mikroKLAN 16

4

1985



# informatyka



Język programowania C
Systemy informacyjne
Metody zarządzania
MACINTOSH

WYDAWNICTWO NOT SIGMA

Cena 100 zł

# informatyka

4 1985

Nr 4

Miesięcznik Rok XX

Kwiecień 1985

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





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. 0127/85. Obj. 4,0 ark. druk. Nakład 6100 egz. N-18.

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:



W NAJBLIŻSZYCH NUMERACH:

- Wojciech Cellary i Jan Węglarz o nauczaniu mikroinformatyki
- Wacław Iszkowski o superkomputerze CRAY
- Jan Bielecki o standardzie operacji wejścia-wyjścia w ADZIE
- Jerzy Karczmarczuk o efektywności obliczeń numerycznych na komputerach SINCLAIRA
- Marian Furman, Zbigniew M. Nowicki i Jerzy Solak o pakiecie do przetwarzania informacji za pomocą MERY 400
- Halina Kontkiewicz-Chachulska o SICOBIE '84
- Jacek Irlik o kwalifikacji prawnej programu komputerowego

## Drodzy Czytelnicy

W związku z licznymi pytaniami, dotyczącymi nieukazywania się w planowanych terminach kolejnych numerów wydawanych przez nas tytułów, uprzejmie zawiadamiamy wszystkich zainteresowanych, że w większości przypadków opóźnienia zostały spowodowane trudnościami natury technicznej, głównie brakiem papieru i farb drukarskich odpowiedniej jakości oraz nierytmicznymi dostawami tych materiałów a także dużą awaryjnością maszyn poligraficznych.

W roku bieżącym, w związku ze zmniejszeniem, o 20% przydziału papieru dla naszego Wydawnictwa zmuszeni zostaliśmy do wydawania zeszytów o zmniejszonej objętości i łączonych, a w niektórych przypadkach musieliśmy także ograniczyć nakłady. Sytuacja, w jakiej znalazła się cała prasa techniczna, znacznie utrudnia realizację podjętych przez tę prasę działań na rzecz wdrażania reformy gospodarczej, aktywizacji postaw społecznych i działalności innowacyjnej inżynierów i techników. O roli i znaczeniu czasopism technicznych świadczy fakt systematycznego wzrostu liczby prenumerowanych pism i ich egzemplarzy. Dla przykładu wzrost ten w 1985 r. wyniósł 15%.

Traktując wydawanie i rozwój czasopism technicznych jako podstawowy cel naszej działalności edytorskiej i działalności Naczelnej Organizacji Technicznej dla środowiska technicznego, podejmujemy i będziemy podejmowali wszelkie działania zmierzające do wywiązania się z przyjętych przez nas zobowiązań.

W sprawie zapewnienia przydziału papieru, umożliwiającego wydawanie czasopism technicznych bez zmiany parametrów wydawniczych i zgodnie z zamówieniami prenumeratorów, zwróciliśmy się do Ministerstwa Kultury i Sztuki, jak również do Komisji Planowania przy Radzie Ministrów. Nasze dotychczasowe wystąpienia nie przyniosły jednak spodziewanych rezultatów. Dalsze negocjacje trwają.

W związku ze złym stanem poligrafii w kraju, Naczelna Organizacja Techniczna i Wydawnictwo NOT SIGMA podjęły dwa lata temu budowę drukarni. Własny zakład poligraficzny zostanie uruchomiony w 1986 r. i umożliwi wydawanie branżowych czasopism niskonakładowych bez opóźnień i na lepszym poziomie edytorskim. W naszych staraniach spotykamy się na co dzień ze rzozumieniem i pomocą ze strony wielu przedsiębiorstw i zrzeszeń, które odstąpiły Wydawnictwu papier lub przekazały własne środki dewizowe na zakup papieru za granicą. Tą drogą wszystkim, którzy wspierają nasze działania dziękujemy.

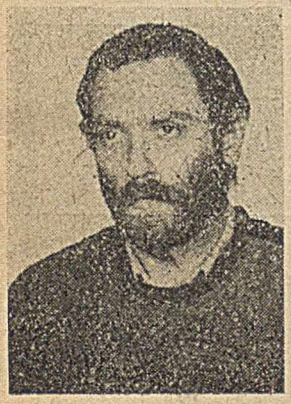
Za zaistniałą sytuację, utrudniającą korzystanie z prasy technicznej, serdecznie przepraszamy naszych Czytelników, Autorów i Współpracowników czasopism.

**Wydawnictwo NOT SIGMA**

**JULIAN WINIEWSKI**
**Instytut Podstaw Informatyki PAN**
**Warszawa**

# Język programowania C (I)

**C jest językiem programowania, który został opracowany jednocześnie z systemem operacyjnym UNIX i służył jako narzędzie do pisania tego systemu. Obowiązujący raport języka opublikowano w 1978 roku, niemniej język jest starszy i wiele kompilatorów akceptuje jego starsze wersje. Język C wywodzi się z języka BCPL i — podobnie jak poprzednik — jest przeznaczony do programowania systemowego.**



Mgr JULIAN WINIEWSKI ukończył Wydział Matematyki i Mechaniki Uniwersytetu Warszawskiego (1971). Zajmuje się językami programowania i ich translatorami. Bierze udział w pracach nad systemem metatranslatora. Obecnie pracuje w IPI PAN — nad kompilatorem języka C.

**Wydaje się, że ideą przyświecającą twórcom języka C było uzyskanie możliwe dużego podobieństwa do języka maszyny. Rzeczywiście, bardzo wiele konstrukcji języka tłumaczy się niemal bezpośrednio na kod maszynowy. Podobieństwo to przejawia się nie tylko w użyciu operatorów, ale również — a może przede wszystkim — w dużej swobodzie, z jaką argumenty różnych typów (także i wskaźniki) mogą być przemieszane w wyrażeniach. Widać w tym ślady pochodzenia od języka beztypowego, jakim jest BCPL.**

**Bardzo silnym narzędziem programowania jest preprocesor będący — co jest rzadkością wśród języków wysokiego poziomu — częścią definicji języka. Najistotniejszą chyba cechą języka C jest osadzenie go w systemie operacyjnym. Standardowa biblioteka C jest bardzo obszerna, a stanowiące ją funkcje umożliwiają m.in. bezpośredni i właściwie nieograniczony, przynajmniej w przypadku UNIXA, dostęp do systemu operacyjnego.**

**Wymienione cechy spowodowały, że w ostatnich latach język C zdobył sobie nie tylko duże uznanie, lecz także stale rozszerzający się rynek.**

Artykuł ten składa się z trzech części. W pierwszych dwóch znajdzie się raczej suchy opis samego języka, natomiast trzecia część będzie poświęcona przykładom, w których zilustrowana zostanie m.in. metodologia programowania w języku C.

## ELEMENTARNE SKŁADNIKI PROGRAMU

Program jest ciągiem identyfikatorów, operatorów, separatorów, stałych i słów zastrzeżonych. Znaki odstępu, tabulacji i nowej linii mogą oddzielać od siebie identyfikatory lub słowa zastrzeżone. W miejscu tych znaków mogą wystąpić również komentarze.

Komentarz jest to dowolny ciąg znaków zaczynający się od sekwencji **/***, a zakończony sekwencją ***/**. Wystąpienie ciągu **/*** wewnątrz komentarza nie oznacza już początku nowego komentarza — komentarze nie mogą się więc zagnieżdżać.

Identyfikatorem jest sekwencja liter i cyfr zaczynająca się od litery. Podkreślenie (**_**) jest uważane za literę. Duże i małe litery są rozróżniane. Identyfikator stanowi nazwę zmiennej, etykiety lub funkcji, bądź jest słowem zastrzeżonym. W niektórych implementacjach długość identyfikatorów zmiennych zewnętrznych może być ograniczona np. do 6—8 znaków, a duże i małe litery mogą być utożsamiane.

Słowa zastrzeżone są to identyfikatory o specjalnym znaczeniu w języku i nie wolno używać ich jako nazw (chyba że przedefiniuje się je w preprocesorze). Lista tych słów jest następująca:

| | | | |
|---|---|---|---|
| **auto** | **double** | **int** | **struct** |
| **break** | **else** | **long** | **typedef** |
| **case** | **extern** | **register** | **switch** |
| **char** | **float** | **return** | **union** |
| **continue** | **for** | **short** | **unsigned** |
| **default** | **goto** | **sizeof** | **while** |
| **do** | **if** | **static** | |

Niektóre kompilatory rezerwują jeszcze słowa **fortran** i **asm**, a nowsze wersje języka również — **entry**, **enum** i **void**. Znaczenie poszczególnych słów omówię w dalszej części artykułu.

W języku C używa się kilku rodzajów stałych. Reprezentują one wartości któregoś z typów podstawowych. Stałe te można zapisywać w różnej postaci.

Stałą całkowitą jest ciąg cyfr. Jeśli pierwszym znakiem tego ciągu nie jest cyfra 0, to traktowany jest on jako liczba dziesiętna, w przeciwnym wypadku — jako ósemkowa. Jeśli natomiast ciąg zaczyna się od 0x lub 0X, to oznacza liczbę szesnastkową. W zapisie szesnastkowym na oznaczenie wartości 10—15 można używać liter a-f lub A-F. Poniżej podano przykłady zapisu stałych całkowitych:

```
037777  0xf19A  655  35
```

Jeśli wartość stałej przekracza maksymalną wartość typu całkowitoliczbowego, to traktowana jest ona jako długa (ang. long). Stałe całkowite mniejsze od maksymalnej wartości, ale zakończone literą l lub L, są także typu **long**, np.:

```
123456  0x90FF21  124L
```

Stałą znakową jest dowolny znak ujęty w apostrofy, np. 'c'. Wartością takiej stałej jest liczbowa wartość kodu tego znaku w maszynie. Znaki tzw. niegraficzne lub specjalne można przedstawiać — jak poniżej — za pomocą znaku / (ang. backslash):

| | | |
|---|---|---|
| **nowa linia** | **LF** | **/n** |
| **tabulacja** | **HT** | **/t** |
| **cofnięcie** | **BS** | **/b** |
| **powrót karetki** | **CR** | **/r** |
| **zmiana strony** | **FF** | **/f** |
| **backslash** | **/** | **//** |
| **apostrof** | **'** | **/'** |

Każdy znak można również zapisać w postaci wartości liczbowej jego kodu, tj. w postaci **/ddd**, gdzie d jest liczbą ósemkową. Jeśli po znaku / nie nastąpi żaden ze znaków wymienionych wyżej, to znak / jest ignorowany.

Stałe zmiennoprzecinkowe składają się z części całkowitej i ułamkowej, kropki dziesiętnej, litery e lub E wraz z wykładnikiem i znakiem. Stałe te mają ogólną postać:

```
cz_całkowita . ułamek e ± wykładnik
```

W takiej stałej musi wystąpić co najmniej jeden element z pary **<część całkowita, ułamek>** oraz co najmniej jeden element z pary <. (kropka) , e>. **Część całkowita, ułamek** i **wykładnik** są stałymi w postaci całkowitej. Przykładowe stałe zmiennoprzecinkowe:

```
0.0  3e10  123.456E-7  3.14159
```

Innym rodzajem stałych są stałe tekstowe (ang. string). Typem takiej stałej jest tablica znaków, a zapisuje się ją w postaci ciągu znaków ujętego w cudzysłów ("..."). Każdy tekst, nawet jeśli jest identyczny z innym, stanowi odrębny obiekt. Konwencje zapisu znaków w ciągu są takie same jak dla stałych znakowych. Znak nowej linii **(/n)** jest ignorowany, co umożliwia pisanie długich tekstów, nie mieszczących się w jednej linii.. Wewnątrz tekstu znak cudzysłowu należy zapisywać jako /". Koniec tekstu jest oznaczany przez znak '/0' (NUL), umieszczany tam automatycznie przez kompilator. Oto przykłady stałych tekstowych:

```
„to jest /n tekst"
"" (tekst pusty)
```

## DEKLARACJE

Program w języku C ma strukturę blokową. Na początku bloku należy zadeklarować wszystkie identyfikatory używane w tym bloku (oprócz etykiet), chyba że deklaracje identyfikatorów pojawiły się w którymś z bloków nadrzędnych. Deklaracja składa się zasadniczo z trzech części: specyfikatora klasy pamięci, specyfikatora typu oraz listy deklaratorów.

### Specyfikator typu

Specyfikator typu określa typ zmiennej, którym może być: typ podstawowy, struktura, unia lub typ zdefiniowany uprzednio w deklaracji **typedef**. Struktury i unie, analogiczne do rekordów w PASCALU, zostaną omówione w drugiej części artykułu. Typy podstawowe są określane następującymi specyfikatorami: **int** — całkowitoliczbowy, **float** — zmiennoprzecinkowy oraz **char** — znakowy. Ponadto, można użyć specyfikatorów **long**, **short** i **unsigned** w kombinacji z poprzednimi, np. **long int**, **short int**, **unsigned int** lub **long float** (ostatni jest równoważny specyfikatorowi **double**). Znaczenie przedrostków **long** i **short** (liczba bitów) zależy od komputera docelowego (p. tabela). Specyfikator **unsigned** oznacza, że w maszynowej reprezentacji wartości liczbowej nie wyróżnia się bitu znaku. Jeśli w deklaracji nie wystąpi specyfikator typu podstawowego, to przyjmuje się, że jest nim **int**. Przykładowo — **long** jest równoważny specyfikatorowi **long int**.

| Specyfikatory | Rodzaj komputera | | | |
|---|---|---|---|---|
| | PDP-11 | HONEY-WELL | IBM/370 | VAX11/780 |
| char | 8 | 9 | 8 | 8 |
| short | 16 | 36 | 16 | 16 |
| int | 16 | 36 | 32 | 32 |
| long | 32 | 36 | 32 | 32 |
| float | 32 | 36 | 32 | 32 |
| double | 64 | 72 | 64 | 64 |

### Deklaratory

Na liście deklaratorów występują identyfikatory zmiennych wraz ze znacznikami — reprezentują zmienną prostą, wskaźnik (ang. pointer), tablicę lub funkcję. W celu właściwego wiązania tych znaczników, wolno używać nawiasów.

Poniższe deklaratory definiują:

```
char x;     — zmienna x typu znakowego
long *a;    — wskaźnik a do typu long int
int m[5];   — tablicę 5-elementową m o wartościach całkowitych
float f( ); — funkcję f obliczającą wartość typu float.
```

Deklaratory mogą być dowolnie złożone, chociaż w kompilatorach na ogół ogranicza się je do kilku znaczników. Przykładowo — deklarator:

```
long unsigned *(*(*a) [15] [μ] ( )
```

definiuje automatyczną zmienną będącą wskaźnikiem do dwuwymiarowej tablicy (o 15 elementach w pierwszym wymiarze) wskaźników do funkcji obliczających wskaźnik do typu **long unsigned int.**

Nie wszystkie jednak kombinacje typów i deklaratorów są dopuszczalne. Przykładowo — funkcje nie mogą obliczać wartości struktur, unii, tablic i funkcji, ale mogą — wskaźnik do takich obiektów. Podobnie, obiekty te nie mogą zawierać funkcji, lecz tylko wskaźniki do nich.

### Inicjowanie

Definiowane zmienne mogą być inicjowane w deklaracjach, np. deklaracja:

```
int x=5;
```

oznacza nadanie wartości początkowej 5 definiowanej zmiennej całkowitej x.

Inicjować można również struktury i tablice — wtedy wartości nadawane poszczególnym składowym zapisuje się jako listę ujętą w nawiasy klamrowe {...}. Elementami tej listy mogą być również inne listy, o ile deklarowany obiekt jest np. tablicą wielowymiarową lub strukturą zawierającą struktury lub tablice. Przykładowo — deklaracja:

```
int t[4] [ ] = {{1, 2, 3}, {4, 5, 6,}, {7, 8, 9}};
```

definiuje 4-elementową tablicę tablic o nieznanej liczbie elementów, ale inicjowanie określa, że są one 3-elementowe i wypełnia pierwsze trzy z tych tablic podanymi wartościami. Inicjowania tablicy znaków (czyli tekstu) dokonuje się znacznie łatwiej:

```
char x [ ] = "abcdef"
```

Wyrażenia inicjujące muszą być wyrażeniami stałymi, tj. mogą zawierać jedynie stale lub uprzednio zdefiniowane zmienne. Zmienne statyczne i zewnętrzne, jeśli nie są inicjowane jawnie, mają domyślną wartość początkową 0:

### Specyfikacja klasy pamięci

W każdej deklaracji może wystąpić co najwyżej jeden spośród pięciu możliwych specyfikatorów klasy pamięci:

**auto** — oznacza zarezerwowanie miejsca w obszarze pamięci zmiennych automatycznych danego bloku; pamięć dla wszystkich takich zmiennych, jest przydzielana przy otwarciu tego bloku, a zwalniana w chwili wyjścia z bloku

**static** — oznacza rezerwację pamięci dla zmiennej w obszarze zmiennych statycznych programu; obszar ten jest przydzielany wraz z rozpoczęciem wykonywania programu, a zwalniany z chwilą jego zakończenia; tak więc otwarcie lub zamknięcie bloku nie powoduje przydziału lub dealokacji pamięci tak zadeklarowanej zmiennej — pamięć ta jest dostępna poza blokiem, w którym wystąpiła deklaracja, niedostępny jest jedynie jej identyfikator

**register** — oznacza zarezerwowanie rejestru maszyny dla tej zmiennej; realizacja takiej specyfikacji zależy od implementacji języka; liczba zmiennych rzeczywiście umieszczanych w rejestrach jest oczywiście ograniczona (czasem nawet do zera); typ tych zmiennych bywa najczęściej ograniczony tylko do wskaźników i całkowitych, niemniej liczba użytych w programie specyfikatorów nie jest ograniczona — zmienne, którym nie starczyło rejestrów, są traktowane jako automatyczne

**extern** — w odróżnieniu od poprzednich specyfikacji nie jest to definicja pamięci, lecz informacja, że dla danej zmiennej istnieje definicja zewnętrzna

**typedef** — specyfikacja ta nie dotyczy w ogóle zmiennych, lecz oznacza, że w deklaracji definiowany jest typ, a występujący w niej identyfikator jest jego nazwą; identyfikator ten może być następnie użyty w deklaracjach zamiast typu.

Jeśli w deklaracji nie wystąpi żaden ze specyfikatorów, to domyślnie przyjmuje się, że jest on **auto.** Po deklaracjach nowych typów:

```
typedef int *INTP;
typedef struct {float rzeczywista, urojona;} zespolona;
```

można z ich pomocą deklarować zmienne, np.:

```
zespolona z, *zp;
INTP a, tab [ ];
```

Odtąd z ma wartość zespoloną, a **zp** jest wskaźnikiem do takiej wartości, natomiast **a** jest wskaźnikiem do wartości całkowitej, a **tab** tablicą takich wskaźników.

## WYRAŻENIA

Wyrażenia w języku C są zbudowane ze stałych, identyfikatorów, wywołań funkcji oraz tzw. „l-wartości", czyli adresów (odwołań do zmiennych), przedzielonych znakami operatorów. „L-wartością" jest wyrażenie, którego wartością jest adres obiektu w pamięci. Angielski termin **left value** (l-value) pochodzi stąd, że wyrażenie tego rodzaju występuje po lewej stronie przypisania (e1=e2).

Każde odwołanie do zdefiniowanego obiektu jest więc adresem. Najprostszym wyrażeniem adresowym jest identyfikator oznaczający odwołanie do zmiennej. Do innych wyrażeń należą:

**e1 [e2]** — odwołanie do elementu tablicy, przy czym wartość **e1** musi być wskaźnikiem a **e2** — całkowitą

**e1.id** — odwołanie do elementu struktury lub unii, gdzie **id** musi być identyfikatorem pola struktury, a wartością **e1** jest adres

**e1—>id** — podobnie jak poprzednio, przy czym wartością **e1** musi być wskaźnik.

Ponadto w języku C istnieje operator * (wyrażenie), którego argumentem jest wskaźnik, a wynikiem adres.

### Operatory jednoargumentowe

Jest to najliczniejsza grupa operatorów w języku C, w której skład wchodzą:

***ptr** — działający na wskaźniki

**&adr** — którego wynikiem jest wskaźnik do obiektu o podanym adresie

**—expr** — minus jednoargumentowy

**!expr** — negacja, której wynikiem jest 1, gdy wartością wyrażenia jest zero, a 0 w przeciwnym przypadku

**~expr** — uzupełnienie bitowe

**++adr** — zwiększenie o 1 wartości pola o podanym adresie, wartością wyrażenia jest wartość pola zwiększona o 1

**——adr** — podobnie jak wyżej lecz wartość jest zmniejszana o 1

**adr++** — zwiększenie o 1 wartości pola o podanym adresie, przy czym wartością wyrażenia jest poprzednia wartość pola

**adr——** — podobnie, lecz dotyczy zmniejszania wartości o 1

**(typ)expr** — operator formowania (ang. cast) powodujący konwersję typu wyrażenia na typ podany w nawiasach

**sizeof expr** — którego wynikiem jest liczba całkowita oznaczająca rozmiar wartości wyrażenie w bajtach (za bajt przyjmuje się jednostkę pamięci, w której mieści się znak)

**sizeof (typ)** — podobnie jak wyżej, lecz w odniesieniu do typu, a nie wyrażenia.

### Operatory arytmetyczne

Do tej grupy zalicza się operatory: dodawania (+), odejmowania (—), mnożenia (*), dzielenia (/) oraz reszty

(%). Operator dzielenia umieszczony między argumentami całkowitymi oznacza dzielenie całkowitoliczbowe. Dla liczb całkowitych dodatnich obcinana jest bardziej znacząca część ilorazu. Jeśli któryś z argumentów całkowitych jest ujemny, to obcięcie wyniku jest zależne od implementacji. Operatory dodawania i mnożenia są łączne i przemienne, dlatego nie jest zapewnione, że wynik ich działania oblicza się począwszy od lewego argumentu.

Wyrażenie **a+b+c** może być więc obliczane jako: **(a+b)+c** lub **a+(b+c)**, zależnie od implementacji.

### Operatory przesunięcia

Do tej grupy należy przesunięcie w lewo << i przesunięcie w prawo >>. Powodują one przesunięcie wartości pierwszego z argumentów (traktowanej jako ciąg bitowy) o **n** pozycji, gdzie **n** jest wartością drugiego argumentu. Wartość ta musi być dodatnia i nie większa od długości lewego argumentu liczonej w bitach. Przy przesunięciu w lewo mniej znaczące bity są wypełniane zerami. Przesunięcie w prawo jest logiczne (bity bardziej znaczące są wypełniane zerami), gdy pierwszy argument jest typu **unsigned**. W przeciwnym wypadku może to być — co zależy od implementacji — przesunięcie arytmetyczne (na bitach bardziej znaczących powiela się bit znaku).

Przykładowo — wartością wyrażenia **x<<16** będzie 0, jeśli zmienna **x** jest 16-bitowa. Gdy jest to zmienna 32-bitowa, to jej dwa mniej znaczące bajty zostaną wyzerowane, a ich poprzednia zawartość zostanie przesunięta do dwóch bardziej znaczących, co jest równoważne wyrażeniu: **x*65536.**

### Operatory porównań

W tej grupie występuje pełny zestaw sześciu operatorów, tj. operatory: mniejszości <, mniejszości lub równości <=, większości >, większości lub równości >=, równości == oraz nierówności !=. Wynikiem operacji jest 1, gdy porównanie jest prawdziwe, lub 0 — gdy jest fałszywe.

Ponieważ wynik operacji jest liczbą całkowitą, to wyrażenie **a<b<c** jest poprawne, lecz oznacza — po wykonaniu pierwszej operacji — **0<c** lub **1<c.**

### Operatory bitowe

Do tej grupy zalicza się trzy operatory: **i** (&), **lub** (|) oraz **lub rozłączne** (^). Działają one na liczby całkowite traktowane jako ciągi bitowe. Odpowiednie operacje są wykonywane na kolejnych odpowiadających sobie bitach z obu ciągów. Ciąg wynikowy jest traktowany jako liczba całkowita i zawiera jedynki na tych pozycjach, na których występowały jedynki w obu argumentach (dla &), przynajmniej jedna jedynka (dla |) lub dokładnie jedna jedynka (dla ^).

Kolejność obliczania wartości wyrażeń z operatorami bitowymi, jako że są one łączne i przemienne, może zostać zmieniona przez kompilator. Przykładowo — wartość wyrażenia **(i&4096)>>12** jest 1, jeżeli zmienna **i** ma na dwunastym bicie jedynkę, w przeciwnym wypadku — 0.

### Operatory logiczne

W języku C występują dwa operatory logiczne: koniunkcja (&&) i alternatywa (||). Wynikiem koniunkcji jest 1, gdy oba argumenty są niezerowe, a 0 — w przeciwnym wypadku. Ponadto, jeśli pierwszy argument ma wartość zero, to drugiego już się nie bierze pod uwagę. Wynikiem alternatywy jest 0, gdy oba argumenty mają wartości zerowe, a 1 — w przeciwnym wypadku. Podobnie, drugiego argumenty nie bierze się pod uwagę, gdy pierwszy ma wartość niezerową.

Tak więc, w odróżnieniu od wyrażeń bitowych, wartości wyrażeń logicznych są obliczane począwszy od lewego argumentu. Przykładowo — wartością wyrażenia.

```
(c-'0') * (c>='1' && c<='9')
```

jest 1, 2, ... lub 9, gdy wartością zmiennej znakowej **c** jest znak cyfry, odpowiednio, '1', '2', ..., '9', a 0 — dla wszystkich innych wartości **c**. Warto też zwrócić uwagę, że wartość wyrażenia nie zależy od implementacji języka C.

### Przypisania

W języku C przypisanie jest również operacją, a wartością wyrażenia jest w tym przypadku wartość przypisywana. Istnieje jedenaście operatorów przypisania. Proste przypisanie ma postać:

```
adr = expr
```

Należy pamiętać, że przypisanie nie zawsze dokonuje konwersji typu prawego argumentu na typ lewego argumentu, jak np. przy przepisaniu wartości całkowitej — zmiennej znakowej. Może to więc być zwykłe przepisanie z jednego pola pamięci do innego, podczas którego może nastąpić obcięcie przepisywanej wartości. Pozostałe operatory przypisania to: +=, -=, /=, %=, *=, >>=, <<=, &=, |=, ^=. Powodują one wykonanie operacji na obu argumentach przed przypisaniem np. **e1 += e2** jest równoważne

```
e1=e1+e2
```

przy czym wartość wyrażenia **e1** jest obliczana tylko raz. Przypisania dokonuje się począwszy od prawego argumentu. Oto przykłady operacji przypisania:

```
i=a[i++]
x += tb[x]
z<<=8
```

### Pozostałe operatory

Operator warunkowy jest jedynym operatorem trójargumentowym. Ma on postać:

```
expr1 ?expr2 : expr3
```

Wynikiem jego działania jest wartość drugiego wyrażenia, jeśli wartość pierwszego była niezerowa, lub trzeciego wyrażenia — w przeciwnym wypadku. Obliczona jest wartość tylko jednego z tych dwóch wyrażeń. Wyrażenia drugie i trzecie muszą mieć ten sam typ lub typy sprowadzalne do tego samego, tak aby typ całego wyrażenia był jednoznacznie określony. Przykładowo wartością wyrażenia:

```
a>=b ? a : b
```

jest większa z liczb a, b.

Para argumentów rozdzielona przecinkiem jest również wyrażeniem. Obliczana jest najpierw wartość pierwszego argumentu a następnie wartość drugiego, która stanowi wartość całego wyrażenia. Przykładowo — w wyrażeniu:

```
x = i>= 0 ? 1 : (i = -i, 0)
```

zmiennej **x** zostanie przypisana jedynka, gdy **i** jest nieujemne, natomiast w przeciwnym przypadku — zero, ale uprzednio **i** stanie się dodatnie.

### Hierarchia operatorów

Operatory można podzielić na 14 grup, zgodnie z malejącym poziomem priorytetu:
— jednoargumentowe *, &, —, ~, ++, ——
— mnożenia *, /, %,
— dodawania +, —
— przesunięcia <<, >>
— porównania <, <=, >, >=
— równości ==, !=
— „i" bitowe &
— „lub" bitowe |
— „rozłączne lub" bitowe ^
— koniunkcja &&
— alternatywa ||
— warunek ?:
— przypisania =, +=, -=, *=, /=, %=, <<=, >>=, &=, |=, ^=
— przecinek ,

Wartości wyrażeń są zasadniczo obliczane począwszy od lewego argumentu. Operatorami grupującymi argumenty od prawej strony są — oprócz jednoargumentowych prefiksowych — przypisania i warunek. Ponadto, obliczanie wartości wyrażeń zawierających operatory łączne i przemienne, tj. +, *, &, |, ^, zależy od kompilatora.

**Konwersja typów**

Jak już stwierdzono, zmiana typu wartości może nastąpić wskutek jawnego użycia operatora formowania (ang. cast). Ponadto jednak, konwersji dokonuje się automatycznie dla każdego argumentu funkcji i operacji, a mianowicie: każdy argument typu **char** lub **short** jest zawsze zamieniany na **int**, a typu **float** — na **double**. Po wstępnej konwersji następuje konwersja typów argumentów wyrażenia, aby sprowadzić je do wspólnego typu, według następującej reguły:

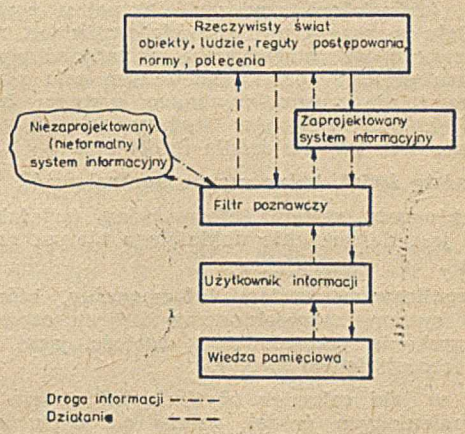
- jeśli jeden z argumentów jest typu **double**, to drugi jest zmieniany na **double** i taki jest też typ wyniku
- jeśli jeden z argumentów jest typu **long**, to drugi jest zmieniany na **long** i taki jest też typ wyniku
- jeśli jeden z argumentów jest typu **unsigned**, to drugi jest zmieniany na **unsigned** i taki jest też typ wyniku
- w każdym innym wypadku oba argumenty muszą być **int** i wynik też jest tego typu.

**Artykuł prof. Landa odbiega od podstawowej tematyki naszego czasopisma, związanej przede wszystkim z technologią informatyczną. Ilustruje on jednak ważny proces: jedną z podejmowanych ostatnio prób naukowego opisu systemów informacyjnych jako decydującego czynnika rozwoju społecznego końca dwudziestego wieku, a także określenia w tych systemach roli i funkcji informatyki — w ramach dyscypliny naukowej. (Red.)**

**FRANK LAND**
**London School of Economics**
**Wielka Brytania**

# Dziedzina systemów informacyjnych

Każda organizacja potrzebuje systemu informacyjnego, który zapewni sprawność jej funkcjonowania. Każdy system informacyjny — począwszy od takiego, który wspomaga właściciela jednoosobowego przedsiębiorstwa lub radę parafialną małej wioski, a kończąc na systemie wspomagającym korporację o międzynarodowym zasięgu lub administrację państwową uprzemysłowionego kraju — składa się z pewnej liczby komponentów. Niektóre z tych komponentów są artefaktami (ang. artifacts) [1]), jak ołówek i papier, procesory tekstów, komputery i sieci teleinformatyczne, systemy operacyjne i podręczniki procedur. Ale jednocześnie każdy system operacyjny potrzebuje również ludzi, którzy tworzą artefakty, współdziałają z nimi, wykorzystują je. Nawet całkowicie zautomatyzowane systemy informacyjne (gdyby stworzenie ich było już możliwe, tak jak możliwe jest zbudowanie całkowicie zautomatyzowanego wydziału produkcyjnego) będą zawsze wymagały udziału ludzi, którzy zapewnią niezbędne zabezpieczenie na wypadek awarii. W praktyce systemy informacyjne opierają się na ludziach posługujących się artefaktami i z nimi współdziałających.




**Model systemu informacyjnego**

Systemy informacyjne są bardzo zróżnicowane pod względem stopnia formalizacji i normalizacji, a także strukturalizacji metod operowania informacją. Metody takie są przeciwieństwem metod niesformalizowanych, tworzonych często doraźnie i w sposób subiektywny. Wszystkie systemy informacyjne zawierają elementy obu typów metod.

FRANK LAND jest profesorem analizy systemów w Londyńskiej Akademii Ekonomicznej (London School of Economics). Po uzyskaniu w 1953 roku dyplomu tej uczelni, rozpoczął pracę w zespole ds. zastosowań komputerów w firmie LEO COMPUTERS, będącej pionierem elektronicznego przetwarzania danych na potrzeby zarządzania oraz seryjnej produkcji komputerów w Wielkiej Brytanii. W zespole tym pracował do 1967 roku, zajmując kolejno stanowiska: programisty, analityka systemu, konsultanta i kierownika ds. marketingu. Następnie powrócił do macierzystej uczelni, gdzie dzięki dotacjom państwowym zorganizował działalność dydaktyczną (studia zawodowe i magisterskie) i badawczą, w dziedzinie analizy systemów (zespół ds. rozwoju metod analizy systemów). Prof. Land jest działaczem Brytyjskiego Towarzystwa Informatycznego (BBC) oraz Międzynarodowej Federacji Przetwarzania Informacji (IFIP).

Oryginalny tytuł artykułu brzmi „The Information Systems Domain". Autor udostępnił nam tekst w formie maszynopisu wraz ze zgodą na publikację.

[1]) Artefakty — termin pochodzenia łacińskiego używany głównie w biologii, oznaczający — według definicji podanej w Wielkiej Encyklopedii Powszechnej PWN (tom I, str. 398) — „struktury obserwowane w preparatach mikroskopowych, powstałe w procesie przygotowania preparatu (czasem przez zniekształcenie istniejących struktur), lecz nie występujące w żywym organizmie".

Model przykładowego systemu informacyjnego pokazany na rysunku ilustruje zależności między różnymi komponentami takiego systemu oraz związane z nim aspekty formalne i nieformalne.

System informacyjny istnieje w rzeczywistym świecie, który składa się z:

- **obiektów** (konkretnych, jak maszyny, zapasy materiałów, budynki, lub abstrakcyjnych, jak budżety, rachunki, prognozy sprzedaży)
- **ludzi** (nabywców, odbiorców, kadry kierowniczej, pracowników biurowych)
- **reguł postępowania** (ucieleśnionych w aktach prawnych, procedurach, instrukcjach i praktycznych wskazówkach działania)
- **norm** (często dotyczących szczegółowych sposobów wykonania przedmiotów oraz trybów rozumowania i praktycznego działania)
- **poleceń** (takich jak zamówienie towaru czy zlecenie produkcyjne).

Osoba pojawiająca się na rysunku — użytkownik informacji — ma wykonać określone zadania, odnoszące się do rzeczywistego świata. Osobą tą może być kierownik podejmujący decyzję, np. tak prostą, jak potwierdzenie zamówienia na pręty stalowe lub — złożoną, jaka jest decyzja dotycząca planowania. Użytkownikiem informacji może być również szeregowy pracownik biurowy, wykonujący określone wyspecjalizowane zadania, np. rejestrowanie szczegółów zleceń przeznaczonych do przetwarzania na komputerze.

Aby wykonać swe zadania, osoba ta potrzebuje informacji dotyczących rzeczywistego świata. Informacje takie często można uzyskać bezpośrednio. Kierownik mógłby więc pójść do magazynu stali, aby sprawdzić, jaki istnieje tam zapas prętów. Mógłby on użyć w tym celu również sformalizowanego, specjalnie do tego celu zaprojektowanego sytemu informacyjnego. System taki może składać się z karty magazynowej zapasu materiału, na której pracownik magazynu ma zapisane wszystkie przychody i rozchody prętów stalowych. Może to być także informatyczny system kontroli zapasów, do którego wspomniany kierownik może kierować za pośrednictwem terminala zapytania, pozwalające stwierdzić aktualny poziom zapasu materiału w magazynie. Ale kierownik ten może również zadzwonić do magazyniera i zapytać go o wielkość zapasu prętów, korzystając w tym celu z niesformalizowanego i nie zaprojektowanego systemu informacyjnego. Większość systemów informacyjnych ma więc wspomniane trzy główne źródła informacji:

- **rzeczywisty świat,** który można kontrolować
- **zaprojektowany system informacyjny** (artefakt), którego celem jest dostarczenie dokładnego obrazu rzeczywistego świata
- **niesformalizowany system informacyjny**, który czasem zastępuje system zaprojektowany i jest często używany do uzyskania jakościowej i wartościującej informacji na temat rzeczywistego świata.

Sposób, w jaki człowiek użytkuje informacje pochodzące z wymienionych źródeł, zależy od wielu czynników. Niektóre z nich mają charakter ogólny, a niektóre odnoszą się do określonego środowiska lub sytuacji, w jakiej uzyskuje się potrzebną informację. Do czynników najważniejszych należy zaliczyć:

- **Cechy poznawcze i preferencje osoby uzyskującej informację.** Niektórzy ludzie preferują i lepiej rozumieją informację zapisaną w postaci obrazów lub wykresów. Inni preferują tekst opisowy lub informację prezentowaną w postaci sformalizowanej, a nawet symbolicznej. Wreszcie inni wybierają postać niesformalizowaną i swobodną. Jedna ze szkół psychologicznych rozróżnia istniejący podział ludzi na tych, którzy wolą odbierać informację w trybie ściśle sekwencyjnym i potrzebują uchwycić każdy jej element, oraz tych, którzy wolą widzieć tylko „kształt" wiadomości i są zdolni pojąć jej treść bez konieczności sekwencyjnego rozpoznawania. Nie zostało jeszcze jednoznacznie stwierdzone, co determinuje określony styl poznawczy konkretnej osoby. Bezsporne jest jedynie to, że pomiędzy poszczególnymi ludźmi występują w tym zakresie znaczne różnice. Każda osoba otrzymująca wiadomość będzie odbierała jej treść poprzez filtr poznawczy, który może wybierać, wzmacniać, odrzucać, osłabiać lub zniekształcać poszczególne cząstki wiadomości. W konsekwencji takiego działania nawet zupełnie proste wiadomości mogą być różnie zinterpretowane przez poszczególne osoby. Wiadomości złożone, a więc takie, które wymagają opisu złożonej sekwencji zdarzeń lub uwzględnienia umownych zasad, otrzymają z pewnością wiele różnych interpretacji.
- **Wiedza zawarta w pamięci człowieka.** I tu znów musimy stwierdzić, że sposób działania mózgu nie jest jeszcze, niestety dostatecznie poznany. Nie znamy również sposobu zmierzenia tego, co znajduje się w pamięci danego człowieka, ani ocenienia stopnia dostępności zawartych tam informacji. Nie możemy też stwierdzić jaką wiedzą dysponuje określona osoba. Tym niemniej istnieje niewątpliwie kojarzenie informacji, odebranej za pośrednictwem zmysłów, z wiedzą zmagazynowaną w pamięci człowieka, która determinuje rodzaj działania i reakcje na otrzymane wiadomości. A więc różne osoby mogą różnie odpowiedzieć w podobnych sytuacjach, nawet wtedy, gdy zadanie, które osoby te zamierzają wykonać, jest identyczne. W pewnym stopniu odpowiedzi mogą być z góry przygotowane lub tak dostosowane, aby w określonym stopniu były one możliwe do przewidzenia. Mechanizm taki jest podstawą szkolenia wojskowego. W tym oczywiście znaczeniu, że taki typ szkolenia próbuje zastąpić procesy poznawcze procesami odruchowymi. Ale tego rodzaju procesy nie mogą oczywiście stawić czoła sytuacjom, w których doniesienia informacyjne są niesformalizowane lub nietypowe. Rezultaty procesów poznawczych polegających na skojarzeniu danych odbieranych z otoczenia z wiedzą zmagazynowaną w pamięci są niemożliwe do przewidzenia.
- **Język.** Całość informacji jest przenoszona do określonej osoby w postaci sygnałów lub wiadomości. Aby mogły one nabrać znaczenia, muszą przybrać postać kodu lub języka. Różne języki mają swoje indywidualne cechy, w tym również odrębne podejście do interpretacji kodów. Języki naturalne są bardzo bogate pod względem skali informacji, jaką mogą one ująć. Z drugiej strony — są one jednak bardzo wieloznaczne. Języki sformalizowane, takie, jak zapis matematyczny lub języki programowania, mogą być bardzo precyzyjne, ale także pozbawione własności interpretowania często bardzo szerokiej skali różnych pojęć. Takie pojęcie, jak np. **dobry,** nie może być wyrażone w języku programowania. Język, sposób percepcji oraz pamięć działają współzależnie. Ich działanie ma często źródło w kulturze, wykształceniu i doświadczeniu danego człowieka. Ogólne modele zachowania się, uwzględniające wymienione trzy czynniki, nie mogą jeszcze dostarczyć praktycznych wskazówek dla zaprojektowania systemów informacyjnych, przystosowanych do efektywnego wykorzystywania przez dowolnego użytkownika. Jednakże zrozumienie ważności tych czynników sugeruje, że należy preferować określone podejście do projektowania takich systemów.

Oprócz wyżej opisanych czynników natury ogólnej, istnieje pewna liczba czynników środowiskowych i sytuacyjnych, które mają istotny wpływ na sposób, w jaki dana osoba zareaguje na otrzymaną informację oraz jaki typ źródła informacji będzie ona w praktyce preferowała. Czynniki te obejmują takie sprawy, jak możliwość zrozumienia oraz zaufanie do zaprojektowanego systemu. Zaufanie powstaje w wyniku oceny zalet i wad systemu, a zwłaszcza osiągalnego czasu uzyskania odpowiedzi lub podjęcia działania, stopnia intensywności pracy, wygody lub trudności obserwowania rzeczywistego świata i wielu inych czynników. Stopień swobody działania, jaki może osiągnąć pojedynczy człowiek, będzie zależał od rodzaju zajmowanego przez niego stanowiska w ramach określonej organizacji. Przykładowo — szeregowy pracownik biurowy, wykonujący zrutynizowane i ściśle sprecyzowane w instrukcji zadanie, może okazać się mniej ograniczony w wyborze potrzebnych mu źródeł informacji oraz własnej metody postępowania, aniżeli kierownik wyższego szczebla zarządzania. Ale w praktyce stosunkowo niewiele procedur działania opiera się na ścisłym przestrzeganiu obowiązujących instrukcji, natomiast znacznie szerzej niż to sugerują istniejące procedury korzysta się z systemu nieformalnego, nie ujętego w ramy sprecyzowanych metod postępowania.

Powyższa analiza wskazuje, że **system informacyjny jest systemem społecznym, który zawiera w sobie technologię informacyjną.** Obserwujemy szybkie rozszerzanie się zakresu oddziaływania oraz wzrostu znaczenia tej technologii. Stwierdzenie to nie wyklucza oczywiście możliwoś-

ci, by każdy system o charakterze ogólnym stał się systemem społecznym. Nie jest również możliwe zaprojektowanie efektywnego systemu informacyjnego, obejmującego znaczny zakres technologii, jeśli nie traktuje się go jako systemu społecznego. Nie wystarczy bowiem zaprojektować system techniczny, a dopiero potem usiłować uczynić go dogodnym dla użytkownika, albo też nakazać projektantowi, aby pamiętał o konieczności uwzględnienia w systemie czynników ludzkich.

Przedstawiony model pomaga zdefiniować również dyscypliny naukowe odpowiadające treści studiów na temat systemów informacyjnych. Obejmują one:
**Teorię organizacji** — ponieważ system informacyjny tkwi wewnątrz organizacji oraz istnieje po to, aby spełniać cele organizacyjne.
**Informatykę** — ponieważ zaprojektowany system informacyjny składa się ze sprzętu komputerowego oraz oprogramowania.
**Semiotykę** — ponieważ system informacyjny zajmuje się przekazywaniem wiadomości za pomocą znaków pisarskich.
**Lingwistykę** — ponieważ większość wiadomości jest przekazywana pod postacią języka.
**Psychologię poznawczą** — ponieważ sposób, w jaki użytkownik informacji przetwarza wiadomości pochodzące z systemu informacyjnego, determinują użyteczność tego systemu.

Oczywiście jeszcze wiele innych dyscyplin naukowych odpowiada problematyce ogólnych badań nad systemami informacyjnymi lub ułatwia zrozumienie konkretnego systemu informacyjnego. Analiza zaprezentowanego modelu wskazuje jednak na wyraźnie interdyscyplinarny charakter dziedziny systemów informacyjnych.

Tłumaczył i opracował:
**WŁADYSŁAW KLEPACZ**

**ZBIGNIEW KIERZKOWSKI**
**JACEK MAŁUSZYŃSKI**
**Poznań**

Zarządzanie współbieżną aktualizacją bazy danych (2)

# Przegląd metod zarządzania

**Zarządzanie współbieżną aktualizacją bazy danych polega na szeregowaniu odwołań do bazy danych pochodzących od procesów współbieżnych; dzięki niemu unika się błędów systemu:**

- **zgubienia efektu aktualizacji**
- **odczytu chwilowo niespójnego obrazu bazy danych**
- **wprowadzenia błędu do bazy danych.**

**Zarządzaniem tym zajmuje się koordynator będący częścią Systemu Zarządzania Bazą Danych (SZBD). Na wejście koordynatora podawany jest strumień wejściowy, złożony z żądań odczytu i zapisu zgłaszanych przez procesy realizujące transakcje zapytania i aktualizacji. Zadaniem koordynatora jest wygenerowanie strumienia wyjściowego zapewniającego szeregowalność typu PSV, a dla systemów, w których dopuszcza się arbitralne transakcje aktualizacji i związane z nimi transakcje martwe — szeregowalność typu PSV lub SFS. Dokładniejsze omówienie powyższych pojęć podano w pierwszej części artykułu. Druga część poświęcona jest przeglądowi mechanizmów będących podstawą działania koordynatorów.**

Przy omawianiu poszczególnych metod założono, że dane zawarte w bazie są identyfikowane przez wskazanie ich unikalnej nazwy. Przypadek asocjacyjnego adresowania danych omówiono odrębnie.

## BLOKOWANIE — TRANSAKCJE DWUFAZOWE

Najszerzej stosowanym mechanizmem sterowania współbieżną aktualizacją bazy danych jest mechanizm zwany blokowaniem. Idea blokowania polega na zapewnieniu pojedynczemu procesorowi wyłączności w dostępie do wskazanej danej. Wyłączność ta jest zapewniona od momentu zrealizowania wydanego przez ten proces żądania blokowania danej — aż do momentu odblokowania tej danej, żądanego przez proces. Jeżeli żąda on zablokowania danej aktualnie zablokowanej przez inny proces, to musi ona czekać. Jedną z metod realizacji mechanizmu blokowania może być związanie z każdą daną semafora binarnego Dijkstry. Można też realizować blokowanie również w inny sposób — tak, aby nie wstrzymywać procesu żądającego blokady, jeżeli realizacja tej blokady nie jest w danym momencie możliwa [7].

Transakcja zapytania odczytująca tylko pojedynczą daną nie wymaga żadnej ochrony, przy założeniu niepodzielności akcji odczytu i akcji zapisu danej. Podobnie jest z arbitralnymi transakcjami aktualizacji dotyczącymi tylko jednej danej. Natomiast normalne transakcje aktualizacji, które czytają i piszą jedną daną, muszą już wykorzystywać mechanizm blokowania, tworząc sekwencję operacji:

**Blokuj(x);Czytaj(x);Pisz(x);Odblokuj(x);**

Dla wszystkich transakcji wykorzystujących więcej niż jedną daną, sposób użycia mechanizmu blokowania nie jest już tak trywialny. W [3] udowodniono, że warunkiem wystarczającym dla zapewnienia szeregowalności jest stosowanie transakcji dwufazowych (ang. 2-phase). Transakcje dwufazowe uzyskuje się, jeżeli:

- proces może żądać dostępu jedynie do danej aktualnie przez niego zablokowanej
- przed zakończeniem transakcji proces musi odblokować wszystkie zablokowane przez siebie dane
- po zrealizowaniu pierwszego odblokowania w danej transakcji, procesowi nie wolno dokonać żadnego blokowania.

W transakcji dwufazowej wyróżniamy dwie fazy:
— **fazę wzrostu**, w czasie której proces blokuje kolejne dane, nie dokonując żadnego odblokowania
— **fazę kurczenia**, w której proces kolejno odblokowuje dane zablokowane w pierwszej fazie transakcji.

Transakcje dwufazowe mogą spowodować wystąpienie zakleszczenia. Jedynym wyjściem jest wtedy wywłaszczenie jednego z procesów z zablokowanych przez niego danych. Problem polega na tym, że dane te mogły już zostać uaktualnione przez wywłaszczany proces, a więc może zajść konieczność odtworzenia ich starych wartości. Takie odtworzenie jest oczywiście możliwe za pomocą modułu odtwarzania bazy danych, ale wnosi dodatkowe opóźnienie.

W [7] zaproponowano więc stosowanie transakcji bezpiecznych. Są to transakcje dwufazowe, które w fazie

wzrostu nie zawierają akcji zapisu. Wycofanie się z zakleszczenia nie wymaga wtedy odtwarzania starych wartości danych, gdyż żadna dana nie uległa jeszcze aktualizacji. Zapisów dokonuje się w fazie kurczenia transakcji, w której nie ma już niebezpieczeństwa wystąpienia zakleszczenia. Wadą tego rozwiązania jest fakt, że teoretycznie w pewnych przypadkach może zachodzić konieczność dwukrotnego czytania tych samych danych.

Jeżeli proces podejmuje decyzję o tym, które dane należy — na podstawie analizy uprzednio odczytanych danych — zablokować, wtedy w fazie wzrostu transakcji bezpiecznych, oprócz blokowania, konieczne jest dokonywanie odczytów. Ponieważ w buforach pamięci operacyjnej nie można przechowywać zbyt dużej liczby danych, może zachodzić konieczność zamazywania wczytanych danych przez dalsze odczyty. Jeżeli w drugiej fazie transakcji bezpiecznej zamazane dane mają być zaktualizowane, konieczny staje się ich powtórny odczyt z bazy danych.

Omówiony już mechanizm blokowania, zwany czasem **blokowaniem dla zapisu** (ang. exclusive locking), można uzupełnić o **blokowanie dla odczytu** (ang. shared locking) lub inaczej **półblokowanie.** Dana półzablokowana przez jeden proces może być półblokowana przez inne procesy. Pozwala to na pełną współbieżność procesów, które tylko odczytują pewne dane, przy jednoczesnym zabezpieczeniu ich przed procesami aktualizacyjnymi, które stosują blokowanie. Półzablokowana jednostka nie może bowiem zostać zablokowana przez proces aktualizujący.

W jednej transakcji można stosować razem blokowanie i półblokowanie, przy czym należy przestrzegać dwufazowości, jeżeli w systemie chce się utrzymać szeregowalność. W [5] wprowadzono szereg dalszych rodzajów blokowania, jak półblokowanie z równoczesnym zapewnieniem sobie prawa do późniejszej zmiany półblokady na pełną blokadę, itd. Jeszcze inne typy blokad wprowadzono w [4] — w celu umożliwienia blokowania grup danych o różnej wielkości (granulacji), co zmniejsza narzut w przypadku blokowania dużej liczby danych.

Jak już wspomniano, transakcje dwufazowe stanowią warunek wystarczający szeregowalności. Nie są one jednak warunkiem koniecznym szeregowalności. Przykładowo — transakcje dwufazowe uniemożliwiają pojawienie się na wyjściu koordynatora strumienia:

$$\begin{array}{lll} T_1: & O(x)\ Z(x) & \qquad O(y)\ Z(y) \\ T_2: & \qquad O(x)\ Z(x) & \qquad O(y)\ Z(y) \end{array} \qquad (3)$$

który jest szeregowalny (jest on równoważny strumieniowi szeregowemu $T_1 T_2$).

## GRAF ZALEŻNOŚCI TRANSAKCJI

Mechanizm ten jest jedną z metod pozwalających na wyższy stopień zrównoleglenia dostępu do bazy danych, niż ma to miejsce w przypadku transakcji dwufazowych. Polega on na wykrywaniu pętli w grafie wyrażającym dynamiczną zależność pomiędzy transakcjami wykorzystującymi wspólną bazę danych. Transakcja $T_j$ staje się zależna od transakcji $T_i$, gdy odczytuje lub zapisuje daną zapisaną przez transakcje $T_i$, a także gdy zapisuje daną odczytaną przez $T_i$. Innymi słowy — zależność nie powstaje jedynie wtedy, gdy obie transakcje tylko odczytują wspólną daną. W każdym innym przypadku transakcja $T_j$, żądająca dostępu do danej wykorzystywanej przedtem przez transakcję $T_i$, staje się od niej zależna, co zapisujemy w postaci $T_i \rightarrow T_j$. W [14] stwierdzono, że każdy strumień, dla którego graf zależności nie zawiera pętli, jest szeregowalny oraz że żaden strumień, którego graf zależności zawiera pętlę, nie jest szeregowalny. Przykładowo — graf dla strumienia szeregowalnego (3) ze względu na obydwie dane zawiera jedynie zależność $T_1 \rightarrow T_2$, a więc nie zawiera pętli, a strumień:

$$\begin{array}{lll} T_1: & O(x)\ Z(x) & \qquad O(y)\ Z(y) \\ T_2: & \qquad O(x)\ Z(x)\ O(y)\ Z(y) & \end{array} \qquad (4)$$

który nie jest szeregowalny — ma w odpowiadającym mu grafie pętlę $T_1 \rightarrow T_2$, $T_2 \rightarrow T_1$. Usuwanie z grafu transakcji zakończonych może nastąpić jedynie w przypadku, gdy nie ma ona poprzedników, bowiem — na przykład — zbyt szybkie usunięcie z grafu odpowiadającego strumieniowi (4) węzła $T_2$ (zaraz po zakończeniu transakcji $T_2$') sugerowałoby, że strumień (4) nie tworzy pętli, a więc jest szeregowalny.

Opisany mechanizm można wykorzystać do sterowania współbieżną aktualizacją bazy danych w ten sposób, że dla każdego nowego żądania dostępu do bazy sprawdza się, czy nie powoduje ono powstania pętli w grafie zależności procesów. Jeżeli wykryje się pętlę, trzeba wycofać jeden z procesów, podobnie jak w przypadku wystąpienia zakleszczenia w metodzie transakcji dwufazowych. Złożoność obliczeniowa koordynatora wykorzystującego tę metodę jest taka, jak złożoność koordynatora wykorzystującego mechanizm blokowania dwufazowego — wynosi $O(n^2)$, gdzie **n** jest liczbą aktualnie wykorzystywanych danych. Obie omówione metody korzystają jedynie z informacji syntaktycznej o transakcji i nie żądają, aby cała informacja była dana z góry. Wobec powyższego, metoda grafu zależności procesów, pozwalająca na wyższy stopień zrównoleglenia dostępu do bazy danych, powinna być oceniona jako lepsza. Niestety, z samej swej istoty metoda ta jest nierozerwalnie związana z efektem domina.

Podamy teraz przykład strumienia, którym wykażemy, iż twierdzenie podane w [14] mówiące, że **nieistnienie pętli w omawianym grafie jest warunkiem koniecznym szeregowalności** — jest błędne. Podany niżej strumień (5) posiada pętlę $T_2 \rightarrow T_3$, $T_3 \rightarrow T_2$ w odpowiadającym mu grafie zależności procesów, a mimo to jest szeregowalny — jest równoważny schematowi $T_3 T_1 T_2$:

$$\begin{array}{lll} T_1: & O(z)\ Z(x) & \\ T_2: & \qquad O(x) & \qquad Z(x) \\ T_3: & \qquad O(y)\ Z(x)\ Z(w) & \end{array} \qquad (5)$$

Jest to niewątpliwie fakt zaskakujący. Zauważmy, że jedynie transakcje $T_2$ i $T_3$ interferują ze sobą i że w przypadku, gdyby w strumieniu nie było transakcji $T_1$, strumień ten nie byłby szeregowalny, gdyż:

- sekwencja $T_2$ $T_3$ pozostawiłaby w bazie danych wartość **x** utworzoną przez transakcję $T_3$, podczas gdy w oryginalnej wersji strumienia (5) w bazie danych pozostaje w końcu wartość utworzona przez $T_2$
- sekwencja $T_3$ $T_2$ spowodowałaby odczytanie przez $T_2$ wartości **x** utworzonej przez $T_3$, podczas gdy w oryginalnej wersji strumienia (5) transakcja $T_2$ odczytuje wartość **x** nie zmienioną jeszcze przez $T_3$.

Dzięki jednak obecności w (5) transakcji $T_1$, możliwe staje się „wyciągnięcie" transakcji $T_3$ przed transakcją $T_2$, gdyż transakcja $T_1$ „zasłoni" przed nią efekty transakcji $T_3$. Zauważmy dalej, że takie zasłonięcie efektu działania transakcji $T_3$ jest dopuszczalne, gdyż jest ona transakcją martwą ze względu na daną **x**.

## GRAF SZEREGOWANIA TRANSAKCJI

W [10] podano metodę rozstrzygania, czy dany strumień jest szeregowalny — polegającą na kolejnym konstruowaniu dla badanego strumienia różnych możliwych grafów szeregowania oraz badaniu, czy zawierają one pętlę. Strumień jest szeregowany wtedy i tylko wtedy, gdy choćby jeden z możliwych grafów szeregowania nie zawiera pętli.

Algorytm tworzenia tych grafów jest następujący:

1. Na początku badanego strumienia dodaje się hipotetyczną transakcję A, która nie odczytuje żadnej danej, natomiast zapisuje wszystkie dane; zaś na końcu strumienia dodaje się hipotetyczną transakcję B, która odczytuje wszystkie dane i nie zapisuje żadnej danej.

2. Węzły, którymi są wszystkie transakcje występujące w strumieniu, łącznie z transakcjami A i B, łączy się krawędziami należącymi do dwóch zbiorów. Zbiory te zdefiniowane są następująco:

$$\textbf{zbiór\_krawędzi\_ czytania} = \{ \langle T_i, T_j \rangle \mid \exists x (T_i \text{—} x \rightarrow T_j) \}$$

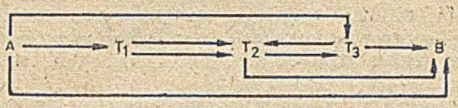
(Notacja $T_i$—x→$T_j$ oznacza, że w badanym strumieniu akcja odczytu danej **x** przez transakcję $T_j$ następuje po akcji zapisu tej danej przez transakcję $T_i$, przy czym pomiędzy tymi akcjami nie występuje żadna inna akcja zapisu **x**.)

$$\textbf{Zbiór\_ krawędzi\_ ingerowania} = \{ \langle T_k, T_i \rangle \vee \langle T_j, T_k \rangle \mid \exists x \times [(T_i \text{—} x \rightarrow T_j) \wedge (T_k : .. Z(x) ...)] \} - \{ \langle T_k, A \rangle \} - \{ \langle B, T_k \rangle \}$$

(Notacja $(T_k$:..Z(x)..) oznacza, że w transakcji $T_k$ występuje akcja zapisu danej x).

Jak widać z definicji **zbioru krawędzi ingerowania**, tran-

sakcję ingerującą można niejako ustawić przed lub po parze transakcji połączonych krawędzią ze zbioru krawędzi czytania, co w rezultacie może dać wiele różnych grafów dla jednego strumienia. Przykładowy graf szeregowania dla strumienia (5) przedstawia rysunek.




Jeden z możliwych grafów szeregowania dla strumienia [3]

Zbiór krawędzi czytania:

1. A,$T_1$, gdyż A—z→$T_1$
2. A,$T_3$, gdyż A—y→$T_3$
3. A, B, gdyż A—y→B
4. $T_1$,$T_2$, gdyż $T_1$—x→$T_2$
5. $T_2$,B, gdyż $T_2$—x→B
6. $T_3$,B, gdyż $T_3$—w→B

Zbiór krawędzi ingerowania:

1. $T_2$,$T_3$, gdyż ($T_1$—x→$T_2$) ∧ ($T_3$:..Z(x)..)
2. $T_3$,$T_2$, gdyż ($T_2$—x→B) ∧ ($T_3$:..Z(x)..)
3. $T_1$,$T_2$, gdyż ($T_2$—x→B) ∧ ($T_1$:..Z(x)..)

Zauważmy, że graf na rysunku zawiera pętlę $T_2 \rightarrow T_3$, $T_3 \rightarrow T_2$. Dla strumienia (5) można jednak zbudować inny graf, w którym zamiast krawędzi $T_2 \rightarrow T_3$ ze zbioru krawędzi ingerowania przyjmie się krawędź $T_3 \rightarrow T_1$, a wtedy powstający graf nie zawiera pętli, co świadczy o szeregowalności strumienia (5).

Przedstawiona tu metoda rozstrzygania o szeregowalności dowolnego strumienia, która teoretycznie zapewnia maksymalną współbieżność, ma niestety dużą złożoność obliczeniową (niewielomianową), co w praktyce uniemożliwia jej stosowanie. Ponadto metoda ta z samej swej istoty zakłada możliwość wystąpienia efektu domina.

## METODA ZNACZNIKÓW CZASOWYCH

Obie przedstawione tu metody grafowe wymagałyby — w przypadku systemu rozproszonego — posiadania w jednym węźle pełnej informacji związanej z przydziałem danych poszczególnym transakcjom. Podobnie w metodzie blokowania, wykrywanie zakleszczeń także wymaga scentralizowanej informacji (w [8] podano algorytm pozwalający wysyłać do węzła centralnego jedynie pewną zbiorczą informację — lokalne zakleszczenia wykrywane są lokalnie w danym węźle). Ponieważ centralizowanie informacji powoduje duży narzut w systemie rozproszonym i obniża jego niezawodność, powszechnie rozważa się stosowanie w systemach rozproszonych metody **znaczników czasowych** (ang. time stamps). Stanowią one unikalne identyfikatory transakcji w systemie. W [6] podano sposób tworzenia unikalnych znaczników czasowych w systemie rozproszonym. Polega on na konkatenowaniu lokalnego czasu z unikalnym w systemie — identyfikatorem węła generującego daną transakcję.

Podstawowy algorytm wykorzystujący znaczniki czasowe podano w [13]. Każda transakcja, wędrując od węzła do węzła, pozostawia przy każdej odczytanej i każdej zapisanej danej swój znacznik czasowy, przy czym — jeżeli napotyka na znacznik czasowy o większej wartości (czyli na ślad pozostawiony przez transakcję późniejszą), to jest wznawiana. Możliwe są różne odmiany tego algorytmu, a nawet łączenie go z ideą transakcji dwufazowych [1, 9]. Łatwo zauważyć, że metoda ta jest bardziej „ostrożna" od metody transakcji dwufazowych. Transakcja jest tu wznawiana, gdy tylko pojawi się możliwość nieszeregowalności, podczas gdy w transakcjach dwufazowych wznowienie następuje jedynie wtedy, gdy rzeczywiście zaistnieje nieszeregowalność, czego wskaźnikiem jest wystąpienie zakleszczenia.

## ADRESOWANIE ASOCJACYJNE A STEROWANIE WSPÓŁBIEŻNĄ AKTUALIZACJĄ BAZY DANYCH

Dotychczas rozważaliśmy model bazy danych, w którym pojedyncze dane adresowane są przez podanie ich unikalnych nazw. Obecnie rozważymy model bazy danych, w którym dane zgrupowane są w jednostki zwane **rekordami**. Adresowanie (selekcja) rekordów może odbywać się przez podanie ich unikalnych identyfikatorów, lecz często również określa się tylko warunki, jakie mają spełniać pojedyncze dane (pola) tych rekordów. Ten drugi sposób określa się mianem adresowania asocjacyjnego. Z punktu widzenia zarządzenia współbieżną aktualizacją bazy danych, konsekwencją adresowania asocjacyjnego są następujące fakty:

- aktualizacja istniejącego rekordu przez jeden z procesów może spowodować, że zrealizowane na rzecz innego procesu zlecenie selekcji staje się nieaktualne
- utworzenie przez jeden z procesów nowego rekordu powoduje, że informacja o nieistnieniu tego rekordu (pusty zbiór selekcji przeprowadzonej na żądanie innego procesu) staje się nieaktualna.

Fakty te określa się czasem mianem **problemu rekordów zjaw** (ang. phantom records).

W [3] zaproponowano takie oto rozwiązanie. Proces zgłasza predykat służący do selekcji z bazy danych interesujących go rekordów, które zamierza czytać i (lub) aktualizować. Dodatkowo, proces zgłasza predykat wyznaczający podzbiór bazy danych, do którego będą należały rekordy aktualizowane przez niego, po dokonaniu aktualizacji. W końcu zgłasza on również predykat obejmujący rekordy, które zamierza utworzyć. Wszystkie te predykaty zgłaszane są równocześnie i wszystkie zapamiętywane są przez oprogramowanie sterujące współbieżną aktualizacją bazy danych. Przedtem jednak sprawdza się, czy nie kolidują one z predykatami zgłoszonymi już wcześniej przez inne procesy. W przypadku kolizji proces jest wstrzymywany.

Jest to więc rodzaj blokowania zasobów przez transakcje dwufazowe, gdyż żadne zwalnianie przydzielonych zasobów nie występuje przed uzyskaniem wszystkich zasobów. Ze względu na charakter tego blokowania, rozwiązanie to spełnia swoją rolę w środowisku adresowania asocjacyjnego. Główną wadą tej metody jest fakt, że nawet w przypadku gdy dwa predykaty określone są na tych samych polach rekordu, nie da się rozstrzygnąć o ich ewentualnej kolizji, jeżeli w tych predykatach występują choćby cztery podstawowe działania arytmetyczne operujące na wartościach tych pól. Ponadto, każde dwa predykaty dotyczące różnych pól także muszą być, na wszelki wypadek, uznane za kolidujące. Przykładowo — w bazie danych mogą nie istnieć rekordy spełniające oba następujące predykaty: „pracownicy zarabiający ponad 20 tys. zł" i „pracownicy posiadający tylko jedno dziecko", niemniej predykaty te w opisanej metodzie będą uznane za kolidujące. W rezultacie narzut związany z analizą i przechowywaniem predykatów szybko może przekroczyć niewielki zysk, związany z dopuszczalną w tej metodzie współbieżnością.

Próbę uniknięcia przedstawionych wad tej metody opisano w [2]. Każdy realizujący transakcję proces może zgłaszać tylko jedno żądanie selekcji rekordów, które zostaną wybrane i zablokowane na rzecz tej transakcji. Wszystkie te rekordy zostają odblokowane równocześnie na końcu transakcji. Dla każdej transakcji system uruchamia osobny proces przeglądający bazę danych i blokujący tylko rekordy spełniające podany predykat selekcji. Ewentualne zakleszczenia, występujące w czasie pracy tych procesów, są łatwo usuwalne, gdyż zablokowane rekordy nie uległy jeszcze aktualizacji (jest to rodzaj transakcji bezpiecznych w sensie używanym w [7]).

Każdy wewnętrzny proces przeglądający bazę danych reaguje ponadto na każdą zakończoną transakcję, sprawdzając, czy zaktualizowane przez nią rekordy nie spełniają obecnie interesującego go predykatu. W ten sposób w momencie zakończenia przeglądania bazy danych na rzecz jakiegoś procesu gwarantuje się, że zobaczy on wszystkie albo nie zobaczy żadnej aktualizacji przeprowadzonej przez dowolny inny proces użytkowy. Nie jest to jednak wystarczający warunek szeregowalności procesów współbieżnych. Rozważmy dwa procesy, których rezultaty działania wzajemnie na siebie wpływają. Oba procesy żądają wyselekcjonowania z bazy danych rekordu spełniającego ten sam predykat i oba uzyskują odpowiedź, że rekord taki nie istnieje. Wtedy obydwa podejmują decyzję utworzenia takiego rekordu. Jak widać, konieczne jest, aby tylko jeden proces użytkowy w systemie mógł znaleźć się w danym momencie w fazie transakcji następującej po zakończeniu przeglądania bazy danych przez wewnętrzny proces SZBD (który na rzecz owego procesu użytkowego prze-

gląda bazę danych). Jest to oczywiście żądanie bardzo kosztowne, gdyż w systemie mogą istnieć procesy, dla których czas analizy i aktualizacji wyselekcjonowanych danych jest długi. Na problem ten zwrócono uwagę w [14].

Adresowanie asocjacyjne jest podstawowym sposobem adresowania danych zawartych w bazie danych. Pozwala ono nie tylko znaleźć rekord o nieznanym identyfikatorze, przez podanie wartości niektórych jego pól, ale pozwala również łatwo tworzyć transakcje dotyczące dużej liczby rekordów. Przykładem takiej transakcji może być zwiększenie stawki godzinowej wszystkim pracownikom, którzy spełniają podany warunek. Blokowanie dużej liczby pojedynczych rekordów związane jest z dużym narzutem. Narzut ten można zmniejszyć poprzez umożliwienie blokowania całego pliku. Z drugiej strony — przyjęcie całego pliku jako jednostkę blokowania spowoduje, że stopień zrównoleglenia działań w systemie zmaleje. Próby znalezienia za pomocą symulacji — optymalnej granulacji jednostek blokowania opisano w [11, 12]. Jeszcze lepszym rozwiązaniem jest wprowadzenie w jednym systemie możliwości blokowania grup danych o różnej wielkości. Dokonanie blokady na poziomie pliku jest jednak poprawne tylko w przypadku, gdy żaden rekord tego pliku nie jest aktualnie zablokowany. Stąd wprowadzenie możliwości blokady grup danych o różnej wielkości wymaga wprowadzenia dodatkowych typów blokowania, które można nazwać **blokowaniem ostrzegawczym.** Przykładowo — proces zamierzający zablokować rekord musi najpierw dokonać blokowania ostrzegawczego na poziomie pliku, aby żaden proces nie mógł dokonać blokowania całego pliku. Problem kompatybilności różnych typów blokad w systemie dopuszczającym różnorodną wielkość blokowania grup danych rozważono w [4].

* * *

Obecnie w oprogramowaniu zarządzającym współbieżną aktualizacją scentralizowanych baz danych prawdopodobnie najczęściej stosowanym mechanizmem są transakcje dwufazowe, przy czym jako jednostkę blokowania przyjmuje się cały plik lub przynajmniej stronę. W rozproszonych bazach danych stosuje się natomiast odmiany algorytmów wykorzystujących ideę znaczników czasowych. W literaturze pojawiają się też kolejne algorytmy, mogące znaleźć zastosowanie w bazach danych o szczególnych strukturach. Nowych rozwiązań można oczekiwać w związku z procesorami baz danych, a także — z nowymi technologiami pamięci masowych, umożliwiających szersze stosowanie baz, w których zachowuje się stare wartości aktualizowanych danych.

LITERATURA


[1] Bernstein P. A., Goodman N.: Ming-Yee Liai: Analizing concurrency control algorithms when user and system operations differ. IEEE Trans. on Soft. Eng., Vol. SE-9. No. 3, May 1983, p. 233—139

[2] Chamberlin D. D., Boyce R. F., Traiger I. L.: A deadlock-free scheme for resource locking in a data-base environment. Inform. Proc. 74, North-Holland Publ. Co. 1974, p. 340—343

[3] Eswarn K. P., Gray J. N., Lorie L. A., Traiger I. L.: The notions of consistency and predicate locks in a database system. Comm. ACM, Vol. 19, No. 11, November 1976, p. 624—633

[4] Gray J. N., Lorie R. A., Putzolu G. R., Traiger I. L.: Granularity of locks and degress of consistency in a shared data base. Modeling in Data Base Systems, G. M. Nijssen (ed), North Holland Publ. Co., 1976

[5] Korth H. F.: Locking primitives in a database system. Journal of the ACM 30 (1), 1983, p. 55—79

[6] Lamport L.: Time clocks and the ordering of events in a distributed system. Comm. ACM, Vol. 21, 1978, p. 558—565

[7] Małuszyński J. T.: Sterowanie współbieżną aktualizacją bazy danych. Praca doktorska, Politechnika Poznańska, 1980

[8] Menasce D. A., Muntz R. P.: Locking and deadlock detection in distributed data base. IEEE Trans. on Soft. Eng., Vol. SE-5, No. 3, May 1979, p. 195—202

[9] Morzy T.: Algorytmy synchronizacji w systemach rozproszonych baz danych. Praca doktorska. Politechnika Poznańska, 1983

[10] Papadimitriou C. H., Bernstein P. A., Rotjnie J. B.: Some computational problems related to database concurrency control. Proc. Conf. Theoretical Computer Science, Waterloo, August 1977

[11] Ries D. R., Stonebraker M.: Effects of locking granularity in a database management system. ACM Trans. on DBS, Vol. 2, No. 3, September 1977, p. 233—246

[12] Ries D. R., Stonebraker M.: Locking granularity revisited. ACM Trans. on DBS, Vol. 4, No. 2, June 1979, p. 210—227

[13] Rosenkrantz D. J., Sterns R. E., Lewis P. M.: System level concurrency control for distributed database systems. ACM Trans. on DBS, Vol. 3, No. 2, June 1978, p. 178—198

[14] Schlageter G.: Process synchronization in database systems. ACM Trans. on DBS, Vol. 3, No. 3, September 1978, p. 248—271.


# Do redakcji

Pragnę zwrócić uwagę na liczne błędy w artykule „FORTH-ASSEMBLER dla mikroprocesora INTEL-8080" w INFORMATYCE nr 8/84 (patrz tabela obok).

W piśmie BYTE słowa FORTHA zawierające znaki interpunkcyjne oraz wszystkie ciągi słów FORTHA, wymieniane w tekście jakiegoś artykułu, wyróżnia się nawiasami klamrowymi, np.:

„... słowo {C,}, które..."

„... za pomocą {SWAP DROP} ..."

Jest to praktyka godna naśladowania.

Ad. * wykonanie słowa, którego nazwa jest mnemoniczną nazwą instrukcji asemblerowej (np. słowa NOP) nie powoduje wcale kompilacji tego słowa (w słowniku nie zostanie umieszczony adres pola GF słowa NOP, lecz kod rozkazu maszynowego NOP — czyli 00).

Ad. ** Tekst źródłowy asemblera FORTHA, mimo iż sprawia solidne wrażenie (wydruk komputerowy), to jednak zawiera błędy przepisywania.

Proszę także zwrócić uwagę na to, że definicja:
: THEN [COMPILE] ENDIF ;

jest poprawna, jednakże słowo [COMPILE] jest tu niepotrzebne, gdyż słowo ENDIF ze słownika ASSEMBLER jest i bez tego kompilowane podczas kompilacji (nie jest ono określone jako IMMEDIATE).

I jeszcze jedno — nagłówki nad wydrukami poszczególnych kadrów powinny chyba być SCR#... (skrót od screen), a nie SRC # 100 itp.

**JACEK NOWAK**
Katowice

| Miejsce | Jest | Powinno być |
|---|---|---|
| str. 4 (lewa kolumna) wiersz: 7 | W słowniku ASSEMBLER istnieje pełny zbiór słów będących mnemonicznymi instrukcjami języka asemblera. Wywołanie słowa powoduje jego kompilację... | * (...) Wywołanie słowa powoduje kompilację rozkazu maszynowego... |
| 17 | HL | HLT |
| 18 | C u C | Cu C |
| 24 | słowo C u | słowo Cu |
| rysunek część c | (; CODE) | (;CODE) |
| str. 5 (lewa kolumna) wiersz: 10 | SO | S O < |
| 15 | CODE BEGIN<br>BEGIN<br>C DCR<br>O = UNTIL<br>B DCR<br>O = UNTIL C; | CODE PETLA<br>BEGIN<br>BEGIN<br>C DCR<br>O = UNTIL<br>B DCR<br>O = UNTIL<br>NEXT JMP C; |
| 32 | Adres zawarty w polu słowa... | Adres zawarty w polu CF słowa... |
| wydruk, kadr 100, wiersz: 1 | OF 1MI RC | ** OF 1MI RRC |
| 10 | FF 4MI CPI | FE 4MI CPI |

ADAM STAWOWY (oprac.)
Kraków

# MACINTOSH
# — nowa jakość na rynku mikrokomputerów

Pierwotna koncepcja MACINTOSHA powstała w 1979 roku. Miał to być przenośny mikrokomputer kompatybilny z APPLE II, oparty na 8-bitowym mikroprocesorze 6809, z małym monitorem, pamięcią na 5¼-calowych dyskach elastycznych i standardową klawiaturą. W następnym roku rozpoczęto prace nad oprogramowaniem systemu. Jednakże w 1981 roku całkowicie zmieniono koncepcję MACINTOSHA. Wykorzystując doświadczenia zdobyte podczas projektowania LISY, postanowiono opracować mikrokomputer tej samej klasy, lecz dużo tańszy i łatwiejszy w obsłudze. Wydzielono specjalny zespół, który z początkiem 1981 roku rozpoczął opracowywanie systemu na bazie mikroprocesora MOTOROLA 68000, z niekonwencjonalną grafiką oraz — myszką. W czerwcu 1981 rozszerzono pamięć RAM z 64 do 128KB. Była to wersja prawie identyczna z tą, która pojawiła się na rynku, tyle tylko że w styczniu 1983 zmieniono dyskietkę 5¼-calową na minidyskietkę 3½-calową.

W roku 1948 firma APPLE przygotowała się do sprzedaży 0,5 mln egzemplarzy MACINTOSHA. W tym celu wybudowała we Fremont (stan Kalifornia USA) specjalny, w pełni zautomatyzowany zakład, w którym proces produkcji obsługiwało początkowo zaledwie 80 pracowników (w końcu 1984 — 540).

Sądzi się, że 70% tych systemów będzie przeznaczonych do prac biurowo-administracyjnych, 20% znajdzie zastosowanie edukacyjne, a pozostałe 10% — jako komputery domowe.

## OPIS SPRZĘTU

Pierwszy syntetyczny opis MACINTOSHA ukazał się na łamach INFORMATYKI w nr. 9/1984. Poniżej, rozszerzę nieco podane wówczas informacje, powtarzając najważniejsze dane techniczne. Krótka charakterystyka MACINTOSHA jest następująca:

— **procesor centralny** 16-bitowy, MC 68000 o częstotliwości zegara 8 MHz

— **pamięć** o pojemności 128 KB RAM i 64 KB ROM (mieszcząca system operacyjny i programy obsługi sprzęgów)

— **monitor** 9-calowy, monochromatyczny, o rozdzielczości 512×342 punkty, z wbudowaną pojedynczą stacją minidyskietek o pojemności 400 KB

— **klawiatura** 58-przyciskowa (kod ASCII) z myszką, jako dodatkowym urządzeniem wejściowym

— **port szeregowy** dla modemu i drukarki oraz czterokanałowy **generator dźwięku**

— możliwość dołączenia **dodatkowej stacji minidyskietek.**

Doświadczenia zdobyte w projektowaniu układów VLSI pozwoliły zredukować liczbę elementów użytych w konstrukcji minikomputera. Monitor jest monochromatyczny, gdyż — jak stwierdzono — „dobry kolor kosztuje trzy razy więcej i wymaga trzy razy większej pamięci". Ciekawą innowacją jest użycie tylko dwóch płytek, jednej — dla funkcji analogowych, drugiej — dla cyfrowych. Klawiatura MACINTOSHA ma standardowy układ QWERTY i pozwala na dołączenie dodatkowego bloku klawiatury numerycznej. Czterokanałowy generator dźwięku może pracować (przy odpowiednim oprogramowaniu) jako syntetyzator mowy.

Myszka (o wielkości talii kart), służąca do sterowania ruchem kursora, ma tylko jeden przycisk do dokonywania wyboru z listy poleceń, gdyż — jak wykazuje praktyka — im więcej przycisków, tym trudniej posługiwać się nimi prawidłowo. Kursor porusza się na ekranie zgodnie z ruchem myszki po biurku.

MACINTOSH jest pierwszym wyrobem APPLE, w którym firma nie montuje swej własnej stacji dyskietek. Dostarcza je firma SONY, która wylansowała minidyskietki 3½-calowe. Współpraca obu firm zaowocowała nową konstrukcją, z reguły nie kompatybilną z innymi, spotykanymi na rynku. Minidyskietka jest cały czas pod kontrolą systemu operacyjnego, który uniemożliwia jej ręczną wymianę w nieodpowiednim momencie, eliminując w ten sposób błędny zapis danych. Jednostronna minidyskietka ma pojemność 400 KB, a zapis i odczyt informacji jest bardzo szybki, prawie taki jak dla dyskietek 8-calowych. Zewnętrzna stacja minidyskietek ma dodatkowe 400 KB pojemności, przy czym planuje się użycie w MACINTOSHU dwustronnej minidyskietki o pojemności 800 KB. Ponadto, firma TECMAR oferuje sztywne dyski o pojemności 5 MB oraz przeznaczone do MACINTOSHA pamięci dyskowe typu Winchester, o pojemności od 10 do 33 MB.

Podobnie jak w przypadku wymienionych elementów mikrokomputera, również cechy jego obudowy świadczą o dążeniu do obniżki kosztów produkcji i eksploatacji. Funkcje poszczególnych złączy z tyłu obudowy są opisane nie przy użyciu słów, lecz — symboli graficznych, np. złącze generatora dźwięku jest oznaczone symbolem nuty, a złącze modemu — rysunkiem telefonu. Pozwala to wykorzystywać tę samą obudowę w produkcji na różnojęzyczne rynki. Ponadto, tzw. chassis MACINTOSHA jest tłoczone z jednego kawałka blachy, co przy wydajności 20 szt./min kosztuje zaledwie ok. 2 dol.

## OPROGRAMOWANIE

Przed rokiem MACINTOSH miał jeszcze niewielką bibliotekę programów użytkowych — niestety, nie zgodnych z programami innych mikrokomputerów tej firmy (np. LISA), a tym bardziej innych wytwórców. Oprogramowanie dostarczane przez producenta zawiera pakiety: MacWrite (pakiet programów do przetwarzania tekstów), MacPaint (pakiet programów do rysowania), pakiet programów zarządzania bazą danych oraz — kompilator języka BASIC. Dostępne są również translatory języków LOGO i ASSEMBLER. Firma APPLE liczy jednak na to, że zdecydowana większość programów użytkowych zostanie opracowanych przez niezależnych wytwórców oprogramowania i na rynku pojawi się wieleset propozycji. Rzeczywiście, czołowi dostawcy oprogramowania, jak MICROSOFT, LOTUS DEVELOPMENT czy SOFTWARE PUBLISHING — pracują intensywnie nad oprogramowaniem nowego mikrokomputera, a np. MULTIPLAN firmy MICROSOFT jest już od dawna dostępny. By ułatwić te prace, firma APPLE dostarcza producentom oprogramowania dokumentację techniczną (tzw. Inside MACINTOSH) z opisem systemu operacyjnego, który — nawiasem mówiąc — nie jest zgodny z innymi systemami spotykanymi na rynku.

MACINTOSH jest mikrokomputerem ukierunkowanym graficznie. Znaczy to, że wszystkie zastosowania i wszystkie zbiory (zwane dokumentami) są reprezentowane przez obrazki-symbole (ang. icons). Ponadto, punktowa grafika MACINTOSHA (ang. bit mapping display) jest nadzwyczaj dokładna, a programista ma dostęp do każdego punktu ekranu.

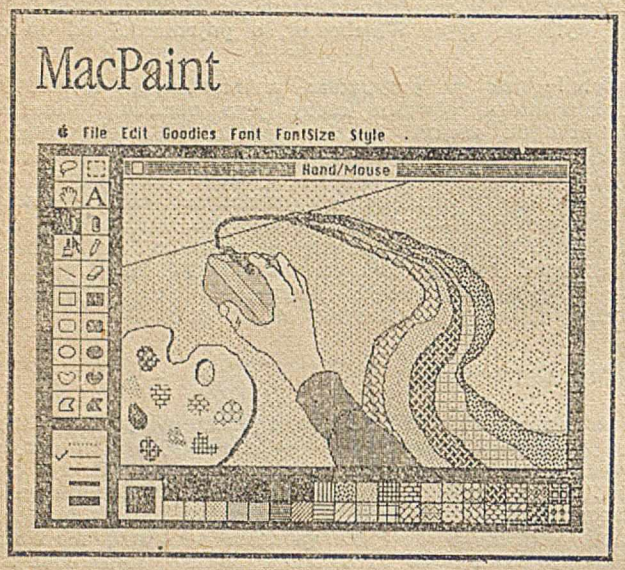
Istotnym elementem oprogramowania MACINTOSHA są tzw. okienka (ang. windows), których główną funkcją jest wyświetlanie aktualnej zawartości zbiorów. Ekran może być podzielony na kilkadziesiąt okienek — każde z inną zawartością, jednak gdy liczba okienek przekracza 4, ekran jest mało czytelny. Z tego względu wygodniej jest utworzyć stos okienek i operować ostatnim z nich. Należy wyraźnie podkreślić, że MACINTOSH nie ma cechy wielo-

programowości (jak np. LISA) i okienka nie służą do wyświetlania przebiegu różnych programów jednocześnie. Okienka mogą natomiast wyświetlać różne dokumenty (zbiory) bądź różnego rodzaju informacje o dokumencie. Ułatwia to wymianę informacji między zbiorami oraz przetwarzanie tekstów. Istnieje również możliwość wyświetlania w jednym okienku przebiegu programu, a w drugim — jego tekstu.

## DZIAŁANIE PROGRAMÓW MACPAINT I MACWRITE

W oprogramowaniu podstawowym najciekawsze są programy MacWrite i MacPaint. Ten pierwszy jest pakietem programów przeznaczonych do przetwarzania tekstów, jednak nie dłuższych niż 10 stron druku (zbiory o większych rozmiarach trzeba dzielić na rozdziały i zapamiętywać w oddzielnych plikach na dyskietce). MacPaint pozwala tworzyć bardzo dokładne rysunki wypełniające stronę o wymiarach $8^1/_2 \times 11$ cali, której połowa jest wyświetlana na ekranie.

Rodzaj czynności w pakiecie MacPaint jest wybierany przez użytkownika z listy zawartej u góry ekranu i z serii symboli. Przesunięcie kursora — ruchem myszki po biurku — wzdłuż listy do pożądanego tytułu i wciśnięcie guzika myszki powoduje wyświetlenie okienka zawierającego dostępne opcje (tzw. menu). Przesuwając kursor do pożądanej opcji, przy wciśniętym guziku, a następnie zwalniając go — uzyskuje się dostęp do jej treści.




Fot. 1. Obraz Hand/Mouse [1]

Na zdjęciu 1 przedstawiono obraz zatytułowany HAND/ /MOUSE. Wokół ekranu widnieją środki ułatwiające rysowanie (opcja „Introduction" wybrana z listy „Goodies"). Funkcje części z nich są oczywiste, np. ołówek służy do rysowania cienkich linii, pędzel — szerokich, zaś litera A pozwala w rysunki wpisywać tekst. Lasso, dłoń i prostokąt służą do przesuwania rysunków po ekranie, a puszka farby i spray — pozwalają wypełnić dowolny obszar wybranym wzorem. Użytkownik ma do dyspozycji 38 różnych wzorów (przedstawionych u dołu ekranu), a wybrany przez niego wzór jest wskazywany w pierwszym okienku. Szerokość linii i obramowań można wybrać z okienka pojawiającego się w lewym dolnym rogu ekranu — symbole nie wymagają wyjaśnień.

Poniżej omówiono krótko funkcje elementów listy przedstawionej na zdjęciu 1, wraz z ich dostępnymi opcjami:
**rysunek jabłka**, siedem opcji, oznaczonych symbolem firmy APPLE, odpowiada standardowym zastosowaniom MACINTOSHA, np. jako kalkulatora, notatnika czy zegara

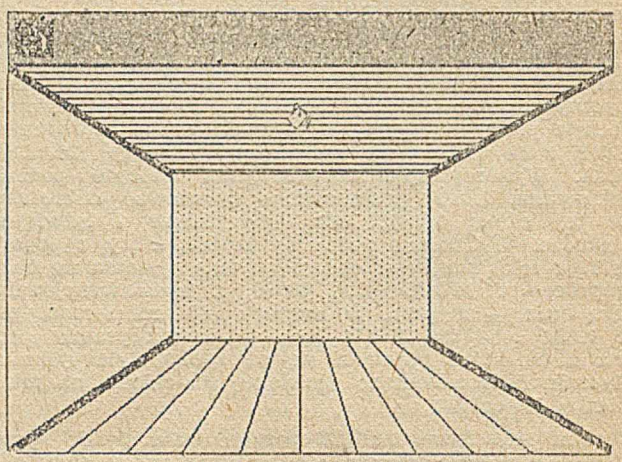
**File** — dziewięć opcji służących do manipulowania zbiorami, jak np. zamykanie, otwieranie i drukowanie dokumentów

**Edit** — dziesięć opcji umożliwiających — na przykład — powrót do stanu przed wykonaniem ostatniego polecenia (opcja UNDO) oraz dokonywanie zmian w już przetworzonym obrazie graficznym

**Goodies** — oprócz opcji „Introduction", którą omówiono powyżej, lista zawiera jeszcze siedem opcji pozwalających tworzyć skomplikowane rysunki (np. tworzenie lustrzanego odbicia lub niestandardowych wzorów)

**Font** — umożliwia pisanie piętnastoma rodzajami czcionek
**Style** — pozwala wybrać jeden spośród sześciu wzorów pisma (dla każdego rodzaju czcionki) oraz pisać tekst prosty bądź pochyły w lewo lub w prawo
**FontSize** — umożliwia wybór dziewięciu różnych rozmiarów czcionki.

Poniżej przedstawiono prosty przykład korzystania z pakietu MacPaint do rysowania wnętrza pomieszczenia (fot. 2):



Fot. 2. Rysowanie pomieszczenia [1]

1. Rysowanie ścian.

Przy użyciu myszki wybiera się symbol wypełnionego prostokąta, szerokość obramowania oraz wymagany wzór, po czym tworzy się ścianę wytyczając przekątną prostokąta z lewego, górnego wierzchołka (w podobny sposób rysuje się też inne figury, np. koła).

2. Wyznaczanie sufitu, podłogi i pozostałych ścian.
Po wyborze symbolu linii należy je wykreślić. Linie zaczynają się w położeniu kursora, w którym wciska się „guzik" myszki, a kończą, gdy po przesunięciu kursora w ostateczne położenie guzik zostanie wyciśnięty.

3. Malowanie sufitu.

Po wyborze wzoru przesuwa się symbol puszki na obraz sufitu i wciska guzik myszki. W ten sposób można wypełnić wzorem dowolny obszar.

W podobny sposób działa pakiet MacWrite. Lista funkcji jest zbliżona do listy MacPaint, aczkolwiek istnieją drobne różnice:

- **File, Edit i Font** zawierają mniej opcji niż ich odpowiedniki w MacPaint
- **Style** jest połączeniem funkcji Style i FontSize z MacPaint
- **Format** jest odpowiednikiem Goodies i służy do wybierania układu tekstu (przy użyciu symboli widocznych wokół ekranu)
- **Search** zawiera opcje wyszukiwania i wymiany wybranych tekstów — nie ma swojego odpowiednika w MacPaint.

W linii pod listą pojawia się nazwa przetwarzanego dokumentu (fot. 3, „Unititled"), a znaczenie poszczególnych symboli jest następujące:
— cyfry — szerokość tekstu w calach
— czarna strzałka — szerokość akapitu
— czarne trójkąciki — lewy i prawy margines
— trójkącik i trójkącik z punktem — standardowa i dziesiętna tabulacja
— trzy prostokąty (w środku) — odstępy między wierszami (pojedynczy, półtora odstępu i podwójny)
— cztery prostokąty z prawej strony (jeden niewidoczny) służą do adiustowania tekstu

dokończenie na str. 21

**Komputer osobisty IBM-PC zaczyna robić karierę i w naszym kraju. Kilkanaście egzemplarzy tego drogiego, jak na polską kieszeń, sprzętu pracuje już u osób prywatnych. Równocześnie kilka firm podjęło montaż komputerów kompatybilnych z IBM-PC i choć trudno mówić o widokach na szerokie upowszechnienie (jak choćby ZX SPECTRUM), to początek został już zrobiony. Mimo wygórowanych cen zestawu 3,5—7 mln zł), co daje przelicznik ok. 3 tys. zł za dolara — chętnych na „złotówkowego" IBM nie brakuje.**

**Zamieszczając w mikroKLANIE 13 tekst o normie GKS, zapowiedzieliśmy dalsze publikacje na ten temat. Obecna publikacja, omawiająca uzależnienia sprzętowe i podająca przykładowe procedury graficzne, przygotowuje grunt pod następne artykuły, w których przedstawimy sposób realizacji jądra systemu graficznego zgodnego ze standardem GKS.**

**Ze względu na objętość tekstu publikujemy go w dwóch częściach: w pierwszej — opis komputera i przykładowe procedury w języku maszynowym, w drugiej — bibliotekę procedur zapisanych w języku C.**

## Grafika IBM-PC (1)

IBM-PC ma bogate możliwości graficzne. Na monitorze ekranowym można zarówno wyświetlać napisy (tryb alfanumeryczny), jak i rysować (tryb graficzny). Wydruki mogą być monochromatyczne (jasne litery na ciemnym tle) lub kolorowe. Ekran dla trybu alfanumerycznego podzielony jest na 25 linii po 80 lub 40 znaków (do wyboru). W trybie graficznym komputer zapewnia rozdzielczość 640×200 (obrazy czarno-białe) i 320×200 (kolor). Tło może być wybrane spośród 16 kolorów, natomiast same rysunki mogą być wykonywane kolorami z dwóch palet. Na palecie 1 dostępne są trzy kolory: zielony, czerwony i żółty, natomiast na palecie 2 kolory: niebiesko-zielony, fioletowy i biały. Opisana koncepcja grafiki kolorowej wykorzystuje możliwie najmniejszy obszar dla pamięci obrazowej. Kolor każdego punktu na ekranie pamiętany jest w dwóch bitach:
**00** — kolor tła
**01** — kolor zielony (paleta 1) lub niebiesko-zielony (paleta 2)
**10** — kolor czerwony (paleta 1) lub fioletowy (paleta 2)
**11** — kolor żółty (paleta 1) lub biały (paleta 2).
W rezultacie cały obraz zajmuje 128 000 bitów (320×200×2), czyli — 16 000 bajtów.

```
        public  clear_image

clear_image :
        push    di
        push    es          ; ochrona rejestrów
        mov     ax,0b800h   ; adres pamięci obrazu
        mov     es,ax
        mov     di,0
        mov     cx,8000     ; ilość słów obrazu
        mov     ax,0        ; kolor tła
        cld
        rep     stosw       ; wpisywanie do pamięci
        pop     es
        pop     di
        ret
```

**Wydruk 1. Procedura clear_image w języku ASSEMBLER 8086**

Adres początku pamięci obrazu równy jest B0000H dla grafiki czarno-białej lub B8000H — dla kolorowej.

Na wydruku 1 przedstawiono procedurę „zamalowywania" obrazu w trybie grafiki kolorowej 320×200, począwszy od lewego górnego rogu ekranu.

W trybie alfanumerycznym, kody wyświetlanych znaków ASCII pamiętane są w obszarach zaczynających się od tych samych adresów, jak w przypadku trybu graficznego. Dla trybu alfanumerycznego kolorowego z każdym znakiem skojarzony jest bajt opisujący kolor znaku i kolor tła. W efekcie na cały ekran przypada 4000 bajtów (80×25×2) dla trybu 80-znakowego oraz 200 bajtów (40×25×2) dla trybu 40-znakowego.

**Bajt opisujący kolor znaku**

| Bajt | | | | | | | | Kolor: |
|---|---|---|---|---|---|---|---|---|
| M | T | T | T | I | Z | Z | Z | |
| | 0 | 0 | 0 | | 0 | 0 | 0 | czarny |
| | 0 | 0 | 1 | | 0 | 0 | 1 | niebieski |
| | 0 | 1 | 0 | | 0 | 1 | 0 | zielony |
| | 0 | 1 | 1 | | 0 | 1 | 1 | niebiesko-zielony |
| | 1 | 0 | 0 | | 1 | 0 | 0 | czerwony |
| | 1 | 0 | 1 | | 1 | 0 | 1 | fioletowy |
| | 1 | 1 | 0 | | 1 | 1 | 0 | żółty |
| | 1 | 1 | 1 | | 1 | 1 | 1 | biały |

**M — migotanie, T — kolor tła, I — intensywność, Z — kolor znaku**

```
        public  clear_screen

clear_screen :
        push    di
        push    es          ; ochrona rejestrów
        mov     ax,0b800h   ; adres pamięci ekranu
        mov     es,ax
        mov     di,0
        mov     cx,2000     ; ilość słów
        mov     ax,0720h    ; biały odstęp na czarnym tle
        cld
        rep     stosw       ; wpisywanie do pamięci
        pop     es
        pop     di
        ret
```

**Wydruk 2. Procedura clear_screen**

Wydruk 2 przedstawia procedurę wypełniania ekranu w trybie alfanumerycznym kolorowym 80×25. Uwaga: kody znaków pamiętane są pod adresami parzystymi, natomiast bajty atrybutów — pod adresami nieparzystymi.

Dysponując podanymi informacjami można już projektować oprogramowanie sterujące monitorem alfanumerycznym i graficznym. Dla uproszczenia programów można wykorzystać procedury zawarte w firmowej pamięci ROM.

Ich wywołanie odbywa się poprzez przerwanie: INT 10H, przy odpowiednio ustawionych wartościach rejestrów.

(cdn)

**ANTONI URBAN**
Computer Studio Kajkowscy
Gdynia

**Po wstępym okresie euforii, prawie każdy użytkownik ZX SPECTRUM ma szczerze dość wykorzystywania magnetofonu jako pamięci masowej. Przyłączenie dysków elastycznych jest jednak rozwiązaniem przewyższającym ceną sam komputer. Dlatego też wiele osób zastanawia się nad alternatywną propozycją firmy SINCLAIR. Sądzimy, że prezentowane artykuły wyjaśnią wiele wątpliwości.**

**Ponieważ otrzymaliśmy niemal równocześnie dwa teksty na temat ZX MICRODRIVE, zawierające częściowo te same informacje, lecz równocześnie znacznie się różniące, przyjęliśmy dość nietypowe rozwiązanie — chyba z korzyścią dla Czytelników. Z obydwu materiałów wydzieliliśmy podstawowe informacje o ZX MICRODRIVE (akapity wybrane z tekstu Jerzego Karczmarczuka zaznaczamy trójkątem). Jako uzupełnienie publikujemy dwugłos wyrażający opinie autorów o tym urządzeniu.**

## ZX MICRODRIVE

ZX MICRODRIVE to zewnętrzna pamięć masowa przeznaczona do współpracy z ZX SPECTRUM. Do zapisu danych wykorzystywane są specjalne mikrokasety (ang. cartridge) o pojemności ok. 85—90 KB każda. W kąsetkach znajduje się taśma magnetyczna połączona w pętlę. Długość taśmy wynosi 5 m, a szerokość 1/16 cala. Czas przewijania pełnej pętli — 7 s. Przewijanie odbywa się z tą samą prędkością co zapis i odczyt. ZX MICRODRIVE jest przyłączony do komputera za pośrednictwem wielozadaniowego interfejsu ZX INTERFACE 1, zawierającego system operacyjny zapisany w pamięci EPROM oraz układy obsługujące transmisję danych (MICRODRIVE, ZX NET, RS-232). Ceny (z końca 1984) MICRODRIVE — 50 funtów, ZX INTERFACE 1 — 50 funtów, mikrokaseta — 4,95 funta.

Zapis na kasetce jest dwuścieżkowy; poszczególne bity są zapisywane na przemian. Dane zgrupowane są w sektorach (512 bajtów + nagłówki) — na jednej kasetce mieści się do 200 sektorów, co oznacza gęstość zapisu ok. 500 bajtów na cal. Struktura zapisu jednego sektora jest następująca:

- blok nagłówka (1,25 ms)

— preambuła bloku 12 bajtów (10 bajtów 00H i 2 bajty 0FFH)

— nagłówek sektora 15 bajtów (numer sektora, nazwa kasetki, bajt sumy kontrolnej

- przerwa (3,75 ms)
- blok danych (25 ms)

— preambuła bloku 12 bajtów

— nagłówek rekordu 15 bajtów (nazwa pliku, numer rekordu, długość, bajt sumy kontrolnej

— dane 513 bajtów (dane, bajt sumy kontrolnej)

- przerwa (7 ms).

Dane zajmują więc ok. 2/3 długości taśmy.

Podczas formatowania w każdym sektorze zapisywany jest w bloku nagłówka (z nazwą kasetki i numerem sektora) oraz rekordy z danymi testowymi. Następnie program formatujący sprawdza, ile jest sektorów sprawnych. Sektory, w których wystąpił błąd odczytu zaznaczane są jako „zepsute" i nie są później używane. Stąd wynika różna pojemność mikrokasetek.

▲ Ponieważ formatowanie jest programowane, można skonstruować logicznie mikrokasetę z zupełnie innymi sektorami, o innej długości, bez podziału na dwa bloki, itp. Nawet można trochę skrócić preambułę oraz przerwy między sektorami, są to jednak bardzo ryzykowne operacje i w ich wyniku mikrokasetą może się nadawać do czytania tylko przez jedną fizycznie stację, a i to nie zawsze. Przerwy między blokami służą do pozostawienia procesowi czasu na wykonanie programu podejmującego decyzję co robić ze znalezionym sektorem. Przerwy są za krótkie, by móc przesłać 512 bajtów z (lub do) bufora i jeszcze „złapać" następny sektor. Tak więc dłuższy plik zapisywany jest w co drugim sektorze (jeśli nie musi przeskoczyć przez sektory już zajęte).

▲ Numery sektorów mają znaczenie jedynie pomocnicze. Przy otwieraniu pliku system konstruuje „mapę": tablicę bitów, w której zaznacza, które sektory są wolne. Tablica ta istnieje w pamięci dopóty, dopóki plik jest otwarty. Nigdy nie jest zapisywana na mikrokasecie (zawsze powstaje na bieżąco).

▲ Prędkość transmisji danych przekracza 2 KB/s. Jednak czas dostępu do pliku i czas jego ładowania może być bardzo długi — wskutek nieefektywności oprogramowania zarządzającego.

▲ MICRODRIVE jest przyłączony do ZX INTERFACE 1 następującymi przewodami: 6 przewodów masy, zasilanie 9 V, 2 linie dwukierunkowej szyny danych, 2 linie sterujące wyborem stacji oraz startem—stopem silnika, linia sterująca odczytem—zapisem, linia sterująca kasowaniem oraz linia kontrolna zabezpieczenia mikrokasety przed zapisem (zabezpiecza się przez wyłamanie ząbka w kasetce). Do INTERFACE 1 można jednocześnie przyłączyć szeregowo do ośmiu stacji. Sygnał selekcji stacji jest generowany programowo w pętli osiem razy: siedem razy „nie", raz „tak" (w odpowiedniej sekwencji) i przekazywany wzdłuż łańcuszka MICRODRIVE'ów. Przy programowaniu w języku maszynowym nie ma zabezpieczenia przed włączeniem kilku stacji lub włączeniem stacji bez włożonej kasetki.

▲ Linie sterujące i kontrolne są podłączone do portu o adresie EFH. Wykonując odczyt, program dowiaduje się czy kasetka jest zabezpieczona przed zapisem (można to zignorować!), dalej — jaki jest stan gotowości urządzenia, a także — czy został rozpoznany znacznik początku sektora.

▲ Wysyłając dane do portu o adresie EFH, program włącza silnik, włącza odczyt lub zapis oraz kasowanie (niezależne od zapisu).

▲ Port o adresie E7H jest wykorzystany do transmisji danych. Konwersja szeregowo-równoległa dokonywana jest sprzętowo; program używa po prostu rozkazów blokowej transmisji: INIR i OTIR.

W ZX INTERFACE 1 umieszczone jest 8 K pamięci EPROM, z systemem operacyjnym — w pewnych sytuacjach przesłania ona 16 K pamięci ROM komputera. Nowa pamięć włącza się przy wykonywaniu cyklu pobrania rozkazu z adresu 08H lub 1708H (druga możliwość pozwala „naprawić" błąd w procedurze CLOSE w interpreterze języka BASIC), a wyłącza się przy pobraniu rozkazu z adresu 700H. Rozkazy języka BASIC sterujące ZX MICRODRIVE znajdują się na klawiaturze SPECTRUM — jedak bez przyłączenia ZX INTERFACE 1 przeważnie powodują komunikat o błędzie: Nonsense in BASIC lub też nie dają się w ogóle wykonać.

▲ Wywołanie procedury inicjującej obsługę błędu (RST 08) wymaga przekazania tej procedurze numeru błędu jako parametru umieszczonego w następnym bajcie. Wartość ta dla standardowych błędów wynosi FFH lub mieści się między 00 a 1AH. Inna wartość powoduje wyświetlenie bezsensownego komunikatu. Przy wpiętym ZX INTERFACE 1 błędy 00 — 1AH rozpoznawane jako „zwykłe" i sterowanie wraca do interpretera języka BASIC, jednak kody 1BH do 31H traktowane są jako wywołania procedur w nowym ROM-ie — czytających lub zapisujących jeden sektor mikrokasety, wysyłających informację do RS 232, itp. Kody błędów są traktowane jako indeksy w tablicy procedur mieszczącej się pod adresem 19A9H w nowym ROM-ie. Można to wykorzystać w programach w kodzie maszynowym. Przy powrocie zawsze włącza się stary ROM.

▲ Aby „obejrzeć" nowy ROM, trzeba przez SAVE zapisać jego zawartość na kasetce, a następnie załadować do pamięci roboczej. Można też postąpić w sposób bardziej wyrafinowany. Kod błędu 32 jest opisany jako „zarezerwowany dla Sinclair Research Ltd.". W rzeczywistości powoduje oddanie sterowania procedurze, której adres mieści się w zmiennej systemowej (nowej!/) HD-11 o adresie 23789

(5CEDH), która nominalnie zawiera numer wiersza programu zapisanego z opcją LINE. Umieszczając własny program obsługujący ten „błąd", możemy bez przeszkód kopiować lub oglądać ROM z ZX INTERFACE 1.

System rezerwuje w pamięci RAM 58 bajtów dla zmiennych systemowych (od adresu 23734) oraz dla każdego otwartego zbioru — 32 bajty na mapę sektorów i 595 bajtów na tzw. kanał, którego częścią jest bufor. Zatem początek programu w BASICU przesuwa się „w górę". Jeśli program nie ma miejsca, aby się przesunąć, następuje błąd.

Przykładowo — zapisanie pliku instrukcją **SAVE * „m"; nr" nazwa pliku"** składa się z następujących czynności:

- alokacja kanału i mapy
- włączenie silnika
- przeczytanie wszystkich sektorów ze sporządzeniem mapy bitowej wolnych sektorów i sprawdzeniem, czy nie ma pliku o tej samej nazwie
- przekazanie 9 bajtów opisu pliku do bufora (typ pliku, długość, adres startu itp.); są to informacje, które na kasecie magnetofonowej zapisywane są wraz z nazwą pliku w oddzielnym nagłówku, tutaj stanowią one pierwsze 9 bajtów danych
- przekazanie do bufora bajtów danych; przy każdorazowym wypełnieniu bufora (tzn. przy próbie przekazania 513 bajtu) zapisywany jest nowy sektor, na co składają się następujące czynności:

— odczytanie nagłówka następnego sektora na taśmie
— sprawdzenie w mapie, czy sektor jest wolny
— zapisanie 512 bajtów danych
— zaznaczenie w mapie sektora jako „zajętego"
— zwiększenie numeru rekordu w buforze

- zapisanie ostatniego rekordu; rekord ten ma ustawiony odpowiedni bit w nagłówku oraz z reguły mniejszą liczbę danych
- wyłączenie silnika
- zwolnienie kanału.

▲ Tworzenie plików z danymi wymaga „otwarcia pliku" przez procedurę OPEN, np. OPEN#4,"M";"Dane". Spowoduje to, że w pamięci zostanie zarezerwowany bufor przypisany strumieniowi wyjściowemu nr 4. Instrukcja PRINT#4;... zapisze dane do bufora, a gdy ten się wypełni, zapisze dane na kasetce w pliku o nazwie "Dane". Tak utworzony plik można oczywiście wymazać lub skopiować. Można go czytać przez INPUT#4; lub INKEY$#4; — wymaga to uprzednio zamknięcia pliku przez CLOSE#4 i ponownego jego otwarcia; plik ponownie otwarty można już tylko czytać, nie można go rozszerzać za pomocą standardowych operacji. Strumień nr 2 jest standardowo przypisany ekranowi — najszybszą metodą przejrzenia zawartości pliku jest skopiowanie go na ekran, np. przez MOVE#4 TO#2.

Przykładowe rozkazy języka BASIC do współpracy z ZX MICRODRIVE:

**FORMAT "m";1;"biblioteka"** — formatowanie kasetki, napęd nr=1, nazwa kasetki="biblioteka"

**CAT 1** — spis plików z podaniem wolnej pamięci w KB

**ERASE "m";1;"dane"** — skasowanie plików "dane"

**SAVE* "m";1;"prog1"LINE 10** — zapisanie programu "prog1" — tak, aby po wczytaniu automatycznie startował od linii nr 10

**LOAD* "m";1;"kod1"CODE 32768** — wczytanie zbioru "kod1" od adresu 32768

**RUN** — wczytanie i zastartowanie programu run"

**VERIFY* "m";1;"tablica2"DATA a()** — sprawdzenie, czy plik zawiera te same wartości co tablica a ( )

**MERGE* "m";1;"prog2"** — dołączenie do programu istniejącego — programu zapisanego pod nazwą "prog2"

**OPEN 4;"m";1;"spis"** — przyporządkowanie do kanału nr 4 pliku „spis"; do tak otwartego pliku możliwy jest dostęp sekwencyjny za pomocą instrukcji PRINT#4, INPUT#4, LIST#4 i INKEY$#4

**MOVE#4 TO#15** — przepisanie pliku przyporządkowanego do kanału nr 4 — do zbioru przyporządkowanego do kanału nr 15.

Pliki zapisane instrukcją SAVE mogą być odczytane tylko instrukcjami LOAD, MERGE i VERIFY. Zbiory zapisane przez PRINT#, LIST# i MOVE mogą być odczytane tylko za pomocą INPUT#, INKEY$# i MOVE.
System operacyjny pozwala na rozszerzenie języka BASIC o nowe instrukcje. Jeżeli dana instrukcja nie daje się zinterpretować ani przez „nowy" ani przez „stary" ROM, to wykonywany jest jeszcze skok do procedury, której adres zapisany jest w RAM w zmiennej systemowej VECTOR (adres: 23735).

▲ W zbliżony sposób można rozszerzać zestaw instrukcji języka BASIC i bez ZX INTERFACE 1 — nawet dla ZX 81. Procedura obsługi błędu pobiera ze zmiennej systemowej ERR-SP adres komórki stosu, która zawiera adres restartu: procedury drukującej diagnozę i wracającej do głównej pętli edytora języka BASIC. Można wymienić adres restartu lub zawartość ERR-SP, wymaga to jednak dobrej znajomości systemu ZX.

**Opracowano na podstawie materiałów nadesłanych przez:**
**JERZEGO KARCZMARCZUKA (▲) z Krakowa oraz TADEUSZA WILCZKA i TOMASZA ZIELIŃSKIEGO z Warszawy**

*

Nie wszystkie programy z kaset magnetofonowych dają się bezpośrednio przenieść na MICRODRIVE. Jeżeli program zostawia mało miejsca na BASIC (tzn. obniża znacznie RAMTOP), to może dojść do sytuacji braku pamięci na zmienne systemowe i kanały (potrzebna rezerwa — ponad 600 bajtów). Ponadto segmenty kodu maszynowego zapisane np. w instrukcjach REM będą się zaczynać — po przesunięciu początku pamięci zawierającej program w języku BASIC — od innego adresu. Ich uruchomienie powoduje z reguły zawieszenie systemu.

Mikrokasety są niestety nośnikiem zawodnym. Zdarza się, że niektóre rekordy po nagraniu i nawet po zweryfikowaniu nie dają się po pewnym czasie odczytać. Należy się przed tym zabezpieczać nagrywając ważniejsze zbiory równolegle na dwie kasetki oraz od czasu do czasu robiąc kopie na kasecie magnetofonowej. Zbiory po nagraniu należy weryfikować (nie dotyczy to zbiorów sekwencyjnych zapisanych przez instrukcje PRINT#, LIST# i MOVE, przy których możliwości tej nie ma).
Błąd w sytemie oprercyjnym spowodował, że jeżeli podczas wykonywania instrukcji SAVE zabraknie pamięci, to system zawiesza się przy włączonym silniku. Instrukcja obsługi zabrania wyjmowania kasetki lub odłączania zasilania, jeżeli ZX MICRODRIVE pracuje. Środkiem zaradczym jest tu krótki (8 bajtów) program, sprawdzający przed rozpoczęciem wykonywania instrukcji liczbę wolnych bajtów.

ZX MICRODRIVE stanowi świetną pomoc szczególnie w programowaniu w kodzie maszynowym. Podczas testowania programów dochodzi często do zawieszania się systemu. Powtórne wczytanie programów ASSEMBLER i DEBUGGER oraz tekstu źródłowego trwa tylko kilkanaście sekund. Czas zapisu zbioru znacznie wzrasta, jeżeli na kasetce jest mało wolnego miejsca.

Istnieje już dzisiaj wiele programów przystosowanych do współpracy z ZX MICRODRIVE. Są to m.in. programy użytkowe: DECPAC (asembler, debuger i disasembler), PASCAL, MASTERFILE (baza danych), TASWORD (procesor tekstu), OMNICALC II (program kalkulacyjny), BETA BASIC (rozszerzenie BASICA), CASH CONTROLLER (budżet domowy).

Programy współpracują z ZX MICRODRIVE w ten sposób, że wczytuje się do pamięci cały zbiór, który po przetworzeniu zapisuje się z powrotem na kasetce. Zazwyczaj nie stosuje się tutaj współpracy interakcyjnej, cha-

rakterystycznej dla dysków elastycznych; poważnym ograniczeniem jest tu czas dostępu i zawodność.

Duże zalety ZX MICRODRIVE w porównaniu z magnetofonem ujawniają się w przypadku zbiorów często modyfikowanych — takich jak kartoteki czy budżet.

Pamięć ZX MICRODRIVE jest cennym uzupełnieniem mikrokomputera ZX SPECTRUM, podnoszącym jego walory użytkowe. Nie bez znaczenia jest tutaj s t o s u n k o w o niska cena stacji i interfejsu.

**TADEUSZ WILCZEK**
**TOMASZ ZIELIŃSKI**

*

MICRODRIVE ma przed sobą przyszłość (choć chyba nie w wersji proponowanej przez SINCLAIRA, dzięki swojej niebywałej prostocie i — co za tym idzie — niskiej cenie. Aktualna cena jest po prostu nieprzyzwoita: urządzenie to jest konstrukcyjnie prostsze od przeciętnego magnetofonu kasetowego. Korzysta z głowicy niczym nie różniącej się od typowej głowicy stereo. Mechanika jest bardzo prosta, elektronika też. A trzeba przy tym pamiętać że stacja dysków 5.25" do SPECTRUM (Viscount) kosztuje 245 funtów.

Bardziej skomplikowana jest konstrukcja mikrokasety. Jednak jej cena również jest zawyżona; wszyscy fachowcy twierdzą, że spadnie, choć niestety — nikt nie wie kiedy.

INTERFACE 1 wykorzystuje porty we—wy o adresach E7H, EFH i F7H, tj. trzecią i czwartą linię adresową. Może to być źródłem kłopotów w przypadku używania razem z INTERFACE 1 niestandardowych urządzeń.

Po około trzech miesiącach dość intensywnej pracy nie miałem ani jednego przypadku zauważalnego przekłamania, nie mówiąc o uszkodzeniu taśmy (odpukać!). Na jednej mikrokasecie mam jeden uszkodzony sektor, którego użycie powodowało, że po kilkudziesięciu sekundach kręcenia taśmą komputer sygnalizował „File not found", mam jednak podstawy sądzić, że ten fragment taśmy nie zepsuł się w trakcie eksploatacji, lecz był od początku zły[1] — być może jest to miejsce klejenia tak niefortunnie wykonane, że prymitywna procedura formatująca przeoczyła błąd. Gorzej, że taki sektor — zepsuty, ale prawidłowo sformatowany — będzie przeszkadzał i trzeba go unieszkodliwić wpisując plik „manekin" za pomocą specjalnie w tym celu napisanego programu.

Moje główne problemy? Jednak strach przed uszkodzeniem taśmy, co powoduje, że wszystkie ważne dane i tak kopiuję na kasetę. Dalej — stosunkowo mała pojemność: moje mikrokasety mają po 90 K pamięci. Wreszcie — powolność odczytu—zapisu i niewygodne korzystanie z plików.

Wczytywanie pliku za pomocą standardowych procedur ładuje sektory o kolejnych numerach. Jeśli plik był wpisany na dość wypełnioną taśmę i sektory zostały umieszczone w kolejności niezbyt naturalnej, to zanim cały plik zostanie wczytany — MICRODRIVE kręci taśmą aż do znudzenia. Oczywiście należało to zorganizować rozsądniej.

Po zapisaniu pliku użytkownik zamyka go. System zapisuje ostatni rekord wraz ze specjalnym znacznikiem końca. Jeśli wskutek błędu powstał na taśmie plik bez znacznika końca, nie dzieje się nic specjalnie złego: wymazywanie takiego pliku oraz próba wczytania nie istniejącego rekordu trwa kilkadziesiąt sekund. Mam na to własne określenie: Młynek Modlitewny Sir Clive'a...

Zapisanego i zamkniętego pliku nie da się rozszerzyć inaczej jak tylko przez skopiowanie całości pliku na nowy i dopisanie doń informacji. Jest to prymitywne i niewygodne rozwiązanie. Nie ma żadnych przeszkód, aby skonstruować znacznie lepszy system obsługi ZX MICRODRIVE.

Można mieć pliki indeksowane albo adresowalne bajtowo. Można prosto napisać program rozszerzający plik; wystarczy wczytać ostatni rekord i wymazać zeń znacznik końca pliku. Można posłużyć się bardziej efektywnymi algorytmami rozmieszczającymi informację na mikrokasecie i optymalizującymi dostęp do plików. Wszystkie te udogodnienia producent zostawił dla użytkowników i niezależnych twórców oprogramowania.

W zamian producent wprowadził „zabezpieczenia", które np. uniemożliwiają MERGE programów zapisanych opcją LINE i nie pozwalają czytać zawartości plików zapisanych przez SAVE* — można je tylko ładować przez LOAD*. Miało to na celu ochronę programów komercyjnych. Oczywiście, nawet dla średnio wykwalifikowanych piratów nie są to duże zabezpieczenia. Co więcej, kopiowanie kasetki jest łatwiejsze niż kasety magnetofonowej (przez komputer, a nie bezpośrednio), gdyż można kopiować sektor po sektorze, a program na kasecie magnetofonowej stanowi fizyczną i logiczną całość.

Warto tu wspomnieć, że samostartujący program zapisany na kasecie również można zabezpieczyć przed załadowaniem przez MERGE, jeśli zawiera on jeden wiersz w BASICU, który został „nadpsuty" przez wpisanie bardzo dużej liczby w pole zawierające jego długość [1]).

Czy zachęcać do kupna MICRODRIVE? Zależy kogo. Nabywca musi zdawać sobie sprawę, że na razie oprogramowanie nie jest najlepsze. ROM w INTERFACE 1 zawiera błędy, choć nieznaczne moim zdaniem — w porównaniu np. z błędem psującym bity zaokrąglenia w operacji dzielenia, mogących jednak wpłynąć na opóźnienie decyzji kupna; a nuż późniejsze wersje będą poprawione? Tyle, że drobny błąd w dzieleniu był sygnalizowany już na początku kariery ZX81, a mimo to algorytmy arytmetyki zmiennoprzecinkowej zostały przepisane na SPECTRUM praktycznie bez zmian...

Niezależne firmy zajmujące się oprogramowaniem dopiero zaczynają oferować programy dostosowane do ZX MICRODRIVE, a ich dotarcie do Polski jeszcze się opóźni. Aktualnie najbardziej rozpowszechnione wersje PASCALA HISOFT: HP4S i HP4M1.6 nie pozwalają efektywnie korzystać z ZX MICRODRIVE. Co prawda można napisać własny program w kodzie i dołączyć do tekstu źródłowego, ale nie można pod kontrolą edytora ani kompilatora korzystać z podprogramów na mikrokasecie. HISOFT jednak stale ulepsza swoje produkty.

Najpopularniejsze wersje FORTHA: firmy ARTIC oraz znacznie wygodniejszy firmy ABERSOFT — również ignorują MICRODRIVE. Tu jednak, ze względu na specyfikę FORTHA, raz napisane rozszerzenie systemu można na stałe w nim „zamrozić" i jest to o wiele łatwiejsze niż próba przeróbki kompilatora PASCALA!

W stosunkowo dobrej sytuacji są programy, które korzystały z magnetofonu za pośrednictwem języka BASIC — wystarczy wtedy wymienić SAVE na odpowiednie SAVE*, itp; oczywiście, o ile jest miejsce na zmienne systemowe i kanał. Tak postępuje — na przykład — prosty, ale dość użyteczny procesor tekstów TASWORD 2.

Polecałbym więc zakup ZX MICRODRIVE osobom, które dobrze poznały komputer, nie boją się pisania programów systemowych, a nie mają możliwości kupna znacznie droższych dysków. Nawet niezbyt zaawansowani programiści, ale piszący bardzo długie programy lub ciągi danych, które trzeba wpisywać na raty, na pewno odetchną po wątpliwych przyjemnościach żonglowania kasetami magnetofonowymi.

**JERZY KARCZMARCZUK**

**LITERATURA**

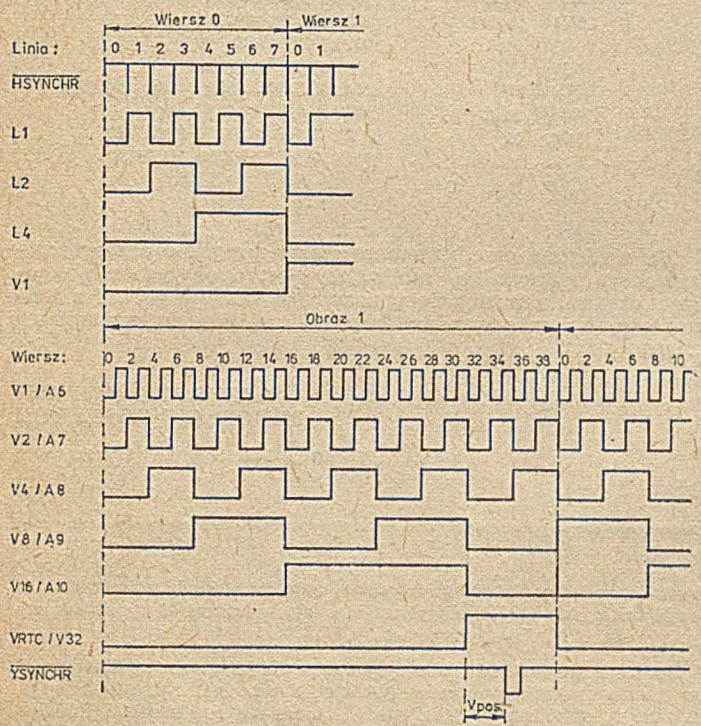
[1] Logan I.: Spectrum Microdrive Book. Melbourne Hause, 1983
[2] Your Computer. No. 12/83, 6/84
[3] ZX SPECTRUM Microdrive and Interface 1 manual.

[1]) Za ten chwyt pragnę podziękować Pawłowi Mąćków z Wrocławia — przypis autora

Pierwsza część artykułu ukazała się w poprzednim numerze. Drugą częścią kończymy opis sterownika zrealizowanego na układach TTL. Aby ułatwić Czytelnikom wykorzystanie układu, zamieszczamy również program obsługi. Niebawem opiszemy zintegrowany sterownik CRT.

# Alfanumeryczny sterownik monitora telewizyjnego (2)

Impulsy synchronizacji poziomej HSYNCHR zliczane są przez liczniki U9 i U10. Wyjścia L1, L2 i L4 (U9) podają numer wyświetlanej linii i adresują odpowiednio pamięć ROM (U1). Wyjścia V1...V16 określają numer wyświetlanego wiersza znaków i dołączone są do wejść adresowych $A_6...A_{10}$ pamięci RAM. Po zliczeniu 32 wierszy (rys.) wytwarzany jest impuls VRTC (ang. verical retrace), wygaszający plamkę na czas powrotu do lewego górnego rogu ekranu. Okres trwania impulsu VRTC jest równy czasowi wyświetlania ośmiu wierszy. Gdy stan licznika wierszy osiąga wartość 32+8, zostaje on wyzerowany (wejścia zerujące $R_{01}$ i $R_{02}$ połączone są z wyjściami V32 i V8) i liczenie wierszy rozpoczyna się od początku. Przednie zbocze impulsu VRTC wyzwala uniwibratory U12 (74123), wytwarzające impuls synchronizacji pionowej VSYNCHR/. Położenie tego impulsu w przedziale czasowym, przeznaczonym na powrót plamki jest regulowane potencjometrem V. Umożliwia to właściwe ustawienie położenia obrazu w pionie.




**Przebiegi czasowe zespołu liczników linii i wierszy**

Całkowita liczba linii obrazu (łącznie z liniami przypadającymi na czas powrotu plamki wynosi 8×(32+8)=320 linii, co nieznacznie różni się od przyjętej w standardzie telewizyjnym liczby 312,5 linii na obraz (bez wybierania międzyliniowego). Przy czasie trwania linii równym 64 µs częstotliwość wyświetlania obrazów wynosi 48,8 Hz, zamiast wymaganej w standardzie TV częstotliwości 50 Hz.

Impulsy synchronizujące HSYNCHR/ i VSYNCHR/ są sumowane z sygnałem wizyjnym (otrzymywanym z rejestru przesuwnego) w obwodzie wejściowym tranzystora T1, pracującego w układzie wtórnika emiterowego. Rezystory $R_1$ i $R_2$ ustalają właściwy stosunek poziomu sygnału wizyjnego i impulsów synchronizacji. Z obwodu emitera T1 wyprowadzany jest zespolony sygnał wizyjny. Potencjometr $R_4$ umożliwia regulację poziomu tego sygnału. Kondensator $C_1$ oddziela obwód emitera $T_1$ od ewentualnej składowej stałej, jaka może wystąpić na wejściu wizyjnym monitora. W zależności od typu monitora, może być wymagana różna biegunowość tego kondensatora.

Sygnały wygaszające plamkę na czas powrotu poziomego i pionowego (HRTC i VRTC), są sumowane w jeden wspólny sygnał wygaszający VSP/ (ang. video suppression). Jest on wpisywany do przerzutnika 7474 (U13) impulsem LOAD/, ładującym również rejestr przesuwny U25. Konieczność takiego synchronizowania sygnału VSP/ (a w szczególności sygnału HRTC) impulsem LOAD/ wynika ze sposobu wykorzystania rejestru. Sygnały składające się na linię danego znaku wprowadzane są do rejestru pod koniec cyklu wyświetlania poprzedniego znaku i wyprowadzane z rejestru do wejścia wizyjnego w cyklu adresowania następnego znaku (por. rys. 2 w pierwszej części). Doprowadzenie sygnału HRTC (VSP/) do wyjścia wizyjnego musi być więc opóźnione o jeden znak. Otrzymywany z wyjścia przerzutnika U13 sygnał VSP/ DEL blokuje przejście dla sygnałów z rejestru przesuwnego i wymusza wygaszanie ekranu na czas trwania impulsów HRTC i VRTC.

Rozdzielanie poszczególnych znaków od siebie — zarówno w pionie, jak i w poziomie — zapewnione jest przez postać znaków umieszczonych w ROM-ie (U1). Użyty w sterowniku krajowy układ MCY 7304AA pozostawia zawsze niezaświetloną pierwszą linię i ostatnią kolumnę pola znaku. Efektywne pole znaków ma wymiary 5×7 punktów.

Kursor realizowany jest w postaci migających dwóch najniższych linii pola znaku i wyświetlany jest w przypadku wpisania jedynki w najbardziej znaczącym bicie bajtu znaku w pamięci obrazu. Wyprowadzany z pamięci RAM sygnał bitu D7 jest synchronizowany impulsem LOAD/ za pomocą przerzutnika U13 — dla uzyskania opóźnienia o jeden znak (podobnie jak dla sygnału VSP/). Uzyskany w ten sposób sygnał D7 RAM DEL jest doprowadzany do wejścia bramki U14 (7430), formującej sygnał włączenia kursora. Bramka U14 mnoży logicznie D7 RAM DEL z sygnałami wyjściowymi L2 i L4 licznika linii i wysyła sygnał włączający kursor tylko w czasie wyświetlania linii 6 i 7 (L2=L4=1). Bramka jest blokowana sygnałem LOAD/ w czasie wyświetlania ostatniego (nieaktywnego) punktu znaku oraz sygnałem CSP/ — na czas powrotów plamki. Sygnał BLINK, uzyskiwany w wyniku podzielenia sygnału BRTC przez licznik U15, powoduje migotanie kursora z częstotliwością ok. 3 Hz. Sygnał CURSOR/ sumowany jest z sygnałem wizyjnym, w sposób wymuszający rozświetlenie ekranu bez względu na poziom sygnału wysyłanego przez rejestr przesuwny. Kursor może zostać zredukowany do najniższej (siódmej) linii pola znaku w przypadku dołączenia do wejścia U14 dodatkowo sygnału L1.

Pamięć obrazu (RAM U16...U19) jest w czasie wyświetlania adresowana przez zespół liczników znaków i wierszy (sygnały H1...V16) w wyniku ustawienia sygnału MUX=1 przełączającego multipleksery U4...U6. Pamięć jest w tym czasie utrzymywana w stanie aktywnym i pracuje w trybie odczytu (WE/=1). Pamięć może być dołączona do systemu mikroprocesorowego przez przełączenie multiplekserów linii adresowych (sygnał MUX=0) oraz uaktywnienie dwukierunkowego bufora szyny danych U20, U21. Dostęp do pamięci możliwy jest jedynie w czasie wygaszania plamki. Dołączanie pamięci do systemu mikroprocesorowego w czasie wyświetlania obrazu zaburza bowiem proces wyświetlania i powoduje występowanie na ekranie wielu dodatkowych, chaotycznie migocących punktów świetlnych.

Możliwość dostępu do pamięci wyznacza sygnał BLK/—VSP=1, wyłączający plamkę. Sygnał ten ustawia jednocześnie MUX=0, dołączając linie adresowe pamięci obrazu do szyny adresowej systemu mikroprocesorowego oraz odblokowuje sygnały uaktywniające pamięć RAM, bufor szyny danych i strob zapisu do pamięci. W przypadku czytania pamięci, sygnał MEMR/ zmienia kierunek przenoszenia sygnałów przez bufor danych (DIEN/ 8216=0).

W czasie wyświetlania, sygnał BLOK/—VSP=0 blokuje dostęp mikroprocesora do pamięci U16...U19, wymuszając MUX=1, CS/ (8216)=1 i WE/=1. W przypadku zaadresowania w tym czasie pamięci obrazu zgłaszanego przez dekoder adresowy (U24) sygnałem MEMSEL=1, praca procesora jest wstrzymana do chwili zakończenia wyświetlania linii przez wysłanie sygnału braku gotowości do współpracy WAITRQ/—READY=0. Sygnał braku gotowości musi być wysłany z wyprzedzeniem, gwarantującym zatrzymanie każdego cyklu dostępu do pamięci, który mógłby być zakończony po rozpoczęciu wyświetlania linii. W tym celu praca procesora jest wstrzymywana już w przypadku próby uzyskania dostępu do pamięci ekranu w czasie ostatniego cyklu znaku przebiegu HRTC. Bramka, wysyłająca sygnał WAITRQ/—READY przy MEMSEL=1, jest blokowana sygnałem wyjściowym Q przerzutnika U22. Przerzutnik ten zerowany jest przez przednie zbocze impulsu HRTC (podanego na wejście T), a następnie ustawiany (Q=1), gdy sygnały wyjściowe H1...H16 liczników znaków są wszystkie w stanie 1, co występuje po zliczeniu 95 znaków (jeden znak przed końcem impulsu HRTC — (p. rys.). Stan H1...H16=1 dekoduje bramka U23, dołączona do wejścia S/ przerzutnika. Bramka ta jest blokowana sygnałmi HRTC i VRTC, co uniemożliwia wysłanie sygnału WAITRQ/ w czasie wygaszania plamki.

Dekoder adresu pamięci obrazu (U24) ustawia MEMSEL==1 przy A11...A15=1. Ustalono w ten sposób adresy pamięci obrazu w zakresie 0F800H...0FFFFH. Przez dołączenie dodatkowego inwertera (NOT) negacji między punkty a—b linii A15, można je zmienić na 7800H...7FFFH.

Dekoder adresowy powinien być bramkowany strobami dostępu do pamięci MEMR/ i MEMW/ dla uniemożliwienia wykrywania przez dekoder zbieżnych adresów urządzeń we-wy i wysyłania zbędnych sygnałów WAITRQ/—READY. Stosowanie takiego bramkowania jest jednak niemożliwe w systemach z procesorem 8080 i kontrolerem 8228 (zbyt późno wysłany sygnał MEMW/ nie umożliwia wstrzymania cyklu zapisu do pamięci sygnałem WAITRQ/ generowanym po odebraniu strobu MEMW/). Takie bramkowanie dekodera jest natomiast możliwe w przypadku stosowania kontrolera 8238 (lub w systemach z innym procesorem, np. Z80 — sygnałem MEMRQ). W systemach z procesorem 8080 i kontrolerem 8228 należy natomiast blokować dekoder adresowy sygnałami I-OR/ i I-OW/.

Pamięć obrazu (2 KB) złożono z czterech układów typu 2114, o organizacji: 1024 bity×4. Pamięć ta może być także zbudowana z układów 8102 (MCY 7102) o organizacji: 1024 bity×1; jednak konieczne jest wówczas użycie 16 takich układów. Dodatkowo, należy połączyć rozdzielone wejścia—wyjścia tych pamięci odpowiednio z wyjściami DO i wejściami DI buforów danych 8216 i wejściem pamięci ROM.

Jako generatora znaków użyto ROM typu MCY 7304AA (inna ostatnia litera symbolu oznacza generator znaków programowany nie w kodzie ASCII!), realizujący przetwarzanie 64 podstawowych znaków ASCII (cyfry, duże litery i znaki pomocnicze) o kodach od 20H do 5FH. Zamiast tego układu można zastosować odpowiednio zaprogramowany EPROM 2708 lub 2716, rozszerzając asortyment znaków do pełnego kodu ASCII, obejmującego także małe litery.

```
;WYGASZANIE EKRANU

WYGAS: LXI   H,POCZEK   ;ADRES POCZ. PAMIECI EKRANU
       LXI   B,800H     ;2048 POZYCJI
WYG1:  MVI   M,20H      ;WPIS SPACJI
       DCX   B          ;DEKREMENTACJA LICZNIKA
       MOV   A, B       ;CZY LICZNIK ZERO?
       ORA   C
       JNZ   WYG1       ;NIE ZERO — PETLA


       ;WPIS ZNAKÓW ASCII

PISZ:  LXI   H,POCZEK   ;ADRES POCZ. PAMIECI EKRANU
       MVI   B,40H      ;64 ZNAKI
       MVI   C,AOH      ;SPACJA + KURSOR
       MOV   M,C        ;WPIS KURSORA
       MVI   A,20H      ;KOD PIERWSZEGO ZNAKU
PISZ1: MOV   M,A        ;WPIS ZNAKU
       INR   A          ;NASTEPNY KOD
       INX   H          ;NASTEPNA POZYCJA WPISU
       MOV   M,C        ;WPIS KURSORA
       DCR   B          ;CZY WSZYSTKIE ZNAKI?
       JNZ   PISZ1      ;NIE WSZYSTKIE PETLA
       END
```

Na wydruku przedstawiono prosty program obsługi opisywanego sterownika, realizujący wstępne wygaszenie całego ekranu, a następnie wpisanie 64 znaków ASCII z jednoczesnym przesuwaniem kursora na pozycję następnego znaku.

**GRZEGORZ STĘPIEŃ**
**MIROSŁAW DOLEŻYCH**

IBSPiE, Politechnika Warszawska

---

● Klub mikrokomputerowy ABAKUS zmienił siedzibę. Z dalekiej Sadyby przeniósł się do śródmiejskiego Centrum Gier i Rozrywek. W nowym lokalu na razie nie jest zbyt „rozrywkowo", bo przecieka dach i kapie woda. Jednak nie zważając na trudności wznowiono działalność i komputery są już dostępne dla klubowiczów. Niezrzeszeni będą mogli „pomacać" komputer raz w tygodniu w tzw. dzień otwarty. Jako nowość przewidziano spotkania dla dzieci w wieku od 2 do 4 lat. Nie jest to zapowiadany w mass-mediach żłobek mikrokomputerowy, bowiem rodzice muszą sami zmieniać pieluchy.

Obecny adres klubu: Al. Jerozolimskie 2, Warszawa (naprzeciwko Muzeum Narodowego) tel. 27-87-73 w. 3.

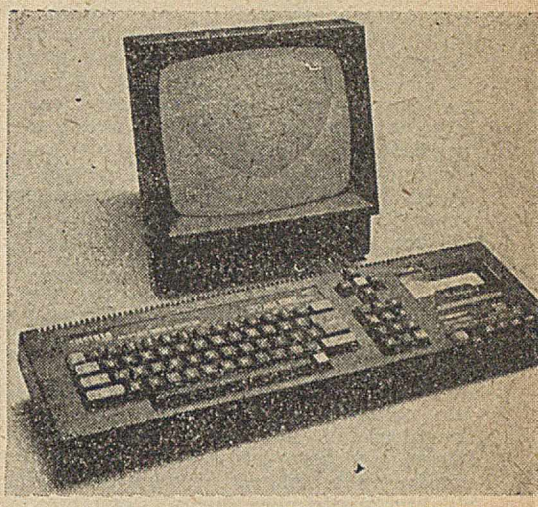
● „ABAKUS" ogłosił kolejny konkurs na grę edukacyjną. Poszukiwane są programy łączące elementy zabawy z przekazywaniem wiedzy lub wyrabianiem pewnych umiejętności (ale nie mięśni lub refleksu). Gra może być adresowana do osób w dowolnym wieku. Nie przewidziano również żadnych ograniczeń w zakresie przekazywanej wiedzy. Wysoko oceniane będą programy uniwersalne — czyli takie, które można wykorzystać dla szerokiej klasy zagadnień, przy niewielkich modyfikacjach.

Program powinien być opracowany dla jednego z dostępnych w klubie mikrokomputerów (COMMODORE C64, ZX SPECTRUM, ZX 81, ORIC 1, lub TI 99/4). Nie wyklucza się możliwości zgłaszania programów przeznaczonych dla innych mikrokomputerów, jednak w tym przypadku autor zobowiązany jest do zaprezentowania programu razem z komputerem (w uzgodnionym z jury terminie).

Programy (na taśmie lub dyskietce) powinny wpłynąć do klubu przed 30 listopada 1985 roku. Po zakończeniu konkursu nie przewiduje się odsyłania nośników (staną się one własnością klubu). Klub zastrzega sobie również prawo do wykorzystywania nadesłanych programów w działalności klubowej (oczywiście bez prawa rozpowszechniania).

Wśród nagród przewidziano m.in. COMMODORE C64, TI 99/4 oraz 50 gier komputerowych.

**Opisujemy komputer, który wprowadzono na rynek bez fanfar i bicia dzwonów, jak to miało miejsce przy pojawieniu się SINCLAIRA QL czy japońskich MSX-ów. Początkowo niedostrzeżony, znika obecnie ze sklepów jak przysłowiowa kamfora. Można przewidywać, że niebawem zrobi nie mniejszą karierę niż ZX SPECTRUM; wielkim jego atutem jest niska cena (po odliczeniu ceny monitora TV — zaledwie o 10 funtów większa od ceny ZX SPECTRUM). Na razie polecamy tę konstrukcję opierając się jedynie na literaturze. Niebawem spróbujemy przedstawić porównanie podanych rewelacji z praktycznymi doświadczeniami.**



WOJCIECH JURZYK

# AMSTRAD CPC 464

Jednym z ciekawszych komputerów domowych, jakie pojawiły się pod koniec ubiegłego roku, jest CPC 464, angielskiej firmy AMSTRAD. Jego konstrukcja nie jest rewelacyjna, uniknięto natomiast niezbyt szczęśliwych rozwiązań stosowanych w innych mikrokomputerach, dając w zamian istotnie lepsze.

Ewenementem jest fakt, że komputer ten jest sprzedawany wraz z monitorem. Użytkownik ma dwie możliwości do wyboru: monitor 14" kolorowy lub monochromatyczny z zielonym ekranem. Większego wyboru nie ma, jako że komputer zasilany jest właśnie z monitora. Dla tych, którzy nie chcą monitora, proponuje się osobny zasilacz wraz z modulatorem do stadardowego odbiornika TV (w systemie PAL), gwarantując jednocześnie dobrą jakość obrazu.

Zaskakujący jest fakt zintegrowania magnetofonu z klawiaturą i procesorem w jednej obudowie. Magnetofon jest zaopatrzony w licznik obrotów i częściowo sterowany z programu. Programista ma możliwość wyboru szybkości zapisu: o podwyższonej niezawodności 1000 lub 2000 bodów (jak wynika z materiałów producenta, nawet ta szybsza zapewnia zadowalającą jakość). Szybkość odczytu wybierana jest automatycznie.

Najistotniejszą niemal częścią komputera jest dla użytkownika klawiatura, z nią bowiem ma on najwięcej do czynienia. I tu AMSTRAD CPC 464 oferuje rozszerzone — w stosunku do komputerów domowych — możliwości; oprócz standardowej co do wielkości i rozkładu klawiszy (QWERTY) „prawdziwej" klawiatury jest 17 dodatkowych klawiszy — 5 przeznaczonych na sterowanie kursorem oraz 12 tworzących klawiaturę numeryczną. Zastrzeżenia budzi natomiast rozkład dodatkowych klawiatur — przy posługiwaniu się kursorem łatwo może dojść do przypadkowego naciśnięcia klawiszy numerycznych.

Użytkownik może również zaopatrzyć się w napęd dysków elastycznych. Jeżeli do tego doda się fakt, że można wtedy pracować pod kontrolą systemu operacyjnego CP/M, zaskoczenie rośnie. Na dyskietkach można mieszać zarówno pliki utworzone pod kontrolą CP/M, jak i AMSDOS (wewnętrzy system operacyjny). Możliwe są transmisje w obu kierunkach tzn. z magnetofonu na dysk i odwrotnie.

Wewnętrzną budowę CPC 464 cechuje brak ekstrawagancji — zastosowano procesor Z80A (istotne dla polskich użytkowników szkolonych na INTELU 8080) z częstotliwością zegara 4 MHz. Pamięć RAM 64 KB i do tego 32 KB pamięci ROM stronicowej po 16 KB i umieszczonej pod tymi samymi adresami co pamięć obrazowa (też 16 KB).

Użytkownik może wybrać jeden z trzech typów: 20, 40 lub 80 znaków w każdym z 25 wierszy i odpowiednio 2, 4 i 16 kolorów (z palety 27) dla k a ż d e g o punktu na ekranie. Z materiałów informacyjnych wynika, iż tryb o najwyższej rozdzielczości daje dobrą jakość tekstów jedynie na monitorze monochromatycznym (tekst w kolorze jest nadal czytelny, ale już nie tak wyrazisty).

Kilka z cech sprzętowej konstrukcji wymaga jednak szerszego opisu.

Po pierwsze — stronicowanie pamięci. Przewidziane jest dołączenie do 240 stron, z których odczyt jest prosty, natomiast ewentualny zapis nieco bardziej skomplikowany, choć także możliwy. Jest to więc prosty sposób na poszerzenie możliwości komputera (dodatkowe pamięci ROM) bez konieczności zmniejszania dostępnej pamięci RAM.

Po drugie — sterownik CRT i procesor są tak zsynchronizowane, że ich dostęp do pamięci nie następuje jednocześnie. Oznacza to, że procesor nie może wykonać więcej niż trzech cykli maszynowych bez stanu WAIT. Tu jednak autorzy oprogramowania (firma LOCOMOTIVE SOFTWARE) popisali się sprawnością — tak je konstruując, aby jak najmniej stracić z tego powodu na szybkości. AMSTRAD CPC 464 jest tylko nieco wolniejszy od BBC/Acorn.

Użytkownik ma do dyspozycji standardowe łącze Centronics dla dołączenia drukarki oraz osobno wyprowadzenie pełnej szyny procesora Z80 z kilkoma dodatkowymi sygnałami, np. do dołączenia pióra świetlnego — w sumie 50 sygnałów, wszystkie zbuforowane. Ponadto istnieje możliwość dołączenia drążka sterowniczego (a następnie do niego drugiego).

Dziwić może nieobecność niezależnego przycisku RESET — zerowanie realizowane jest na zasadzie trzech klawiszy: SHIFT, CONTROL, ESCAPE.

Do pełnego opisu „kontaktu ze światem" dodać trzeba jeszcze informacje o możliwościach dźwiękowych: trzy kanały siedmiooktawowe z szerokim zakresem sterowania wyprowadzane na wzmacniacz stereo lub bezpośrednio przez wewnętrzny głośnik komputera o regulowanym natężeniu dźwięku (słyszalnym jako mono).

Nie mniej godne uwagi jest zaimplementowane w AMSTRADZIE CPC 464 oprogramowanie — zarówno w części dotyczącej interpretera języka BASIC, jak i systemowej. BASIC oparty jest na standardzie przemysłowym opracowanym przez znaną firmę MICROSOFT, a dodatkowo zawiera w sobie dużo nowych funkcji do sterowania dźwiękiem, grafiką, wejściem—wyjściem, redagowaniem programów oraz (uwaga!) przerwaniami czasu rzeczywistego.

Z nowym rozwiązaniem spotykamy się już w edytorze. Oprócz standardowego edytora wierszowego, mamy do czynienia z kursorem kopiującym, dzięki któremu tekst może być kopiowany z jednego miejsca ekranu na inne. Istotnymi pomocami przy uruchamianiu programu w języku BASIC są: tryb śledzenia oraz instrukcja ON ERROR. Program można zatrzymać naciskając klawisz ESCAPE, zaś przerwać kolejnym naciśnięciem tegoż klawisza. Podobnie, tryb automatycznego przenumerowania linii programu jest dla użytkownika istotną pomocą.

Podstawowy zestaw instrukcji języka BASIC został poszerzony ponadto o takie instrukcje, jak WHILE...WEND oraz IF...THEN...ELSE, zaś instrukcja PRINT USING znacznie ułatwia pracę przy formatowaniu informacji wyjściowej.

Zdecydowanie większe niż w innych mikrokomputerach są możliwości pro-

gramowania grafiki. Standardowy zbiór instrukcji DRAW, PLOT, MOVE (przesunięcie kursora bez śladu) i TEST (podaje kolor punktu na ekranie) występuje również w postaci relatywnej, zgodnej z nowym układem współrzędnych definiowanych w instrukcji ORIGIN. Można też wyprowadzać teksty na obrazy graficzne. Pracując w języku BASIC, użytkownik ma do dyspozycji 42 KB RAM.

Kolejne dwie cechy oprogramowania: to ostatni krzyk mody w tej branży: okna i strumienie (ang. windows & streams). Można tu zdefiniować osiem okien, z czego jedno graficzne, i mogą one się nawzajem nakładać. Okna powiązane są ze strumieniami jako ich obiekty wyjściowe, podobnie jak drukarka i magnetofon lub napęd dyskowy. Jako obiekty wejściowe służą: klawiatura, magnetofon i dysk. Obsługa strumieni jest wyjątkowo prosta, ale poważną wadą jest zerowanie wszystkich okien przy zmianie trybu pracy (np. z 40 na 80 kolumn).

Cechą niezwykle rzadko spotykaną w tego typu konstrukcjach są przerwania. CPC 464 ma cztery wewnętrzne zegary dostępne dla programisty, które mogą być użyte do generowania przerwań. Dotychczas przerwaniami zajmowali się konstruktorzy i programiści rozwiązujący zagadnienia typu systemowego, zaś za pomocą AMSTRADA CPC 464 każdy może odbyć „kurs programowania" w czasie rzeczywistym. Instrukcja AFTER generuje pojedyncze uprzednio zdefiniowane przerwanie, zaś EVERY czyni to w regularnych odstępach czasu.

Autorzy oprogramowania dołożyli dużo starań, aby ułatwić użytkownikowi dostęp do oprogramowania firmowego. Jego pełny opis znajduje się w odpowiednim podręczniku. Producent z zasady stara się udostępnić użytkownikom wszelkie informacje mogące służyć lepszemu wykorzystaniu lub poszerzeniu możliwości mikrokomputera.

Reasumując — AMSTRAD CPC 464 jest przykładem dobrego rozwiązania pod względem technicznym — zarówno od strony sprzętowej, jak i programowej. Niemal grzechem byłoby używać tego komputera do tak prymitywnych zastosowań jak gry. Przecież to prawie profesjonalny sprzęt: wygodna klawiatura, 80-kolumnowy ekran, dysk elastyczny pracujący pod systemem CP/M (dostarczanym wraz z językiem LOGO — wersja opracowana przez firmę DIGITAL RESEARCH specjalnie dla AMSTRADA CPC 464) oraz bardzo tani „backup" — w polskich warunkach cechy o niebagatelnym znaczeniu. Można się w krótkim czasie spodziewać dużej ilości oprogramowania na kasetach i dyskach (do końca 1984 roku oferowano ponad 200 tytułów).

Wielu producentów oprogramowania dostosowuje swoje programy (napisane dla innych komputerów) dla CPC 464, nie wspominając już o niezliczonych zasobach programów pracujących pod nadzorem CP/M.

**MAREK GÓRECKI**
Warszawa

---

**Akademia mikroKLANU stara się uczyć elegancji w pisaniu programów. Jej lekcje przeznaczone są dla osób mających podstawowe doświadczenie w programowaniu. Liczymy na propozycje tematyczne od „uczniów".**

# Akademia mikroKLANU (5)

## Wyrażenie

Kolejną część naszego cyklu poświęcimy pewnemu drobiazgowi... Spróbujmy wykonać program z wydruku 1. Jest to procedura mogąca być częścią dużego programu numerycznego.

```
10 A=0
20 B=1
30 C=2
40 A=A+B
50 B=SQR(1+A*A)
60 C=C+C
70 X=3*C/(A+B+A)
80 PRINT X
90 GOTO 40
```

Wartość X, powinniśmy skończyć obliczać gdy osiągniemy już pełną dokładność numeryczną komputera (patrz odcinek 1). Najbardziej prymitywną metodą jest utworzenie pętli explicite[1]) w tym celu dodajemy instrukcje:

```
35 FOR D=1 TO 8
90 NEXT D
```

Po poprawkach procedura wydrukuje osiem wartości (zbieżnych do ..., trzeba się samemu przekonać do czego!) i zakończy się.

```
34 X=0
35 EPS=1/1000000
36 Y=X
80 IF ABS(X-Y)>EPS THEN GOTO 36
90 PRINT X
```

Gdy dysponujemy komputerem o dużej dokładności obliczeń wolelibyśmy jednak wyliczać wartość X — aż do otrzymania pożądanej dokładności. Zatem niech EPS oznacza żądaną dokładność bezwzględną. Wtedy, wracając do pierwotnej wersji procedury i dokonując poprawek jak na wydruku 2, możemy spokojnie (podstawiając różne wartości EPS) obliczać wartość X. Jeśli algorytm obliczania naszego wyrażenia jest skończony (już raz mieliśmy do czynienia z tym problemem przy Zasadzie Ulama — odcinek 3), to zawsze otrzymamy odpowiednią liczbę z żądaną dokładnością. Może to jednak trwać zbyt długo, jeżeli X jest duże, a zbieżność procedury powolna (EPS wybraliśmy ambitnie małe!). Dlatego zawsze powinno się określić liczbę iteracji, po wykonaniu których — niezależnie od osiągniętej dokładności — program zakończy obliczenia. Zrobimy to wprowadzając modyfikacje z wydruku 3.

```
32 I=0
33 M=50
71 I=I+1
80 IF ABS(X-Y)>EPS AND I<M THEN GOTO 36
```

Jeszcze powinniśmy poprawić wyjście: nie zawsze potrzeba drukować wszystkie cyfry. Poza tym nie mamy komunikatu, że program skończył obliczenia z powodu wykonania 50 iteracji.

```
LIST

10 A=0
20 B=1
30 C=2
32 I=0
33 M=50
34 X=0
35 EPS=1/1000000
36 Y=X
40 A=A+B
50 B=SQR(1+A*A)
60 C=C+C
70 X=3*C/(A+B+A)
71 I=I+1
80 IF ABS(X-Y)>EPS AND I<M THEN GOTO 36
90 IF I=M THEN PRINT "WYKONANO 50 ITERACJI"
100 X0=INT(X/EPS+0.5)*EPS
110 PRINT "X=";X0
READY.
```

Tak więc prosta procedura nieco się rozrosła (wydruk 4) — za to spokojnie możemy umieścić ją jako przetestowany blok naszego superprogramu!

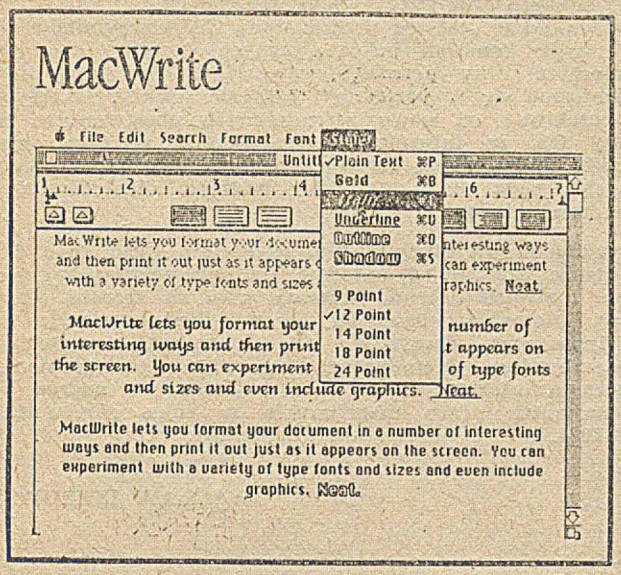
Pozostaje pytanie: dlaczego ten program daje właśnie taki rezultat. Nie wiemy tego — może Czytelnicy potrafią zrekonstruować algorytm?

**JAKUB TATARKIEWICZ**

[1]) Przykład zaczerpnięty z POPULAR COMPUTING, maj 1984

dokończenie ze str. 12

— strzałki z prawej strony ekranu umożliwiają wybór kierunku przeglądania tekstu (ang. scrolling), a biały prostokąt poruszający się wzdłuż szarej kolumny wskazuje bieżący wiersz tekstu.




Fot. 3. Dokument „Untitled"; wokół ekranu widoczne środki do wybierania układu tekstu, w okienku — zawartość listy Style [1]

* * *

Analizując rozwiązanie konstrukcyjne i oprogramowanie MACINTOSHA należy stwierdzić, że jest to pierwszy na światowym rynku mikrokomputer tej klasy. (Cena zestawu bazowego ok. 2400 dol.) Elastycznością oprogramowania i łatwością komunikowania się z użytkownikiem nowy wyrób APPLE znacznie przewyższa ewentualnych konkurentów (firma podaje, iż wystarcza 15 min., aby nauczyć się obsługi MACINTOSHA). Choć wprowadzono w nim wiele ciekawych innowacji technicznych i technologicznych, w małym stopniu wykorzystano zdolność adresową procesora centralnego (MC 68000 może adresować do 16 MB pamięci), a co gorsze — konstrukcja MACINTOSHA nie pozwala użytkownikowi dołączyć dodatkowych modułów pamięci.

Niektóre symbole graficzne (ang. icons) są źle dobrane i trudno zorientować się od razu, co reprezentują. Trzeba jednak zaznaczyć, że sam dobór jest na ogół zadaniem bardzo trudnym. Myszka jest niewątpliwie ciekawym elementem sprzętu, lecz chyba mocno przereklamowanym. W czasie pracy łatwo o przypadkowe poruszenie myszką i w konsekwencji — przesunięcie kursora. Użytkownicy są już przyzwyczajeni do sterowania ruchem kursora z klawiatury, bez odrywania od niej rąk w celu poruszania myszką. Ponadto barwne monitory stają się standardem przemysłowym i chyba jest błędem producenta wykorzystanie w MACINTOSHU monitora monochromatycznego, nawet jeśli w ten sposób znacznie obniżono koszt komputera.

Ten rodzaj oprogramowania, jaki tu zastosowano, mógłby być ciekawą ofertą na naszym rynku, gdzie potencjalni użytkownicy wykazują słabą znajomość klasycznego oprogramowania i nikłą umiejętność posługiwania się nim. Niewątpliwie warto się nad tym głębiej zastanowić, gdyż produkowane w Polsce systemy mikrokomputerowe są wyposażone co najwyżej w kompilatory języków wysokiego poziomu. Brak dobrego oprogramowania do przetwarzania tekstów i grafiki (na Zachodzie żaden szanujący się wytwórca nie sprzedaje systemów bez takiego oprogramowania) w istotny sposób ogranicza wykorzystanie mikrokomputerów oraz odstrasza użytkowników nie posiadających umiejętności programowania. Należy mieć nadzieję, iż wcześniej czy później podejmie się takie próby i doczekamy się rozwoju prawdziwego rynku oprogramowania.

LITERATURA

[1] Miller M. J.: Introducing Lisa-like technology in a personal package. POPULAR COMPUTING, vol. 3, No. 5, 1984
[2] Shelton J.: The mice. SOFTOLK, vol. 4, 1984
[3] Tommervik A.: Another winner from Apple. SOFTOLK, vol. 4, 1984
[4] Was leistet der Macintosh. CHIP, Nr 5 1984.

*Dydaktyka*

# Nowy dział

Studia informatyczne (specjalność „Budowa i oprogramowanie maszyn matematycznych") ukończyłem ponad dziesięć lat temu. Zakres otrzymanej wtedy wiedzy obejmował podstawy organizacji i elementy oprogramowania ówczesnych nowoczesnych komputerów (a przepraszam — maszyn matematycznych, bo taką wówczas nosiły nazwę). Na laboratoriach sprzętowych korzystaliśmy z układów logicznych zrealizowanych w technice tranzystorowej. Teoretycznie opanowaliśmy zasady działnia układów serii TTL. Zaś same te układy były niedostępne — z wyjątkiem jednego, ale nie użytkowanego przez studentów, jako że mogliby go spalić. Poznawaliśmy organizację maszyn UMC-1, UMC-10 i ODRA 1204. Programowaliśmy w języku wewnętrznym UMC-10, kodzie W-20 bazujących na składaniu rozkazów z mikrooperacji. O szybkości działania tych maszyn niech świadczy fakt, że obliczenie wartości prostego równania różniczkowego metodą Rungego-Kutty trwało ponad trzy godziny. Program śledziło się zaś dzięki wyświetlanym zawartościom rejestrów adresowych w trakcie jego działania. Z języków wysokiego poziomu poznaliśmy język ALGOL 60 oraz częściowo FORTRAN, niestety — oba „na sucho". Większą konfigurację z taśmami magnetycznymi zobaczyłem dopiero podczas pierwszych lat pracy. Jako pracę dyplomową wykonałem wspólnie z kolegą interpreter języka BASIC na ODRZE 1204.

Od tego czasu informatyka nauczana na studiach zmienia się dynamicznie z szybkością niespotykaną w innych dziedzinach. Rozwój informatyki na świecie i — nieco wolniejszy — w kraju był i jest uwzględniany w ciągle modyfikowanych programach studiów. Praktycznie nie zdarzyło się, aby dwa kolejne roczniki miały dokładnie takie same przedmioty. Pojawiały się nowe wykłady, laboratoria, zmieniał się sprzęt oraz użytkowane komputery (niestety, nie należą one do najnowocześniejszych). Obecnie, pomimo ciągłej pracy na uczelni i stałego dokształcania się, z dużą trudnością zdałbym egzaminy z niektórych prowadzonych dzisiaj przedmiotów.

W podobnej sytuacji znajdują się w kraju wszyscy informatycy (oprócz oczywiście najmłodszej generacji). Informatycy ci, pracujący i nabierający doświadczenia w użytkowaniu i projektowaniu konkretnych systemów informatycznych, mają coraz większe luki w wiadomościach z innych działów. Informatyka, jak żadna inna dziedzina, wymaga więc stałego pogłębiania i odświeżania wiedzy.

Z drugiej strony — w ciągu tych kilkunastu lat historii nauczania informatyki na uniwersytetach, politechnikach i innych uczelniach programy studiów ewoluowały różnorodnie. Powstało też wiele nowych specjalności informatycznych, wyodrębniających się organizacyjnie z wydziałów elektronicznych, elektrycznych, matematycznych i innych. W podstawowym zakresie kierunku „Informatyka" można wyróżnić uniwersytecką informatykę teoretyczną i politechniczną — techniczną (praktyczną?), przy czym podział ten nie

jest jednoznaczny. Istnienie takiego podziału nie zawsze jest korzystne dla rozwoju i chyba jedynie chemia jest w podobnej sytuacji. Ostatnio można też zauważyć trend wydzielania z informatyki zagadnień związanych tylko z mikrokomputerami. Pojawiają się też specjalności informatyczne w powiązaniu z innymi dziedzinami nauki i techniki — mechaniki, automatyki, ekonomiki, organizacji itp. W rezultacie obecni absolwenci, otrzymując tytuł informatyka, a często elektronika czy elektryka, różnią się między sobą zasobem i zakresem nabytej na studiach wiedzy informatycznej.

Dlatego też dla poznania aktualnego stanu nauczania informatyki na studiach, proponujemy reaktywowanie istniejącego przed kilku laty na łamach INFORMATYKI działu („Nauczanie i Szkolenie") pod nową nazwą „Dydaktyka". Tematyka tego działu będzie obejmowała prezentacje programów studiów w różnych uczelniach oraz profile zawodowe ich absolwentów. Dział ten jest przeznaczony dla wymiany informacji dydaktycznych między pracownikami uczelni. Interesujące tu będzie skonfrontowanie metodyk prowadzenia wybranych zajęć teoretycznych i praktycznych, przejrzenie wyposażenia laboratoriów oraz streszczeń najciekawszych prac magisterskich i doktorskich. Zwrócimy też uwagę na programy studiów podyplomowych oraz kursów dokształcających.

„Dydaktyka" jest też przeznaczona dla kierowników różnego typu komórek informatycznych w ośrodkach obliczeniowych i w przedsiębiorstwach, którzy będą mogli zorientować się w poziomie i zakresie wiedzy, jakiej mogą oczekiwać od nowo zatrudnionych absolwentów. Sądzimy też, że informacje zamieszczane w tym dziale, a także w pozostałej części czasopisma, choćby w „Samotestach", będą dla czytelników bodźcem dla odświeżenia posiadanej wiedzy.

Truizmem jest stwierdzenie, że poziom treści tego działu będzie zależny od autorów prezentujących osiągnięcia swoje i swoich uczelni. Liczymy więc na szeroką współpracę. Interesujące będą dla nas wszelkie propozycje, także dyskusyjne. Zaczynamy od prezentacji sprawozdania z Seminarium Metodycznego poświęconego problemom nauczania informatyki, a także z przyznania nagród PTI za najlepsze prace magisterskie w roku akademickim 1983-1984.

WACŁAW ISZKOWSKI

# Rozwój mikroelektroniki a nauczanie

Po kilkuletniej przerwie, dydaktycy nauczający informatyki spotkali się na VII Seminarium Metodycznym Informatyki, zorganizowanym tym razem przez Instytut Informatyki Politechniki Gdańskiej (22—26 września 1984). Uczestnikami seminarium byli przedstawiciele uczelni kształcących informatyków — politechnik: Gdańskiej, Poznańskiej, Warszawskiej i Wrocławskiej, uniwersytetów: Jagiellońskiego i Warszawskiego oraz Akademii Górniczo-Hutniczej.

Podstawowym tematem spotkania była prezentacja aktualnych programów studiów w zakresie specjalności „Informatyka", które ze względu na rozwój tej dziedziny muszą się zmieniać prawie co roku. W prezentowanych programach było widać obciążenie (na szczęście już malejące) specjalistycznymi przedmiotami wydziałów, na których pojawiła się ta specjalność. W rezultacie młodsi, nierzadko zapaleni studenci mają często kontakt z prawdziwą informatykę dopiero po dwóch, trzech latach studiów. Istotne były więc też tutaj pytania, ile i jakiej matematyki oraz fizyki należy nauczać przyszłych inżynierów informatyków. Matematyka jest potrzebna do zmniejszenia dystansu między informatyką teoretyczną a praktyczną. Fizyka zaś jest istotna w poznawaniu i tworzeniu nowych technologii, mających znaczenie w rozwoju mikroukładów.

Innym problemem było określenie profilu absolwenta. Czy należy kształcić dokładnie wyszkolonych specjalistów w wąskich dziedzinach sprzętowych i oprogramowania, czy też mniej dokładnie, ale uniwersalnie. Unikając jednoznacznej odpowiedzi na to trudne pytanie, wprowadzono do programów przedmioty fakultatywne, wybierane — zależnie od uczelni — w sposób dowolny, sterowany lub ograniczony przez studentów. Ta grupa przedmiotów pozwala też na wcześniejsze przygotowanie wykładowców do nowych zajęć, które w przyszłości będą stanowiły podstawowy trzon programów.

Na tle standardowych programów pojawiły się nowe propozycje, związane przede wszystkim z ekspansją mikroprocesorów. Zaproponowano specjalność „Mikroinformatyka", obejmującą całość zagadnień — począwszy od analizy modelowej struktur systemów, przez projekty i realizacje jedno- i wieloprocesorowych systemów, aż do oprogramowania standardowego i aplikacyjnego mikrokomputerów. W sumie, celem tej specjalności byłoby nauczanie inżynierii projektowania specjalizowanych cyfrowych układów sterowania.

Podział informatyki na mikro i makro (?) może się wydawać dyskusyjny. Wiadomo przecież, że stary już FORTRAN, COBOL czy PL/I będą wykorzystywane również w XXI wieku. Będą też istniały duże komputery. Przy tej okazji wypłynęło pytanie, czy powstanie Piątej Generacji maszyn zmieni sposób kształcenia informatyków w Polsce. Pierwszą odpowiedzią było — nie. Trzeba jednak przynajmniej mówić studentom o tych problemach, jak też i o nieosiągalnych dla nas superkomputerach. Zwrócono też uwagę, że istnieje już potrzeba rozróżnienia informatyki tworzącej narzędzia informatyczne (ang. computer science) od informatyki zastosowań (ang. computational science — informatyzacja).

Przy okazji przedstawiono park komputerowy, jakim dysponują uczelnie. A więc — dość już stare ODRY 1304, 1305, 1325, trochę młodszy R-32, kilka nierzadko psujących się SM3 i SM4, MERA 400 i różnego rodzaju sprzęt mikrokomputerowy od ZX80 do PSPD 90. W stosunku do potrzeb wyraźny jest brak nowoczesnego sprzętu oraz oprogramowania, co jest odczuwalną barierą w efektywnym nauczaniu współczesnej informatyki i jej zastosowań. Dało się też odczuć, że wiele uzyskanego sprzętu jest bardziej wynikiem zaradności poszczególnych instytutów, niż planową działalnością Ministerstwa.

Spotkanie można podsumować następującymi wnioskami.

1. Koncepcja kształcenia inżynierów informatyki jest — na miarę aktualnych możliwości — nowoczesna. Daje się już jednak odczuć istnienie stale powiększającej się luki informacyjnej pomiędzy wiedzą dostępną w kraju, a istniejącą w ojczyźnie superkomputerów czy komputerów Piątej Generacji. Konieczne jest więc szersze nawiązywanie kontaktów z zagranicą i zdobywanie dla uczelni literatury, sprzętu i oprogramowania.
2. Należy dążyć do wyodrębnienia samodzielnego kierunku kształcenia informatyki o logicznie ułożonym, w pełni informatycznym programie studiów — już od pierwszego semestru nauki. Jednocześnie nie można zmniejszać limitów przyjęć na te studia. Należy też zastrzec używanie tytułów mgr inż. informatyki i mgr informatyki tylko dla absolwentów w pełni informatycznych kierunków studiów.

3. Nauczanie informatyki nie może się ograniczać tylko do specjalistów, ale powinno być udziałem pozostałych kierunków technicznych i humanistycznych oraz szkolnictwa stopnia średniego. Pożądany jest przy tym nadzór nad tym nauczaniem ze strony informatyków. Konieczne jest też kształcenie kadry nauczającej informatyki na tych kierunkach oraz w szkolnictwie średnim i zawodowym.

4. Właściwe kształcenie w zakresie mikrokomputerów, które jest nieodzowne dla informatyków i specjalistów z wielu innych dziedzin, jest uwarunkowane prowadzeniem przez odpowiednie instytuty uczelniane badań, umożliwiających nadążanie za światowym postępem. Kadra nauczająca jest przygotowana do prowadzenia takich badań, pod warunkiem zapewnienia jej odpowiedniego warsztatu naukowego. Konieczne są więc działnia mające na celu import najnowocześniejszego sprzętu, oprogramowania i literatury naukowej. Działania te powinny być koordynowane — dla uzyskania największej efektywności wykorzystania zainwestowanych środków. W związku z tym uznano za celowe uruchomienie w ramach badań resortowych problemu „Mikrokomputery w nauczaniu".

W. I.

# Konkurs PTI na najlepsze prace dyplomowe z informatyki

Zakończył się I Ogólnopolski Konkurs Polskiego Towarzystwa Informatycznego na najlepsze prace dyplomowe z dziedziny informatyki, zainicjowany i zorganizowany przez Koło PTI we Wrocławiu. Komisja Konkursowa PTI w składzie: doc. Czesław Daniłowicz (przewodniczący), prof. Andrzej Blikle, dr Piotr Kociatkiewicz, doc. Maciej Sysło i prof. Władysław M. Turski — ustaliła, że do konkursu będą dopuszczone prace obronione w wyznaczonym przez uczelnie terminie w okresie 1 października 1983 — 30 września 1984. Prace bronione później będą mogły wziąć udział w następnym konkursie.

Do konkursu dopuszczono 21 prac magisterskich. Plon konkursu jest więc obfity, a poziom zgłoszonych prac — zdaniem Komisji Konkursowej i powołanych przez nią recenzentów — dobry. Zwracał uwagę duży udział prac o charakterze wybitnie praktycznym. Zgłoszone na konkurs prace zostały wykonane w uczelniach Warszawy, Wrocławia, Katowic, Szczecina, Łodzi i Poznania.

Na posiedzeniu Zarządu Głównego PTI 10 grudnia 1984 Komisja ogłosiła wyniki konkursu i rozdała nagrody.

Nagrodę I, w wysokości 16 tys. zł, otrzymała **mgr Wiesława Nizioł** za pracę pt. **„Metody projektowania i specyfikacji algorytmów systolicznych"** (Instytut Informatyki Uniwersytetu Warszawskiego, opiekun dr Wojciech Rytter).

Nagrody II nie przyznano.

Dwie równorzędne nagrody III, po 8 tys. zł, otrzymali: **mgr inż. Janusz Machowski** za pracę pt. **„Węzeł teledacyjny sieci komputerowej z wykorzystaniem mikrokomputera"** (Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, opiekun dr inż. Ireneusz Jóźwiak) oraz **mgr inż. Krzysztof Szwed** za pracę pt. **„Uniwersalny asembler dla mikrokomputerów"** (Instytut Informatyki Politechniki Warszawskiej, opiekun mgr inż. Ryszard K. Kott).

Komisja Konkursowa przyznała ponadto cztery wyróżnienia, po 4 tys. zł: **mgr. Zdzisławowi Jarzębowskiemu** za pracę pt. **„Pakiet makrodefinicji ogólnego przeznaczenia"** (Instytut Informatyki Uniwersytetu Wrocławskiego, opiekun dr Ewa Gurbiel), **mgr. inż. Leszkowi Misiurze** za pracę pt. **„Adaptacja drukarki DZM 180 do grafiki o dużej rozdzielczości z użyciem mikrokomputera ZX 81"** (Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, opiekun dr inż. Jan Łakomski), **mgr. inż. Jerzemu Sasowi** za pracę pt. **„Problemy zastosowań metody VDL w implementacji języków programowania"** (Centrum Obliczeniowe Politechniki Wrocławskiej, opiekun dr inż. Zbigniew Huzar) oraz **mgr. inż. Andrzejowi Szewczukowi** za pracę pt. **„Quasi-równoległość i symulacja w PASCALU — język SIMPAS i jego implementacja w systemie OS/JS"** (Studium Zaoczne Matematyki Uniwersytetu Warszawskiego, opiekun dr Paweł Gburzyński).

**BARBARA OSUCHOWSKA**
Rzecznik prasowy PTI

# 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*

**W czasach gdy ochrona danych powinna być wyjątkowo solidna — jest ona — wskutek postępu techniki przetwarzania danych — ciągle niedostateczna. Autor artykułu, profesor Uniwersytetu we Frankfurcie n. Menem uważa, że środki zabezpieczające powinny być powiązane bezpośrednio z komputerem — przez uwzględnienie ich w konstrukcji maszyn cyfrowych. Tłumaczenie artykułu prof. Spirosa Simitisa, będącego jednocześnie komisarzem ds. ochrony danych w Hesji (RFN) przedstawiamy za kwartalnikiem „FORUM. Conseil de l'Europe" nr 3/1984, wydawanym przez Radę Europejską w Strasburgu. (Red.)**

# Jak zapewnić ochronę danych?

Ochrona danych — od czasów pierwszej regulacji ustawowej dokonanej w Hesji w 1970 roku do projektów ustaw złożonych ostatnio w Szwecji, Włoszech i Portugalii — przebyła długą i miejscami bardzo trudną drogę. Podobnie jak uprzednio, również dzisiaj sprawa ta jest przedmiotem kontrowersji. Różnice poglądów co do konkretnych sposobów ochrony danych są wyraźne — zarówno gdy rzecz w teoretycznym określeniu tzw. czułych danych (franc. données sensibles), jak i w zrównaniu osób prawnych z osobami fizycznymi — by przywołać tylko te dwa przykłady. Jednocześnie wszyscy są zgodni co do konieczności uregulowań prawnych.

Rezolucja Rady Europejskiej z 28 stycznia 1981 szczególnie wyraźnie zaakcentowała fakt, że problem ochrony danych nie jest tylko szczególnym zagadnieniem określonego porządku prawnego, lecz jednym z warunków funkcjonowania współczesnej demokracji. W każdym bowiem przypadku, kiedy obywatel nie wie lub nie może wiedzieć kto gromadzi o nim informacje i w jakich okolicznościach informacje te są zbierane, dochodzi do zawieszenia jego podstawowych praw obywatelskich. Brak kontroli, a także utajnienie procesu przetwarzania informacji o osobach — prowadzą nieuchronnie do manipulacji i nadużyć.

### Odpowiedź polityczna

To paradoks, że właśnie w chwili gdy ochrona danych powinna być w pełni uregulowana aktami prawnymi, przeżywa ona poważny kryzys. Żadne ustawodawstwo — czy to już obowiązujące, czy dopiero przygotowywane — nie może być właściwie zrozumiane, jeżeli nie uwzględni się technicznego kontekstu sprawy. Akty prawne zapewniające ochronę danych wypływają bowiem ze zmian spowodowanych przez całkowicie nowe sposoby przetwarzania danych.

Od chwili, gdy stało się możliwe gromadzenie niemal nieograniczonej liczby danych, ich wyszukiwanie w dowolnym momencie, a ponadto — ich kojarzenie pod dowolnymi względami, ograniczenia prawne stały się niezbędne. W tej sytuacji prawną ochronę danych należy widzieć jako **polityczną odpowiedź** na zautomatyzowane przetwarzanie danych.

Związek między ochroną danych a rozwojem techniki ich przetwarzania implikuje konieczność stałego poszukiwania adekwatnych rozwiązań prawnych. Przepisy dostosowane do pewnego stadium rozwoju techniki komputerowej mogą normować zagadnienie przetwarzania danych tylko tak długo, jak długo sposób przetwarzania mieści się w ramach tego stadium. Inaczej mówiąc — każda zaawansowana technika stawia pod znakiem zapytania realność ochrony danych. Stąd również i ustawodawca zachodnioniemiecki nie przestał traktować swych dotychczasowych dokonań jedynie jako pierwszego etapu drogi. Jeszcze bardziej logiczna jest reakcja ustawodawcy islandzkiego: ustawę dotyczącą tych spraw uchwala się jedynie na określony czas i z zastrzeżeniem bieżącej jej nowelizacji. Ochronę danych traktuje się tam więc jako nieustanny proces praktykowania (franc. un processus d'apprentissage continu), którego ustawodawca nie może swoimi decyzjami zakłócać.

### Sterowanie zachowaniem się jednostki

Nie ulega wątpliwości, że techniczne uwarunkowania przetwarzania informacji zmieniły się w sposób zasadniczy i to co najmniej w dwóch punktach. Jeżeli porówna się obecną sytuację z latami sześćdziesiątymi i siedemdziesiątymi, łatwo stwierdzić, że dysponujemy dzisiaj bardzo zróżnicowanym i efektywnym instrumentarium gromadzenia i wyszukiwania informacji.

Systemy interaktywne, związane z wykorzystaniem monitorów ekranowych, są tego szczególnym przykładem. Pozwalają one wykonywać na monitorze wiele operacji życia codziennego: od zleceń zakupów dokonywanych na podstawie katalogów sprzedaży wysyłkowej — do rezerwacji biletów i operacji bankowych. Jednostka, za pośrednictwem monitora, ujawnia część swoich zwyczajów, preferencji i oczekiwań, a każda z tych reakcji zostaje w sposób precyzyjny utrwalona. Podobnie ma się rzecz z systemami „tele-zleceń" (franc. les systemes de télécommande), pozwalającymi odczytywać na odległość zużycie energii elektrycznej lub wody, doglądać chorych i dzieci, regulować ogrzewanie i wykonywać wiele innych zajęć domowych. Skoncentrowanie informacji pozwala dokładnie odtworzyć dzienny rozkład zajęć każdej osoby.

Nigdy dotąd nie było tak dużych możliwości kreowania mniej lub bardziej dokładnego wizerunku człowieka; nigdy też warunki nie sprzyjały w takim stopniu ukierunkowywaniu jego zachowania. Tymczasem żadne z obowiązujących ustawodawstw nie pozwala na przeciwdziałanie temu stanowi rzeczy. Monitor ekranowy i system „tele-zleceń" są instrumentami, które pojawiły się na marginesie regulacji prawnych i nadają przetwarzaniu danych osobistych całkowicie nowy wymiar.

Ponadto, ustawy dotyczące ochrony danych opierają się na idei scentralizowania informacji uzyskiwanej w wyniku automatycznego przetwarzania danych. Punktem wyjścia dla przyjętych przez te ustawy rozwiązań prawnych były duże banki danych, przeważnie państwowe; z tego też powodu przewidziane w tych aktach formy kontroli dostosowane są do modeli takich banków. Tymczasem — począwszy od lat osiemdziesiątych — poszczególne, coraz liczniejsze operacje przeprowadza się w ośrodkach obliczeniowych. Rozproszone przetwarzanie danych (ang. distributed data processing), to podstawa autonomicznych (ang. stand alone) ośrodków informatycznych. Dzięki rozwojowi minikomputerów nie ma żadnych trudności w tworzeniu autonomicznych ośrodków informatycznych, np. w szpitalach. Zresztą, tradycyjne urządzenia biurowe — od maszyny do pisania po dalekopis, przez telefon, telekopiarkę i kalkulator — są również doskonalone i coraz częściej wiązane z wielofunkcyjnymi terminalami. Korzyść, jaka z tego wynika, to zwiększenie elastyczności procesów przetwarzania danych.

Proces informatyczny może być realizowany z dowolnego stanowiska pracy; niezbędne po temu sprzężenia mogą być tworzone bądź likwidowane w identyczny sposób. Korzyść ta ma wszakże swą przeciwwagę, która w nie mniejszym stopniu zasługuje na podkreślenie: kontrola przetwarzania staje się coraz trudniejsza, a w końcu niemożliwa. Tradycyjne środki kontroli tracą swoje znaczenie z chwilą, gdy ślady przetwarzania mogą być zatarte i to tym prędzej, że proces przetwarzania jest zdecentralizowany, a posługiwanie się sprzętem informatycznym wymaga w coraz mniejszym stopniu wiedzy technicznej. Inaczej mówiąc — tradycyjne sposoby przetwarzania podzielają los środków kontroli będących do niedawna w użyciu. Jedne i drugie skazane są na wymarcie.

### Elastyczność regulacji prawnej

Należy jednak przestrzec przed możliwym nieporozumieniem: warunki

przetwarzania i kontroli powszechnie obecnie akceptowane mogą tracić swą użyteczność; w najmniejszym jednak stopniu nie dezaktualizuje to problemu ochrony danych. Traci walor jedynie pewna koncepcja ochrony, która na pewnym etapie odpowiadała rozwojowi technologii przetwarzania danych. Trzeba więc mieć nadzieję, iż ustawodawca wykaże elastyczność równą tej, z jaką zmienia się technologia przetwarzania. Problem, przed którym obecnie stoimy nie polega zatem na załamaniu się systemu ochrony danych w ogóle, ale na znalezieniu innego rozwiązania prawnego, bardziej adekwatnego do nowej techniki przetwarzania.

Nie poddajmy się wszakże złudzeniom: o ile konieczność nowych unormowań prawnych nie budzi żadnych wątpliwości, o tyle nie można zasadnie twierdzić, że mamy już pełną jasność co do podstawowych kierunków przyszłej regulacji. Rysuje się jedynie jej ogólny kształt. Do chwili obecnej ustawy o ochronie danych dotyczyły głównie ośrodka zbierającego dane i abstrakcyjnie określały warunki przetwarzania oraz ogólne procedury kontrolne; przyszłe akty — przeciwnie — **powinny dotyczyć bezpośrednio maszyny cyfrowej**, aby określić, czy w konsekwencji danego urządzenia zastosowano elementy techniczne uwzględniające normy ochronne, a jeżeli tak, to w jakim stopniu. W ten sposób normy te mogłyby wyrażać się w substancji rzeczy, dzięki przepisom technicznym dotyczącym konstrukcji komputerów. W pełni rozwinięta ochrona tego typu weszłaby na drogę, która — zdaniem ustawodawcy — wydaje się narzucać np. przy ochronie naturalnego środowiska człowieka. Tymczasem — powtórzmy to jeszcze raz — rozważanie tego zagadnienia zostało dopiero zapoczątkowane; a przecież nie można zbyt długo zwlekać z jego pogłębioną analizą. Jeżeli prawdą jest, że ochrona danych jest jednym z warunków funkcjonowania współczesnego społeczeństwa, to jego zdolność przetrwania będzie się mierzyć zdolnością dostosowywania prawnych unormowań przetwarzania danych do obiektywnych uwarunkowań.

Tłumaczyła **ZOFIA SZEWC**

# Informatyka a suwerenność

Dominacja amerykańska w zakresie produkcji i zastosowań komputerów oraz wyzwanie japońskie — słynne hasło Piątej Generacji — wywołują poważne obawy działaczy latynoamerykańskich. Grupa redakcyjna MANIFESTU KALIJSKIEGO, podpisanego w maju ub. r. przez ponad 60 czołowych osobistości 15 krajów Ameryki Południowej i Środkowej, dopatrzyła się bowiem wynaturzonej koncentracji postępu informatycznego przez nieliczne przedsiębiorstwa ponadnarodowe.

Można się spierać, czy te potężne korporacje kontrolują rzeczywiście aż 90% światowego rynku informatycznego; i czy rzeczywiście 95% światowego parku komputerowego przypada tylko na USA i Japonię, a także — czy 98% wszystkich nakładów na badania i rozwój informatyki finansowanych jest z wyrachowaniem przez te dwa mocarstwa przemysłowe. Bo takie właśnie liczby padały podczas trzydniowej konferencji na temat INFORMATYCZNEJ STRATEGII INTEGROWANIA REGIONU LATYNOAMERYKAŃSKIEGO, odbytej w maju 1984 w mieście Cali, silnym kolumbijskim ośrodku przemysłowym. Ale gdyby nawet uwzględnić to, co się dzieje z informatyką w krajach RWPG — względna wymowa liczb procentowych byłaby tak samo zła: brak tego, co można nazwać suwerennością informatyczną, której nie należy odnosić do pojedynczych krajów, ale całego subkontynentu.

Oto niektóre osobistości, które zainicjowały wydanie Manifestu Kalijskiego: prezes brazylijskiego towarzystwa producentów komputerów i urządzeń peryferyjnych, pełnomocnik rządu chilijskiego ds. informatyki, wiceprezes centrum latynoamerykańskiego humanistyki i informatyki w Kolumbii, wiceprezes kubańskiego instytutu systemów zautomatyzowanych i techniki obliczeniowej, generalny dyrektor ds. polityki informatycznej meksykańskiego urzędu statystycznego, nikaraguański dyrektor generalny departamentu informatyki w resorcie planowania, generalny dyrektor międzynarodowego biura informatycznego w Rzymie, przedstawiciel kolumbijskiego stowarzyszenia użytkowników systemów komputerowych. Właśnie z racji swych czynności zawodowych osoby te odczuwają niepokojące pogłębianie się przepaści oddzielającej „kolonie informatyczne" subkontynentu latynoamerykańskiego i Wysp Karaibskich od ośrodków komputerowych, w których zlokalizowane są bazy danych dla tych regionów. Dane o zasobach naturalnych i życiu gospodarczym są bardzo często przetwarzane poza terytoriami zainteresowanych krajów. W ten sposób mocarstwa informatyczne mogą mieć bardziej szczegółowe i aktualne informacje o danym kraju latynoamerykańskim niż jego władze.

Na razie działacze latynoamerykańscy skonstatowali zgodnie niepokojące objawy i niekorzystne rokowania na przyszłość — stwierdzając generalną konieczność podjęcia radykalnych kroków zapobiegawczych. Ale nie wyszli jeszcze poza ogólniki mówiące o niezbędności podejmowania wspólnych wysiłków, a także koordynacji wprowadzania metod informatycznych i stworzenia obiektywnych kryteriów rozwojowych. Zwykłe porównania prowadzą bowiem do wręcz obezwładniających wniosków. Jeżeli mówi się o ponad dziesięcioletnim opóźnieniu informatycznym Europy Zachodniej w stosunku do czołówki amerykańsko-japońskiej, to na ile lat należy szacować opóźnienie krajów Trzeciego Świata? Tymczasem nie trzeba być mędrcem, aby zauważyć, że wzorzec amerykański nie jest jedynym modelem rozwojowym — ani też najbardziej oszczędnym. Inaczej mówiąc — kraje odleglejsze od czołówki informatycznej muszą sobie wypracować jakąś własną strategię „suwerennościową", aby nie dać się wciągnąć w obłędny wyścig nowych technologii informatycznych i nie dać się oplątać tylko samymi końcówkami odległych systemów komputerowych.

Dotychczas nikomu — jak widać — nie udało się sformułować rozsądnego, skutecznego programu komputeryzacji — owej koncepcji minimum, dającej szansę zaspokojenia podstawowych potrzeb informatycznych zainteresowanych krajów. Gorzej — potrzeby te nie zostały jeszcze ujęte ilościowo. Z drugiej zaś strony, katastrofalne zadłużenie subkontynentu latynoamerykańskiego nie daje nadziei na przezwyciężenie opóźnienia informatycznego. Mimo wszystko, inwestycje informatyczne są tak łakomym kąskiem rozwojowym, że trudno sobie wyobrazić, aby jakikolwiek koncern kapitalistyczny zgodził się uszczuplić swe perspektywiczne zyski, oddając prawa do korzystania z nowych osiągnięć.

Być może jednak utworzony na wspomnianej konferencji KLUB CALI doprowadzi przynajmniej do stworzenia czegoś w rodzaju wspólnego funduszu rozwojowego badań informatycznych.

**Oprac. A.B.E. na podst.**
**IBI NEWSLETTER 16/1984**

## Mikrokomputery w automatyce i technice systemów

Największym obszarem zastosowań systemów mikrokomputerowych jest automatyka i ogólnie pojęte systemy sterowania. Zaprezentowanie aktualnego rozwoju tej dziedziny było celem konferencji naukowo-technicznej „Mikrokomputery w automatyce i technice systemów", która odbyła się w dniach 18—24 września 1984 we Wrocławiu. Organizatorem konferencji był Instytut Sterowania i Techniki Systemów Politechniki Wrocławskiej przy współudziale Komitetu Automatyki PAN, Polskiego Komitetu ds. Pomiarów i Automatyki NOT, Instytutu Komputerowych Systemów Automatyki i Pomiarów z Wrocławia, Zakładów Elektronicznych ELWRO oraz Przedsiębiorstw AMEPROD i NOWATECH.

Konferencja zgromadziła około czterystu uczestników z różnych krajowych ośrodków nauki i przemysłu. Na jedenastu sesjach zaprezentowano 150 opracowań w formie referatów, komunikatów i sesji plakatowych. Prace zostały wydrukowane w dwóch tomach materiałów konferencji (przygotowywany jest tom trzeci grupujący referaty plenarne i nie opublikowane wcześniej).

Duża część prac dotyczyła ogólnych zagadnień projektowania i realizacji mikrokomputerów jednoprocesorowych 4-, 8-, 16-bitowych, oraz wieloprocesorowych systemów rozproszonych. W tej grupie część opracowań prezentowała też proste sieci lokalne, metody dołączania magistrali i urządzeń zewnętrznych, projekty systemów uruchomieniowych oraz implementacji oprogramowania podstawowego i wspomagającego. Prezentowane rozwiązania były przeważnie standardowe — dostosowywane do konkretnych praktycznych zastosowań. Pojawiały się też nowinki, np. komputery drzewiaste.

Osobny dział stanowiły prace omawiające testowanie, diagnostykę i symulację działania elementów oraz całych systemów mikrokomputerowych.

W dziale zastosowań w automatyce prezentowano realizację regulatorów cyfrowych, algorytmy sterowania, sterowanie procesami technologicznymi, tunelowymi, czy chemicznymi w zakładach przemysłowych. Pokazano też wykorzystanie mikrokomputerów w automatyzacji energetyki, nawigacji i sterowaniu ruchem ulicznym. Były też dwie prace dotyczące robotyki.

Osobny dział był poświęcony urządzeniom pomiarowo-kontrolnym, wykorzystywanym w sterowaniu eksperymentami, pomiarami objętości i szumów, badaniu właściwości materiałów oraz analizie widm elektronowych.

Spośród zastosowań telekomunikacyjnych przedstawiono realizację sterowania małych systemów komutacyjnych, zespołów aparatów telefonicznych i sieci teleksowych.

W dziale zastosowań w biomedycynie omawiano podstawy rozpoznawania mowy, problemy fizjologii człowieka (ergonomia) oraz systemy automatycznych analiz i nadzorów w medycynie. Pokazano też wykorzystanie informatyki w przetwarzaniu danych medycznych.

W dziedzinie dydaktyki zaprezentowano laboratoria mikrokomputerowe, przeważnie powiązane z systemami CAMAC.

Podstawowym sprzętem wykorzystywanym do realizacji omawianych systemów były układy serii INTEL z procesorami 8080, Z80 i czasem już 8086 — wraz z elementami towarzyszącymi. Dają przy tym znać ograniczenia bazy elementowej i dostępności niektórych układów. Metody zdobywania tych elementów pozostały tajemnicą realizatorów. W zakresie gotowych systemów wykorzystywano i modyfikowano oraz przystosowywano do konkretnych potrzeb mikrokomputery typu ZX81 oraz PSPD-90.

Niektóre referaty zrodziły poczucie, że dany zespół nigdy by się nie zajął projektowaniem i samodzielnym zrealizowaniem sytemu mikrokomputerowego, jeżeli możliwe byłoby jego zakupienie. Celem działania takiego zespołu było bowiem wykorzystanie systemu do konkretnego zastosowania.

Widoczne były prywatne firmy komputerowe, które na wystawie prezentowały swoje produkty-błyskotki, a także oferowane oprogramowanie — w trakcie sesji. Uczestnicy zaś wykazywali duże zainteresowanie, czasem wprost żądzę zdobycia jakiegokolwiek systemu za każdą cenę. Niestety, i te firmy borykają się z typowymi problemami — płytki, elementy, urządzenia zewnętrzne... Pesymistyczne jest też stwierdzenie, że właściwie nie pojawili się przedstawiciele państwowego przemysłu.

Podsumowując — konferencja na dość wysokim poziomie zaprezentowała szeroki wachlarz prac. Większość wniosków końcowych pokrywała się z poprzednio prezentowanymi [1]). Głównym jednak efektem była wymiana informacji między uczestnikami. Niekiedy dało się bowiem zauważyć, że to co jedni z dużym wysiłkiem realizowali w ostatnim czasie — inni osiągnęli wcześniej. Brak informacji o działaniach poszczególnych zespołów prowadzi do niepotrzebnego rozproszenia sił i elementów. Systemy mikrokomputerowe, ze swą bardzo szeroką gamą zastosowań, stały się już podstawową częścią informatyki i dlatego regularne organizowanie konferencji tego typu — jak to postulowali uczestnicy imprezy wrocławskiej — jest nad wyraz pożądane.

**WACŁAW ISZKOWSKI**

[1]) Janusz Zalewski: O mikroprocesorach po polsku. INFORMATYKA nr 7/84

## O robotyce

Można powiedzieć, że „dwunastu gniewnych samurajów informatyki i cybernetyki technicznej..." — zdominowało nowy kwartalnik naukowy poświęcony wyłącznie sprawom robotyki [1]). Japończycy bowiem stanowią w 30-osobowym komitecie redakcyjnym Journal of Robotics Systems grupę przytłaczającą. Zewnętrznym wyrazem tego są streszczenia japońskie zamieszczane przy każdym artykule tego wydawanego w USA periodyku. Niemniej — jak dotąd — większość zamieszczanych tutaj prac autorstwa japońskiego powstała na uniwersytetach amerykańskich. Zaś w komitecie redakcyjnym można znaleźć także dwóch Węgrów, dwóch Anglików, dwóch Włochów oraz po jednym Australijczyku i Belgu. Wśród dziewięciu Amerykanów można znaleźć aż dwóch przedstawicieli przemysłowej firmy AUTOMATIX z Billerica w stanie Massachussetts oraz jednego z firmy LORD z Cary w stanie North Carolina. Czterej reprezentują różne uniwersytety, a jeden — politechnikę. Jest wreszcie ekspert NBS z Waszyngtonu. Brak nazwisk polskich można jeszcze zrozumieć, ale wyraźnie uderza brak przedstawicieli francuskiej strefy językowej.

Journal of Robotics Systems można więc traktować jako amerykańsko-japońskie forum wymiany myśli naukowej, do którego zechcą się może włączyć autorzy innych nacji. W każdym razie komitet redakcyjny gorąco zachęca do nadsyłania (oryginalnych!) artykułów — c/o Dept of El. & Computer Eng., University of California, Santa Barbara, Cal. 93106, USA, na ręce Dr. Gerardo Beni lub dr Susan Hackwood. Ciekawe, kiedy na łamach tego 111-dolarowego (w prenumeracie rocznej) czasopisma można będzie przeczytać pracę autora polskiego.

Przykładem profilu zainteresowań nowego czasopisma mogą być objętoś-

[1]) Journal of Robotics Systems (kwartalnik, prenumerata roczna 95 dol. + 16 dol. + koszty przesyłki lotniczej — 39 dol.); Wiley Journals, John Wiley & Sons, Inc., 605 Third Ave., New York, NY 10158; ISSN 0741-2223

ci (liczba stron) poszczególnych pozycji zeszytu jesiennego — Fall 1984:

51 stron — Systematyzacja receptorów obrazu z punktu widzenia perspektyw rozwojowych robotyki

19 stron — Analiza redundantnych kinematycznie układów wieloprzegubowych realizujących manipulatory przykładowe o siedmiu stopniach swobody

12 stron — Wyznaczanie właściwych długości palców w dłoniopodobnych chwytakach robotów przemysłowych

12 stron — Wykreślanie konturów wielościennych jako podstawa syntezy mechanizmów postrzegania przestrzennego.

Redakcja Journal of Robotics Systems liczy na autorów będących w stanie ściśle wywiązać się z dosyć drobiazgowych wytycznych wydawniczych. W szczególności wytyczne te dotyczą precyzyjnych danych bibliograficznych literatury cytowanej, konsekwentnego stosowania we wzorach matematycznych notacji wykładniczej zamiast pierwiastków algebraicznych, stosowania jednolitego liternictwa w rysunkach, zmniejszanego w procesie poligraficznym do standardowej wysokości 1,5 mm, jak również zwrotu odbitek kontrolnych w ciągu ... 48 godzin. Miejmy nadzieję, że ten wymóg czasowy będzie ewentualnym (oby!) autorom polskim liczony tylko do momentu nadania przesyłki na poczcie.

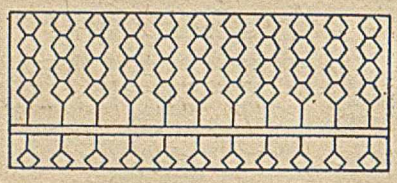
(A.B.E.)

## Listy

# Liczyć rozumniej

Do napisania tego listu skłoniła mnie krótka informacja [1]), napisana na podstawie notatki w tygodniku NEWSEEK, a dotycząca rosnącej roli liczydła japońskiego — **soroban.** NEWSWEEK bardzo często nadaje niemal sensacyjną formę informacjom o zjawiskach znanych i najzwyklejszych, zwłaszcza gdy pochodzą one z odległych i tajemniczych zakątków świata, a za taki uchodzi nadal w wielu kręgach Japonia. Chciałbym rozwiać jeszcze jedną legendę o tym kraju, obdzierając z sensacji informację o sorobanie. Przez 1,5 roku byłem stypendystą w Japonii i od tego czasu moja fascynacja tym krajem stale rośnie. Daleki jednak jestem od tworzenia legend, staram się raczej dociec źródeł zachodzących tam zjawisk.

Zacznę od historii. Soroban przywędrował do Japonii z Chin w XVI wieku w postaci tam używanej i dopiero na przełomie XIX i XX wieku został udoskonalony do obecnej postaci, która przypomina liczydła używane już przez Rzymian [2]).



Rysunek przedstawia soroban w pozycji przygotowanej do rozpoczęcia obliczeń. W każdym rzędzie wyróżniony jest jeden krążek odpowiadający pięciu jednostkom, cztery pozostałe są krążkami jednostkowymi. Za pomocą sorobanu można wykonywać cztery podstawowe działania arytmetyczne na dowolnych liczbach dziesiętnych, a bardziej zaawansowani użytkownicy potrafią także pierwiastkować. Umiejętność posługiwania się sorobanem w zakresie czterech podstawowych działań arytmetycznych Japończycy nabywają już w szkole podstawowej. Jest to jeden z ich uczniowskich obowiązków; a to, że opanowują go do perfekcji przypisuję ich mentalności i sumienności. Podstawowe umiejętności zdobyte za młodu można pogłębić na kursach „jukus", które dodatkowo umożliwiają zdobycie kolejnych stopni świadczących o poziomie doskonałości. Cudzoziemca, który kojarzy sobie Japonię z największą fabryką kalkulatorów, może to zaskoczyć, Japończycy jednak z pieczołowitością pielęgnują swoje tradycje.

Czy soroban może być antidotum na błędy i opieszałość w rachowaniu? Raczej nie, może z wyjątkiem działań addytywnych. W czasie swojego pobytu w Japonii zaobserwowałem, że liczydło używane jest przede wszystkim w okienkach urzędów (np. pocztowych) i w małych sklepikach. Ciekawe jednak, że w czasie zakupów związanych z targowaniem się o ceny — a tak właśnie kupuje się w Japonii większe rzeczy — sprzedawcy, nawet ci najstarsi, rezygnują z sorobanu na korzyść kalkulatora, który jest wygodniejszy do obliczania procentów, wielokrotnie liczonych przy ustalaniu ostatecznej ceny. (Zauważmy, że soroban musi być trzymany w odpowiedniej pozycji i trudno ukryć go przed kupującym, podczas gdy kalkulator można trzymać w ręce poza zasięgiem wzroku klienta). Szybkość liczenia nie może być jednak jedynym kryterium wyboru narzędzia obliczeń. Soroban ma wadę, której nie sposób usunąć ani nadrobić nie tracąc nic na szybkości obliczeń — nie pozostawia śladu wykonywanych operacji, wyników pośrednich i końcowego. Soroban nie wydaje pokwitowań i nie prowadzi buchalterii. Jak każde liczydło, soroban pozostać więc musi tylko pomocniczym narzędziem rachunkowości.

Notatka, którą komentuję, zwraca uwagę na kapitalną cechę stosowania wszelkich nieautomatycznych urządzeń wspomagających obliczenia. Pozwalają one bowiem liczyć i jednocześnie lepiej rozumieć liczby i proces liczenia. Nieprzypadkowo pada w tym kontekście nazwa koncernu MATSUSHITA EI, do którego należą takie potęgi rynku elektronicznego, jak NATIONAL czy PANASONIC. Koncern ten założył Konsuke Matsushita, który jest także założycielem Instytutu PHP (skrót od słów Peace, Happiness and Prosperity), którego głównym celem jest zespalanie wysiłków ludzi dobrej woli dla dobra wszystkich, w szczęściu i w pokoju. K. Matsushita — niestrudzony humanista — cieszy się w Japonii olbrzymim autorytetem intelektualnym i moralnym. Jego obecna działalność idzie w parze z produkcją koncernu, który założył, i ma na celu zachowanie w użytkownikach tego, czego nie posiadają urządzenia: rozumienia procesów (w szczególności — obliczeniowych) i możliwości kierowania nimi.

MACIEJ M. SYSŁO

[1]) K. I.: Liczyć lepiej. Informatyka, nr 9/1984

[2]) Por. H. Zemanek: Abacus: The Word and the Device. Abacus 1, no. 3 1983, p. 22—27

# Skoordynować

Przedsiębiorstwa, które korzystały dotąd ze sprawnych systemów informatycznych, z reguły nadal je użytkują. Ale ile dobrych nie wykorzystanych opracowań deaktualizuje się na półkach? Nakład finansowy i czasowy na ich aktualizację jest niewspółmiernie mały w porównaniu z rozpoczęciem prac projektowych od początku. Od chwili opracowania koncepcji systemu informatycznego, sporządzenia projektu, oprogramowania i wdrożenia aż do jego eksploatacji upływa niekiedy kilka lat. Kto jednak potrafi udzielić odpowiedzi o funkcjonujących obecnie systemach i gotowych opracowaniach informatycznych?

Częściowo czyni to redakcja INFORMATYKI. Wskazana byłaby cykliczna (półroczna, roczna) informacja typu katalogowego — wydawana przez zespół redakcyjny. Katalog taki powinien zawierać prezentację wybranych systemów powielarnych z podaniem zakresu tematycznego, użytkowników oraz kosztów wdrożenia i eksploatacji systemu na konkretnym sprzęcie informatycznym.

Informacje te przeznaczone dla potencjalnych użytkowników systemu powinny być zwięzłe i zrozumiałe również

dla osób, które nie mają przygotowania informatycznego. Nie zawsze bowiem kadra kierownicza, zainteresowana eksploatacją systemu, zna specyficzny język zasady technologii informatycznej. Często też nie ufa ona młodym informatykom zatrudnionym w przedsiębiorstwie, że potrafią trafnie ocenić przydatność systemu. Dlatego częściej wolą rezygnować niż ryzykować. Przydałaby się w tym zakresie również instytucja doradztwa organizacyjnego, ponosząca pełną odpowiedzialność za sporządzone ekspertyzy.

**HALINA TOWARNICKA**

**Od Redakcji:** Postulat ten informatycy kierują do nas już od wielu lat. Widzimy w tym dowód bezradności ludzi, którzy widząc tak daleko idącą dezorganizację chcieliby w małym choć stopniu system ponaprawiać. Ale nie mają się do kogo zwrócić, piszą więc do redakcji. Powtarzamy więc z równą bezradnością: nie jesteśmy w stanie wziąć na siebie tak gigantycznego zadania — istniejąca obsada etatowa dostosowana jest jedynie do redagowania czasopisma. Uważamy, że instytucją, która takiego zadania mogłaby się podjąć, jest Polskie Towarzystwo Informatyczne, a jeszcze lepiej Państwowa Agencja Informatyki, o utworzenie której PTI od dawna bezskutecznie zabiega. (Z.G.)



# Słownik pojęć i terminów z dziedziny grafiki komputerowej (3)

**PODZIELNOŚĆ EKRANU** (screen split) — cecha wielu monitorów, zwłaszcza alfanumerycznych, polegająca na podziale pola ekranu monitora na część bierną i część aktywną, z których pierwsza zawiera dane nie zmieniające się, a druga — zmieniające się w czasie pracy. Zwykle tylko dane zmienne są przesyłane między monitorem a komputerem.

**RASTROWE ROZWINIĘCIE OBRAZU** (raster scan) — technika komórkowego (punktowego) generowania lub zapisywania obrazu punkt po punkcie wzdłuż poziomych linii równoległych, radialnych lub spiralnych. Jest wykorzystywana do generowania obrazu np. w odbiornikach telewizyjnych, urządzeniach radarowych itp.

**RASTROWE URZĄDZENIE GRAFICZNE** (raster display device) — GRAFICZNE URZĄDZENIE WYJŚCIOWE wykreślające obraz metodą RASTROWEGO ROZWINIĘCIA OBRAZU, np. MONITOR RASTROWY, DRUKARKA GRAFICZNA.

**REJESTRATOR MIKROFILMOWY** (microfilm recorder, computer output mikrofilmer, COM device) — urządzenie do rejestrowania obrazu generowanego przez komputer na mikrofilmie lub mikrofiszy.

**RODZAJ LINII** (line type, line style) — ATRYBUT wykreślonej linii związany z jej ciągłością (np. linia nieprzerywana, kreskowana, kropkowana, osiowa itp.)

**ROZDZIELCZOŚĆ** (resolution) — 1) KROK KREŚLAKA lub JEDNOSTKA RASTRA. 2) Miara rozróżnialności szczegółów na ekranie monitora, zależna w równym stopniu od właściwości oka widza, jak też od jakości sprzętu, np. ostrości PIÓRA (zogniskowania strumienia elektronów), ziarnistości podłoża (LUMINOFORU) ekranu, na którym kreślony jest obraz, dokładności ustawienia, POWTARZALNOŚCI i rozmywania (dyfuzji) obrazu.

**ROZJAŚNIANIE, BŁYSZCZENIE** (highlight) — przyciąganie uwagi do pewnego ELEMENTU OBRAZU przez MIGOTANIE lub zmianę intensywności świecenia.

**ROZKAZ GRAFICZNY** (display order) — instrukcja interpretowalna przez GRAFICZNE URZĄDZENIE WYJŚCIOWE (GENERATOR OBRAZU).

**RZUTOWANIE, ODWZOROWANIE** (mapping function) — TRANSFORMACJA, która przekształca elementy jednego systemu reprezentacji na drugi, jak — układy współrzędnych lub struktury danych logicznych (np. PRZESTRZEŃ MODELU na PRZESTRZEŃ OBRAZOWANIA).

**RYSOWNICA** (tablet) — rodzaj DYSKRETYZATORA (czytnika rysunków) o niewielkich wymiarach (ok. 30×30 cm), z PIÓREM ULTRADŹWIĘKOWYM lub NAPIĘCIOWYM, wykorzystywana zazwyczaj do PLASOWANIA ELEMENTÓW OBRAZU na ekranie monitora.

**RYSUNEK CIĄGŁY** (continuous drawing) — rysunek, którego elementy składowe są opisane współrzędnymi względnymi z wyjątkiem początkowego położenia ELEMENTU KREŚLĄCEGO OBRAZ (strumienia elektronowego, GŁOWICY KREŚLAKA). Rysunki ciągłe można przesuwać zmieniając jedynie ich punkty początkowe.

**SKANING** (flying spot scan) — w dziedzinie GRAFIKI KOMPUTEROWEJ system kodowania obrazu za pomocą techniki omiatania rastrowego, uwzględniający poziom jaskrawości obrazu.

**STRONICA** (page) — pojedynczy OBRAZ EKSPONOWANY lub ich zbiór (zazwyczaj tekst alfanumeryczny).

**STRONICOWANIE** (paging) — w GRAFICE KOMPUTEROWEJ proces zastępowania jednej strony inną w MONITORZE ALFANUMERYCZNYM. Zazwyczaj STRONICE są wyświetlane kolejno.

**SZABLON** (form overlay) — wzór stanowiący tabelę, siatkę lub mapę, używany zazwyczaj jako element TŁA OBRAZU.

**SZKIELETOWE PRZEDSTAWIENIE OBIEKTU** (wire frame representation) — sposób zobrazowania obiektu trójwymiarowego, polegający na ukazaniu wszystkich jego krawędzi, bez usuwania LINII UKRYTYCH.

**TŁO (PODSTAWA) OBRAZU** (display background, static image, background image) — fragment OBRAZU FIZYCZNEGO, który w trakcie zmian treści obrazu nie ulega zmianie (np. SZABLON). Bywa nazywany stałą częścią obrazu.

**TRANSFORMACJA** (transformation function) — jedna z następujących funkcji: skalowanie, obrót, przesunięcie lub RZUTOWANIE.

**TRYB BIERNY** (passive mode) — w GRAFICE KOMPUTEROWEJ sposób wykorzystania KONSOLI GRAFICZNEJ, który nie dopuszcza żadnej interakcji użytkownika za pomocą WEJŚCIOWYCH URZĄDZEŃ GRAFICZNYCH w trybie bezpośrednim (ang. on-line).

**TRYB INTERAKCYJNY** (interactive mode) — w GRAFICE KOMPUTEROWEJ metoda wykorzystania KONSOLI GRAFICZNEJ do komunikacji bezpośredniej (ang. on-line) człowieka z komputerem, powszechnie używana do wprowadzenia danych i sterowania przebiegiem programu.

**WEKTOR BEZWZGLĘDNY** (absolute vector) — wektor zadany za pomocą punktów skrajnych o współrzędnych mierzonych w jednostkach bezwzględnych, np. od pozycji bieżącej ELEMENTU KREŚLĄCEGO OBRAZ do punktu zwanego końcem tego wektora.

**WEKTOR WZGLĘDNY, PRZYROSTOWY** (incremental vector) — wektor, którego punkt początkowy jest punktem końcowym poprzedniego elementu graficznego, a punkt końcowy jest zdefiniowany jako przemieszczenie od jego punktu początkowego.

**WIĄZANIE ELASTYCZNE** — technika przemieszczania wspólnego wierzchołka zbioru odcinków prostych ELEMENTU OBRAZU lub całego obrazu w taki sposób, że inne punkty końcowe tego elementu pozostają bez zmian.

**WIELKOŚĆ OBRAZU, ROZMIAR RYSUNKU** (drawing size, display scope) — maksymalne wymiary obrazu, który można przesłać do GENERATORA OBRAZU w urządzeniu graficznym, określone przez pojemność rejestrów pozycji x i y w GENERATORZE OBRAZU. Maksymalna wielkość obrazu, który można wyświetlić na ekranie lampy monitora, może być mniejsza.

**WIELKOŚĆ (ROZMIAR) POWIERZCHNI OBRAZOWANIA** (viewing area, drawing area) — wymiary POWIERZCHNI OBRAZOWANIA wyrażone w jednostkach fizycznych (cm, cal).

**WODZENIE** (dragging) — przesuwanie wybranego elementu obrazu wzdłuż ścieżki określonej przez GRAFICZNE URZĄDZENIE WEJŚCIOWE (np. pióro świetlne).

**WRAŻLIWOŚĆ NA PIÓRO ŚWIETLNE** (light pen sensivity) — cecha elementu obrazu umożliwiająca jego WYKRYWANIE za pomocą PIÓRA ŚWIETLNEGO.

**WSPÓŁRZĘDNE** (coordinates) — uporządkowany zbiór wartości, bezwzględnych lub względnych, które określają punkt adresowalny w PRZESTRZENI MODELU, PRZESTRZENI OBRAZOWANIA lub PRZESTRZENI URZĄDZENIA GRAFICZNEGO.

**WSPÓRZĘDNE ZNORMALIZOWANE** (normalized device coordinates) — WSPÓŁRZĘDNE PRZESTRZENI OBRAZOWANIA (zwykle z zakresu 0—1), które są niezależne od urządzenia.

**WYBIERALNOŚĆ** (detectability) — cecha ELEMENTU OBRAZU pozwalająca na WYBIERANIU go za pomocą GRAFICZNEGO URZĄDZENIA WEJŚCIOWEGO.

**WYBIERANIE** (pick) — identyfikowanie ELEMENTU OBRAZU wykreślonego na POWIERZCHNI OBRAZOWANIA, za pomocą GRAFICZNEGO URZĄDZENIA WEJŚCIOWEGO, np. PIÓRA SWIETLNEGO (przeciwieństwo PLASOWANIA). Przykładowo, wybieranie za pomocą pióra świetlnego następuje, gdy wykreślany przez strumień elektronowy lampy element graficzny znajdzie się w polu widzenia PIÓRA (będzie wskazany przez PIORO). Wtedy PIÓRO ŚWIETLNE generuje impuls elektryczny powodujący przerwanie, interpretowane przez program sterujący monitora graficznego, w celu określenia pozycji lub innych ATRYBUTÓW wskazanego elementu graficznego.

**WYGASZANIE, WYMAZYWANIE WYBIÓRCZE** (blanking, selective erase) — usunięcie jednego lub większej liczby ELEMENTÓW OBRAZU EKSPONOWANEGO bez zmiany pozostałych ELEMENTÓW OBRAZU.

**WYKAZ (SPIS) FUNKCJI** (menu, function menu) — w GRAFICE KOMPUTEROWEJ zestaw funkcji urządzenia graficznego dostępnych na określonym etapie realizacji zadania, których wybieranie dokonuje się za pomocą urządzenia wejściowego. W MONITORZE GRAFICZNYM, WYKAZ FUNKCJI na ogół ma postać zbioru KLAWISZY ŚWIETLNYCH.

**WZIERNIK** (viewport) — ograniczony obszar w obrębie PRZESTRZENI OPERACYJNEJ, który prezentuje zawartość OKNA. Może być rozszerzony do całej PRZESTRZENI OPERACYJNEJ.

**ZAWIJANIE** (wraparound) — dotyczy pracy urządzenia graficznego (zwykle monitora), gdy współrzędne wykreślanych elementów obrazu przekraczają pojemność rejestrów pozycji x i y, i oznacza kontynuację wykreślania począwszy od przeciwległego brzegu ekranu (p. OBCINANIE).

**ZBLIŻANIE** (zooming) — stopniowe skalowanie ELEMENTÓW OBRAZU obserwowanego za pomocą WZIERNIKA tak, aby sprawić wrażenie zmiany odległości względem obserwatora.

**ZNIEKSZTAŁCENIE BECZKOWATE** (barrel distortion) — zniekształcenie obrazu polegające na tym, że linie równoległe mają tendencje do odchylania się od siebie w części środkowej ekranu monitora. Wynika z niedoskonałości urządzeń z LAMPĄ KATODOWĄ.

**ZNIEKSZTAŁCENIE PODUSZKOWATE** (pin-cushion distortion) — zniekształcenie obrazu polegające na tym, że linie równoległe mają tendencje do zbiegania się ku sobie w środkowej części ekranu. Wynika z niedoskonałości urządzeń z LAMPĄ KATODOWĄ.

**LENA WALKIEWICZ**
**WOJCIECH MOKRZYCKI**

# „Nasz drugi język"

Dyskusja nad tym, czy nieznajomość języka angielskiego jest poważną przeszkodą w programowaniu i korzystaniu z maszyn cyfrowych traci już dzisiaj wiele ze swojego znaczenia, gdyż coraz trudniej jest znaleźć osobę, która nie zetknęłaby się przynajmniej z elementami tego języka. Nie kwestionuję, że język angielski zdominował konferencje i publikacje informatyczne adresowane do międzynarodowego grona. Jeśli zaś mowa tylko o korzystaniu z oprogramowania, to autor „Naszego drugiego języka" [1]) przywiązuje chyba zbyt wielką wagę do pochodzenia słów kluczowych. Prawdą jest, że nie pojawił się dotychczas żaden bardziej uniwersalny język — źródło słów kluczowych.

Autor wymienił najważniejsze z cech zapewniających językowi angielskiemu taką pozycję, przemilcza jednak, że przecież:

(1) liczba słów kluczowych jest zwykle bardzo niewielka; i tylko

(2) częściowo zachowują one swoje znaczenie z języka mówionego; gdyż

(3) znaczenie dyrektyw jest najczęściej precyzyjnie definiowane bez oglądania się na ich znaczenie w języku angielskim; co więcej

(4) dłuższe słowa kluczowe lub ich konstrukcje ulegają skróceniu lub zastępowane są akronimami, przez co tracą swoje znaczenie z języka angielskiego; i niestety

(5) bardzo często to co uważane jest przez cudzoziemców za poprawną konstrukcję w języku angielskim razi osoby, które wyrosły w tym języku, nie mniej niż ich polskie odpowiedniki — i nie ma swojego znaczenia poza wąskim kręgiem fachowców.

Właściwym podejściem do nauczania np. języka programowania jest traktowanie dyrektyw i słów kluczowych jako ciągów znaków, które powinny być przyswajane w całości. Zwięzłość języka angielskiego może jedynie w tym pomóc, nieprecyzyjność jednak nie powinna być większą przeszkodą. Ze swoich lat uniwersyteckich pamiętam kurs języka ALGOL 60 prowadzony przez prof. Stefana Paszkowskiego, autora najpopularniejszej książki o ALGOLU 60 w języku polskim, w czasie którego wykładowca ani razu nie wypowiedział słowa w obcym języku. Była to znakomita lekcja języków: ojczystego i ALGOLU 60, a także i angielskiego.

**MACIEJ M. SYSŁO**

PS

1. Przykładem niewłaściwego posłużenia się pozornym podobieństwem słów jest użycie przez autora słowa oportunista, którego znaczenie jest niestety przeciwne do pierwszego skojarzenia, jakie nasuwa jego brzmienie. Posługiwanie się słowami obcego języka zwiększa szanse takich potknięć.
2. Anglicy mówią raczej mother tongue lub native language zamiast mother language.

[1]) Wacław Iszkowski: Nasz drugi język, INFORMATYKA nr 9/1984

Winiewski J.: Język programowania C (1)

INFORMATYKA 1985, nr 4, s. 1

Pierwsza część charakterystyki języka programowania C, przeznaczonego głównie do programowania systemowego. Omówiono elementarne składniki programu, deklaracje oraz wyrażenia.

---

Land F.: Dziedzina systemów informacyjnych

INFORMATYKA 1985, nr 4, s. 5

Charakterystyka istoty oraz podstawowych funkcji systemu informacyjnego. Omówiono główne źródła informacji oraz czynniki uzależniające ich wykorzystanie przez człowieka.

---

Kierzkowski Z., Małuszyński J.: Zarządzanie współbieżną aktualizacją bazy danych (2). Przegląd metod zarządzania

INFORMATYKA 1985, nr 4, s. 7

Druga część charakterystyki rozwiązań zapewniających zarządzanie współbieżną aktualizacją bazy danych. Omówiono mechanizmy działania koordynatora, stanowiącego część składową systemu zarządzania bazą danych.

---

Виневски Ю.: Язык программирования С (1)

INFORMATYKA 1985, № 4, стр. 1

Первая часть характеристики языка программирования С, в основном предназначенного для системного программирования. Обсуждены составные элементы программы, описание данных и выражения.

---

Ланд Ф.: Область информационных систем

INFORMATYKA 1985, № 4, стр. 5

Характеристика сущности и основных функции информационных систем. Обсуждены главные источники информации и факторы влияющие на использование их человеком.

---

Кежковски З., Малушиньски И.: Управление параллельным обновлением базы данных (2). Обзор методов управления

INFORMATYKA 1985, № 4, стр. 7

Вторая часть характеристики решений обеспечивающих управление параллельным обновлением базы данных. Обсуждены механизмы функционирования координатора, составляющего часть системы управления базой данных.

**Winiewski J.: C programming language (1)**

INFORMATYKA 1985, Nr. 4, p. 1

First part of characteristics of the C language, devoted mainly for system programming. Basic program components, declarations and expressions are discussed.

---

**Land F.: The information systems domain**

INFORMATYKA 1985, Nr. 4, p. 5

Characteristics of essence and fundamental functions of all information systems. Main information sources and factors, which condition their utilization by a human being, are discussed.

---

**Kierzkowski Z., Małuszyński J.: Management of concurrent data base updating (2). Management methods survey**

INFORMATYKA 1985, Nr. 4, p. 7

Second part of characteristics of solutions, which secure management of concurrent data base updating. Working mechanisms of the coordinator, a component of data base management systems, is discussed.

---

**Winiewski J.: C-Programmiersprache (1)**

INFORMATYKA 1985, Nr 4, s. 1

Erster Teil einer Charakteristik von C-Programmiersprache, die hauptsächlich für Systemprogrammierung bestimmt ist. Es wurden elementare Programmbestandteile, Deklarationen und Ausdrücke besprochen.

---

**Land F.: Bereich von Informationssystemen**

INFORMATYKA 1985, Nr 4, s. 5

Eine Charakteristik des Wesens und der Grundfunktionen von einem Informationssystem. Es wurden Informationshauptquellen und Faktoren, die ihre Ausnützung durch Menschen abhängig machen, besprochen.

---

**Kierzkowski Z., Małuszyński J.: Verwaltung von simultaner Aktualisierung einer Datenbank (2). Ein Übersicht von Verwaltungsmethoden**

INFORMATYKA 1985, Nr 4, s. 7

Zweiter Teil einer Charakteristik von Lösungen, die die Verwaltung simultaner Aktualisierung einer Datenbank sichern. Es wurde Wirkungsmechanismus des Koordinators, der den Bestandteil eines Datenbankverwaltungssystems bildet, besprochen.

# Zasady prenumeraty

**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 dostarczone 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!**

Przedpłata na dwumiesięczniki przyjmowana jest 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.

**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 prenumeraty w złotych | | | | | |
|---|---|---|---|---|---|
| kwartalna | | półroczna | | roczna | |
| normalna | ulgowa | normalna | ulgowa | normalna | ulgowa |
| 300,— | 105,— | 600,— | 210,— | 1200,— | 420,— |

# Ceny ogłoszeń

**Od 1 stycznia br. obowiązują następujące ceny ogłoszeń publikowanych na naszych łamach:**

**ogłoszenia duże** (zależnie od objętości):
cała strona — 35 tys. zł, 3/4 — 30 tys., 1/2 — 25 tys., 1/4 — 20 tys., 1/8 — 15 tys.

**ogłoszenia drobne** (zależnie od liczby słów):
jedno słowo — 30 zł

**Dodatki do ceny podstawowej:**
— za dodatkowy kolor (na okładce) +30%
— za zamieszczenie ogłoszenia na czwartej stronie okładki +100%
— za zamieszczenie ogłoszenia na trzeciej stronie okładki +50%

**Zniżki:**
— za ogłoszenie 3—5-krotne —5%
— za ogłoszenia 6—10-krotne —10%
— za ogłoszenia 11-krotne i powyżej —20%
— za artykuły reklamowe i wkładki wykonane przez zleceniodawcę —40%
— za bloki i biuletyny wykonane przez zleceniodawcę — maks. —60%

**W przypadku dostarczenia przez zleceniodawcę materiału ilustracyjnego nie odpowiadającego warunkom technicznym druku lub tekstu wymagającego redakcyjnego opracowania, do powyższych cen doliczane będą koszty odpowiednich usług fotograficznych, graficznych lub przygotowania tekstów.**

# Mikrokomputery i nauczyciele

W swoim czasie ogłoszone zostało hasło: informatyka kluczem do dobrobytu. Kopalnie, huty i inne wielkie zakłady przemysłowe instalowały komputery, które miały rychło dokonać cudu, przede wszystkim — gospodarczego. Niestety — jak wiemy — nic się takiego nie stało. Nadzieje wiązano także z wprowadzeniem komputerów do szkół. Spodziewano się przede wszystkim znacznego podniesienia poziomu nauczania i nagłego zlikwidowania różnych niedomagań występujących w systemie szkolnym. Na szczęście komputery do naszych szkół nie dotarły — uniknęliśmy więc dzięki temu jeszcze jednego rozczarowania. Aż strach pomyśleć co by to było, gdyby w owym czasie postanowiono wyposażyć nasze szkoły, chociażby niektóre, w komputery typu ODRA czy nawet MERA.

Ale „groźba" komputera w szkole wcale się nie oddaliła, wprost przeciwnie — stała się bardzo realna. Można powiedzieć. że dzisiaj mikrokomputery są już w zasięgu szkoły i lada dzień się w niej znajdą.

Zaczynam się obawiać, czy niebawem ktoś nie ogłosi nowego hasła „mikrokomputery jedyną szansą polskiej szkoły" lub podobnego. Hasłem takim można bardziej zaszkodzić niż pomóc. Mikrokomputery mogą być wprawdzie kluczem do rozwiązywania wielu problemów szkolnych, ale pod warunkiem, że będzie się je traktować jako klucz, a nie jako wytrych.

Na polskim rynku pojawił się pierwszy rodzimy mikrokomputer: MERITUM I, który — jak głoszą zapowiedzi — ma być produkowany głównie na potrzeby szkół. Bardzo popularny jest u nas ZX SPECTRUM, który już niebawem będzie dostępny za złotówki. Organizuje się już kluby lub kursy — niemal wyłącznie dla dzieci i młodzieży, gdzie 10—15-latki uczą się programowania w asemblerze i BASICU lub zabawiają się różnymi grami komputerowymi. Wiele gier i programów opracowanych jest głównie z myślą o nich. Wielu uczniów ma już własny sprzęt i oprogramowanie.

A ilu nauczycieli ma własny mikrokomputer? Czy istnieje gdzieś klub mikro dla nauczycieli? Czy w ogóle ktokolwiek zainteresował się nauczycielem w kontekście spraw mikroinformatycznych?

Prasa codzienna podała niedawno informację o przekazaniu 300 egzemplarzy ZX SPECTRUM polskim szkołom przez austriacką firmę Reitera. Komu i w jaki sposób został przydzielony ten sprzęt? Czy po jednym egzemplarzu do szkoły, czy też po dziesięć do wytypowanych szkół? A jakiej informatyki ma się uczyć w tych szkołach? Programowanie w BASICU czy może w asemblerze? I kto będzie tego uczył, może uczniowie?

Myślę, że nadeszła już pora, aby uświadomić sobie wreszcie, że informatyka nie oznacza znajomości tego czy innego komputera, ani też programowania w tym czy innym języku, lecz że jest to n a u k a o r o z w i ą z y w a n i u p r o b l e m ó w z p o m o c ą k o m p u t e r a. Komputer jest tutaj jedynie narzędziem, wykonawcą. Rozwiązywanie problemów wymaga zaś głębokiej wszechstronnej wiedzy...

Co zatem robić, aby komputer był należycie traktowany przez ucznia, aby zdobył uznanie wśród nauczycieli i zajął właściwe sobie miejsce w szkole?

Najpierw trzeba uwzględnić założenie, że sam komputer, nawet najwspanialszy, nic pożytecznego, a nawet w ogóle nic nie jest w stanie zrobić, jeśli nie znajdzie się we właściwych rękach. Drugie założenie: te właściwe ręce, to nie są ręce uczniów, lecz ręce nauczycieli. Trzecie — dotyczy zasady budowania piramidy: każdą piramidę, a więc też informatyczną, należy budować od podstawy.

Podstawą jest tu przede wszystkim kształcenie informatyczne, szeroko pojęte, a więc obejmujące także edukację nauczycieli. Wygląda ona gorzej niż słabo. Nie kształci się nauczycieli informatyki, najbliżej informatyki są więc nauczyciele matematyki. Ale w programie studiów matematycznych na Uniwersytecie Warszawskim — na przykład — jest tylko jeden semestr (wykład z ćwiczeniami) informatyki oraz jeden semestr laboratorium informatycznego. A w programie studiów podyplomowych dla nauczycieli matematyki — jeden semestr wykładu z informatyki (bez ćwiczeń) oraz 15 godzin zajęć praktycznych, ale tylko na papierze, bo w rzeczywistości nie uwzględniono ich w siatce godzin. Nie można — moim zdaniem — budować piramidy na tak wątłej podstawie.

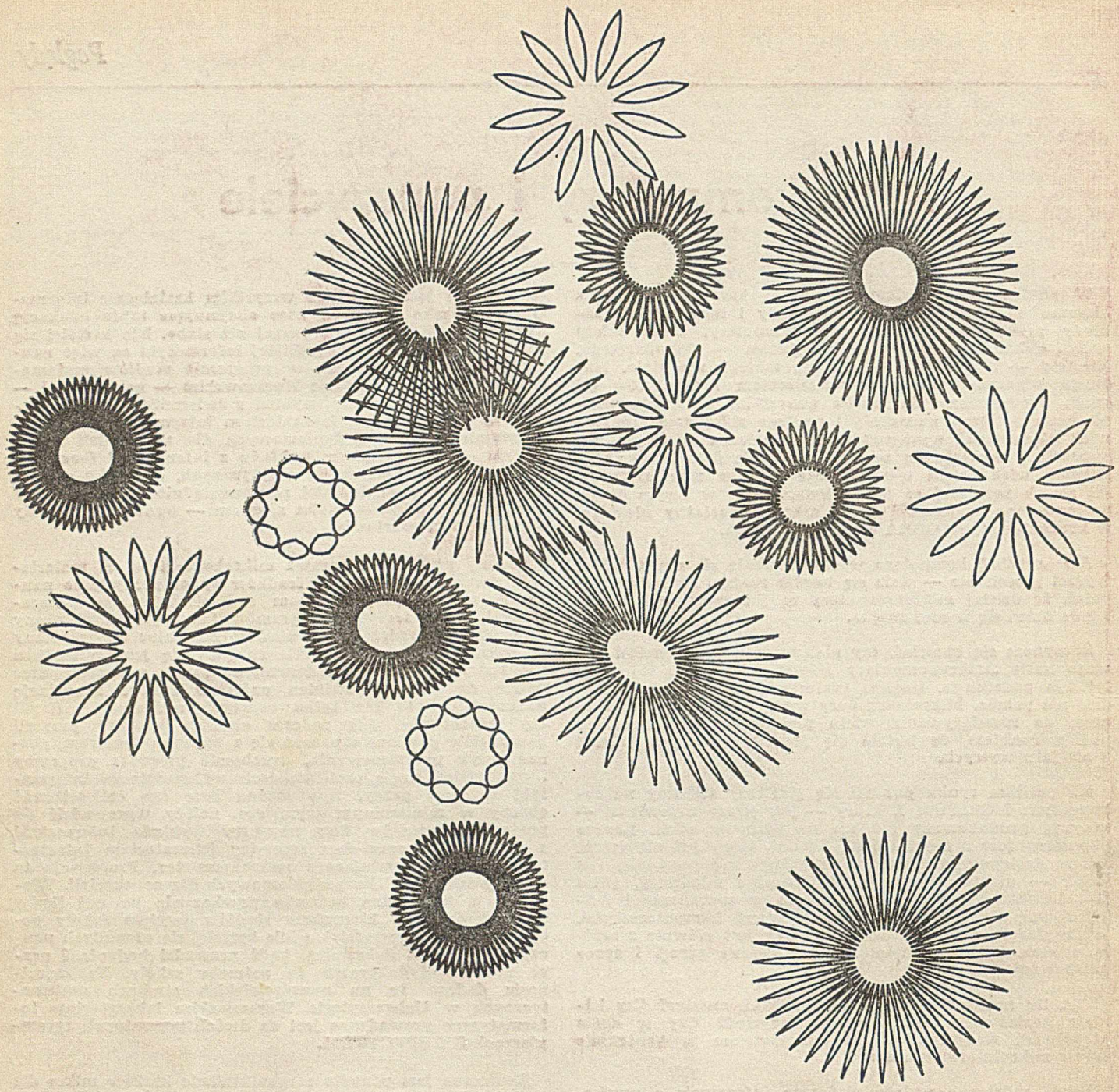
Osobny problem to sprawa mikrokomputera. W materiałach zawierających wykaz środków dydaktycznych do nauczania matematyki w liceum ogólnokształcącym, opracowanych przez Instytut Programów Szkolnych, znajdujemy tylko dwa narzędzia obliczeniowe: kalkulator elektroniczny i... suwak logarytmiczny. Nie znalazło się jeszcze miejsca dla komputera. I chyba dobrze, bo zanim mikrokomputer dotrze do ucznia, powinien najpierw zdobyć akceptację nauczycieli i to nie tylko uczących matematyki, fizyki czy informatyki. Już podczas studiów wszyscy przyszli nauczyciele powinni zapoznać się z mikrokomputerem, poznać język programowania, uruchomić pierwsze programy i dowiedzieć się o możliwościach wykorzystania informatyki w swej pracy. Aby można było ten cel osiągnąć choćby w minimalnym wymiarze, należy wprowadzić do programów studiów dwa semestry wykładu informatyki z ćwiczeniami oraz dwa semestry laboratorium informatycznego wykorzystującego mikrokomputer. Propozycja ta dotyczy także studiów podyplomowych dla nauczycieli. Wiąże się z tym pilna potrzeba przekazania pewnej liczby komputerów tym kierunkom studiów (wyższe szkoły pedagogiczne, uniwersytety), gdzie kształci się przyszłych nauczycieli lub ich dokształca, bądź prowadzi badania i prace naukowo-dydaktyczne na potrzeby szkoły. Na marginesie dodam, że na nauczycielskich studiach matematycznych w Uniwersytecie Warszawskim laboratorium informatyczne prowadzone jest na dwóch prywatnych egzemplarzach ZX SPECTRUM.

Konieczne jest ponadto zorganizowanie klubów mikro dla nauczycieli. W klubach tych jedni doskonaliliby posiadane już umiejętności informatyczno-dydaktyczne, inni — zdobywaliby je. Te mikrokomputery ofiarowane przez Reitera polskim szkołom przeznaczyłbym właśnie na zorganizowanie klubów nauczycielskich. Oczywiście kluby te w miarę rozwoju oraz istniejących możliwości mogłyby służyć także uczniom, ale zawsze w drugiej kolejności.

Uważam, że nie należy zbyt pochopnie oddawać komputerów w ręce ucznia. Do powodów, które podałem wyżej, trzeba dołączyć i ten, że urządzeń tych jest ciągle za mało, aby obdzielić nimi sensownie wszystkie szkoły w kraju.

Rozwiązywanie problemów edukacji mikroinformatycznej należy rozpocząć od nauczyciela, a skończyć na uczniu — to wydaje mi się oczywiste. Tymczasem w praktyce dzieje się odwrotnie. Czy nie mamy więc do czynienia z dylematem: n a u c z y c i e l z k m p u t e r e m d l a u c z n i a czy u c z e ń z k o m p u t e r e m p r z e c i w n a u c z y c i e l o w i? Jeśli chcemy rzeczywiście pomóc szkole, to między innymi ten dylemat trzeba rozwiązać. A rozwiązać w informatycznym i matematycznym sensie oznacza zawsze rozwiązać dobrze i prawidłowo.

EMANUEL CZYŻO



# KWIATY

Przedstawione rysunki są przykładowymi wynikami eksperymentów z systemem graficznym zaimplementowanym na minikomputerze SPERRY V77-600, wyposażonym w plotter CALCOMP 960/925. W systemie graficznym wykorzystano koncepcję syntezy obrazów w oparciu o reprezentację grafową i gramatyki grafowe (koncepcja została opisana w pracy doktorskiej Ewy Grabskiej „Synteza obrazów z wykorzystaniem teorii grafów", IM UJ, 1982).

Rysunki są wynikiem wielopoziomowego przetworzenia łuku jednostkowego. Najpierw, w wyniku złożenia dwóch transformacji takiego łuku, wygenerowany został płatek. Następnie, poddając płatek kolejno różnym transformacjom i powielając go za każdym razem inną liczbę razy, uzyskaliśmy kwiaty o różnym wyglądzie. W końcu, korzystając z tego samego zbioru wzorcowego — przez manipulowanie wyłącznie wartościami wiązań generatorów otrzymaliśmy obrazy o odmiennych kompozycjach.

**EWA GRABSKA**
**WOJCIECH GRABSKI**