwać warunkowymi operatorami logicznymi (przez analogię do instrukcji warunkowych), ponieważ są operatorami logicznymi a przebieg wykonywania operacji zależy od spełnienia pewnego warunku. Tak więc, operacja and then jest koniunkcją warunkową, a operacja or else — alternatywą warunkową.

Operatorem rzadko spotykanym w innych językach programowania jest tzw. alokator (ang. allocator), którego działanie polega na utworzeniu obiektu i udostępnieniu wartości typu wskaźnikowego, która oznacza ten obiekt. Wykonanie odpowiedniej operacji nazywa się po angielsku **execution of the allocator**, co po polsku powinno się nazywać u ż y c i e m a l o k a t o r a, jako że nie można powiedzieć wykonanie alokatora (niezgodność semantyczna), ani wykonanie alokacji (w informatyce znaczy co innego). Mówi się natomiast — użycie operatora dodawania, mnożenia itp.

Przy omawianiu nazw operatorów warto też zwrócić uwagę na dość marginalną, ale nie wyjaśnioną sprawę polskiej nazwy operacji **catenation**. Odrzucając nazwę katenacja i tradycyjny termin matematyczny — konkatenacja, a także — złożenie (gdyż w matematyce oznacza zupełnie co innego) i łączenie (ma inne znaczenie informatyczne), proponuję używanie nazwy s k l e j a n i e, choć padały także inne propozycje, jak np. spinanie.

Kilka poważniejszych uwag terminologicznych odnosi się do struktur sterowania w ADZIE. Główna zmiana w stosunku do terminologii „Słowniczka ADY" polega na upowszechnieniu się przymiotnika r o d z a j o w y na oznaczenie angielskiego **generic**. Tak więc, mówimy: j e d n o s t k i r o d z a j o w e, p o d p r o g r a m y r o d z a j o w e, p a r a m e t r y r o d z a j o w e itp., zamiast jak poprzednio — generyczne. Z użyciem jednostek rodzajowych wiąże się tworzenie ich konkretnych realizacji w procesie kompilacji. Ze względu na nieadekwatność określeń użytych w „Słowniczku ADY" i ich powszechną krytykę, na ogół używa się terminów zapożyczonych z rosyjskiego wydania książki P. Wegnera „Programmirowanije na jazykie ADA". Określony egzemplarz jednostki rodzajowej, powstały wskutek kompilacji, nazywa się k o n k r e t e m (ang. instance), a sam proces jej tworzenia na podstawie wartości parametrów rodzajowych — k o n k r e t y z a c j ą (ang. instantiation). Inne odpowiedniki polskie słów angielskich **instance** i **instantiation**, odpowiednio — w c i e l e n i e i p o w o ł a n i e, są dość trafne, jeżeli rozpatrujemy je oddzielnie, ale po pierwsze — nie mają wspólnego rdzenia, co tu jest niepożądane, a po drugie — obydwa są rzeczownikami odsłownymi, co również jest niepożądane w przypadku, gdy jedno pojęcie i słowo angielskie (instantiation) wywodzi się z drugiego (instance).

Dość kontrowersyjne jest powszechne używanie w środowisku zajmującym się ADĄ, nazwy c i a ł o na oznaczenie pojęcia nazwanego po angielsku **body**. Mówi się więc: c i a ł o p r o g r a m u, c i a ł o p r o c e d u r y, c i a ł o f u n k c j i, a nawet c i a ł o i n s t r u k c j i (np. instrukcji accept). Co bardziej ortodoksyjni zwolennicy czystości języka używają słowa t r e ś ć. Osobiście preferuję ten pierwszy termin, gdyż jak nietrudno dostrzec jest to najnaturalniejszy odpowiednik polski słowa body. Słowo treść jest obciążone tyloma innymi znaczeniami, że próba nadawania mu precyzyjnego znaczenia informatycznego jest nad wyraz sztuczna. Nawet jeśli słowo ciało wywołuje zbyt dosłowne skojarzenia (coś podobnego można zarzucić także „treści"), to nie mam wątpliwości, że wkrótce upowszechni się w nowym znaczeniu, a w każdym razie już teraz jego użycie nie jest błędem.

O ile informacje o działaniu jednostki programowej są zawarte w jej ciele, to specyfikacja zawiera opis jej współdziałania z innymi jednostkami, a więc stanowi tzw. sprzężenie (ang. interface). Właśnie to słowo wydaje mi się najodpowiedniejsze na oznaczenie roli odgrywanej przez specyfikację jednostki programowej w ADZIE.

Najbardziej chyba kontrowersyjną propozycją terminologiczną, której z nikim — poza „Słownikiem poprawnej polszczyzny" pod red. prof. Doroszewskiego — nie konsultowałem, jest użycie słowa e w e n t u a l n o ś ć (zamiast wybór) na oznaczenie angielskiego **choice**, oznaczającego m.in. element konstrukcji case. W ADZIE, wartość choice wyznacza alternatywę instrukcji case, a jednym z potocznych znaczeń słowa choice jest także — alternatywa. Jednak z oczywistych względów, słowa alternatywa nie można tu użyć jako odpowiednika polskiego (ma już w tym języku inne, precyzyjne znaczenie). Proponowany w „Słowniczku ADY" wyraz wybór wydał mi się tu wyjątkowo nie na miejscu, gdyż może on oznaczać:

1) wybranie czegoś, a więc dokonanie wyboru, co jest niezgodne ze znaczeniem słowa choice w ADZIE, gdzie oznacza ono jedną z możliwości, ale nie sam wybór tej możliwości,

2) zestaw wybrany pod pewnym kątem, co także nie odpowiada znaczeniu słowa choice w ADZIE, gdzie oznacza ono pojedynczą możliwość.

Skoro więc wyraz wybór jest nieodpowiedni, a nie można użyć wyrazu alternatywa, to nie pozostaje nic innego jak znaleźć wyraz inny, mający potoczne znaczenie alternatywy. Takim wyrazem jest e w e n t u a l n o ś ć.

Wiele uwag wywołało użycie polskiego słowa w e j ś c i e na oznaczenie angielskiego **entry**. Podstawowym argumentem przeciwko „wejściu" jest możliwość pomyłek i nieporozumień wynikających z używania wyrazu wejście na oznaczenie angielskiego input. Jest tak niewątpliwie, **input** znaczy po polsku w e j ś c i e lecz tylko w języku potocznym. W informatyce wyraz **input**, jeżeli występuje sam jako termin, oznacza w p r o w a d z a n i e d a n y c h, natomiast w połączeniach wyrazowych ma znaczenie przymiotnika wejściowy, np. **input device** — u r z ą d z e n i e w e j ś c i o w e. Jedynym użyciem wyrazu **input**, w którym znaczy on w e j ś c i e, jest połączenie wyrazowe **input-output**, po polsku — w e j ś c i e-w y j ś c i e. W tym przypadku są to jednak dwa słowa występujące łącznie tak, że nie powinno być nieporozumień co do znaczenia słowa w e j ś c i e jako **entry** w ADZIE. Można jeszcze dodać, że nawet w elektronice słowo **input** występujące jako samodzielny człon zdania oznacza s y g n a ł w e j ś c i o w y. Jeżeli zaś oznacza wejście, to na ogół nie jako termin lecz jako wyraz potoczny. Uważam więc, że obawy co do nieporozumień mogących wynikać z przyjęcia polskiego odpowiednika słowa **entry** w ADZIE są mocno przesadzone, gdyż w zasadzie nie dochodzi tu do żadnych kolizji.

Jeżeli powyższe wyjaśnienia nie są przekonywujące, to ciekawym rozwiązaniem mogłoby być przyjęcie nazwy dojście na oznaczenie angielskiego entry. Wtedy zachowalibyśmy pokrewieństwo obu nazw, uzyskując jednocześnie ich odrębność i unikając w ten sposób używania nazwy wieloznacznej. Jest to jednak propozycja dość odważna, a wynikiem jej przyjęcia mogłoby być, na przykład, użycie nazwy odejście na oznaczenie angielskiego exit.

Mówiąc o nazewnictwie struktur sterujących w ADZIE warto dodać, że polski odpowiednik słowa exception utrwalił się jako wyjątek. Tak więc, nie ma obecnie sensu używanie w tym znaczeniu innych słów, jak: protest, wypadek itd.

Kilka innych terminów odnosi się zarówno do struktur danych jak i struktur sterujących. Pewną nieznaczną korektę proponuję wprowadzić do „Słowniczka ADY" w odniesieniu do wyrazu w i d z i a l n o ś ć. Sądzę, że bardziej właściwą nazwę odpowiadającą angielskiemu **visibility** jest — w i d o c z n o ś ć. Za używaniem wyrazu widoczność, a nie widzialność, przemawia jego związek z przesłanianiem (a nie — przysłanianiem) obiektów języka. Jeżeli obiekt jest przesłonięty, to może być widoczny z jednego miejsca programu a niewidoczny z innego, natomiast niewidzialność obiektów oznacza ich całkowitą niewidoczność.

Pewne wątpliwości budzi interpretacja terminów angielskich **range** i **scope** oznaczających odpowiednio:

range — ciągły zbiór wartości typu skalarnego,
scope — obszar tekstu programu, w którym skojarzenie identyfikatora z obiektem (deklaracja) wywiera skutek.

Choć, być może, będzie to niezgodne z pewnym zwyczajowym nazewnictwem informatycznym, termin **range** należy zastąpić słowem z a k r e s a termin **scope** — słowem z a s i ę g. Otóż, zakres zmiennej o znanej nazwie jest określony przez dolną i górną granicę wartości tej zmiennej, natomiast zasięg deklaracji tej zmiennej obejmuje pewien fragment programu. Tak więc, zakres zmiennej i zasięg jej deklaracji dotyczą dwóch różnych przestrzeni — przestrzeni wartości i przestrzeni tekstu programu, przy czym druga z tych przestrzeni, w odróżnieniu od pierwszej, obejmuje obiekty zewnętrzne wobec samej zmiennej. Wydaje mi się więc zupełnie naturalne, aby obszar, w którym występują obiekty mogące przybierać wartości określonej zmiennej, nazywać jej zakresem, a obszar, w którym jest ona rozpoznawalna przez nazwę — jej zasięgiem.

→

Waligórski S.: LOGO (I). Grafika żółwia

INFORMATYKA 1985, nr 7, s. 1

Pierwsza część charakterystyki języka programowania LOGO. Omówiono historię oraz podstawowe właściwości języka. Możliwości języka oraz jego elementy przedstawiono za pomocą tzw. grafiki żółwia, ilustrując prostymi przykładami i rysunkami.

Irlik J.: Kwalifikacja prawna programu komputerowego

INFORMATYKA 1985, nr 7, s. 5

Ocena sytuacji w zakresie ochrony praw autorskich twórców programów komputerowych w Polsce, w oparciu o istniejące ustawodawstwo.

Bielecki J.: FORTH — definiowanie kompilatorów i instrukcji strukturalnych

INFORMATYKA 1985, nr 7, s. 9

Przykłady definiowania złożonych kompilatorów oraz instrukcji strukturalnych języka FORTH w wersji fig-FORTH.

Suraj Z.: Problemy nauczania informatyki w szkołach średnich

INFORMATYKA 1985, nr 7, s. 11

Prezentacja wyników badań ankietowych na temat nauczania informatyki w szkołach średnich. Badania przeprowadzono wśród nauczycieli informatyki oraz kandydatów na studia wyższe. Podano propozycje zmierzające do doskonalenia procesu dydaktycznego.

Валигурски С.: LOGO (I). Графика черепахи

INFORMATYKA 1985, № 7, с. 1

Первая часть характеристики языка ЛОГО. Описание теории и основных свойств языка. Возможности языка и его элементы, представленные при помощи так наз. графики черепахи, проиллюстрированные простыми примерами и чертежами.

Ирлик Я.: Юридическая квалификация вычислительной программы

INFORMATYKA 1985, № 7, с. 5

Оценка положения в области охраны авторских прав разработчиков вычислительных программ в Польше на основании существующего законодательства.

Белецки Я.: FORTH — определение компиляторов и структурных инструкций

INFORMATYKA 1985, № 7, с. 9

Примеры определения сложных компиляторов и структурных инструкций языка ФОРТ в варианте фиг-ФОРТ.

Сурай З.: Проблемы обучения вычислительной технике в средних школах

INFORMATYKA 1985, № 7, с. 11

Представление результатов анкетных опросов на тему обучения вычислительной технике в средних школах. Опросы велись среди учителей — информатиков и кандидатов в вузы. Предложения относительно совершенствования процесса обучения.

Waligórski S.: LOGO (1). Schildkröte Graphik

INFORMATYKA 1985, Nr. 7, S. 1

Erster Teil einer Charakteristik von LOGO-Programmiersprache. Es wurden Geschichte und Grundeigenschaften der Sprache angegeben. Mit Verwendung von sog. Schildkröte-Graphik, illustriert mit einfachen Beispielen und Zeichnungen, wurden die Möglichkeiten der Sprache, sowie ihre Elemente, besprochen.

Irlik J.: Gesetzliche Qualifizierung eines Computerprogrammes

INFORMATYKA 1985, Nr. 7, S. 5

Eine Beurteilung der Lage im Beireich des Autorenrechteschutzes für Computerprogrammenurheber in Polen mit bestehender Gesetzgebung.

Bielecki J.: FORTH — Definierung von Kompilatoren und strukturellen Anweisungen

INFORMATYKA 1985, Nr. 7, S. 9

Beispiele der Definierung von zusammengesetzten Kompilatoren und strukturellen Anweisungen der FORTH-Sprache in fig-FORTH-Version.

Suraj Z.: Probleme der Informatikunterricht in Oberschulen

INFORMATYKA 1985, Nr. 7, S. 11

Eine Präsentation von Ergebnissen einer Umfrage über Informatikunterricht in Oberschulen. Die Umfrage wurde in Informatiklehrer- und Hochschulkandidatenkreisen durchgeführt. Es wurden Vorschläge zur Vervollkommnung des didaktischen Prozesses angegeben.

Waligórski S.: LOGO (1). Turtle graphics

INFORMATYKA 1985, No. 7, p. 1

First part of the characteristics of the LOGO programming language. Language's history and its main features are discussed. Using so called turtle graphics, illustrated with simple examples and drawings, possibilities of the language and its elements are presented.

Irlik J.: Legal qualification of computer program

INFORMATYKA 1985, No. 7, p. 5

Situation of copyright protection for computer programs in Poland and existing legislation.

Bielecki J.: FORTH — compiler and structural statements definition

INFORMATYKA 1985, No. 7, p. 9

Definition exaples for complex compilers and structural statements of the FORTH language in fig-FORT Version.

Suraj Z.: Problems of teaching informatics in secondary schools

INFORMATYKA 1985, No. 7, p. 11

Presentation of inquiry results oriented towards teaching informatics in secondary schools. The inquiry was realised in informatics teachers and university candidates environment. Proposals for didactic process improvement are presented.

Terminologia

Oprócz terminów dotyczących bezpośrednio konstrukcji ADY, w języku tym występuje pewna grupa terminów ogólnych, dotyczących programowania w ogóle. Część z nich została z różnych względów skorygowana, głównie w tym celu, aby poszczególne nazwy bardziej odpowiadały duchowi języka polskiego. Uwaga ta dotyczy terminów: ścisła typizacja (ang. strong typing) zamiast silna typizacja, zdefiniowany pierwotnie (ang. predefined) zamiast predefiniowany, byt (ang. entity) zamiast rzecz.

JANUSZ ZALEWSKI

WARUNKI PRENUMERATY NA 1986 R.

Prenumeratorzy zbiorowi — jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat na blankiecie „polecenie przelewu" rozszerzonym dla potrzeb Wydawnictwa o część dotyczącą zamówienia. Blankiety te będą dostarczane przez Zakład Kolportażu.

Prenumeratorzy indywidualni — osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie Wydawnictwa lub blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty.
Wpłacać należy na konto NBP III O/M Warszawa 1036-7490-139-11.

Prenumerata ulgowa — przysługuje wyłącznie osobom fizycznym — członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły.

Sposób zamawiania prenumeraty taki sam jak dla prenumeraty indywidualnej.

Prenumerata ze zleceniem wysyłki za granicę — zamawia się tak jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy. Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Przedpłaty na prenumeratę przyjmowane są w terminach:
— do 10 listopada na I kwartał, I półrocze i cały rok następny,
— do 28 lutego na II, III, IV kwartał i II półrocze,
— do 31 maja na III, IV kwartał i II półrocze,
— do 31 sierpnia na IV kwartał.

U w a g a!

Wpłaty na dwumiesięczniki przyjmowane są na okresy półroczne lub roczne.

Informacji o prenumeracie udziela — Zakład Kolportażu Wydawnictwa NOT-SIGMA, ul. Bartycka 20, 00-716 Warszawa, lub skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 w. 249, 293, 297, 299 oraz 40-35-89 i 40-30-86.

Egzemplarze archiwalne czasopism — można nabyć za gotówkę w Klubie Prasy Technicznej w Warszawie ul. Mazowiecka 12, tel. 27-43-65 oraz w Dziale Handlowym Wydawnictwa ul. Bartycka 20 skr. poczt. 1004, 00-950 Warszawa, na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena miesięcznika INFORMATYKA została ustalona na 120 zł za numer (35 zł — cena ulgowa).

Cena prenumeraty wg cennika					
kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
360	105	720	210	1440	420

FRAKTALE

Rysunek na czwartej stronie okładki przedstawia kolejny fraktal — zbiór Julia. Tym razem iterowano funkcję c sin (z), gdzie c = 1 + 0.23i. Niewielka zmiana (patrz nasza grafika w Informatyce 2/1985), a o ile, różny efekt.

RAFAŁ PIETRAK
JAKUB TATARKIEWICZ

CSK—Computer Studio Kajkowscy

81-505 GDYNIA ORŁOWO, ul. Balladyny 3B, tel. 29-00-18

Komputer osobisty może być przydatny niemal na każdym stanowisku pracy. Wymaga jednak odpowiedniego oprogramowania użytkowego. W ramach tego oprogramowania oferujemy zainteresowanym dostawę uniwersalnych pakietów programowych:

BANK DANYCH CSK, TABPLAN CSK, TEKST CSK, TRANSCOM CSK

To doskonałe narzędzia pracy dla każdego. Aby z nich korzystać, nie trzeba być informatykiem! Zupełnie samodzielnie można tworzyć złożone systemy zarządzania przedsiębiorstwem, każdym przedsiębiorstwem; nawet najbardziej specyficzne uwarunkowania nie są przeszkodą.
To jednak jeszcze nie wszystko... Kiedy dotychczasowe problemy łatwo i szybko zostały rozwiązane — pojawiają się zupełnie nowe. Można wtedy bez kłopotów samemu udoskonalić dotychczasowy system!

BANK DANYCH CSK, TABPLAN CSK, TEKST CSK, TRANSCOM CSK

składają się w zakładowe systemy płacowe, osobowe, finansowo-księgowe lub magazynowe. Korzystając z nich, z łatwością można prowadzić planowanie, kalkulacje i sprawozdawczość. Można też sporządzać kosztorysy i oferty, a nawet prowadzić „automatyczną" korespondencję czy redagować dowolne teksty. Można wreszcie skorzystać z już zgromadzonych zasobów na komputerze ODRA (pod nadzorem systemu GEORGE-3), wykorzystując komputer osobisty jako inteligentny terminal — stację lub emulator TTY.

Nowość:

Oferujemy system operacyjny kompatybilny z CP/M 2.2. dla mikrokomputerów ROBOTRON 5120/5130 oraz systemy finansowo-księgowe FK dla dowolnych mikrokomputerów

---

Szczegółowych informacji udziela:

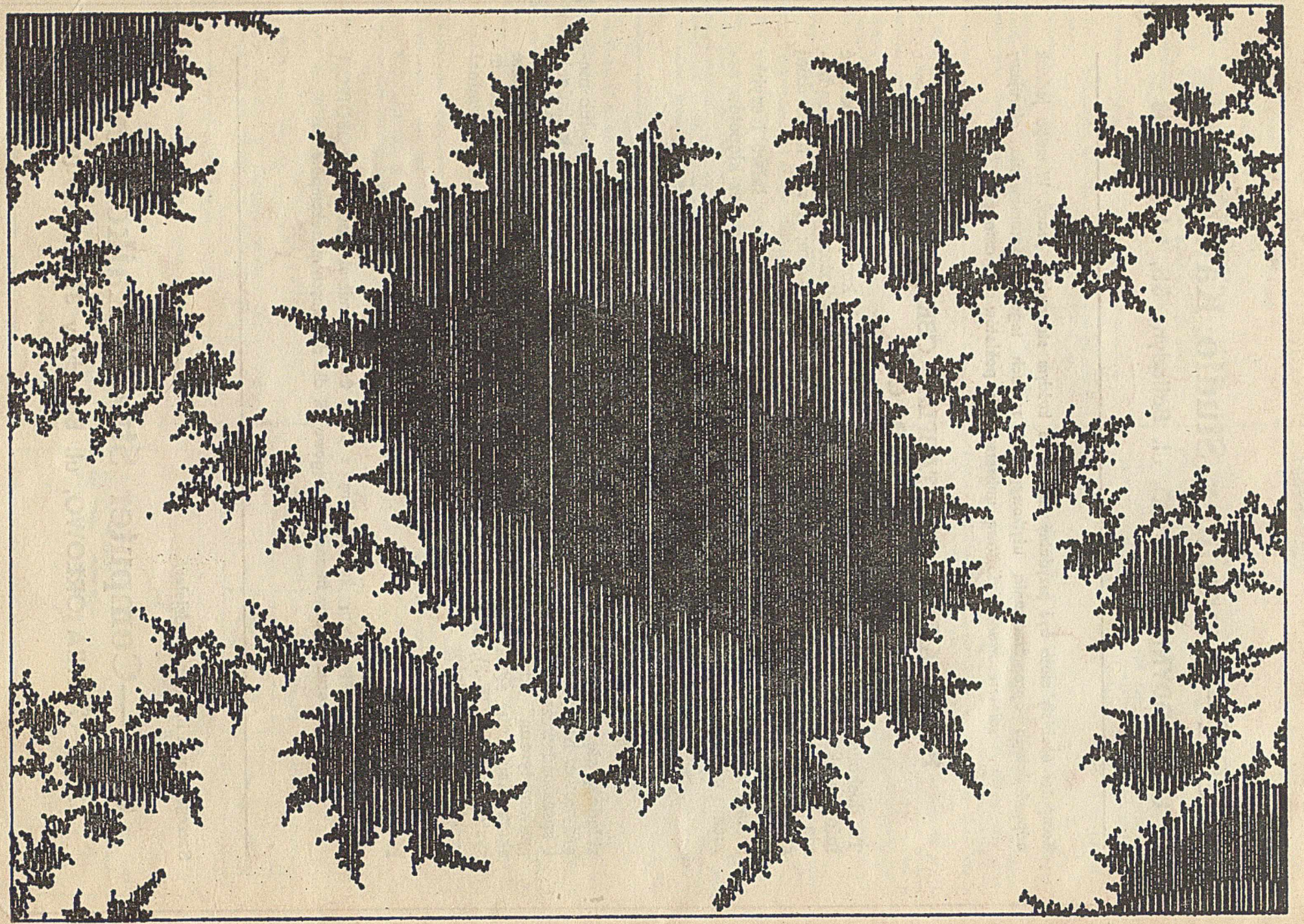
CSK—Computer Studio Kajkowscy

81-505 GDYNIA ORŁOWO, ul. Balladyny 3B, tel. 29-00-18

EO/129/K/85

FRAKTALE

RAFAŁ PIETRAK, JAKUB TATARKIEWICZ



Opis grafiki na str. 32