1 1986

informatyka

Prof. Ian Pyle o Adzie

Abstrakcje w programowaniu

Modularny system wyszukiwawczy

Pakiet symulacyjny DESA

Ratfor dla Mery 400

WYDAWNICTWO NOT SIGMA

Cena 120 zł

informatyka

1 1986

Nr 1
Miesięcznik Rok XXI
Styczeń 1986

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. 1322-1300/85. Obj. 4,0 ark. druk. Nakład 7200 egz. P-70.

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza 120 zł
Prenumerata roczna 1200 zł

NACZELNA ORGANIZACJA TECHNICZNA
WYDAWNICTWO
CZASOPISM I KSIĄŻEK TECHNICZNYCH
SIGMA
00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

W NASTĘPNYCH NUMERACH:

- Niklaus Wirth o zasadach wieloprogramowości i ich implementacji w Moduli-2
- Michał Kleiber, Maciej Leśny i Romuald Szuniewicz o komputerach osobistych w zastosowaniach profesjonalnych
- Zbigniew Szkaradnik o operacjach zmiennoprzecinkowych w języku FORTH
- Jerzy Zakręcki o abstrakcjach w programowaniu
- Jan Bielecki o zaawansowanych konstrukcjach języka C
- Zdzisław Płoski o Discologo
- Jerzy Karczmarczuk o języku programowania Icon

IAN C. PYLE
Computing Service
University of York
Heslington, hrabstwo York
Wielka Brytania

Pakiet do specyfikacji programów w Adzie

W niniejszym artykule przedstawiono jedną z kilku aktualnie badanych metod wyrażania specyfikacji programów w Adzie [2]. Podstawy tej metody polegają na założeniu, że ciąg instrukcji (np. w ciele procedury) można opisać semantycznie w trzech częściach, podając:

- warunki początkowe (ang. precondition), opisujące relacje między odpowiednimi zmiennymi przed rozpoczęciem wykonywania ciągu instrukcji
- warunki końcowe (ang. postcondition), opisujące relacje między odpowiednimi zmiennymi po zakończeniu wykonywania ciągu instrukcji
- skutki, opisujące relacje między dwoma wymienionymi stanami.

Każdy z tych trzech rodzajów relacji można wyrazić w zapisie zbliżonym do Ady, deklarując odpowiednie podprogramy w części deklaracyjnej poprzedzającej ciąg instrukcji. Jedynie bardzo proste, statyczne warunki początkowe i końcowe mogą być wyrażone jako ograniczenia na parametry podprogramu. Technika ta ma tę zaletę, że do kontroli ważności (ang. validity check) specyfikacji funkcjonalnej w jej kontekście stosują się zwykłe reguły zasięgu kompilatora i identyfikowania nazw. Metoda sama w sobie nie zapewnia jednak niesprzeczności (ang. consistency) między specyfikacją a odpowiadającym jej ciągiem instrukcji, choć można utworzyć przeznaczone do tego celu narzędzia programowe.

ZARYS METODY POSTĘPOWANIA

Pakiet SPECYFICATIONS (wydruk 1) jest podzielony na trzy podpakiety odpowiadające różnym stopniom formalizacji. Projektowanie jednostki programowej polega na stopniowym korzystaniu z tych pakietów.

Pierwszy pakiet PSEUDO_CODE umożliwia projektantowi tradycyjne użycie pseudokodu, polegające na nadaniu jednostce struktury formalnej i nieformalnym określeniu jej zawartości. Do opisania czynności oraz zdarzeń przy użyciu konstrukcji ACT i IT_IS (ang. czyń, to jest) stosuje się napisy (ang. string).

Drugi pakiet OBJECTS pozwala na zwiększenie formalizacji w określaniu identyfikatorów. Projektant może opisywać obiekty przy użyciu typów pochodnych od ENTITY (ang. byt) i konkretów procedury rodzajowej ACTION (ang. czynność). Prowadzi to do uzyskania nowego opisu jednostki programowej, której wybrane byty i czynności są określone w odrębnym pakiecie zgodnie ze zwykłymi regułami Ady.

```
package SPECIFICATIONS is

  package PSEUDO_CODE is
    procedure ACT (EFFECT : STRING) ;
    function IT_IS (CONDITION : STRING) return BOOLEAN;
    generic
      type OPTIONS is (<>);
    function WHICH (SELECTION : STRING) return OPTIONS;
  end PSEUDO_CODE;

  package OBJECTS is
    type ENTITY is private;
    generic
    procedure ACTION;
  private
    type ENTITY is (NOT_DEFINED_YET);
  end OBJECTS;

  package SEMANTICS is
    function PRE_CONDITION return BOOLEAN;
    function POST_CONDITION return BOOLEAN;
    function INVARIANT return BOOLEAN;
    procedure EFFECT;
    procedure ASSERT (B: BOOLEAN);
    procedure DENY (B: BOOLEAN);
    procedure NOT_DESIGNED_YET;

    generic
      type ELEMENT is (<>);
      with function PREDICATE(T:ELEMENT) return BOOLEAN;
    package UNIVERSAL is
      function FOR_ALL return BOOLEAN;
      function THERE_EXISTS return BOOLEAN;
      function COUNT_OF return INTEGER;
    end UNIVERSAL;

  end SEMANTICS;

end SPECIFICATIONS;
```

Wydruk 1. Pakiet służący do specyfikacji programów w Adzie

Trzeci pakiet wewnętrzny umożliwia projektantowi określenie semantyki tych bytów i czynności (ang. SEMANTICS) w postaci ciał, których części wykonywalne są jeszcze nie zaprojektowane (ang. NOT_DESIGNED_YET). Implementowanie tych ciał na tak niskim poziomie jednostki programowej można rozpocząć również od wyrażenia ich w postaci pseudokodu.

Zauważmy, że wymienione pakiety obejmują różne stopnie formalizacji opisu, od całkowicie nieformalnego tekstu w postaci napisu, przez zadeklarowanie identyfikatora, do pełnej deklaracji z parametrami w Adzie (specyfikacja syntaktyczna) i formalnego wyrażenia semantyki specyfikacji funkcjonalnej.

Metoda polega na zadeklarowaniu funkcji PRE_CONDITION i POST_CONDITION oraz procedury EFFECT w

Prof. IAN C. PYLE studiował nauki przyrodnicze w Cambridge i uzyskał doktorat z fizyki teoretycznej. Od roku 1965 pracował w laboratoriach jądrowych AERE w Harwell, gdzie m.in. zajmował się obliczeniami reaktorowymi, prowadził zespół opracowujący kompilator Fortranu, opracował system podziału czasu dla komputera IBM 360 oraz spędził kilka lat nad projektem dużego systemu wielokomputerowego do przetwarzania danych w czasie rzeczywistym.

W roku 1972 został mianowany profesorem na uniwersytecie York, gdzie stworzył Wydział Informatyki, ucząc podstaw informatyki i inżynierii oprogramowania oraz propagując stosowanie technik komputerowych w działalności uniwersyteckiej. Brał aktywny udział w projekcie amerykańskiego Departamentu Obrony dotyczącym języków wysokiego poziomu, czego wynikiem było powstanie Ady. Był konsultantem zespołu projektowego Ady i zainicjował prace nad jej kompilatorem na Uniwersytecie York. Obecnie jest dyrektorem Computing Service na tym uniwersytecie.

części deklaracyjnej podprogramu, bezpośrednio po wyrażeniu specyfikacji w Adzie, wprowadzającej parametry formalne. W przypadku funkcji w Adzie, należy zadeklarować także zmienną zawierającą wynik w celu nadania mu nazwy. Poniżej przedstawiono właściwości, które muszą mieć te podprogramy specyfikacyjne.

OPIS WARUNKÓW I SKUTKÓW

Funkcje PRE_CONDITION i POST_CONDITION opisują relacje zachodzące między odnośnymi zmiennymi, odpowiednio na początku i końcu ciągu instrukcji. Udostępniają one wartość TRUE, jeżeli określona relacja zachodzi, a wartość FALSE w przypadku przeciwnym. Procedura EFFECT opisuje relacje zachodzące między początkowymi i końcowymi wartościami odnośnych zmiennych, ustanawiając odpowiednią zależność. Używa się ich tak, jakby początkową instrukcją było wywołanie:

```
ASSERT (PRE_CONDITION)
```

a instrukcją końcową wywołanie:

```
ASSERT (POST_CONDITION)
```

procedury ASSERT zadeklarowanej w pakiecie SPECIFICATIONS.SEMANTICS. Kompilator optymalizujący może stwierdzić, że podprogramy w praktyce nie są wywoływane i powstrzymać się od wygenerowania kodu. Taki sam efekt dla mniej doskonałego kompilatora można uzyskać używając pragmy:

```
pragma IN_LINE (PRE_CONDITION,
                POST_CONDITION,
                EFFECT);
```

Prostą relację można opisać zwykłym wyrażeniem w Adzie. Przykładowo, w celu stwierdzenia, że dwie liczby całkowite, A i B, są uporządkowane w kolejności rosnącej (z równością włącznie) można użyć funkcji:

```
function POST_CONDITION return BOOLEAN is
begin
   return A<=B;
end POST_CONDITION;
```

Jednakże, do wyrażenia bardziej złożonych specyfikacji konieczne jest użycie kwantyfikatorów ogólnych i szczegółowych. Można to osiągnąć stosując pętle i jednostki rodzajowe w Adzie, o ile kwantyfikowany element jest typem skalarnym. Tak więc, w celu wyrażenia faktu, że wszystkie elementy tablicy A, gdzie:

```
type INDEX is range 1..10;
A : array (INDEX) of FLOAT;
```

są dodatnie, można użyć funkcji:

```
function POST_CONDITION return BOOLEAN is
begin
   for I in INDEX loop
      if not A(I) 0.0 then
         return FALSE;
      end if;
   end loop;
   return TRUE;
end POST_CONDITION;
```

lub konkretu jednostki rodzajowej:

```
function A_POSITIVE(I : INDEX) return BOOLEAN is
begin
   return A(I) 0.0;
end A_POSITIVE;
function POST_CONDITION is new FOR_ALL(INDEX,
A_POSITIVE);
```

[A_POSITIVE — dodatnie A, FOR_ALL — dla wszystkich]

Zauważmy, że wymagane w Adzie podawanie nazwy funkcji jako rodzajowego parametru aktualnego jest niezbyt zgrabne. Bardziej wygodne byłoby bezpośrednie napisanie dla funkcji wyrażenia lambda.

Tę samą notację można stosować dla typów nieskalarnych, nie przestrzegając zasady, że indeks pętli w Adzie musi być skalarny. Niesprzeczność łączników listy dwukierunkowej o elementach zdefiniowanych poniżej:

```
type LIST_ELEMENT;
type POINTER is access LIST_ELEMENT;
type LIST_ELEMENT is
record
   PRIOR, NEXT : POINTER;
   ——
end record;
```

[LIST_ELEMENT — element listy, POINTER — wskaźnik, PRIOR — poprzednik, NEXT — następnik]

opisuje się przez podanie warunku niesprzeczności:

```
function CONSISTENT(P : POINTER) return BOOLEAN is
begin
   return (P = null
   or else (P.NEXT.PRIOR = P and P.PRIOR.NEXT = P));
end CONSISTENT;
```

Takie sformułowanie zakłada listę kołową. Mimo że dla innych rodzajów list wyrażenie logiczne jest bardziej skomplikowane, zasada pozostaje niezmieniona. W celu stwierdzenia, że warunek jest spełniony dla wszystkich elementów listy, tj. dla wszystkich wartości wskaźników, należy dokonać konkretyzacji:

```
function POST_CONDITION is new FOR_ALL (POINTER,
      CONSISTENT);
```

Choć jest to zgodne ze składnią Ady, stanowi rozszerzenie języka, ponieważ typ POINTER nie jest typem dyskretnym, jak wymaga się w definicji rodzajowego parametru formalnego.

Inne rozszerzenie jest konieczne do zdefiniowania procedury EFFECT, w której należy powiązać początkowe i końcowe wartości zmiennych. Proponuje się użycie w tym celu takiej notacji jak dla kwalifikatorów w Adzie [1]. Zapis X'IN oznacza wartość początkową, a X'OUT wartość końcową, co sugeruje użycie trybu wejściowego (in) lub wyjściowego (out) dla odpowiedniego parametru. W ten sposób można zdefiniować skutek wymiany wartości dwóch zmiennych, np.:

```
procedure EFFECT is
begin
   ASSERT ((X 'IN=Y 'OUT) and
           (X 'OUT=Y 'IN));
end EFFECT;
```

Zwróćmy uwagę na zasadniczą różnicę między specyfikacją a algorytmem. Specyfikacja skutków dotyczy operowania wartościami IN i OUT bez żadnych ograniczeń, natomiast algorytm musi prowadzić do uzyskania wyniku jedynie przy użyciu wartości początkowych (IN). Występuje tu pewna analogia do operowania równaniem:

$$X + 1 = 0$$

w celu uzyskania rozwiązania:

$$x = -1$$

Ponadto, określenie specyfikacji nie zapewnia ani istnienia, ani jednoznaczności wyniku.

Prowadzi to do określenia nowego składnika specyfikacji programu — niezmiennika dla typu danych (ang. INVARIANT). Po wprowadzeniu typu prywatnego w pakiecie podaje się zbiór operacji stosowanych do obiektów tego typu. Informacja, która odpowiada skutkom podprogramu, stanowi niezmiennik typu. Jest to właściwość każdego obiektu tego typu w całym programie z wyjątkiem ciała pakietu.

Podprogramy wewnątrz ciała muszą traktować niezmiennik jako warunek wstępny i końcowy jednocześnie, choć zasadę tę można złamać w ciągu instrukcji stanowiącym implementację podprogramu.

PRZYKŁAD

Do przedstawienia specyfikacji według powyższych zasad służy pakiet SPECIFICATIONS, który umożliwia programiście przejście od nieformalnego opisu do specyfikacji syntaktycznej, a następnie do wyrażenia semantyki i algorytmu. Przykładowo, rozpoczęcie opisu może polegać na wyrażeniu stwierdzenia:

```
ACT („Control position of shaft
      — sterowanie położeniem wału");
```

Następny krok polega na zdefiniowaniu odnośnych obiektów i ich działania, np.:

```
type SHAFTS is new ENTITY;
procedure ROTATE is new ACTION;
```

[SHAFT — wał, ROTATE — obróć]

przy użyciu identyfikatorów wprowadzonych uprzednio nieformalnie w celu werbalnego przedstawienia opisu. Takie definicje zastępuje się następnie przez deklaracje pakietów, zawierające szczegóły zależności strukturalnych, jak np.:

```
package SHAFT_ASSEMBLY is
   type DIRECTION is (FORE, BACK, STOP);
   package MOTOR is
      procedure ROTATE (GO : DIRECTION);
   end MOTOR;
end SHAFT ASSEMBLY;
```

[SHAFT_ASSEMBLY — zespół wału, DIRECTION — kierunek, FORE — do przodu, BACK — do tyłu, MOTOR — silnik]

```
package PHYSICAL is
  type ANGLE is new FLOAT;
  type DURATION is new FLOAT;
  type LENGTH is new FLOAT;
  type MASS is new FLOAT;
  type SPEED is new FLOAT;
  type ENERGY is new FLOAT;
  type CURRENT is new FLOAT;
  type VOLTAGE is new FLOAT;
  type RATE is new FLOAT;
  type ANY is new FLOAT;
  procedure TRANSDUCER ;
  PRESENCE : BOOLEAN;
  OCCURRENCE : BOOLEAN;
end PHYSICAL;
```

Wydruk 2. Pakiet pomocniczy PHYSICAL
PHYSICAL — właściwości fizyczne, ANGLE — kąt, DURATION — czas trwania, LENGTH — długość, MASS — masa, SPEED — prędkość, ENERGY — energia, CURRENT — prąd, VOLTAGE — napięcie, RATE — tempo, ANY — dowolny, TRANSDUCER — przetwornik, PRESENCE — obecność, OCCURENCE — wystąpienie

```
with PHYSICAL;
package body SHAFT_ASSEMBLY is
  use PHYSICAL;
  package ROTATING is
    type SHAFT_PROPERTIES is
    record
      CURRENT_POSITION : ANGLE := 0.0 ;
      RATE_OF_CHANGE : RATE := 0.0 ;
    end record;
    SHAFT : SHAFT_PROPERTIES;
  end ROTATING;
  package body MOTOR is
    procedure ROTATE ( GO : DIRECTION ) is
      function POST_CONDITION return BOOLEAN is
      begin
        case GO is
        when FORE =>
          return ROTATING.SHAFT.RATE_OF_CHANGE > 0.0;
        when BACK =>
          return ROTATING.SHAFT.RATE_OF_CHANGE < 0.0;
        when STOP =>
          return ROTATING.SHAFT.RATE_OF_CHANGE = 0.0;
        end case;
      end POST_CONDITION;
    begin
      PHYSICAL.TRANSDUCER;
    end ROTATE;
  end MOTOR;

  package body ROTATING is
    task ROTATION is
      entry CHANGE ( SPEED : RATE);
      entry SENSE ( THETA : out ANGLE);
    end ROTATION;

    task body ROTATION is
      DT : PHYSICAL.DURATION;
    begin
      loop
        select
        accept CHANGE ( SPEED : RATE) do
          SHAFT.RATE_OF_CHANGE := SPEED;
        end CHANGE;
        or
        accept SENSE ( THETA : out ANGLE) do
          THETA := SHAFT.CURRENT_POSITION;
        end SENSE;
        end select;
        delay DT;
        SHAFT.CURRENT_POSITION := SHAFT.CURRENT_POSITION + SHAFT.RATE_OF_CHANGE
      end loop;
    end ROTATION;
  end ROTATING;

end SHAFT_ASSEMBLY;
```

Wydruk 3. Ciało pakietu SHAFT_ASSEMBLY
ROTATING — obracanie, SHAFT_PROPERTIES — właściwości wału, CURRENT_POSITION — położenie bieżące, RATE_OF_CHANGE — tempo zmian, GO — ruch, ROTATION — obrót, CHANGE — zmiana, SENSE — czujnik, DT (Delay Time) — czas opóźnienia

W kolejnym kroku należy sprecyzować właściwości obiektów, np. w postaci ciała pakietu przedstawionego na wydruku 3, posługując się dodatkowym pakietem PHYSICAL z wydruku 2.

* * *

W przedstawionej pracy kierowano się dążeniem do możliwie najpełniejszego użycia koncepcji i notacji języka Ada, w celu jak najlepszego wykorzystania istniejących narzędzi utworzonych dla Ady i wyjaśnienia zależności między specyfikacjami a algorytmami, między deklaracjami a ciągami instrukcji oraz między językami deklaracyjnymi a imperatywnymi. Stwierdzono, że właściwości ekspresyjne Ady są wystarczające do spełnienia większości wymagań specyfikacji funkcjonalnych, lecz stosowana notacja wydaje się raczej niezgrabna i zbyt rozwlekła.

W celu wyrażenia kwantyfikatorów ogólnych i szczegółowych dla dowolnego typu konieczne jest uogólnienie reguł obowiązujących w Adzie, polegające na zniesieniu ograniczeń dotyczących typu indeksu pętli. Obowiązujące reguły mają sens w zastosowaniu do programów przeznaczonych do wykonania, lecz można je interpretować również dla typów dowolnych w kontekście specyfikacji. Ponadto, w celu odróżnienia początkowych i końcowych wartości zmiennych pożądane jest wprowadzenie odpowiednich atrybutów.

W artykule nie poruszono innych istotnych zagadnień specyfikacji, dotyczących ograniczeń czasowych i pojemności pamięci oraz wejścia-wyjścia włącznie ze współpracą człowiek-maszyna.

Pierwotną wersję artykułu opublikowano w czasopiśmie ACM Ada Letters, Vol. 3, No. 5, pp. 63—68, 1984. Obecna postać ukazuje się po raz pierwszy.

Tłum. i oprac. JANUSZ ZALEWSKI

LITERATURA

[1] Mc Gettrick A. D.: Program Verification Using Ada. Cambridge University Press, 1982
[2] Pyle I. C.: Using Ada for Specification and Design. IUCC Bulletin, Vol. 5, pp. 74—80, 1983.

ASSOCIATION OF SIMULA USERS

Ostatnio powstały nowe kompilatory SIMULI dla komputerów VAX, firmy DEC, z systemem operacyjnym Berkeley UNIX 4.1 i 4.2 i wielu komputerów firmy Data General:

- komputery osobiste Desktop z systemem operacyjnym RDOS i AOS
- średnie minikomputery Eclipse S/C z systemem operacyjnym RDOS i AOS
- 16-bitowe minikomputery MV z systemem operacyjnym AOS/VS.

W ubiegłym roku odbyły się dwie ważne konferencje poświęcone wyłącznie SIMULI:

- Groningen, Holandia, 23—25 kwietnia 1985 r., tematyka: programowanie strukturalne,
- Calgary, Kanada, 28—30 sierpnia 1985 r., tematyka: pojęcie obiektu.

Dokładniejszych informacji związanych z SIMULĄ udziela: **The Association of Simula Users, POB 335, Blindern, Oslo 3, Norwegia.**

L. Szpor

JERZY ZAKRĘCKI
Instytut Podstaw Informatyki
Polska Akademia Nauk
Warszawa

Abstrakcje w programowaniu (I)

Historia rozwoju programowania polega w znacznej mierze na rozwoju stosowanych w nim abstrakcji i narzędzi do abstrahowania.

Terminy takie jak abstrakcja, abstrakcyjny, abstrahowanie stosuje się w informatyce już od wielu lat [1, 2], jednak w większości wypadków unika się ich ścisłego definiowania. Najczęściej wyjaśnia się je nieformalnie, podając przykłady lub odwołując sę do intuicji czytelnika. Związane jest to zapewne z faktem, że pojęcie abstrakcji w programowaniu jest szczególnym przypadkiem ogólnego pojęcia abstrakcji, którego definicję można znaleźć w każdej encyklopedii. Dla realizacji celu tego opracowania wystarczająca jest następująca definicja, będąca zawężeniem definicji abstrakcji z Małej Encyklopedii PWN: *abstrakcja (w programowaniu) jest czynnością polegającą na pomijaniu pewnych składników lub cech programu, a wyodrębnianiu innych uznawanych za istotne.*

Abstrakcja w programowaniu powstaje wskutek pominięcia szczegółów technicznych programu, nieistotnych (w danym momencie) dla opisu zadania realizowanego przez program.

PRZEGLĄD MECHANIZMÓW ABSTRAHOWANIA

Pierwszym narzędziem umożliwiającym stosowanie abstrakcji w programowaniu był język asemblera, pozwalający na tworzenie zapisu symbolicznego, np.

```
LOAD  X
ADD   Y
STORE Z
```

Dzięki temu możliwe stało się abstrahowanie od konkretnej reprezentacji rozkazów maszyny oraz adresów, a także od alokacji programu i jego zasobów.

Następnym etapem wprowadzania abstrakcji do programowania było automatyczne przekształcenie formuł na kod maszynowy. Pierwszym szeroko rozpowszechnionym językiem posiadającym takie możliwości był Fortran. Programista pisząc np.:

```
Z = X + Y
```

abstrahował dany fragment programu od wszelkich szczegółów jego maszynowej reprezentacji, takich jak: kody operacji, rejestry, adresy, alokacja zasobów programu.

Fortran umożliwiał też programowanie przy użyciu prostych abstrakcji sterowania, takich jak instrukcje IF i DO:

```
      A = 0.0
      DO 20 I = 1. 10
      IF (D(I))20, 20, 10
10    A = A + D(I)
```

Instrukcje te pozwalają na ukrycie takich elementów maszynowej reprezentacji programu jak: kody operacji, rejestry warunków itp., aczkolwiek punkty kontrolne programu (w naszym przykładzie etykiety 10 i 20) pozostają nadal widoczne.

Ta niewątpliwa wada języka Fortran została usunięta w późniejszych językach programowania. Powstały takie instrukcje jak: for. if-then-else (Algol 60), while, repeat, case (Pascal). W następującym fragmencie programu wyrażonym w języku Algol 60:

```
A := 0.0;
for I := 1 step 1 until 10 do
  if D[I] > 0.0 then A := A + D[I];
```

przebieg sterowania jest opisany w taki sposób, że punkty kontrolne programu nie są ani nazywane, ani numerowane.

Pierwsze języki programowania dostarczały też prostych środków do deklarowania danych programu — tzw. typów, np.:

```
FLOAT A, D(10)
INTEGER I
```

w języku Fortran lub

```
real A;
real array[1:10] D;
integer I;
```

w języku Algol 60.

Typ rozumiany jest tutaj jako zbiór wartości i zestaw operacji do działań na tych wartościach. Typy pozwalają na opis zasobów programu w oderwaniu od ich reprezentacji maszynowej.

Warto zwrócić uwagę na konstrukcje do deklarowania tablic: D(10) i **array** [1:10]. Są to przykłady typów sparametryzowanych. W przypadku korzystania z typów sparametryzowanych (zwanych też konstruktorami typów) konkretny typ danych (tzn. konkretna abstrakcja służąca do opisu zasobów programu) otrzymuje się poprzez podstawienie do konstruktora typów parametrów aktualnych. Przykładowo, w celu zdefiniowania konkretnego typu tablicowego w języku Pascal:

```
VECT = array [1..10] of REAL
```

konieczne jest określenie typu indeksowego i typu elementów tablicy (w naszym przykładzie są to odpowiednio 1..10 i REAL).

Wszystkie wymienione abstrakcje charakteryzują się tym, że są zdefiniowane przez projektanta języka programowania, w którym występują, a programista nie ma żadnego wpływu na ich właściwości. Istnieją też konstrukcje językowe pozwalające programiście na definiowanie jego własnych abstrakcji.

Pierwszą konstrukcją tego rodzaju była procedura. W różnych językach programowania konstrukcja ta występuje

Dr JERZY ZAKRĘCKI ukończył w 1980 r. studia na Wydziale Matematyki Stosowanej i Cybernetyki Uniwersytetu Moskiewskiego. Pracuje w Instytucie Podstaw Informatyki PAN. Zajmuje się zagadnieniami modularyzacji i projektowania programów. Specjalizuje się w programowaniu mikrokomputerów.

pod różnymi nazwami: w Fortranie — SUBROUTINE i FUNCTION, w Algolu 60 — **procedure** itd., jednak we wszystkich przypadkach jej sens jest jednakowy. W języku Pascal deklaracja i wywołanie procedury ma następującą postać:

```
procedure SUMPOSEL(V: WEKTOR; var WYNIK: REAL);
  var I: 1..10;
begin
  WYNIK := 0.0;
  for I := 1 to 10 do if V[I]>0 then WYNIK := WYNIK + V[I]
end;

SUMPOSEL(D,R)
```

Procedura pozwala na zwięzłe, parametryczne odwołanie się do pewnego fragmentu programu, tzw. ciała procedury. Tym samym istnieje możliwość wyabstrahowania funkcji realizowanej przez procedurę.

Najnowszymi rodzajami abstrakcji stosowanymi w programowaniu są abstrakcyjne typy danych i obiekty abstrakcyjne. W niektórych nowych językach programowania występują narzędzia do definiowania abstrakcji sterowania (np. konstrukcja iterator w języku CLU [3]), jednak propozycje te nie są jeszcze wystarczająco dopracowane.

Typ abstrakcyjny jest to typ całkowicie zdefiniowany przez programistę. Definiuje się reprezentacje typu, jak również zbiór operacji na danych tego typu. Definicja abstrakcyjnego typu danych dla liczb wymiernych zapisana w języku Ada ma postać następującą:

```
package RATIONAL-ARITHMETIC is
  type   RATIONAL   is private -- deklaracja typu z niedostepna
                               -- reprezentaca

  function CREATE(N: INTEGER; D: POSITIVE) return RATIONAL;
  function SUM(I, J: RATIONAL) return RATIONAL;
  function MULT(I, J: RATIONAL) return RATIONAL;

  private
    type RATIONAL is record
                       N: INTEGER;
                       D: POSITIVE
                     end record;

end RATIONAL_ARITHMETIC;
```

Specyfikacja ta zawiera opis zasobów dostarczanych przez pakiet. Opis składa się z dwóch części: jawnej i prywatnej.

Część jawna zawiera informacje niezbędne dla użytkownika pakietu, nazwę typu RATIONAL i nagłówki procedur (operacji wykonywanych na danych typu RATIONAL) CREATE. SUM. MULT.

Część prywatna zawiera informacje dla kompilatora, a użytkownik pakietu nie może z niej korzystać. Ta niezbyt elegancka konstrukcja została włączona do języka Ada wyłącznie ze względu na wymaganie dotyczące dużej efektywności kodu wynikowego.

Obecnie, tylko niewiele języków programowania posiada konstrukcje ułatwiające definiowanie abstrakcyjnych typów danych i obiektów abstrakcyjnych, jednak stosując konsekwentnie określoną dyscyplinę kodowania programu można programować w tym stylu i w konwencjonalnych językach programowania.

UŻYCIE ABSTRAKCJI W FORTRANIE I W ADZIE

Istniejące języki programowania można poddać ocenie z punktu widzenia dostarczanych przez nie środków do programowania w abstrakcjach. Poniżej rozważymy Fortran i Adę. Ich wybór nie jest przypadkowy — Fortran jest pierwszym szeroko rozpowszechnionym językiem wysokiego poziomu, a Ada ma bogaty repertuar środków do programowania w abstrakcjach, będący w pewnym sensie zakończeniem linii rozwoju proceduralnych języków programowania zapoczątkowanej przez Fortran. W każdym z tych języków napiszemy program złożony z dwóch części:

- definicji typu abstrakcyjnego dla liczb wymiernych
- przykładu użycia zdefiniowanej abstrakcji.

Wybór przykładu podyktowany był przeświadczeniem, że ocena zastosowania rozpatrywanych języków do zaprogramowania dość złożonej abstrakcji najpełniej uwydatni modularyzacyjne właściwości tych języków.

JĘZYK FORTRAN

Program w Fortranie składa się z sześciu podprogramów. Podprogramy CREATE, SUM, MULT, EQ i ASSIGN odpowiadają operacjom typu abstrakcyjnego, podprogram GCD służy do wyznaczania największego wspólnego dzielnika dla dwóch liczb całkowitych i jest wywoływany tylko z podprogramu CREATE. Ostatni podprogram jest przykładem wykorzystania zdefiniowanej abstrakcji.

```
C  wartosci  wymierne reprezentowane sa  jako  dwuelementowe
C  tablice typu INTEGER;  pierwszy element  tablicy  zawiera
C  licznik, drugi mianownik

C  operacja tworzenia liczby wymiernej
      SOBRUTINE CREATE(N, D, RES)
      INTEGER N, D, RES(2), A
      IF (D)  77, 77, 1
    1 IF (N) 13, 11, 12
   11 RES(1) = 0
      RES(2) = 1
      RETURN
   12 CALL GCD(N, D, A)
      GOTO 14
   13 CALL GCD(-N, D, A)
   14 RES(1) = N/A
      RES(2) = D/A
      RETURN
   77 CALL ERROR
      END

C   pomocniczy   podprogram  do   wyznaczania   najwiekszego
C   wspolnego dzielnika
      SUBROUTINE GCD(N, D, RES)
      . . . . . .
      END

C  operacja sumowania liczb wymiernych
      SUBROUTINE SUM(I, J, RES)
      INTEGER I(2), J(2), RES(2)
      CALL CREATE (I(1)*J(2)+J(1)*I(2),  I(2)*J(2), RES)
      RETURN
      END

C  operacja mnozenia liczb wymiernych
      SUBROUTINE MULT(I, J, RES)
      INTEGER I(2), J(2), RES(2)
      CALL CREATE(I(1)*J(1), I(2)*J(2), RES)
      RETURN
      END

C  operacja porownywania liczb wymiernych
      LOGICAL FUNCTION EQ (I, J)
      . . . . . .
      END

C  operacja przypisania liczb wymiernych
      SUBROUTINE ASSIGN(L, R, RES)
      . . . . . .
      END


C program korzystajacy z abstrakcji
      INTEGER A(2), B(2), C(2)
      CALL CREATE(5, 6, A)
      CALL CREATE(1, 2, B)
      CALL SUM(A, B, C)
      IF EQ(A, B) CALL ASSIGN(B, C)
      END
```

Zaletą tego programu (i języka, w którym został napisany) jest duża niezależność zdefiniowanych w nim abstrakcji od innych wykorzystujących programów. Użytkownik i implementator abstrakcji powinni uzgodnić tylko sposób reprezentowania wartości wymiernych i nagłówki operacji do ich przetwarzania. Po dokonaniu tego, definicje abstrakcji i programy korzystające z nich mogą być tworzone, modyfikowane, kompilowane i testowane całkowicie niezależnie. Oznacza to, m.in., że wszelkie modyfikacje w definicji abstrakcji nie wywołujące zmian w nagłówkach operacji nie powodują konieczności zmian w podprogramach korzystających z abstrakcji. Korzystne jest też istnienie konstrukcji SUBROUTINE i FUNCTION umożliwiających tworzenie abstrakcji proceduralnych.

Program ten ma jednak wiele wad (z punktu widzenia modularyzacji), związanych przede wszystkim z niedoskonałością języka, w którym został napisany.

1. Język, a raczej jego implementacja, nie posiada żadnych środków do kontroli prawidłowości odwołania się do podprogramów. W naszym przykładzie możliwe byłoby odwołanie się do operacji CREATE w następujący sposób:

— CALL CREATE,
— CALL CREATE(5),
— CALL CREATE(5, 21, 0, 32, R, S) itp.

2. W podprogramach definiujących typy abstrakcyjne i w podprogramach odwołujących się do nich konieczne jest powtarzanie opisu reprezentacji danych tego typu. W naszym przykładzie, w celu zadeklarowania zmiennej A o typie „liczba wymierna" konieczne było zdeklarowanie tej zmiennej jako dwuelementowej tablicy typu INTEGER:

INTEGER A(2)

Właściwość ta, poza jej oczywistą niewygodą, może być przyczyną błędów powstających przy przepisywaniu opisu reprezentacji, zwłaszcza w przypadku złożonych struktur danych. Błędy te nie będą w żaden sposób sygnalizowane przez kompilator.

3. Możliwe jest nielegalne działanie na strukturach danych przeznaczonych do przechowywania wartości typów abstrakcyjnych i obiektów abstrakcyjnych, polegające na modyfikowaniu zawartości tych struktur przy użyciu operacji różnych od przewidzianych do tego celu w definicji abstrakcji. Działania tego rodzaju mogą doprowadzić do całkowitego zaburzenia poprawności funkcjonowania abstrakcji, np. po wykonaniu przypisania:

A(1) = 3, A(2) = 6, B(1) = 1, D(2) = 2

funkcja EQ (A, B) udostępni wartość FALSE., natomiast po wykonaniu analogicznej sekwencji:

CREATE (3,6,A), CREATE (1,2,B)

funkcja EQ (A, B) udostępni wartość TRUE.

4. Definicja abstrakcji nie stanowi zwartej całości, składa się z kilku podprogramów, które mogą znajdować się w różnych miejscach tekstu programu.

5. Niemożliwe jest zabronienie dostępu do niektórych składowych programu. Podprogram GCD może być wywoływany z każdego miejsca programu, natomiast naszą intencją było wywoływanie go tylko z podprogramu CREATE.

6. Niemożliwy jest opis operacji na liczbach wymiernych za pomocą konstrukcji FUNTION (jej użycie w przykładzie byłoby bardziej naturalne). Związane jest to z ograniczeniami języka — w Fortranie typem wyniku funkcji może być tylko: REAL, DOUBLE PRECISION, INTEGER lub LOGICAL.

LITERATURA

[1] Barnes J. G. P.: Programming in Ada. Addison-Wesley Publishing Company, London, 1982

[2] Dijkstra E. W.: A discipline of programming. Prentice-Hall, Englewood Cliffs (NJ), 1976

[3] Liskov B. et al.: CLU Reference Manual. Lecture Notes in Computer Science No. 114, Springer-Verlag, Berlin, 1981.

WITOLD ABRAMOWICZ

Modularny system wyszukiwawczy

Informację można traktować jako surowiec lub środek do własnej działalności, a także jako produkt, na którym można zarobić. Dlatego zbieranie, przechowywanie i ponowne wyszukiwanie informacji jest przedmiotem zainteresowania specjalistów różnych dziedzin. Metody umożliwiające efektywne przetwarzanie informacji w powyższym sensie są przedmiotem zainteresowania teorii systemów wyszukiwawczych.

Celem opracowania koncepcji modularnego systemu wyszukiwawczego MIRS (ang. Modular Information Retrieval System) było umożliwienie studentom informatyki Politechniki Federalnej w Zurychu praktycznego poznania metod stosowanych w systemach wyszukiwawczych. Modułowa budowa systemu umożliwia wymienianie części oprogramowania. Dzięki temu można pokazać praktycznie działanie różnych metod i w konsekwencji porównać je.

Podstawowym celem pracy było przygotowanie odpowiedniego środka dydaktycznego, jednak w praktyce wykorzystano system także do modelowania struktur informacji, badania algorytmów wyszukiwania oraz testowania rozwiązań dialogu człowieka z komputerem.

W trakcie pracy ze studentami podkreślałem też bardzo mocno metodologię programowania i problematykę organizacji pracy w grupie projektowej. Te aspekty nie będą jednak poruszane w tym artykule. Są one wszakże często niedoceniane, choć w dużej mierze decydują o sukcesie zawodowym inżyniera informatyka. Moim celem było zatem również zaprezentowanie różnych metod projektowania i programowania systemów informatycznych [22, 30].

Warunki realizacji

System MIRS był zaimplementowany przez studentów Wydziału Informatyki Politechniki Federalnej w Zurychu w ramach zajęć nieobowiązkowych „Systemy wyszukiwawcze". Zajęcia te składały się z trzech części:

1) wykładu (2 godziny w tygodniu),

2) seminarium, na które zapraszano przedstawicieli przemysłu w celu prezentacji gotowych systemów informacyjnych (głównie systemów wspomagania prac biurowych np. [12, 20, 23]),

3) zajęć projektowych związanych bezpośrednio z systemem MIRS.

W trakcie opracowywania koncepcji zajęć współpracowałem początkowo z jednym, a później z dwoma asystentami. Eksperymentalnej realizacji systemu dokonał w ramach pracy semestralnej student, posiadający kilkuletnią praktykę projektową w firmach software'owych. Pełne przygotowanie koncepcji, testy oraz opracowanie odpowiednich pomocy dla studentów [3] zajęło ok. 850 godzin pracy (nie licząc pracy semestralnej — ok. 400 godzin pracy).

Fakultatywność zajęć pozwalała założyć wysoką motywację studentów, a w konsekwencji zwiększać wymagania wobec nich. Studenci dobierali się w 2—4-osobowe grupy. W przeprowadzonej pod koniec semestru ankiecie oświadczyli, że na realizację systemu poświęcili od 150 do 520 godzin pracy (licząc na grupę). Ze zdziwieniem stwierdziłem, że studenci nie uważali obciążenia zajęciami za zbyt duże, mimo iż wielu z nich budowało równolegle kompilator języka PL/O [29]. W wyniku otrzymaliśmy 9 bardzo różnych systemów od najprostszych po rozbudowane, charakteryzujących się starannością modelowania struktur danych lub bardzo oryginalnymi rozwiązaniami dialogu człowieka z komputerem.

System MIRS był wykonywany na komputerach osobistych ze względu na powszechną dostępność tych komputerów dla studentów. Ponadto, ponieważ system opracowano głównie dla celów dydaktycznych, nie było istotne zapewnienie możliwości organizacji bardzo dużych zbiorów informacji jakie normalnie występują w systemach wyszukiwawczych. Innym powodem użycia komputerów osobistych było duże doświadczenie zespołu w pracach nad systemami wyszukiwawczymi realizowanymi na tych komputerach w sieciach lokalnych (np. [4, 5, 9, 14]).

Wszystkie systemy zrealizowano na komputerze Lilith [28], wyposażonym w cztery procesory segmentowe Am2901,

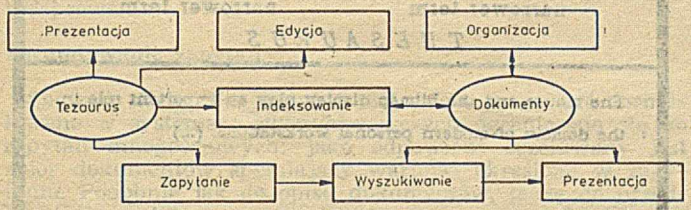
Dr inż. Witold ABRAMOWICZ jest absolwentem kierunku: cyfrowe systemy sterowania Politechniki Poznańskiej. Zajmuje się systemami wyszukiwawczymi i sztuczną inteligencją. Praca doktorska (z wyróżnieniem) dotyczyła kontekstowo zorientowanego rozpowszechniania i organizacji informacji w sieciach komputerowych. Interesuje się także inżynierią oprogramowania oraz problematyką dialogu człowiek-maszyna.

pamięć operacyjną o pojemności 128K słów 16-bitowych, wymienną pamięć dyskową o pojemności 10 MB lub stałą o pojemności 15 MB i pamięć mikrokodu o pojemności 2K instrukcji. Bardzo istotne były urządzenia wejścia-wyjścia mające duży wpływ na rozwiązanie dialogu człowieka z komputerem: monochromatyczny monitor graficzny o rozdzielczości 892×640 punktów (ang. bit map) i mysz.

Komputery pracowały pod nadzorem systemu operacyjnego Medos-2 [17], który jest systemem otwartym, umożliwiającym użytkownikowi dostęp do wszystkich zasobów. Do programowania użyto Moduli-2 [2]. Wykorzystano dwie techniki dialogowe, których zastosowanie w komputerze Lilith można zaliczyć do pionierskich, a które obecnie są stosowane w wielu innych systemach (np. MacIntosh, SunStation): technikę okien i technikę menu.

Struktura systemu MIRS

Strukturę systemu MIRS przedstawiono na rys. 1. Owalnym kształtem oznaczono struktury danych: opis wiedzy i gromadzone informacje. Do opisu wiedzy zastosowano w systemie kontrolowaną strukturę danych (tzn. strukturę zawierającą z góry określony zasób termów i powiązań pomiędzy nimi), niezależną od gromadzonych informacji. Jako kompromis między prostym (np. listy lub klasyfikacja) a rozbudowanym opisem wiedzy (np. sieci semantyczne [27] lub sieci Crofta [11]) wybrano tezaurus [8]. Informacje gromadzone są w postaci wzajemnie niezależnych dokumentów. Dokument nie posiada własnej struktury (traktowany jest jako ciąg znaków).



Rys. 1. Struktura systemu MIRS

Funkcje realizowane w systemie MIRS są reprezentowane na rys. 1 przez prostokąty. Najprostszą funkcją jest prezentacja struktury tezaurusa. W celu jego dostosowania do indywidualnych potrzeb i wyobrażeń użytkowników przewidziano możliwość edycji. Są to działania dotyczące tylko pierwszej klasy danych — opisu wiedzy. Gromadzone dokumenty muszą być również udostępniane użytkownikowi (prezentacja dokumentów) oraz zarządzane (np. usuwane z systemu — organizacja). Połączenie pomiędzy obiema klasami danych — dokumentami i tezaurusem — tworzone jest przez opisywanie dokumentów pojęciami tezaurusa (indeksowanie). Użytkownik pragnący znaleźć interesujące go dokumenty formułuje zapytanie wybierając odpowiadające

```
LOOP
  CASE CommandsOfMIRS OF
  | Thesaurus-Editor :
  LOOP
    CASE CommandsOfThesaurus-Editor OF
    | GoToTerm:
    | DeleteTerm:
    | InsertTerm:
    | ....
    | Quit: EXIT (*from Thesaurus-Editor *)
    END (* case *);
  END (* loop *);
  | AnotherMode :
  LOOP
    CASE CommandsOfAnotherMode OF
    | ....
    | Quit: EXIT (*from AnotherMode *)
    END (* case *);
  END (* loop *);
        ....
  | Quit: EXIT (*from MIRS *)
 END (* case *);
END (* loop *);
```

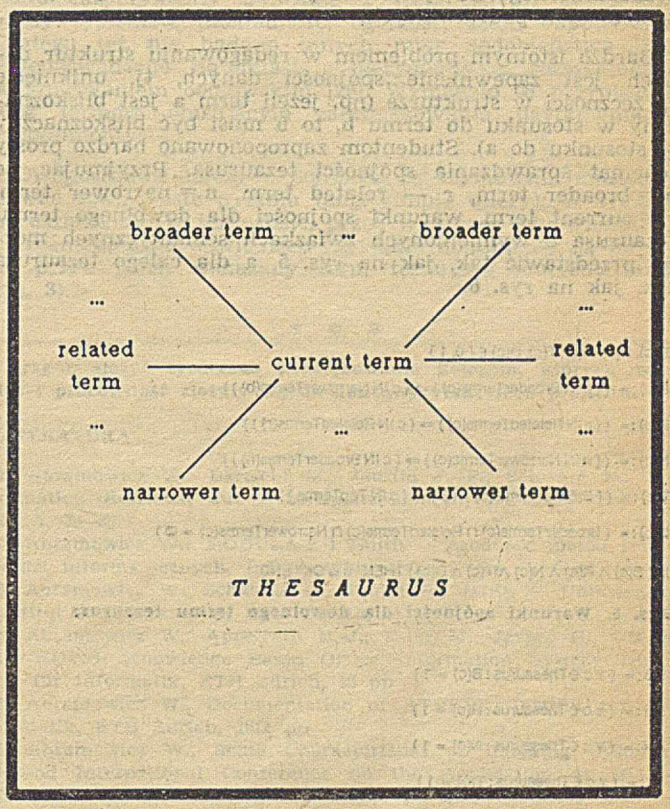
Rys. 2. Interpreter poleceń

mu pojęcia z tezaurusa. Na tej podstawie zostają znalezione odnośne dokumenty (wyszukiwanie).

Dialog w systemie jest sterowany przez moduł zwany interpreterem poleceń. Jego główną część przedstawiono w skróconej formie na rys. 2. Interpreter poleceń jest także odpowiedzialny za komunikacje ze wszystkimi innymi modułami systemu. Jest on jedyną częścią systemu, która nie powinna ulegać zmianom.

Prezentacja opisu wiedzy

Wiedza w systemie jest opisywana tezaurusem, w którym zgromadzone są termy. Term jest słowem lub frazą (szeregiem słów). Tezaurus zawiera także związki semantyczne pomiędzy termami. Dla każdego termu (pojęcia) może istnieć wiele podpojęć, pojęć nadrzędnych, synonimów, antonimów, homografów, homonimów oraz termów bliskoznacznych. Spośród wielu dostępnych wybrano tezaurus ERIC (Educational Resources Information Center) [13] opisujący wiedzę z wielu dziedzin. Do celów dydaktycznych oryginalny tezaurus o objętości ok. 5 MB zredukowano do 96 kB korzystając z edytora struktur nie będącego częścią składową systemu MIRS [1]. ERIC przedstawia nieliczne związki semantyczne pomiędzy termami (rys. 3).



Rys. 3. Związki semantyczne między termami; zbiór poleceń tezaurusa obejmuje m.in.: go_to_term, show_top_term, quit itp.

Tezaurus jest pamiętany w bardzo prosty sposób. Termy nie posiadające pojęć nadrzędnych są zawarte w liście tzw. TopTerms. Właściwa struktura tezaurusa jest przedstawiona jako tekst o formacie podanym na rys. 4.

```
Terms :: = { TFlag TermField EOL } .
TermField :: =
        Term EOL
        [ BFlag BroaderTerm EOL { BroaderTerm EOL} ]
        [ NFlag NarrowerTerm EOL { NarrowerTerm EOL } ]
        [ RFlag RelatedTerm EOL { RelatedTerm EOL } ] .
BFlag :: = 'b' .
NFlag :: = 'n' .
RFlag :: = 'r' .
TFlag :: = 't' .
EOL :: = EndOfLine .
```

Rys. 4. Format tezaurusa

Lista ta umożliwia graficzne przedstawienie tezaurusa według rys. 3. Po inicjalizacji systemu użytkownikowi przedstawiona jest lista TopTerms, którą można przeglądać (ang. scrolling). Z niej wybiera się pożądany term (ang. current

term, rys. 3). Zostaje on pokazany na monitorze z graficznym wskazaniem termów będących z nim w związku semantycznym. Użytkownik może poznać strukturę tezaurusa wybierając jako current term coraz to inny term.

Inna bardzo ciekawa i nowatorska metoda przedstawiania tezaurusa jako tzw. grid file [19], bazuje na pracy [16]. Rozwiązanie to, opracowane przez studentów, umożliwia bardzo szybkie poruszanie się w tezaurusie.

Edycja opisu wiedzy

Redagowanie tezaurusa sprowadza się do umiejętności przeprowadzenia czterech podstawowych operacji:

- wprowadzenie termu do TopTerms,
- usunięcia termu z tezaurusa,
- wprowadzenia związku semantycznego między dwoma wybranymi termami (rozpatrujemy tylko dwuargumentowe związki semantyczne),
- usunięcia związku semantycznego między dwoma wybranymi termami.

Edytor musi zatem wspomagać wprowadzanie i usuwanie węzłów (termów) i krawędzi grafu (związki semantyczne) przedstawionego na rys. 3. Wszystkie inne operacje są złożeniem tych podstawowych.

Bardzo istotnym problemem w redagowaniu struktur danych jest zapewnienie spójności danych, tj. uniknięcie sprzeczności w strukturze (np. jeżeli term **a** jest bliskoznaczny w stosunku do termu **b**, to **b** musi być bliskoznaczny w stosunku do **a**). Studentom zaproponowano bardzo prosty schemat sprawdzania spójności tezaurusa. Przyjmując, że **b** = broader term, **r** — related term, **n** = narrower term, **c** = current term, warunki spójności dla dowolnego termu tezaurusa o wymienionych związkach semantycznych można przedstawić tak, jak na rys. 5, a dla całego tezaurusa tak, jak na rys. 6.

```
B(c), R(c), N(c), T(c), D(c) ∈ { 0, 1 }
B(c) := ((b IN BroaderTerms(c)) ⇒ (c IN NarrowerTerms(b)))
R(c) := ((r IN RelatedTerms(c)) ⇒ (c IN RelatedTerms(r)))
N(c) := ((n IN NarrowerTerms(c)) ⇒ (c IN BroaderTerms(n)))
T(c) := ((BroaderTerms(c) = ∅) ↔ (c IN TopTerms))
D(c) := (BroaderTerms(c) ∩ RelatedTerms(c) ∩ NarrowerTerms(c) = ∅)
IF (B(c) ∧ R(c) ∧ N(c) ∧ T(c) ∧ D(c)) THEN "c is OK" END
```

Rys. 5. Warunki spójności dla dowolnego termu tezaurusa

```
BB := (∀c ∈ Thesaurus : B(c) = 1)
RR := (∀c ∈ Thesaurus : R(c) = 1)
NN := (∀c ∈ Thesaurus : N(c) = 1)
TT := (∀c ∈ Thesaurus : T(c) = 1)
DD := (∀c ∈ Thesaurus : D(c) = 1)
IF (BB ∧ RR ∧ NN ∧ TT ∧ DD) THEN "thezaurus is OK" END
```

Rys. 6. Warunki spójności dla całego tezaurusa

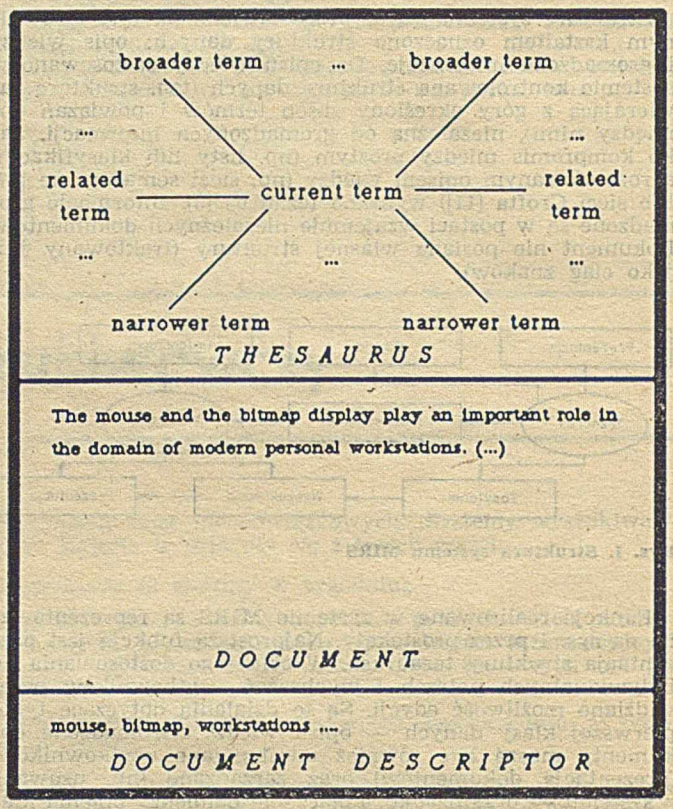
Z tak sformułowanych warunków spójności wynikają następujące zachowania:

- wprowadzenie termu do TopTerms nie pociąga za sobą żadnych innych działań,
- usuwając term **c** z tezaurusa należy usunąć wszystkie połączenia z innymi termami, tzn. Broader-Terms(c). RelatedTerms(c), NarrowerTerms(c), a także wszystkie odwrotne związki semantyczne (por. następne dwa warunki),
- wprowadzenie związku semantycznego pomiędzy dwoma wybranymi termami (rozpatrujemy tylko dwuargumentowe związki semantyczne) pociąga za sobą wprowadzenie także odwrotnego związku semantycznego, tzn.
  bIN BroaderTerms(c)→CIN NarrowerTerms(b)
  rIN RelatedTerms(c)→CIN RelatedTerms(r)
  nIN NarrowerTerms(c)→CIN BroaderTerms(n)
- usunięcie związku semantycznego między dwoma wybranymi termami pociąga za sobą usunięcie odwrotnego związku semantycznego.

Zarządzanie dokumentami

W systemie MIRS zorganizowane są dokumenty będące krótkimi tekstami bez określonej wewnętrznej struktury. Dokumenty są dostępne w formie czytelnej dla systemu liczącego. Przykładowymi dokumentami mogą być opisy płyt z domowej kolekcji, opisy patentowe gromadzone przez radcę patentowego czy historie chorób w prywatnej praktyce lekarskiej.

System musi umożliwić przeprowadzenie kilku podstawowych operacji związanych z zarządzaniem dokumentami. Są one przedstawione na rys. 7. W górnej części rysunku znajduje się okno, w którym przedstawiona jest struktura tezaurusa (por. rys. 3). W środkowym oknie przedstawiony jest każdorazowo jeden dokument. Znaczenie dolnego okna zostanie objaśnione w następnym punkcie.



Rys. 7. Ilustracja zarządzania zbiorem dokumentów
— polecenia dotyczące struktury tezaurusa: thesaurus_editor — delete_term, ..., guit; indexing — choose_term_for_descriptor, ..., quit
— polecenia dotyczące struktury dokumentów: insert_document, delete_document, next_document, previous_document, jump_to_document
— polecenie dotyczące deskryptora dokumentu: delete_term

Użytkownik musi mieć możliwość włączenia dowolnego dokumentu do systemu (polecenie insert_document). Redagowanie dokumentów nie jest przedmiotem zainteresowania w systemie MIRS. Korzysta się tu ze standardowego systemu redagowania dokumentów Lara [15]. Dokumenty, które nie są użyteczne, usuwa się (polecenie delete_document). Podobnie jak przy poznawaniu tezaurusa użytkownik może także przeglądać dokumenty przez żądanie następnego (polecenie next_document) lub poprzedniego (polecenie previus_document). Studenci określili w bardzo różny sposób porządek w strukturze dokumentów: od chronologicznego do wynikającego z częstotliwości wyszukiwania dokumentów.

Indeksowanie dokumentów

Aby opisać dokument w możliwie krótkiej formie, przyporządkowywane są mu deskryptory odzwierciedlające jego wartość. Przydzielanie dokumentowi deskryptorów nazywa się indeksowaniem. Rozważono tylko deskryptory będące termami tezaurusa. W systemie MIRS nie przewidziano deskryptorów wolnych (wyrażeń języka naturalnego) i deskryptorów formalnych (np. autor i objętość dokumentu).

Indeksowanie wolnymi deskryptorami nie jest zgodne z założeniami systemu (tezaurus jako pomost łączący potrzeby użytkownika i informacje zgromadzone w systemie). Indeksowanie deskryptorami formalnymi jest charakterystyczne raczej dla baz danych.

Indeksowanie jest procesem subiektywnym, mającym decydujące znaczenie dla poprawnego działania systemów wyszukiwawczych. Bez poprawnego indeksowania jednostki informacji nie mogą być poprawnie znalezione w systemie. Deskryptory mogą zostać wybrane automatycznie lub samodzielnie przez użytkownika. W systemie MIRS przewidziano obie sytuacje. Było to uzasadnione zamiarem prezentacji wad różnych sposobów indeksowania. Omówienie tych eksperymentów wykracza jednak poza ramy tego artykułu [6].

Użytkownik samodzielnie indeksujący dokument wybiera z tezaurusa termy (polecenie choose_term). Jeżeli użytkownik chce opisać dokument termami nie należącymi do tezaurusa, to najpierw musi z pomocą edytora wprowadzić term do tezaurusa, a następnie wybrać go jako deskryptor. Takie rozwiązanie odpowiada ścisłemu rozgraniczeniu funkcji systemu (por. rys. 1). Deskryptory są przedstawione w dolnym oknie na rys. 7. Niewłaściwie deskryptory można usunąć z listy (polecenie delete_term).

Do automatycznego indeksowania dokumentu zaproponowano powszechnie znane, proste metody wyszukiwania termów w tekstach [7, 10, 18]. Ze względu na poważne wady tych metod (np. nieuwzględnienie analizy semantycznej, można je stosować tylko jako wspomaganie interakcyjnego indeksowania.

Formułowanie zapytań

Spośród wielu klas zapytań [21], które mogą być formułowane w systemach wyszukiwawczych ograniczono się do zapytań mnogościowych: jako odpowiedź oczekiwany jest zbiór dokumentów spełniający warunki określone w zapytaniu. Podobnie jak do opisu dokumentów, także do przedstawienia zainteresowań użytkownika używa się termów tezaurusa. W celu uniknięcia pisania interpreterów zapytań zaproponowano studentom bardzo prosty schemat dla dopuszczalnych zapytań (np. 8).

```
query :: = simplequery [ operator simplequery].
simplequery :: = [NOToperator] term.
NOToperator :: = NOT.
operator :: = OR / AND.
```

Rys. 8. Format zapytań

Użytkownik formułując zapytanie porusza się w tezaurusie podobnie jak w trakcie poznawania tezaurusa lub interakcyjnego indeksowania dokumentów. Interesujące termy wybiera się do zapytania przez wskazanie odpowiedniego polecenia. Operatory logiczne określane są przez wskazanie polecenia w menu.

Wyszukiwanie

Najbardziej kontrowersyjną funkcją systemów wyszukiwawczych jest odszukiwanie odnośnych dokumentów, tzw. relewantnych w stosunku do zapytania. Kontrowersyjność ta wynika z subiektywności oceny relewantności informacji (nawet jeżeli dwóch użytkowników sformułuje jedno zapytanie, to dla jednego wyszukana informacja jest relewantna, a dla drugiego nie). Do udzielenia odpowiedzi na zapytania opracowano wiele algorytmów. Relewantność dokumentu w stosunku do zapytania może zostać określona dwoma sposobami.

- Zbiór dokumentów zostaje podzielony na dwa rozłączne podzbiory: dokumentów relewantnych w stosunku do zapytania i dokumentów nie mających znaczenia.
- Zbiór dokumentów zostaje uporządkowany w listę na podstawie podobieństwa pomiędzy poszczególnymi dokumentami i zapytaniem. Na początku listy znajdują się dokumenty, które są najbardziej podobne do zapytania, na końcu umieszczone są te, które są najmniej podobne. Następnie decyduje się, ile dokumentów z początku listy będzie pokazane użytkownikowi (tzn. ile będzie uznane za relewantne).

Aby umożliwić wyszukiwanie relewantnych dokumentów na pierwszy sposób, w praktyce stosuje się najczęściej metodę list inwersyjnych. Składają się one z listy dokumentów zawierającej cały ich zbiór oraz z listy termów zawierającej wszystkie termy oraz wskaźniki do dokumentów indeksowanych przez każdy z tych termów. W przypadku zapytania złożonego z jednego termu, w liście termów wyszukuje się wskaźniki od tego termu do dokumentów. Wskazane dokumenty zostają pokazane użytkownikowi jako relewantne. W przypadku zapytania złożonego z większej liczby termów, na listach wskaźników przeprowadza się operacje zgodne z operatorami logicznymi wiążącymi termy zapytania. Inna szybka metoda korzystania ze struktury termów i dokumentów określana jest mianem list łańcuchowych. Od każdego z termów wiedzie jeden wskaźnik pokazujący pierwszy dokument opisany przez ten term. Od tego dokumentu wskaźnik wskazuje na drugi dokument indeksowany przez ten sam term itd. W celu udzielenia odpowiedzi na zapytanie, przeprowadza się operacje na listach łańcuchowych, prowadzących od termów zapytania zgodnie z operatorami logicznymi zapytania.

W wyszukiwaniu relewantnych dokumentów drugim sposobem zastosowano cztery podstawowe miary podobieństwa pomiędzy zapytaniem i dokumentem: cosinus, miarę nakładania, miarę Dice'a i Jaccarda [25]. Wobec założenia bardzo małego zbioru dokumentów, można przeprowadzić tylko eksperymenty związane z technicznymi kryteriami oceny systemów (dokładność, ogólność, szum odpowiedzi). Dlatego też nie badano innych miar podobieństwa [24]. Przeprowadzono natomiast eksperymenty z rozszerzonym wyszukiwaniem boolowskim [26]. Wymagało to przypisania wag termom opisującym zainteresowania użytkowników i dokumenty.

Dla porządku należy wspomnieć, że użytkownik może zadawać tzw. proste pytania (ang. simple query). W trakcie poruszania się w tezaurusie, przez wybranie odpowiedniego polecenia można dowiedzieć się, jakie dokumenty są opisane przez jeden wskazany term tezaurusa (current term rys. 3).

* * *

Pragnę złożyć serdeczne podziękowanie kolegom, których wiedza i pracowitość złożyły się na końcowy efekt tego projektu.

LITERATURA

[1] Abramowicz W., Bärtschi M., Jauslin J. F., Schwarz E.: Information Retrieval auf Arbeitsplatzrechnern, OUTPUT Vol. 13, No. 7, 33—36
[2] Abramowicz W.: MODULA-2 i Lilith — zgodność metod i narzędzi informatycznych, Informatyka, Vol. 19, No. 4, 1—15
[3] Abramowicz W., Schäuble P., Teufel B.: MIRS — Unterlagen, Institut fur Informatik, ETH Zürich, 26 pp
[4] Abramowicz W., Appelrath H,-J., Ester M., Jasper H., Ultsch A.: KOFIS: Knowledge Based Office Information System, Institut für Informatik, ETH Zürich, 18 pp
[5] Abramowicz W., Documentation of ATHENE, Institut fur Informatik, ETH Zurich, 1012 pp
[6] Abramowicz W., Some Characteristics of Indexing Systems, Second International Conference on the Application of Micro-Computers in Information, Documentation and Libraries, Baden-Baden (do druku)
[7] Aho A. V., Carasick M. J., Efficient String Matching: an Aid to Bibliographic Search, Communications of the ACM, Vol. 18, June
[8] Aitchison J., Gilchrist A., Thesaurus Construction, Aslib, London, 95 pp
[9] Appelrath H.-J. ODIR: Optical Disk Information Retrievel, Datenbanksysteme in Buro, Technik und Wissenschaft, Karisruhe, 20—22. Marz
[10] Boyer R. S., J. S. Moore, A Fast String Searching Algorithm Communications of the ACM, Vol. 20, October, pp. 762—772
[11] Croft W. B., Wolf R., Thompson R., A Network Organization Used for Document Retrieval, Proceedings of the Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM SIGIR Forum, Vol. 17, No. 4
[12] Data General Corporation, CEO: ein Uberblick. Das umfassende elektronische Burosystem, Report No. 069-000039-00, 43 pp
[13] Thesaurus of ERIC: Educational Resources Information Center of the National Institute of Education, U.S. Departament of Education, ORYXPress, 419 pp
[14] Frei H. P., Bartschi M., Jauslin J.-F., Caliban: its User-Interface and Retrieval Algorithm, ETH Zurich, Institut fur informatik, Report No. 62
[15] Gutknecht J., A Concept of Rext Editing: LARA, ETH Zurich Institut fur Informatik
[16] Hinrichs K., The Grid File System: Implementation and Case Studies of Applications, Diss, ETH Nr. 7734, Zurich

[17] Knudsen S. E., A Modula-2 Oriented Operating System for the Personal Computer Lilith, Diss. ETH Nr. 7346, Zurich 43
[18] Knuth D. E., Morris J. H., Pratt V. R. Fast Pattern Matching in Strings, SIAM Journal of Computing, Vol. 6, 323—350
[19] Nievergelt J., Hinterberger H., Servik K. C., The Grid File: An Adaptable, Symmetric Multikey File Structure, ACM Transactions on Database Systems, Vol. 9, No. 1, March
[20] Ing. C. Olivetti C., S.p.A., OWS-2, Presentation Kit, 114 pp
[21] Pawlak Z., Systemy informacyjne. Podstawy teoretyczne, Wydawnictwa Naukowo-Techniczne, Warszawa, 188 pp
[22] Pomberger G., Softwaretechnik und Modula-2, Carl Hanser Verlag, Munchen, Wien, 231 pp
[23] Prime Computer, Inc., Information TM, 16 pp
[24] Sager W. K., Lockermann P.C., Classification of Ranking Algorithms, International Forum on Information and Documentation, Vol. 1, No. 4, 12—15
[25] Salton G., McGill M. J., Introduction to Modern Information Retrieval, McGraw-Hill Book Company, 448 pp
[26] Salton G., The Use of Extended Boolean in Information Retrieval, B. Yormark (ed.), Proceedings of Annual Metting SIGMOD'85, SIGMOD Recod Vol. 14, No. 2
[27] P. Shoval, Knowledge Representation in Cosultation Systems for Users of Retrieval Systems, C. Keren, L. Perimutter (ed.), The Application of Mini- and Micro-Computers in Information, Documentation, and Libraries, North-Holland, 631—643
[28] Wirth N., The Personal Computer Lilith, Berichte des Instituts fur Informatik No. 40 ETH Zurich
[29] N. Wirth, Compilarbau, Unterlagen, Institut fur Informatik, ETH Zurich
[30] C. A. Zehnder, Informatik-Projektentwicklung, ETH Institut fur Informatik, 156 pp.

JAN BIELECKI
Instytut Informatyki
Politechnika Warszawska

Błędy implementacji w kompilatorze języka C firmy Supersoft

Jednym z najbardziej popularnych kompilatorów języka C dla mikrokomputerów 8-bitowych jest kompilator firmy Supersoft, przystosowany do pracy pod nadzorem systemu operacyjnego CP/M. Chociaż w większości przypadków generuje on poprawny kod wynikowy, warto przyjrzeć się jego zachowaniu podczas kompilowania nietypowych konstrukcji, gdyż analiza taka ujawnia trudności, które muszą pokonać implementatorzy języka.

Przystępując do omówienia ważniejszych błędów implementacji należy nadmienić, że użyta tu terminologia jest w zasadzie zgodna ze zwyczajową terminologią dla języka C, z jednym ważnym wyjątkiem sprowadzającym się do zastąpienia angielskiego słowa „Pointer" polskim odpowiednikiem „wskazanie". Autorowi wydaje się, że taki termin polski właściwie oddaje znaczenie oryginału, a ponadto jest wygodniejszy w użyciu od stosowanego przez niektórych autorów słowa „wskaźnik", stanowiącego odpowiednik innych terminów angielskich, jak „flag" lub „indicator".

Odwołania do tablic i zmiennych wskazujących

Odwołania do tablic są w języku C traktowane inaczej niż w pozostałych językach programowania. Jeśli w wyrażeniu występuje napis reprezentujący tablice (np. arr), to zostaje on niejawnie przekształcony w ujęty w nawiasy okrągłe napis reprezentujący wskazanie na pierwszy element tej tablicy (&arr[0]). Takie ujednolicenie operacji na tablicach oraz zmiennych wskazujących, a w szczególności istnienie tożsamości, w myśl której a[i] jest równoważne * (a+i) powoduje, ze skutkiem wykonania programu:

```
char arr[ ][2] = ("j","b" );
main() {
    printf("%c%c%c", arr[0][0],arr[1][0]);
}
```

jak również wykonania programu:

```
char arr[ ][2] = ("j","b" );
    (*ptr)[2] = arr,
    *ref = ptr[1];
main() {
    printf("%c%c%c",**arr,*ref);
    }
```

jest wydrukowanie napisu jb.

W implementacji Supersoft drugi z przytoczonych programów jest wykonywany niepoprawnie, gdyż błędnie zrealizowano tam konstrukcje zawierające pary sąsiadujących operatorów wyłuskania [1]) (np. **arr). Ponadto, za błędne uznano indeksowanie i wyłuskiwanie dotyczące literałów tekstowych (np. "jan[2] oraz *"jan"). Błędnie w pewnych kontekstach zrealizowano także operatory postinkrementacji. Zachodzi to, na przykład, w programie:

```
char *ref = "jb";
main( ) {
   printf("%c", (ref++)[0]);
}
```

którego wykonanie powoduje wyprowadzenie napisu b, a nie jak być powinno, napisu j.

Realizowanie konwersji

W implementacji Supersoft, podobnie jak w szeregu innych implementacji języka C, zaniechano niejawnej konwersji wyrażeń występujących w instrukcji return. Z tego powodu funkcja:

```
char
fun(par)
{ return par; }
```

nie jest traktowana jak:

```
char
fun(par)
{ return (char)par; }
```

lecz jak:

```
int
fun(par)
{ return par; }
```

Należy nadmienić, że o ile rozwiązanie to ma pewne uzasadnienie i w znacznej mierze wynika z niedomówień w

[1]) Jeśli ref jest wyrażeniem reprezentującym wskazanie na obiekt, to wyrażenie *(ref), w którym * jest operatorem wyłuskania, reprezentuje ten obiekt. Przykładowo, jeśli var jest nazwą symboliczną obiektu, to &var jest wyrażeniem reprezentującym wskazanie na var, a *(&var) reprezentuje var.

standardzie języka C, o tyle naganne jest nieprzestrzeganie przez implementację Supersoft niektórych innych niejawnych konwersji, jak np. w programie:

```
char c;
int i;
main( ) {
   i = c = i = 258;
   printf("%d", i);
}
```

którego wykonanie powoduje wyprowadzenie liczby 258 zamiast liczby 2.

Podobnie jak konwersje niejawne, w wielu przypadkach źle zrealizowane są konwersje jawne. W szczególności w programie:

```
char (*ptr)[3][4],
     arr[3][4];
main( ) {
   ptr = arr;
   ptr = (char (*)[3][4])arr;
}
```

za błędną uznano instrukcję zawierającą konwersję:

```
(char(*)[3][4]
```

a za poprawną instrukcję

```
ptr = arr;
```

mimo iż powinno być akurat odwrotnie.

Bezkrytyczne utożsamianie danych wskazujących z danymi całkowitymi powoduje, że w implementacji Supersoft skutkiem wykonania programu:

```
char (*ptr)[3][4],
     (*ref)[4],
     arr[3][4];
main( ) {
   ptr = /* (char (*)[3][4]) */ arr;
   ref = ptr[1];
   printf("%d", (int)ref — (int)arr);
}
```

jest wyprowadzenie liczby 3, mimo iż program ten byłby poprawny dopiero po uwzględnieniu konwersji przytoczonej w komentarzu, kiedy to skutkiem wykonania byłoby wyprowadzenie liczby 12.

Realizowanie instrukcji

Z nieznanych przyczyn w implementacji Supersoft poprawnie zrealizowano funkcję:

```
set(t, s)
 char *t, *s;
(for ( ; *t++ = *s++, *s; ); }
```

natomiast za niepoprawną uznano równoważną jej funkcję:

```
set(t, s)
 char *t, *s;
(while(*t++ = *s++, *s; ); }
```

Sygnalizowanie błędów

Sygnalizowanie błędów w implementacji Supersoft budzi wiele zastrzeżeń. Wiele konstrukcji niepoprawnych, jak np.:

```
char *ref;
...
fun(ref++[0]);
```

oraz

```
struct (
  char var,
       arr[2];
) str;
...
fun(str->arr);
```

jest kompilowanych bez sprzeciwu, natomiast wiele konstrukcji poprawnych, jak np.

```
char *ptr,
     var;
...
var = (int)++ptr;
```

oraz

```
char arr[3][4],
     len;
main( ) {
   len=arr[2] — (char *) arr;
   printf("%d", len);
}
```

jest odrzucanych. Stanowi to znaczne utrudnienie procesu uruchamiania programów.

WARUNKI PRENUMERATY NA 1986 R.

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

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

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

Sposób zamawiania prenumeraty taki sam jak dla prenumeraty indywidualnej.

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

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

U w a g a!

Wpłaty na dwumiesięczniki przyjmowane są na okresy półroczne lub roczne.

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

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

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

Cena prenumeraty INFORMATYKI (w złotych)					
kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
360,—	105,—	720,—	210,—	1440,	420,—

MAREK SIKORA
Przedsiębiorstwo Zagraniczne STARCOM
Warszawa

Modułowy system mikroprocesorowy kompatybilny z systemem Mikroster

Celem opracowania niżej opisanego systemu było wypełnienie luki rynkowej, jaka powstała po stosunkowo szerokim rozpropagowaniu systemu Mikroster w wydawnictwach Przemysłowego Instytutu Elektroniki oraz prasie fachowej, przy jednocześnie małej jego podaży.

Przystępując do prac przyjęto poniższe założenia:

- wszystkie podstawowe sygnały magistrali systemowej powinny być zgodne z objętym normą branżową systemem Mikroster
- zestaw oferowanych modułów musi być dostosowany do potrzeb maksymalnej liczby potencjalnych użytkowników
- system powinien łączyć w sobie zarówno cechy otwartego, jak i zamkniętego systemu mikroprocesorowego
- poszczególne pakiety powinny być zbudowane w oparciu o łatwo dostępne, dobrze znane, a jednocześnie nowoczesne układy LSI
- zastosowane rozwiązania muszą zapewniać łatwość rozbudowy i zmian konfiguracji systemu
- możliwość pełnego wykorzystania zalet modułowego systemu mikroprocesorowego nie powinna zależeć od posiadania trudno dostępnych urządzeń zewnętrznych, pamięci masowych, programatorów EPROM oraz skrośnego oprogramowania systemowego.

Opracowane moduły można podzielić na cztery główne grupy: jednostki centralne, pamięci, moduły do współpracy z urządzeniami we-wy, moduły wspomagające rozbudowę i serwis systemu.

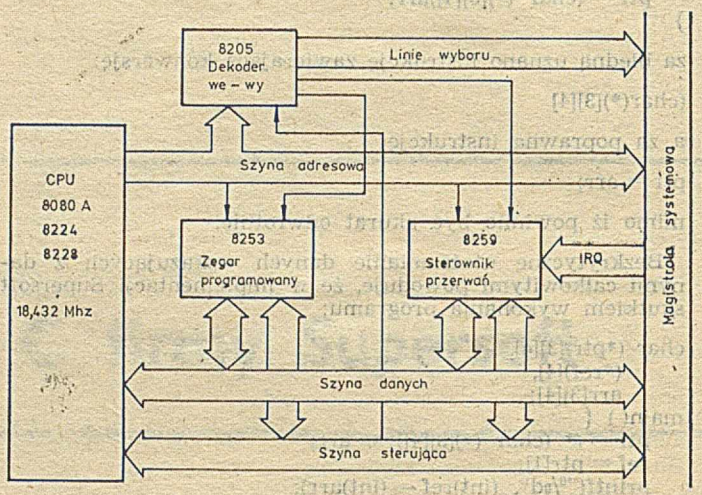
OMÓWIENIE SYSTEMU

Poniżej przedstawiono podstawowe informacje charakteryzujące poszczególne moduły systemu.

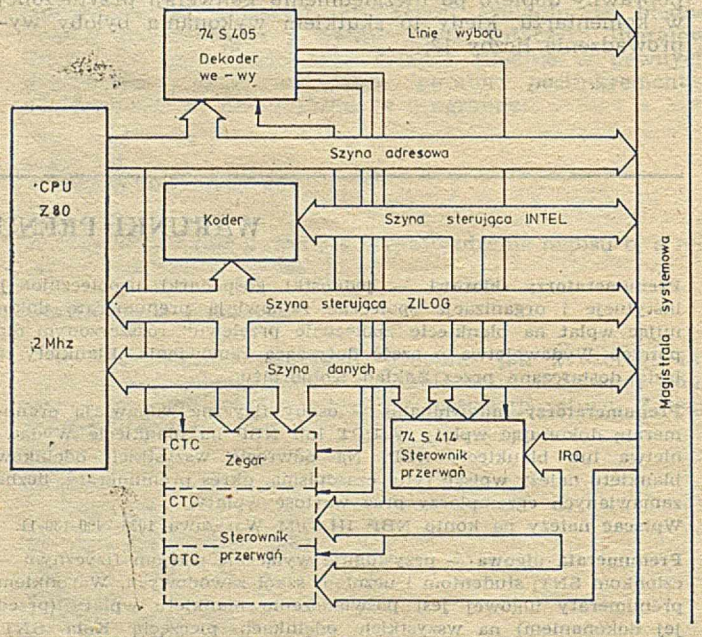
Jednostki centralne

Jednostka centralna z mikroprocesorem Intel 8080 (rys. 1) jest zbudowana w oparciu o następujące elementy: procesor 8080, generator impulsów zegarowych 8224, sterownik systemu 8228/38, sterownik przerwań 8259, programowy licznik i zegar czasu rzeczywistego 8253, bufory szyny adresowej, danych i sterowania 8216, dekoder urządzeń we-wy 8205.

Mgr inż. MAREK SIKORA ukończył w 1982 r. Wydział Elektroniki Politechniki Warszawskiej. Pracował w zakresie problematyki mikroprocesorowej w Instytucie Chemii i Techniki Jądrowej w Warszawie. Aktualnie prowadzi zakład rzemieślniczy, w którym są produkowane między innymi opisane w artykule urządzenia.



Rys. 1. Jednostka centralna z mikroprocesorem INTEL 8080



Rys. 2. Jednostka centralna z mikroprocesorem Z80

Jednostka centralna z mikroprocesorem Z-80 (rys. 2) jest wyposażona w układ przekodowania podstawowych sygnałów identyfikujących cykl maszynowy mikroprocesora Z-80 ($\overline{M1}$, $\overline{RD}$, $\overline{WR}$, MREQ, $\overline{IORQ}$) na sygnały charakterystyczne dla mikroprocesora INTEL 8080 ($\overline{INTA}$, $\overline{MEMR}$, $\overline{MEMW}$, $\overline{I/OR}$, $\overline{I/OW}$). Dzięki temu moduł jest całkowicie kompatybilny ze wszystkimi pozostałymi pakietami, zbudowanymi w oparciu o programowane układy LSI firmy Intel. Sterownik przerwań jest wytwarzany w dwóch wersjach: z układem 8214 oraz z układem Z-80 CTC. Zastosowanie

układów Z-80 CTC w roli sterownika przerwań pozwala na maskowanie poszczególnych linii zgłoszeń, przyjmowanie przerwań zgłaszanych zboczem impulsu, a przede wszystkim — na obsługę przerwań przez procesor pracujący w trybie 2. Ważną zaletą pakietu jest jednonapięciowe zasilanie. Jego koszt jest mniejszy od jednostki centralnej z mikroprocesorem INTEL 8080.

Pamięci

Pamięć 4 KB RAM/16 KB ROM jest zbudowana z elementów 2114 i 2716. Ze względu na stosunkowo małą pojemność jest przeznaczona do małych systemów.

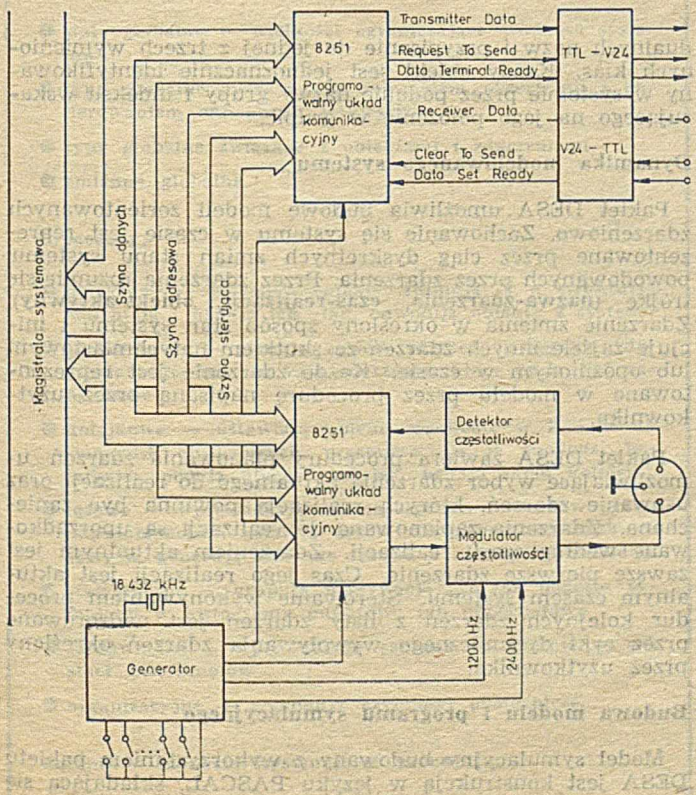
Pamięć 16 KB RAM/32 KB ROM jest wytwarzana z elementów 6116 (150 ns, 2K×8 STATIC C-MOS RAM) oraz 2764 (150 ns, 8K × 8 EPROM). Znikomy pobór prądu przez pamięć RAM pracującą w trybie rezerwowym, ang. stand by (200μA, 16 KB), umożliwia podtrzymywanie zawartości pamięci po wyłączeniu zasilania.

Pamięć 64 KB DRAM/32 KB ROM nie ma autonomicznego odświeżania pamięci DRAM (4164) i z tego powodu jest przeznaczona do współpracy tylko z jednostką centralną opartą na mikroprocesorze Z-80. Ponieważ łączna pojemność tej pamięci (96 KB) jest większa niż przestrzeń adresowa jednostki centralnej, wprowadzono układ zarządzania pamięcią (ośmiobitowy rejestr w przestrzeni adresowej we-wy).

Moduły współpracy z urządzeniami we-wy

Sterownik monitora alfanumerycznego jest przeznaczony do współpracy z monitorem telewizji użytkowej (zalecany — NEPTUN 156). Opcjonalnie dodawany jest modulator pozwalający na pracę ze zwykłym telewizorem. Pamięć obrazu zainstalowana bezpośrednio na pakiecie jest widziana przez procesor jako ostatnie 2 KB przestrzeni adresowej. Znaki alfanumeryczne są wyświetlane w 24 wierszach po 64 znaki w każdym wierszu.

Pakiet wejść-wyjść równoległych umożliwia, dzięki zainstalowaniu programowanych układów 8255, realizację praktycznie dowolnego protokołu transmisji równoległej. Porty układów 8255 mogą być buforowane elementami 8216. Kierunek pracy buforów może być zmieniany programowo. Na życzenie, bezpośrednio na płytce montowana jest podstawka zaciskowa i przetwornica 5V/25V pozwalająca na wykorzystanie jednego z układów 8255 do programowania pamięci EPROM (2716, 2732, 2764, 27128).



Rys. 3. Schemat blokowy pakietu transmisji szeregowej

Pakiet transmisji szeregowej (rys. 3) działa w oparciu o dwa układy 8251, dla których źródłem impulsów zegarowych jest generator kwarcowy o częstotliwości 18,432 MHz. Jeden z układów jest wyposażony w sprzęg V-24, a drugi w modulator i detektor częstotliwości. Szybkość transmisji układu wyposażonego w sprzęg V-24 jest ustalana za pomocą zwór. Manipulowanie zworami oraz wewnętrznym dzielnikiem układu 8251 pozwala na uzyskanie następujących szybkości transmisji: 9600, 4800, 2400, 1200, 600, 300, 200, 150, 75 oraz 50 bodów. Szybkość transmisji układu wyposażonego w modulator i detektor częstotliwości, którego głównym przeznaczeniem jest zapisywanie i odczyt danych i programów przy użyciu magnetofonu powszechnego użytku, wynosi 300 lub 1200 bodów. W zmodulowanym częstotliwościowo sygnale jedynkę logiczną reprezentuje częstotliwość 2400 Hz, a zero logiczne częstotliwość 1200 Hz. Linia RTS układu 8251 służy do sterowania silnikiem magnetofonu.

Sterowanie pracą systemu

Do kontroli pracy służy pakiet obserwacji magistrali umożliwiający obserwację stanu szyny adresowej, danych i sterującej za pomocą diod świecących zamontowanych na płytce czołowej. Poza stanem linii adresowych i linii danych wyświetlany jest stan następujących linii sterujących: $\overline{\text{INTA}}$, $\overline{\text{MEMR}}$, $\overline{\text{MENW}}$, $\overline{\text{I/OR}}$, $\overline{\text{I/OW}}$, READY, HLDA.

Sterowanie systemem odbywa się za pomocą następujących przełączników: RESET, STEP/RUN, SINGLE STEP, D_0 — D_7, PUSH. Naciśnięcie klawisza RESET wprowadza system w stan początkowy. Przycisk SINGLE STEP umożliwia wykonywanie pojedynczych cykli maszynowych (po odpowiednim ustawieniu przełącznika STEP/RUN). Naciśnięcie klawisza PUSH powoduje przepisanie stanu przełączników D_0 — D_7 do znajdującego się w module sterowania buforu 8212 i otwarcie jego wejść na szynę danych systemu (PUSH jest aktywny tylko przy pracy krokowej w cyklach typu MEMR lub I/OR). Moduł sterowania pozwala na zdeterminowanie działania procesora niezależnie od oprogramowania. Jednoczesna obserwacja systemu przy użyciu pakietu obserwacji magistrali oraz sondy TTL umożliwia praktycznie pełny serwis jednostki centralnej i współpracujących z nią modułów.

Obecnie opracowywany jest sterownik monitora graficznego oraz sterownik dysków elastycznych 5 1/4 cala. Co najmniej jeden z tych pakietów powinien być gotowy w chwili opublikowania artykułu.

Z opisanych urządzeń w postaci pojedynczych modułów można tworzyć kompletne zestawy wraz z oprogramowaniem systemowym. Typ procesora, rodzaj pamięci oraz modułów sprzęgających urządzenia we-wy zależy od rodzaju wymagań użytkowych. Rezydentne oprogramowanie podstawowe umożliwia używanie zestawu następujących dyrektyw:

AS — asembler
DS — disasembler
LT — odczyt zawartości pamięci w postaci szesnastkowej
LD — ładowanie do pamięci liczb szesnastkowych
BP — ustawianie pułapki
GO — skok do programu użytkownika
PSW, BC, DE, HL, SP — odczytanie i modyfikacja stanu rejestrów procesora po wejściu w pułapkę lub przed wykonaniem programu
WT — zapis danych lub programów z pamięci operacyjnej na kasetę magnetofonową
RT — zapis danych lub programów z kasety magnetofonowej do pamięci operacyjnej
LR — odczyt pamięci EPROM umieszczonej w podstawce programatora
KK — kontrola kasowania pamięci EPROM
PGM — programowanie pamięci EPROM
KZ — kontrola poprawności zapisu pamięci EPROM

ZASTOSOWANIA

Opisany system może stanowić podstawę konstrukcji mikrokomputerów lub sterowników mikrokomputerowych przeznaczonych do realizacji zadań kontroli, sterowania i rejestracji danych w obiektach przemysłowych, procesach technologicznych lub do przeprowadzania prac badawczo-doświadczalnych, a także do wspomagania projektantów urzą-

dzeń mikroprocesorowych. Zastosowanie gotowych modułów przyspiesza opracowywanie złożonych urządzeń, gdyż dzięki elastyczności systemu, dostosowanie go od strony sprzętowej do wykonywanego zadania wymaga jedynie dobrania odpowiedniej konfiguracji modułów i urządzeń zewnętrznych.

W pracach badawczo-doświadczalnych system wykorzystuje się do budowy wielofunkcyjnych przyrządów pomiarowych. Wielowejściowy przyrząd programowany, z monitorem, drukarką i przetwornikami wielkości fizycznych na napięcie lub częstotliwość, może dokonywać pomiarów temperatury, ciśnienia, przepływu cieczy lub gazów, ciężaru i wymiarów, spełniając jednocześnie wiele funkcji przetwarzania, jak:

- automatyczna kalibracja, korekcja zera, zmiana zakresów pomiarowych
- filtracja cyfrowa wyników pomiaru
- obliczenia i korekcja błędów pomiarowych
- wyznaczanie wartości wielkości mierzonych pośrednio
- obliczenia i korekcja nieliniowości charakterystyk czujników, przetworników lub sond pomiarowych
- obrazowanie graficzne i tabelaryzowanie wyników pomiarów.

System można wykorzystać w konstrukcji urządzeń kontroli technicznej parametrów mechanicznych (wymiary, ciężar) lub elektrycznych (prąd, napięcie, częstotliwość) wyrobów produkowanych seryjnie, gdzie wymagane jest sortowanie dużej liczby detali na grupy oraz odrzucanie detali wadliwych.

Oprócz funkcji wspomagania pomiarów i rejestracji wyników w sprzęcie pomiarowym, system może wykonywać zadania nadzorowania i sterowania procesów technologicznych, a także zadania programowanej regulacji, np. regulacji temperatury lub prędkości obrotowej silników elektrycznych.

Wykorzystując ten system można konstruować sterowniki mikroprocesorowe dla maszyn, urządzeń i procesów technologicznych. W takich zastosowaniach sterownik nadzoruje i steruje pracą elementów pomiarowych i wykonawczych, a jednocześnie może rejestrować i aktualizować informację o liczbie wytworzonych detali, stanie zapasów lub ilości surowca w podajnikach wejściowych, a także przeprowadzać niezbędne obliczenia.

Ten sam system może służyć do wspomagania pracy projektanta przyrządów i urządzeń oraz do budowy oprogramowania dla zadań realizowanych przez sterownik mikroprocesorowy.

Zastosowanie opisanego systemu w budowie przyrządów pomiarowych czy w sterowaniu procesami technologicznymi pozwala na zwiększenie funkcjonalności urządzeń, upraszcza ich obsługę i wpływa na polepszenie parametrów procesów i produkowanych wyrobów.

WIESŁAW NOSOWSKI
BOLESŁAW SZOMAŃSKI
Instytut Organizacji i Zarządzania
Politechnika Warszawska

Pakiet symulacyjny DESA

Pakiet DESA jest zbiorem programów w języku PASCAL przeznaczonych do budowy komputerowych modeli symulacyjnych dla dyskretnych systemów obsługi. Powstał w wyniku prac nad oprogramowaniem wspomagającym modelowanie systemów produkcyjnych. Wynikiem tych prac były m. in. dwa pakiety wcześniejsze, przeznaczone do modelowania w językach ALGOL 60 [1] i ALGOL 1900 [2].

Struktura modelowanego systemu

Wiele systemów działających w takich dziedzinach jak przemysł, rolnictwo, handel, transport, można traktować jako systemy obsługi. Typową ich cechą jest występowanie stanowisk obsługi, do których zgłaszają się pewne obiekty w celu ich obsłużenia według określonych reguł. Proces zgłaszania się i obsługi podlega z reguły licznym zakłóceniom losowym. W modelowanym przez pakiet DESA systemie wyróżnia się trzy klasy obiektów:

- stanowisko obsługi — są to obiekty, w których odbywa się obsługa innych zgłaszających się w tym celu obiektów
- obiekty ruchowe — są to obiekty obsługiwane, przemieszczające się pomiędzy stanowiskami obsługi lub obiekty wchodzące w skład stanowisk, lecz mogące zmieniać swoje miejsce (np. obsługa stanowisk, wyposażenie)
- reguły — są to obiekty zawierające dane o parametrach i zasadach obsługi.

Obiekty każdej z tych trzech klas są organizowane w wielopoziomowe struktury typu drzewa binarnego. Pozwala to na odwzorowanie szerokiej klasy struktury systemów obsługi przez odwzorowanie zróżnicowania obiektów obsługiwanych i różnorodności reguł obsługi. Użytkownik może zdefiniować wiele grup obiektów przez nadanie im indywidualnych nazw i przypisanie do jednej z trzech wymienionych klas. Każdy obiekt jest jednoznacznie identyfikowany w systemie przez podanie nazwy grupy i indeksu wskazującego na jego położenie w grupie.

Dynamika modelowanego systemu

Pakiet DESA umożliwia budowę modeli zorientowanych zdarzeniowo. Zachowanie się systemu w czasie jest reprezentowane przez ciąg dyskretnych zmian stanu systemu, powodowanych przez zdarzenia. Przez zdarzenia rozumie się trójkę (**nazwa-zdarzenia**, **czas-realizacji**, **obiekt-aktywny**). Zdarzenie zmienia w określony sposób stan systemu i inicjuje zajście innych zdarzeń ze skutkiem natychmiastowym lub opóźnionym w czasie. Każde zdarzenie jest reprezentowane w modelu przez procedurę napisaną przez użytkownika.

Pakiet DESA zawiera procedury planowania zdarzeń umożliwiające wybór zdarzenia aktualnego do realizacji oraz usuwanie zdarzeń, których realizacja powinna być zaniechana. Zdarzenia zaplanowane do realizacji są uporządkowane według czasu realizacji. Zdarzeniem aktualnym jest zawsze pierwsze zdarzenie. Czas jego realizacji jest aktualnym czasem systemu. Sterowanie wykonywaniem procedur kolejnych zdarzeń z listy zdarzeń jest nadzorowane przez cykl dynamicznego wywoływania zdarzeń określony przez użytkownika.

Budowa modelu i programu symulacyjnego

Model symulacyjny budowany z wykorzystaniem pakietu DESA jest konstrukcją w języku PASCAL, składającą się z następujących części:

● **Stałe modelu** — występują tu na ogół wielkości ograniczające wielkość modelowanego systemu.

● **Typy wartości w modelu** — określone są tu takie elementy, jak nazwy zdarzeń dyskretnych, stany obiektów we wszystkich klasach, atrybuty obiektów.

● **Zmienne modelu** — powinny tu być określone wielkości odpowiadające typom globalnym i typom modelu. W szczególności określić należy nazwy charakterystyk rozkładów prawdopodobieństwa dla wyróżnionych w modelu zmiennych losowych oraz nazwy histogramów do zbierania obserwacji.

● **Procedury zdarzeń** — treść wyróżnionych w systemie zdarzeń dyskretnych reprezentowana jest w modelu przez procedury. Istotą każdej procedury reprezentującej zdarzenie jest zmiana stanu w pewnej części systemu, planowanie następnych, możliwych do określenia zdarzeń i zbieranie obserwacji powstałych w wyniku zajścia zdarzenia.

● **Procedury pomocnicze** — oprócz procedur zdarzeń, w modelu występują na ogół inne procedury niezbędne do jego funkcjonowania. Powinna wśród nich wystąpić procedura zawierająca dynamiczny cykl wywoływania zdarzeń. Na ogół występuje też potrzeba ustalenia warunków początkowych do symulacji. Procedura ustalania warunków początkowych powołuje stałe obiekty systemu, przygotowuje histogramy do zbierania obserwacji, planuje zajście pierwszych zdarzeń. Oprócz tego w modelu występują na ogół procedury czytania danych i wyprowadzanie wyników.

● **Część operacyjna modelu** — będzie zawierać na ogół wywołanie procedur czytania danych i ustalania warunków początkowych. Następnie powinien zostać uruchomiony dynamiczny cykl wywoływania zdarzeń. Przerwanie tego cyklu określonym warunkiem zakończenia symulacji pozwala przejść do wyprowadzenia wyników symulacji.

Budowa pakietu DESA

Pakiet DESA jest grupą procedur operującą na określonej strukturze danych globalnych i danych związanych bezpośrednio z modelowanym systemem. Jego budowę oparto na koncepcji symulacji algorytmicznej Winkowskiego [3].

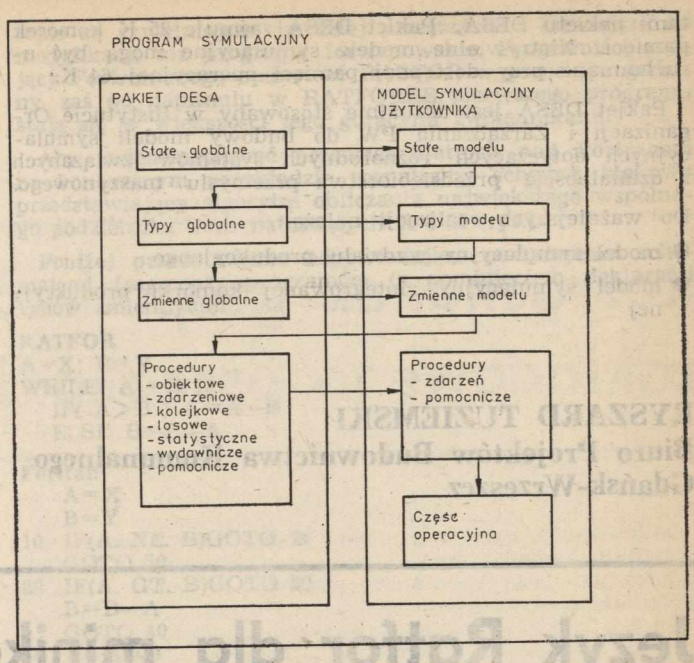
Struktura danych globalnych:

- **stałe globalne — wielkości ograniczające wielkość systemu**
- **typy globalne związane ze zbieraniem obserwacji i generowaniem zmiennych losowych**
- **typy globalne związane z obiektami i zdarzeniami**
- **zmienne globalne.**

Procedury:

- **obiektowe — powołują obiekty, formują je w struktury, zmieniają i badają ich stan, odwołują obiekty z systemu**
- **zdarzeniowe — planują zajście zdarzeń, wybierają zdarzenie aktualne, usuwają zdarzenia, które mają być zaniechane**
- **kolejkowe — ustawiają obiekty ruchome w kolejki do stanowisk obsługowych, przemieszczają je pomiędzy kolejkami, podają długość kolejek, badają skład kolejek**
- **losowe — wprowadzają charakterystyki rozkładów prawdopodobieństwa, ustalają źródła dla generatora liczb losowych, generują zmienne losowe z zadanych rozkładów**
- **statystyczne — grupują obserwacje zebrane podczas symulacji w histogramy punktowe lub przedziałowe**
- **wydawnicze — wprowadzają zebrane obserwacje w postaci histogramów**
- **organizacyjne — wykonują czynności pomocnicze.**

Łącznie w pakiecie występuje 45 procedur.



Struktura programu symulacyjnego złożonego z elementów pakietu DESA i modelu użytkownika

Program symulacyjny jest konstrukcją złożoną z elementów pakietu DESA oraz elementów modelu symulacyjnego przygotowanych przez użytkownika. Elementy pakietu DESA są omówione w tabeli, a logiczna struktura programu symulacyjnego jest pokazana na rysunku.

Metodyka modelowania

Przebieg modelowania z wykorzystaniem pakietu DESA jest podobny w głównych swoich etapach do modelowania innymi technikami. Etapy i kroki modelowania są następujące:

1. Sformułowanie problemu
2. Opis systemu
3. Wybór metody rozwiązania
4. Projektowanie modelu symulacyjnego
   - — cele poznawcze modelu
   - — grupy obiektów
   - — stany obiektów
   - — zmienne losowe
   - — atrybuty obiektów
   - — sposób zbierania i wyprowadzania obserwacji
   - — procedury zdarzeń
   - — warunki początkowe
   - — procedury pomocnicze
   - — dane do symulacji
5. Budowa modelu symulacyjnego

Etapy (1), (2), (3) są typowe dla metodyk rozwiązywania problemów z wykorzystaniem podejścia systemowego. Jeśli na etapie (3) zapadnie decyzja o zastosowaniu metody symulacyjnej, to realizowane są etapy (4) i (5). Przebieg tych etapów jest z kolei typowy dla współczesnej metodyki programowania. Tak więc, metodyka modelowania łączy w sobie metodykę systemowego rozwiązywania problemów z metodyką programowania.

Eksploatacja i realizacja

Pakiet DESA został opracowany w języku PASCAL 1900. Podstawowa wersja pakietu może być eksploatowana na maszynie ODRA 1305 w systemie operacyjnym GEORGE 3. Zmiana maszyny lub systemu operacyjnego wymaga odpowiednich adaptacji. Korzystanie z pakietu jest ułatwione dzięki opracowaniu specjalnych makrokomend zabezpieczających model użytkownika oraz ułatwiających jego utworzenie i uruchomienie. Poszczególne elementy modelu symulacyjnego (stałe, typy, zmienne, procedury zdarzeń, procedury pomocnicze, część operacyjna) przechowywane są w oddzielnych zbiorach, skąd za pomocą makrokomendy są automatycznie łączone podczas kompilacji z elemen-

tami pakietu DESA. Pakiet DESA zajmuje 25 K komórek pamięci. Nietrywialne modele symulacyjne mogą być uruchamiane przy dostępnej pamięci operacyjnej 64 K.

Pakiet DESA jest aktualnie stosowany w Instytucie Organizacji i Zarządzania PW do budowy modeli symulacyjnych dotyczących różnorodnych systemów związanych z działalnością przedsiębiorstwa przemysłu maszynowego.

Do ważniejszych realizacji należą:

- model symulacyjny wydziału produkcyjnego
- model symulacyjny zintegrowanej komórki produkcyjnej
- model obsługi komórki produkcyjnej przez komórkę remontową
- gra przemysłowa dotycząca kierowania pracą oddziału produkcyjnego.

LITERATURA

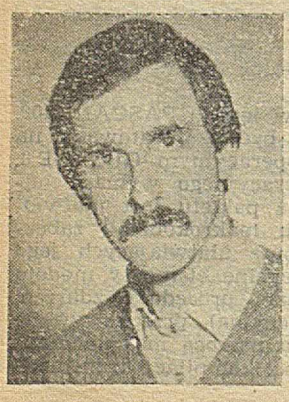
[1] Nosowski W.: Modelowanie symulacyjne w języku ALGOL 60. Wyd. IOZ, Zeszyt nr 10, Warszawa 1976
[2] Nosowski W.: Modelowanie procesów produkcyjnych. System MOSP. Wyd. IOZ, Zeszyt nr 22, Warszawa 1979
[3] Winkowski J.: Programowanie symulacji procesów. WNT, Warszawa 1974.

RYSZARD TUZIEMSKI
Biuro Projektów Budownictwa Komunalnego
Gdańsk-Wrzeszcz

Język Ratfor dla minikomputera Mera 400

Moja przygoda z Ratforem zaczęła się w 1981 roku po przeczytaniu w „Communications of the ACM" artykułu o realizacji projektu wirtualnego systemu operacyjnego, u którego podstaw leżało szereg idei systemu UNIX. Wspomniano tam, że implementacji systemu dokonano wykorzystując RATFOR, a wykaz literatury zawierał odsyłacz do artykułu B. W. Kernighana "RATFOR — A Preprocessor for a Rational Fortran" w "Software Practice and Experience". Po pewnym czasie dotarłem do tego artykułu, zacząłem go czytać i skończyłem lekturę z wypiekami na twarzy. Korzystając z zamieszczonego w nim opisu RATFORU napisałem preprocesor dla MERY 400 pracującej pod nadzorem systemu SOM-3. Byłem zafascynowany możliwościami stosowania strukturalnego stylu programowania w RATFORZE, a jednocześnie zdumiony, że nikt w Polsce nie zwrócił na niego wcześniej uwagi (wszak artykuł Kernighana pochodził z 1975 roku!). Było to tak proste i tanie narzędzie ułatwiające życie programistom, zwiększające ich wydajność i propagujące właściwy styl programowania — wszyscy przecież dysponowali Fortranem, zaś o Pascalu i Loglanie słyszeli wówczas jedynie bywalcy ośrodka obliczeniowego Instytutu Informatyki Uniwersytetu Warszawskiego.

Niniejszy artykuł informuje o istnieniu w kraju sprawdzonych w działaniu preprocesorów Ratforu (dla minikomputera Mera 400, działających pod nadzorem systemów operacyjnych SOM-3 i CROOK-4) być może pozwoli to uniknąć wyważania kolejnych otwartych drzwi (o ile to już nie nastąpiło). Być może również aktualna będzie informacja, że z uwagi na przyjętą przeze mnie technikę realizacji preprocesora w Fortranie możliwe jest w ciągu tygodnia przeniesienie go na maszyny serii ODRA, SM, RIAD i inne posiadające kompilatory Fortranu. Bliższe dane można otrzymać pisząc pod adresem: mgr inż. R. Tuziemski, BPBK, ul. H. Sawickiej 27, 80-237, Gdańsk-Wrzeszcz (tel. 41-40-11, w. 13).



Mgr inż. RYSZARD TUZIEMSKI ukończył w 1978 r. Wydział Elektroniki Politechniki Gdańskiej (specjalność — informatyka). Pracuje w Biurze Projektów Budownictwa Komunalnego w Gdańsku jako projektant ds. informatyki. Zajmuje się oprogramowaniem podstawowym mini i mikrokomputerów.

Poniżej opisano krótko główne zaimplementowane konstrukcje języka RATFOR, które czynią go tak wygodnym narzędziem programowania.

- Swobodny format pisania instrukcji. Instrukcja może zaczynać się od dowolnej pozycji wiersza, a w jednym wierszu można umieszczać kilka instrukcji, np.

```
A=1; B=2; C=3
```

- Matematyczny zapis relacji w instrukcjach warunkowych.

```
IF( A<=B & C>D )...
```

- Automatyczna kontynuacja instrukcji arytmetycznych, list formatu, deklaracji typów i list DATA.

```
X=A*B+
  C/D—E
900 FORMAT(1X, "BARDZO DŁUGI TEKST-CZĘŚĆ I",
"BARDZO DŁUGI TEKST-CZĘŚĆ II")
```

- Automatyczne rozpoznawanie i odpowiednie przetwarzanie etykiet. Pozwala to na umieszczanie instrukcji opatrzonych etykietami w dowolnym miejscu wiersza, np.

```
WRITE(5,100); 100 FORMAT(" HELLO !")
```

- Zmodyfikowana postać instrukcji DO. Nie wymaga ona podawania etykiety końca cyklu i instrukcji CONTINUE.
- Możliwość umieszczania komentarzy w jednym wierszu z instrukcjami.

```
A = 1; B = 2 # ZAINICJOWANIE A I B
```

- Pseudoinstrukcja DEFINE. Umożliwia ona przypisanie nazwie wartości, co oprócz zwiększenia czytelności programów pozwala na ich parametryzację, np.:

```
DEFINE ROWS 100 # MAKS. LICZBA WIERSZY
DEFINE COLS 50 # MAKS. LICZBA KOLUMN
DIMENSION A(ROWS),B(ROWS, COLS)
IF( I <= ROWS & J <= COLS)...
# INDEKSY O. K.
```

● Pseudoinstrukcja INCLUDE. Umożliwia ona dołączanie do segmentów programowych wspólnych dla nich plików zawierających, na przykład, deklaracje bloków COMMON lub wspólne definicje.

● Możliwość tworzenia instrukcji złożonych ograniczonych parą nawiasów [...]. Odpowiada to konstrukcjom begin ... end w językach Algol i Pascal. Wewnątrz instrukcji złożonej mogą występować dowolne instrukcje, a więc również instrukcje złożone. Instrukcja złożona może wystąpić wszędzie tam, gdzie pojedyncza instrukcja.

● Instrukcja FOR. Jest podobna do analogicznej instrukcji języków Algol i Pascal, lecz bardziej uniwersalna z uwagi na fakt, że jej krok nie jest ograniczony do postępu arytmetycznego.

● Instrukcja NEXT. Umożliwia wymuszenie następnej iteracji w dowolnym miejscu pętli DO lub FOR, np.:

```
DO I=1,80 # PRZETWARZANIE ZNAKÓW RÓŻNYCH OD
SPACJI
[ IF(CARD(I)==BLANK ) NEXT
  ...
  # BARDZO DUŻO INSTRUKCJI
  ...; ]
```

● Instrukcja BREAK. Pozwala na opuszczenie pętli DO lub FOR przed wyczerpaniem zadanej liczby iteracji, np.:

```
DO I=1,80 # POMINIĘCIE POCZATKOWYCH SPACJI
  IF (CARD(I)!=BLANK) BREAK
```

● Instrukcja IF ... ELSE. Umożliwia warunkowy wybór kolejności obliczeń. Należy podkreślić, że pomiędzy IF a ELSE oraz po ELSE może wystąpić dowolna instrukcja, a więc także instrukcja złożona zawierająca inne instrukcje IF... ELSE. Oczywiście część ELSE jest opcjonalna, np.:

```
IF( A<=B )
  [ SW=0; IF(LO==1)WRITE(5,900)A, B; ]
ELSE [ SW=1; WRITE(5,900)B, A; ]
```

● Instrukcja WHILE. Służy analogicznie jak w języku Pascal do organizacji pętli o liczbie powtórzeń uzależnionej od warunku logicznego testowanego na początku pętli, np.:

```
WHILE( A(I) >B(J) ) [ I=I+1; J=J-1; ]
```

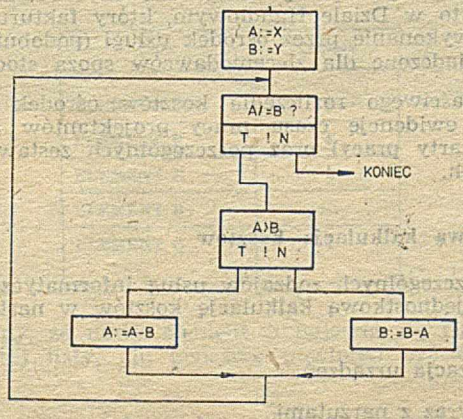
● Instrukcja REPEAT... UNTIL. Jest używana podobnie jak w języku Pascal do organizacji pętli o liczbie powtórzeń uzależnionej od warunku logicznego testowanego na końcu pętli, np.:

```
REPEAT
  WHILE( A>B ) A=A-B
  WHILE( B>A ) B=B-A
UNTIL( A==B )
```

Ze względu na powyższe cechy adaptacja programów z języków Algol i Pascal na język RATFOR staje się zadaniem banalnym (o ile programy te nie zawierają jawnej lub niejawnej rekursji).

* * *

Na zakończenie wypada stwierdzić, że Ratfor pozwala na wykorzystanie wszystkich możliwości języka Fortran oraz dodatkowo posiada szereg udogodnień ułatwiających i przyspieszających pisanie programów. Programy pisane w Ratforze są czytelniejsze, bardziej przejrzyste i łatwiejsze do modyfikacji niż programy fortranowskie. Programiści znający Fortran mogą opanować RATFOR w przeciągu godziny, zaś po napisaniu w RATFORZE pierwszego programu stają się w większości jego gorącymi zwolennikami.



Algorytm obliczania największego wspólnego podzielnika liczb naturalnych x i y

Aby zademonstrować przewagę Ratforu nad Fortranem na konkretnym przykładzie rozpatrzmy schemat blokowy przedstawiający algorytm obliczania największego wspólnego podzielnika liczb naturalnych X i Y (rys.).

Poniżej przedstawiono odpowiadające powyższemu schematowi fragmenty programów (z pominięciem deklaracji typów zmiennych).

RATFOR

```
A=X; B=Y
WHILE( A!=B )
  IF( A>B ) A=A-B
  ELSE B=B-A
```

Fortran

```
      A=X
      B=Y
10    IF(A. NE. B)GOTO 20
      GOTO 50
20    IF(A. GT. B)GOTO 30
      B=B-A
      GOTO 40
30    A=A-B
40    GOTO 10
50    ...
```

Wniosek z powyższego porównania jest oczywisty — Ratfor jest niezbędnym narzędziem dla każdego, kto chce programować lepiej i wydajniej.

JERZY SUKIENNIK
Stocznia Marynarki Wojennej
im. Dąbrowszczaków
Gdynia

Rozrachunek gospodarczy zakładowego ośrodka informatyki

Podstawowym celem wprowadzenia wewnętrznego rozrachunku gospodarczego (WRG) w zakładowym ośrodku informatyki jest dążenie do obniżenia poziomu kosztów ogólnozakładowych — zwłaszcza w związku z obowiązującymi od 1984 r.[1]) nowymi przepisami finansowo-księgowymi oraz zwiększenie dyscypliny zarówno w samym ośrodku, jak i u użytkowników informatyki. Pozwala to eliminować różne postacie marnotrawstwa (wdrażanie i eksploatację systemów mało przydatnych, zawyżanie opłat przez ośrodek polegające na wielokrotnym powtarzaniu przebiegów obliczeniowych).

Idea rozrachunku wewnętrznego

Zakładowy Ośrodek Informatyki w Stoczni Marynarki Wojennej od 1 stycznia 1979 r. podporządkowany jest Dyrektorowi Technicznemu (podobnie jak Biuro Projektowo-Konstrukcyjne, będące od dawna na rozrachunku, a także inne komórki organizacyjne, takie jak Szefostwo Ruchu, Służba Przygotowania Produkcji, Szefostwo Inwestycji itp.). Podporządkowanie to zdeterminowało rozwój informatyki zakładowej: większość eksploatowanych podsystemów dotyczy konstrukcyjno-technologicznego przygotowania produkcji — z emisją dokumentacji włącznie (kart technologiczno-zadaniowych, kwitów RW, kosztorysów materiałowych, planów zużycia materiałów itp.). Usługi świadczone dla Pionu Technicznego, Zaopatrzenia i Kooperacji oraz dla Pionu Produkcji przekraczają 60% mocy obliczeniowej ośrodka i wykazują tendencję rosnącą. Stosunkowo mały udział usług informatycznych świadczonych dla komórek zarządu stoczni (zgrupowanych w pionach: ekonomicznym, księgowym i pracowniczym) nasunął myśl odmiennego podejścia do rozliczenia kosztów informatyki i wydzielenia z kosztów ogólnozakładowych co najmniej wspomnianych 60%.

W tym celu Zakładowy Ośrodek ETO wydzielono organizacyjnie z komórek zarządu, tworząc dodatkowy wydział produkcji pomocniczej — zbliżony profilem do innych wydziałów produkcji pomocniczej w stoczni. Jednocześnie wprowadzono ewidencjonowanie kosztów informatyki w grupie zleceń produkcji pomocniczej w miejsce zleceń kosztów ogólnozakładowych.

Ewidencjonowanie kosztów informatyki

Budowa symbolu zlecenia kosztownego (źródłowego) opiera się na schemacie
7-MPK-RR
gdzie:
7 — kierunkowy symbol kosztów wydziałowych w stoczni
MPK — miejsce powstawania kosztów
RR — pozycja analityczna kosztów w układzie rodzajowym
Natomiast budowa symbolu kont księgowych, na których rozliczane są wtórnie koszty, jest następująca:
531-MPK

Dla ośrodka przewidziano następujące symbole MPK:
350 — koszty ogólnowydziałowe ośrodka
551 — prace projektowo-programowe
352 — przygotowanie danych
353 — przetwarzanie komputerowe.

Powyższy podział na MPK nie odpowiada ściśle wewnętrznej strukturze organizacyjnej ośrodka, bowiem przykładowo „Prace projektowo-programowe" obejmują nie tylko projektowanie i programowanie nowych podsystemów w Dziale Projektowania i Programowania, ale również konserwację oprogramowania użytkowego i podstawowego w Dziale Przetwarzania Danych, natomiast „Przygotowanie danych" obejmuje nie tylko prace personelu przygotowania maszynowych nośników informacji, ale również płace operatorek nadzorujących i operatorów systemów (kontrolerów wejścia-wyjścia), koszty konserwacji i napraw oraz amortyzacja systemów MERA 9150, zużycie energii i nośników danych itp.

Przyjęto zasadę bezpośredniego obciążania kosztami „produkcyjnych" MPK, a minimalnego kosztów ogólnowydziałowych ośrodka (płace kierownictwa, referenta administracyjnego, koszty ogólnobiurowe, amortyzacja budynku, zużycie energii elektrycznej, CO, woda, telefony itp.).

W ślad za wydzieleniem MPK ośrodka dokonano szczegółowego przydzielenia poszczególnych stanowisk pracy do odpowiednich sfer działalności. Umożliwia to automatyczne rozliczanie kosztów robocizny na poszczególne MPK. Poważne trudności wystąpiły m. in. przy oszacowaniu kosztów zużycia nośników energii oraz amortyzacji części biurowca zajmowanego przez ośrodek. Trudności te pokonano przez zastosowanie normatywów odnoszących się do liczby zatrudnionych pracowników, zajmowanej przez ośrodek powierzchni, liczby punktów energetycznych itp. Główne odbiorniki energii, jakimi są komputery, zostały wyposażone w liczniki pomiaru zużycia.

Rozliczanie kosztów informatyki

Podobnie jak w przypadku Biura Projektowo-Konstrukcyjnego i innych wydziałów produkcji pomocniczej, wprowadzono w stoczni obowiązek formalnego zamawiania usług informatycznych oraz otwierania zleceń wewnętrznych dla poszczególnych podsystemów lub użytkowników. Odbywa się to w Dziale Handlowym, który fakturuje należności za wykonanie przez ośrodek usługi (podobnie jak za usługi świadczone dla zleceniodawców spoza stoczni).

Dla właściwego rozliczenia kosztów ośrodek prowadzi dokładną ewidencję czasu pracy projektantów i programistów (karty pracy) oraz poszczególnych zestawów komputerowych.

Jednostkowa kalkulacja kosztów

Dla poszczególnych rodzajów usług informatycznych sporządzono jednostkową kalkulację kosztów w następującym układzie:

- amortyzacja urządzeń
- płace wraz z narzutami
- remonty i konserwacje urządzeń

[1]) Całość kosztów ogólnozakładowych z danego roku kalendarzowego musi być rozliczona na produkcję zakończoną (a nie jak dotychczas, również na produkcję w toku, co dawało możliwość pewnych przerzutów kosztów z roku na rok)

- zużycie energii
- obsługa programowa
- usługi obce
- koszty wydziałowe.

Suma powyższych składników stanowi techniczny koszt wytworzenia, który powiększony o narzut kosztów ogólnozakładowych stoczni daje fabryczny koszt wytworzenia, który powiększony o zysk daje cenę zbytu.

W oparciu o powyższy schemat wykonano kalkulację obowiązujących w stoczni cen jednostkowych dla następujących rodzajów usług informatycznych [2]):

- jedna godzina pracy komputera ODRA 1305 (w przeliczeniu na pracę jednoprogramową) 3500 zł
- jedna godzina pracy systemu MERA 9150 1200 zł
- przygotowanie[3]) 10 000 znaków za pomocą systemu MERA 9150 500 zł
- jedna godzina pracy projektanta lub programisty 370 zł

Powyższe ceny jednostkowe są więc zbliżone do poziomu cen stosowanych w przodujących ośrodkach sieci ZETO, a znacznie niższe od poziomu cen w byłych ośrodkach resortowych lub branżowych, które obecnie nie korzystają z dotacji.

Warto nadmienić, że ośrodek stoczni ma obsadę jednozmianową, a komputer ODRA 1305 został uruchomiony dopiero w 1983 r.

[2]) Poziom cen obowiązujących w I półroczu 1985 r.

[3]) Wraz z kontrolą poprawności, wyjaśnianiem i kontrolą błędów przez operatorów systemów

* * *

Wprowadzenie dla ośrodka ograniczonego wewnętrznego rozrachunku miało na celu nie tylko typową „żonglerkę" księgową mającą na celu zmianę sposobu rozliczenia kosztów. Podstawowym celem kierownictwa Stoczni (zwłaszcza Dyrektora Ekonomicznego i Głównego Księgowego) było zahamowanie dynamiki wzrostu kosztów ogólnozakładowych, w czym pewien udział miał również ośrodek informatyki.

Dodatkowym motywem tych działań było podejrzenie, że część zestawień komputerowych jest w rzeczywistości mało przydatna i może być realizowana taniej metodami tradycyjnymi. Wewnętrzne zamawianie i fakturowanie usług informatycznych miało na celu skuteczne zahamowanie procesu „inflacji" wydruków komputerowych, wykonywanych dotąd bezpłatnie na każde żądanie użytkownika.

Rozrachunek wewnętrzny ośrodka działa w stoczni od stycznia 1984 r., trudno więc ocenić wszystkie jego zalety i wady. Można jednak wskazać na szereg przypadków „otrzeźwienia" użytkowników, gdy dowiedzieli się, ile będzie kosztować przetwarzanie określonego pliku dokumentów lub zbiorów. Poprawiła się również dyscyplina i jakość pracy w samym ośrodku, ponieważ za błędne wyniki użytkownik nie zapłaci, a zmarnowany czas pracy ludzi lub maszyn obciąży koszty wydziałowe ośrodka, co z kolei rzutuje na wysokość jego funduszu premiowego.

Niestety, nie poprawiła się dyscyplina „kooperantów" ośrodka (magazynów, technologów itp.). Dokumenty nadal spływają nierytmicznie, co powoduje okresowe przestoje, a następnie konieczność pracy w godzinach nadliczbowych.

Mankamentem opisanego rozrachunku jest pewne zwiększenie zakresu pracy administracyjnej w ośrodku wskutek wprowadzenia ewidencji i rozliczania wykonywanych usług.

IV/C. Organizacja i przetwarzanie plików

Samotesty, opracowane i opublikowane przez Association for Computing Machinery, nie stanowią standardu egzaminacyjnego i są przeznaczone wyłącznie do samodzielnego sprawdzania własnej wiedzy [1].

PYTANIA

14. Rozważmy następującą listę jednokierunkową, zrealizowaną łącznikowo (ang. linked list), w której pierwszym elementem jest R:

Dana	Łącznik
ELEMENT L	2
ELEMENT P	4
ELEMENT R	5
ELEMENT B	pusty
ELEMENT Q	1

Załóżmy, że nowy element S pojawia się fizycznie u dołu tej listy, ale logicznie powinien być dowiązany między elementami R oraz Q. Jak po wykonaniu tego dołączenia będzie wyglądać prawidłowa kolumna łączników?

a) 2, 4, 6, pusty, 1, 5

b) 2, 4, 5, 6, 1, pusty

c) 2, 4, 5, pusty, 6, 1

d) 2, 4, 1, pusty, 6, 5.

15. Ważną cechą listy dwukierunkowej (ang. two-way, mniej poprawne — doubly), nie występująca w listach pierścieniowych (ang. circular list), jest łatwość wskazywania poprzednika każdego elementu składowego bez względu na kierunek obiegu. W której z wymienionych niżej operacji ta cecha list dwukierunkowych najwyraźniej się akcentuje?

a) uzyskanie dostępu do wskazanego elementu

b) kopiowanie całej listy

c) odtwarzanie zagubionego łącznika

d) scalanie dwu list.

16. Wyobraźmy sobie nieposortowany plik 14 400 rekordów. Załóżmy, że program wstępnego sortowania wewnętrznego odczytuje w kolejnych krokach tylko po 100 rekordów. Pierwsze 100 rekordów sortuje i umieszcza w pliku roboczym A, drugie 100 rekordów sortuje i umieszcza w pliku roboczym B — itd. aż do wyczerpania pliku wejściowego, za każdym razem posortowane 100 rekordów umieszczając naprzemiennie w wymienionych dwu plikach roboczych. Ile razy trzeba będzie wykonać dwukierunkowe przebiegi scalenio-

[1]) Communications of the Association for Computing Machinery, vol. 21 (No. 2 — February 1978), pp. 112—114

we (ang. two-way merge), aby uzyskać prawidłowo posortowany cały plik wejściowy?

a) 140 b) 70 c) 8 d) 2.

17. Które z wymienionych metod są praktycznie użyteczne przy generowaniu adresów rozproszonych (ang. hashing scheme) według klucza rekordu?

a) metoda reszty z dzielenia (ang. division-remainder)

b) metoda zaginania (ang. folding)

c) metoda podwajania (ang. multiplication by 2)

d) metoda zmiany podstawy numeracji (ang. radix transformation).

18. Rozważmy następującą alokację adresów w pliku indeksowo-sekwencyjnym:

Nr ścieżki	Indeksy ścieżek			Zawartość obszaru podstawowego			
	Największy klucz na ścieżce	Największy klucz w obszarze nadmiarowym	Adres pierwszego rekordu nadmiarowego				
1	20	20	pusty	10	14	18	20
2	60	60	pusty	26	34	41	60
3	75	75	pusty	72	73	75	pusty

Podaj, która z poniższych postaci indeksów ścieżki nr 2 — po dołączeniu do pliku rekordu o wartości klucza równej 55 — jest prawidłowa?

a) 60 60 adres rekordu 55
b) 60 55 adres rekordu 55
c) 55 60 adres rekordu 60
d) 55 55 adres rekordu 60.

ROZWIĄZANIA

14-d; 15-c; 16-c; 17-a, b, d; 18-c.

LITERATURA

ad 14. Kroenke D.: Database Processing. Science Research Associates, pp. 57—59

ad 15. Kroenke, ibid. pp. 60—61
Knuth D.: The Art of Computer Programming, Vol. I: Fundamental Algorithms. Addison-Wesley, pp. 278—279

ad 16. Burch J. and Strater F.: Information Systems: Theory and Practice. Hamilton Publ., pp. 211—213.
Flores I.: Data Structures and Management, IInd Ed. Prentice-Hall, pp. 133—136

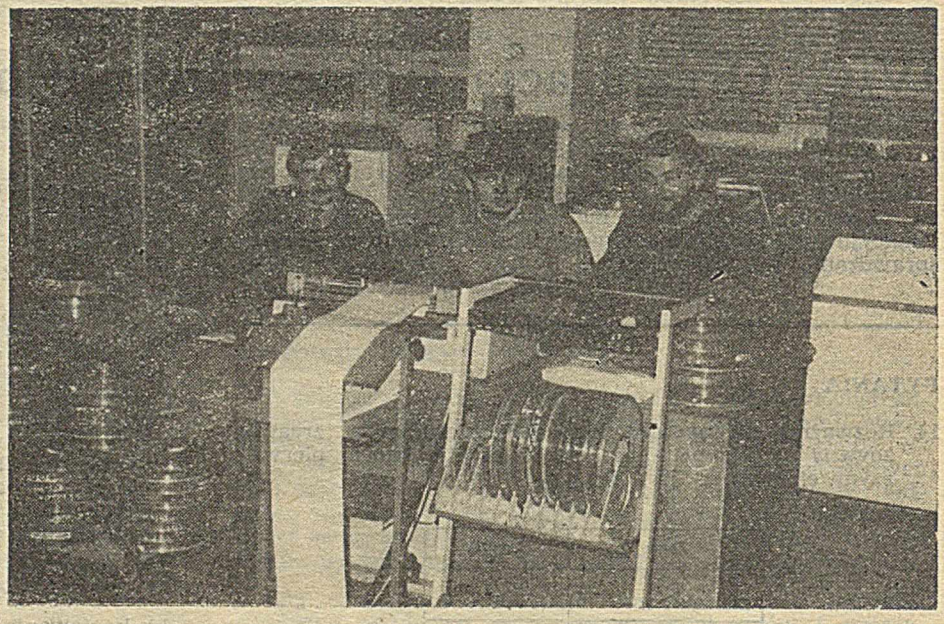
ad 17. Kroenke, ibid. pp. 44—46
Flores, ibid. pp. 250—254

ad 18. Kroenke, ibid. pp. 40—43

ADAM B. EMPACHER
JERZY L. ROSSOWSKI

Z kraju

10 lat Zakładu Obliczeniowego ETOB w Lublinie



Praca w sali komputera

18 października 1985 r. w Lublinie odbyła się uroczystość jubileuszu 10-lecia istnienia Zakładu Obliczeniowego ETOB.

Za datę narodzin Zakładu przyjmuje się 1 stycznia 1976 r., kiedy to zarządzeniem Dyrektora Warszawskiego Przedsiębiorstwa Informatyki Przemysłu Budowlanego ETOB oficjalnie powołano Zakład Obliczeniowy ETOB w Lublinie, jako oddział przedsiębiorstwa warszawskiego.

Początkowo Zakład mieścił się w kilku miejscach, korzystając z gościnności przedsiębiorstw budowlanych. Pierwsze jego wyposażenie stanowiły jedynie urządzenia do przygotowania danych typu SAM i Soemtron. Obliczenia komputerowe wykonywano w macierzystym przedsiębiorstwie w Warszawie, dowożąc dokumenty źródłowe, karty perforowane i tabulogramy. Zakończenie tego okresu nastąpiło w 1979 r., kiedy oddano do zagospodarowania obiekt z prawdziwego zdarzenia, mieszczący się przy ulicy Wieniawskiej 14 (p. Informatyka, nr 8—9'1979).

W kwietniu 1979 r. zainstalowano i uruchomiono pierwszy zestaw komputerowy ODRA 1305 w podstawowej konfiguracji. Choć początkowo pracowano tylko na jedną zmianę, z czasem uruchomiono drugą i trzecią zmianę, ze względu na duże zapotrzebowanie na usługi informatyczne ze strony przedsiębiorstw lubelskich. Rozbudowano

też zestaw komputerowy o dodatkowe urządzenia. Zainstalowano i uruchomiono urządzenia do przygotowania danych na taśmie magnetycznej typu Mera 9150.

Od 1 lipca 1981 r. zakład lubelski, w związku z reorganizacją sieci przedsiębiorstw ETOB, wszedł w skład wielozakładowego Przedsiębiorstwa Centrum Informatyki Przemysłu Budowlanego ETOB. W nowej strukturze organizacyjnej nastąpił dalszy rozwój zakładu w zakresie sprzętu i technologii.

W roku jubileuszu zainstalowano i uruchomiono drugi zestaw komputerowy Odra 1305, a także wdrożono system operacyjny GEORGE 3, znacznie efektywniejszy niż egzekutor.

Załoga zakładu ma w swoim dorobku szereg zgłoszonych i wdrożonych wniosków racjonalizatorskich, których zastosowanie przynosi korzyści ekonomiczne. Z inicjatywy Naczelnego Dyrektora Przedsiębiorstwa, doc. dr. inż. Marka Grochowskiego, lubelski zakład jako pierwszy podjął produkcję pamięci półprzewodnikowych do komputerów (przedtem produkowały je tylko firmy polonijne).

Zakład nie zaniedbuje też rozwoju usług dla przedsiębiorstw, proponując nowe systemy informatyczne i rozbudowując już eksploatowane. W dniu jubileuszu Zakłady Elektroniczne Elwro zrobiły zakładowi piękny „prezent" przysyłając urządzenie do transmisji danych, co stwarza nowe możliwości obsługi przedsiębiorstw.

Doceniając wkład pracy i uzyskane wyniki Ministerstwo Budownictwa i Dyrekcja Przedsiębiorstwa wyróżniły pracowników Zakładu odznaczeniami państwowymi i resortowymi przyznając: srebrny krzyż zasługi i medal 40-lecia PRL — ob. Bogumile Dybko, srebrny krzyż zasługi — mgr. Adamowi Ziółkowi, medal 40-lecia PRL i złotą odznakę Zasłużony dla Budownictwa — ob. Krystynie Hradilowej, medal 40-lecia PRL — ob. ob. Emilii Szabla i Halinie Szewera, srebrną odznakę Zasłużony dla Budownictwa — inż. Halinie Muszyńskiej i mgr. Ryszardowi Burskiemu.

Ponadto Zarząd Lubelskiego Oddziału TNOiK przyznał srebrny medal profesora Adamickiego Dyrektorowi Zakładu, mgr. Ryszardowi Burskiemu, a zakładowi — dyplom uznania.

Dyplomami uznania za wieloletnią pracę, działalność społeczną i wyniki w pracy wyróżniono 25 pracowników. W uroczystościach jubileuszu uczestniczyli też odznaczeni pracownicy innych zakładów przedsiębiorstwa.

R.B.

Techniki komputerowe w zarządzaniu produkcją

W dniach 3—5 października 1985 r. w Ciechocinku odbyła się konferencja naukowa na temat: **Techniki komputerowe w zarządzaniu produkcją**, zorganizowana przez Polskie Towarzystwo Ekonomiczne, Oddział Wojewódzki w Bydgoszczy (Komisja Informatyki oraz Klub Użytkowników ETO) przy współudziale: Instytutu Badań Systemowych PAN, Instytutu Organizacji i Projektowania Systemów Produkcyjnych Politechniki Gdańskiej oraz Zakładu Informatyki Stosowanej Akademii Techniczno-Rolniczej w Bydgoszczy.

Referat wprowadzający (I. Durlik, Politechnika Gdańska) obejmował całość powyższej tematyki. Jego świetnym uzupełnieniem był opis stanu i kierunków zastosowań informatyki w zarządzaniu przedsiębiorstwami węgierskimi (G. Markus, K. Faqquas).

Zdecydowana większość referatów dotyczyła zastosowań komputerów w konkretnych przedsiębiorstwach. Przykładowo, przedstawiono algorytm wyznaczania optymalnego harmonogramu produkcji i jego zastosowanie w sterowaniu produkcją w TELKOM-ZWUT oraz komputerowe wspomaganie organizacji systemów produkcyjnych na przykładzie rozwiązań zastosowanych w tych samych zakładach (M. Potrzebowski, E. Michalewski i in., Instytut Badań Systemowych PAN). W innym referacie omówiono organizację przetwarzania rozproszonego na komputerze Honeywell Bull w zarządzaniu produkcją ZWUT (K. Dutkowski, ZWUT).

Duża grupa referatów, dotycząca użycia komputerów w Zakładach Teleelektronicznych TELKOM-TELFA w Bydgoszczy została przedstawiona przez pracowników tych Zakładów. Omówiono zastosowanie systemu MERA 9150 do zarządzania produkcją (E. Fladrowska, M. Trzęsowska) oraz zarządzanie produkcją Wydziału Mechanicznego przy wykorzystaniu komputera, minikomputerów i mikrokomputerów (A. Hanuszek, S. Woźniak). Dwa inne referaty tej grupy dotyczyły wykorzystania komputera i minikomputerów w rozliczaniu zużycia materiałów produkcyjnych (St. Rafiński, D. Woźniak) oraz zastosowanie mikrokomputerów IMP-85 w kalkulacji kosztów i obsłudze pionu ekonomicznego (A. Dzianott i in.).

W innych referatach szczegółowych omówiono wykorzystanie minikomputera MERA 9150 do planowania i sterowania zadań na Wydziale Mechanicznym w Zakładach Radiowych Radmor w Gdyni (M. Wirkus, Politechnika Gdańska) oraz wielodostępny system on-line obsługujący Narodowy Bank Polski w rejonie bydgoskim (K. Wawrzyniak, Wojewódzki Ośrodek Obliczeniowy NBP).

W dwóch ogólniejszych referatach przedstawiono ocenę stanu rozwiązań planowania operacyjnego produkcji w świetle badań wykonanych w niektórych przedsiębiorstwach województwa bydgoskiego oraz ocenę możliwości wykorzystania wyników wielowariantowych optymalnych planów produkcji do realizacji planowania strategicznego w przedsiębiorstwie (Z. Jaskulski, L. Drelichowski i in., Akademia Techniczno-Rolnicza, Bydgoszcz).

W konferencji uczestniczyło 330 osób z różnych przedsiębiorstw z całej Polski: dyrektorzy, szefowie produkcji, kierownicy wydziałów produkcyjnych, kierownicy służb gospodarki materiałowej i profesjonalni informatycy.

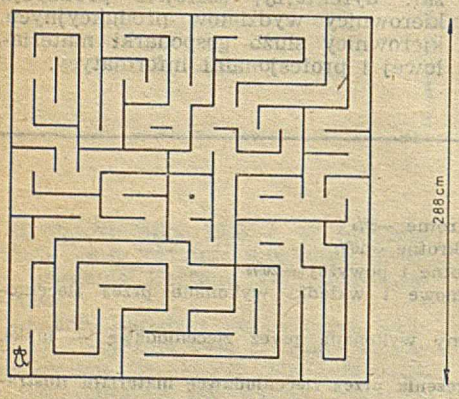
Mikromyszy na start!

Konstruowanie robotów przejawiających rozmaitego rodzaju inteligencję ma już dość długą historię i zdążono odnotować na tym polu poważne osiągnięcia. W czasach prehistorycznych, tj. około 10—15 lat temu, była to jednak zabawa zarezerwowana praktycznie tylko dla dużych zespołów badawczych, dysponujących silnym zapleczem badawczym i finansowym.

Z chwilą pojawienia się układów mikroprocesorowych z problemem mógł zmierzyć się każdy. Możliwość upakowania całego mikrokomputera na płytce mniejszej niż tabliczka czekolady zafascynowała konstruktorów. Po raz pierwszy można było bez kłopotu ulokować we wnętrzu robota namiastkę ludzkiego mózgu. Skądinąd problem był ciekawy od strony czysto poznawczej i naukowej. Zastosowanie wbudowanego komputera do sterowania maszyną obiecywało bardzo wiele — co najmniej dobrą zabawę. Tak narodziła się koncepcja konkursów, łącząca przyjemne z pożytecznym. Konkursów takich było już wiele. W dalszej części opiszemy dwa z nich i proponujemy wziąć w nich udział!

Elektroniczna mysz w labiryncie

Idea tego konkursu narodziła się w drugiej połowie lat siedemdziesiątych w USA. Patronuje mu tak poważna organizacja jak IEEE, znana u nas głównie jako wydawca wielu znakomitych czasopism naukowych z dziedziny elektroniki. Pierwsze zawody europejskie zorganizowano w 1980 roku pod egidą EUROMICRO i IEEE, przy okazji najpoważniejszej europejskiej konferencji poświęconej technice mikrokomputerowej — EUROMICRO' 80. Ponieważ w Europie obowiązuje nieco odmienny w szczegółach regulamin konkursowy, skupimy się na wersji bliższej nam geograficznie.



Labirynt konkursowy z finałów brytyjskich, Londyn 1984 (proszę zauważyć, że proste reguły, np. „zawsze w prawo", nie zdają tu egzaminu).

Zadaniem uczestników turnieju MICROMOUSE (ang. mikromouse — mikromysz) jest skonstruowanie robota potrafiącego znaleźć najkrótszą drogę w labiryncie w ograniczonym czasie. Elektroniczna mysz zastępuje więc tutaj tradycyjnego szczura używanego powszechnie w licznych laboratoriach naukowych. Podobnie jak w doświadczeniach z żywym zwierzęciem, zadanie polega na znalezieniu drogi do środka nie znanego labiryntu. Sam labirynt jest dość skomplikowany (rys.).

Problem jest dość trudny, gdyż robot poruszając się w labiryncie musi dysponować odpowiednimi czujnikami umożliwiającymi postrzeganie otoczenia, musi zapamiętywać drogę, którą przebył, a ponadto — konstruować sobie plan labiryntu. Ponieważ celem jest znalezienie najkrótszej drogi, algorytm sterujący ruchami robota musi być aktywny poznawczo (badać nie rozpoznane jeszcze korytarze labiryntu). Dopiero w drugim etapie można w zbadanym labiryncie znaleźć najkrótszą drogę do celu, którym jest środek labiryntu. Organizacyjnie wygląda to w ten sposób, że każda mysz dysponuje pewnym czasem przeznaczonym na eksplorację labiryntu oraz na zasadniczy, konkursowy przebieg. Wygrywa ta mysz, która dotarła do celu w najkrótszym czasie.

Dodatkowym utrudnieniem jest wymaganie całkowitej niezależności. Każdy robot musi mieć własne zasilanie (baterie) i napęd, i musi być kompletnie samosterowny. Jedyne, co może zrobić jego konstruktor, to umieścić go w labiryncie i nacisnąć przycisk START.

Oprócz właściwego konkursu, organizuje się także pokazy. Sędziowie i publiczność podziwiają wtedy dodatkowe możliwości robotów. Dla przykładu, można wspomnieć o myszach, które potrafią się podpisać, tańczyć czy śpiewać. Szczególną popularnością cieszą się cyfrowe syntezatory dźwięku. Mysz wyposażona w taki układ obwieszcza na przykład *a teraz skręcam w prawo* albo *żeby tylko tu nie było kota!*, a na koniec oczywiście *hurra! udało się.*

Podczas dwóch pierwszych finałów europejskich, w których brało udział po kilkanaście konstrukcji, okazało się, że algorytmy poruszania się w labiryncie nie są banalne. W 1980 roku oprócz zwycięzcy (Stirling Mouse, konstruktor Nick Smith), tylko dwie inne myszy wykazały się inteligentnym zachowaniem. W roku 1981 tylko połowa myszy znalazła środek labiryntu. Jednak większość z nich przejawiała już zdecydowanie celowe zachowanie, tak że zarówno publiczności, jak i uczestnikom konferencji często zapierało dech w piersiach, a brawa rozlegały się co chwila. W ostatnich latach problemy te przezwyciężono i coraz istotniejszą rolę zaczęła odgrywać konstrukcja mechaniczna i elektroniczna. Najlepsze myszy dysponują obecnie czujnikami podczerwieni, radarem ultradźwiękowym, możliwością przyspieszania na długich prostych i hamowania przed zakrętami. Takie właśnie są konstrukcje Dave Woodfielda — aktualnego mistrza kontynentu. Wszystko to jednak musi mieć wsparcie w odpowiednio rozbudowanym algorytmie. W ostatnich finałach jego mysz Enterprise momentami osiągała szybkość 3 m/sek! Niewiarygodne, gdy przypomnimy, że jest to tylko automat wielkości radia tranzystorowego, zasilany typowymi bateriami. (co prawda matka natura może pochwalić się lepszymi osiągnięciami!).

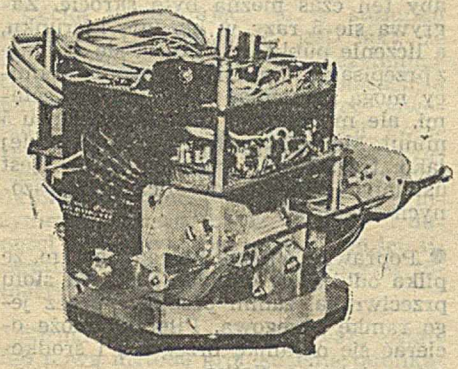
W historii dotychczasowych konkursów dominowali Brytyjczycy i Finowie, zdobywając większość nagród. W 1980 roku zwyciężył Nick Smith ze swoją Stirling Mouse. Dave Woodfield wygrał trzykrotnie: w 1981 r. z Thumper oraz w 1984 i 1985 r. z Enterprise. Aż dwa lata pod rząd, w 1982 i 1983 roku, najlepszą konstrukcją był Microsaurus Fina Hannu Matti Jarvinena z Tampere University of Technology. Smith, Woodfield i Jarvinen, a także Alan Dibley (również z Wielkiej Brytanii) zdobywali także miejsca medalowe, przywożąc na zawody z reguły po kilka konstrukcji.

Zaskakująco wielu uczestników konkursu było amatorami. Przykładowo, w finałach 1981 roku autor oglądał w Paryżu studentów (m.in. cała ekipa przybyła z Tampere), a nawet uczniów szkoły średniej (z Worthing, Wielka Brytania).

Fot. 1. Nick Smith

Najciekawszym przykładem amatora jest chyba zwycięzca pierwszego konkursu i jednocześnie zdobywca drugiej nagrody w 1981 roku — Nick Smith (fot. 1). Jest on z wykształcenia ekonomistą i pracuje w Gas Board — instytucji rządowej, będącej odpowiednikiem wydziału naszego Ministerstwa Energetyki. W swojej pracy zawodowej, obejmującej opracowywanie założeń finansowych projektowanych gazociągów, na co dzień korzysta z dużych systemów komputerowych. Najpierw zafascynowała go możliwość

zbudowania mikrokomputera w domu, a potem usłyszał o konkursie. O tym, jak wielkim był zapaleńcem, niech świadczy fakt, że uruchomił i przetestował swój 8-bitowy mikrokomputer oraz całą elektronikę robota mając do dyspozycji tylko miernik uniwersalny (uważał, że oscyloskop był trochę za drogi, jak na takie hobby), a wprowadzenie i wyprowadzenie danych przez długi czas było tylko binarne (klawisze i lampki). Zirytowany niewygodą takiej współpracy z maszyną, Nick zrobił sobie własny programator pamięci PROM.



Fot. 2. Stirling Mouse

Jego Stirling Mouse (fot. 2) jest sterowana przez 8-bitowy mikroprocesor RCA CDP1802, z kostką CDP1851 jako układem we-wy. Pamięć zawiera tylko dwie kostki: 1K PROM i 1K RAM, w sporej części niewykorzystane; zasadniczy algorytm zajmuje ok. 200 bajtów. Konstrukcję Stirling Mouse cechuje wielka prostota i funkcjonalność, przy dużej niezawodności. Mysz ma dwa koła napędzane niezależnymi silniczkami elektrycznymi i odpowiednio sprytne wsporniki, pozwalające na poprawne zachowanie się jednostki napędowej na niezbyt równej powierzchni oraz zapewniające brak poślizgów. Wszystkie czujniki położenia są mechaniczno-elektryczne, przy czym najważniejsze z nich są wielostopniowe, z odpowiednim kodowaniem napięciowym położenia za pomocą przełączanych przez czujnik rezystorów. W tym rozwiązaniu bardzo wygodne okazało się zastosowanie odpowiednio wyprofilowanych skrzydełek, ślizgających się po ściankach labiryntu (zgodnie z regulaminem). Kiedy odwiedziłem Nicka pod koniec 1984 roku, wyjęta z pudełka Stirling Mouse działała po założeniu baterii jak nowa!

Micro ping-pong

Propozycję tego konkursu zapowiadano już od około roku w kołach uczestników turnieju mikromyszy. W 1985 roku ukazał się tymczasowy regulamin zawodów, a finały europejskie odbyły się we wrześniu w Brukseli, podczas konferencji EUROMICRO'85. Zwyciężył robot Charly, któremu udało się dwukrotnie odbić piłeczkę. Ten konkurs stanowi prawdziwe wyzwanie dla konstruktorów. Zadaniem jest ni mniej ni więcej tylko skonstruowanie robota grającego w nieco zmodyfikowaną wersję tenisa stołowego. Stół co prawda znacznie zmniejszono (głównie zwężono), ale za to gra się zwykłą piłeczką ping-pongową!

Ten bardzo trudny konkurs będzie również powtarzany co roku przy okazji konferencji EUROMICRO, być może przy niewielkich zmianach regulaminowych. Osobiście po cichu podejrzewam, że wcześniej czy później może dojść do konfrontacji człowieka z maszyną i na tym polu. Podobnie jak w szachach prawdopodobny rezultat będzie taki, że tylko mistrz będzie mógł wygrać z robotem, tak jak tylko mistrzowie szachowi mogą obecnie wygrać z najlepszymi programami szachowymi. Co prawda szachy programuje się od 30 lat, więc na elektronicznego mistrza tenisa stołowego będziemy musieli prawdopodobnie zaczekać do 2000 roku. Na razie wszystko dopiero się zaczyna, toteż organizatorzy zachęcają do przywiezienia każdego robota, który potrafi choćby zamachnąć się na piłeczkę — bez względu na to, czy potrafi ją odbić czy nie! Żeby jeszcze nie te koszty wyjazdu...

REGULAMINY

Dla osób, które pragną wziąć udział w jednym z tych konkursów podajemy kompletne regulaminy konkursowe.

Turniej mikromyszy (Euromouse Contest)

Wymiary

Labirynt składa się z 16×16 kwadratów, rozłożonych na siatce o boku 18 cm (7 cali). Ściany labiryntu mają grubość 12 mm (pół cala), w związku z czym korytarze mają szerokość 16,5 cm (6,5 cala). Ściany mają wysokość 5 cm (2 cale), pomalowane są na biało, natomiast ich wierzch na czerwono. Celem jest słupek umieszczony w środku labiryntu o wymiarach: podstawa kwadratowa o boku 2,5 cm (1 cal), wysokość 20 cm (8 cali), który w razie potrzeby może być usunięty. Kwadratem startowym jest kwadrat znajdujący się w lewym dolnym rogu labiryntu. Mysz początkowo jest ustawiona tak, że środek znajduje się po przekątnej z jej prawej strony. Powierzchnia labiryntu, po której porusza się mysz jest z laminatu pomalowanego na czarno.

Tolerancje

Nie można zakładać większej dokładności wymiarów niż 5%. Labirynt może być wykonany przy użyciu jednostek metrycznych albo cali i stóp, a podane liczby mogą być przybliżeniami (do 5%). Połączenia części labiryntu nie będą powodowały uskoków większych niż 0,5 mm — mogą być pokryte taśmą. Należy jednak się liczyć z odkształceniami powierzchni labiryntu podczas transportu, co może powodować zmianę kąta nachylenia na połączeniach do 4 stopni.

Zasady szczegółowe

- Każda mysz dysponuje czasem łącznym 15 min (limit może ulec zmianie). Jury może jednak zadecydować o wycofaniu myszy przed upływem tego czasu, jeśli nie czyni ona żadnych postępów albo jeśli jej zachowanie zagraża stanowi labiryntu.

- Gdy myszy udaje się znaleźć drogę do środka labiryntu notuje się czas. Właściciel może ponownie umieścić wtedy mysz na polu startowym, tak że może ona wykorzystać swoją nabytą wiedzę o labiryncie podczas drugiego przebiegu. W ramach 15-minutowego przydziału czasu można wykonać dowolną liczbę przebiegów.

- Jeśli mysz nie może znaleźć drogi, to właściciel ma prawo zwrócić się do Jury z prośbą o pozwolenie na przerwanie trwającego przebiegu i ponowne umieszczenie jej na polu startowym. Jury może nałożyć karę czasową za ponowny start. Mysz nie może wrócić na pole startowe tylko dlatego, że skręciła w niewłaściwą stronę, a od decyzji Jury nie ma odwołania. Jury może ukarać uczestnika karą czasową za wszelkie nieuprawnione manipulacje według własnego uznania, a mysz ukarana w ten sposób przegrywa z każdą inną myszą, która osiągnęła środek bez takich manipulacji.

- W uzasadnionych przypadkach oraz za zgodą Jury, można dokonać wymiany baterii oraz drobnych napraw. Nie wstrzymuje to jednak zegara odmierzającego upływ 15 minut.

- W przypadku, gdy żadna z myszy nie osiągnęła celu, o zwycięstwie decyduje Jury, biorąc pod uwagę takie czynniki jak przebyty dystans, celowość ruchów w porównaniu z zachowaniem przypadkowym i jakość sterowania.

- Jeśli mysz ma awarię techniczną uniemożliwiającą jej kontynuowanie zawodów, to Jury może zezwolić na ponowny start w późniejszym terminie. Jednakże od pozostałego jeszcze czasu odlicza się 3 minuty karne.

- Jury przydziela oprócz nagrody głównej także inne nagrody według własnego uznania, w szczególności za najciekawszą konstrukcję, najlepszą prezentację itp.

- Wszystkie myszy muszą zostać oddane pod opiekę Jury zanim nastąpi odsłonięcie labiryntu. Właściciel może zabrać mysz na start tylko na oficjalne polecenie Jury.

- Nie ma ograniczeń na wysokość konstrukcji myszy, może ona wystawać ponad ścianki labiryntu, ale nie może być wywrotna. Natomiast jej długość i szerokość nie może przekraczać 25 cm. Myszy muszą być całkowicie samowystarczalne i nie jest dozwolona żadna zewnętrzna pomoc. Konstruktor ma swobodę wyboru techniki postrzegania ścianek przez mysz, jednakże mysz nie może oddziaływać na ściankę siłą, która mogłaby powodować uszkodzenie labiryntu. Napęd można zastosować dowolny, przy zastrzeżeniu, że źródło zasilania nie zanieczyszcza oto-

czenia (zasada ta wyklucza silniki spalinowe). Sędziowie mogą nie dopuścić myszy do zawodów, jeśli uważają, że istnieje istotne ryzyko uszkodzenia lub zniekształcenia labiryntu. Mysz nie może zostawiać żadnych pozostałości w labiryncie. Można zastosować dowolny układ lokomocyjny poruszający mysz, o ile zachowane będą zasady pokonywania drogi w labiryncie — niedozwolone jest przechodzenie ponad ściankami ani przechodzenie przez nie w inny sposób.

● Sposób uruchamiania myszy musi być prosty i nie może dawać właścicielowi możliwości wyboru strategii stosowanej przez mysz. Dla przykładu, decyzja, że należy dokonać szybkiego przebiegu do środka labiryntu, gdyż kończy się przydzielone 15 minut, musi być podjęta przez samą mysz.

● Żadna część jednej myszy (za wyjątkiem baterii) nie może być przenoszona do innej. Przykładowo, gdy ta sama obudowa używana jest z dwoma różnymi układami sterującymi, stanowią one tę samą mysz i mają do dyspozycji tylko pojedynczy przydział 15 minut.

Turniej automatycznego ping-ponga (Robot Ping-Pong Contest)

Poniższy regulamin jest regulaminem tymczasowym*) i niektóre jego punkty mogą ulec zmianie po turniejach wstępnych. W szczególności do ustalenia pozostaje, czy piłeczka może odbijać się od ramki brzegowej itp. Organizatorzy proszą więc o uwagi i sugestie.

● Piłeczka jest standardową piłeczką do tenisa stołowego, bez żadnych specjalnych oznaczeń.

● Stół ma 2 metry długości i 0,5 metra szerokości, a jego powierzchnia znajduje się 75 cm nad poziomem podłogi.

● Powierzchnia stołu jest gładka, matowoczarna, bez żadnych linii brzegowych. W turnieju próbnym będzie to gładka płyta laminowana pomalowana czarną emulsją.

● Na każdym końcu stołu jest umieszczona pionowa ramka brzegowa, o powierzchni wewnętrznej 0,5 m². Materiał, z którego wykonana jest ramka, ma grubość najwyżej 2 mm (dla zmniejszenia przekroju widocznego utrudniającego postrzeganie), a jego szerokość wzdłuż stołu nie przekracza 1 cm. Na zewnątrz ramki brzegowej może być zamontowana przezroczysta siatka dla ułatwienia oceny czy odbicie było poprawne, czy nie.

● Na środku stołu zamontowana jest trzecia pionowa ramka o wymiarach wewnętrznych 0,5 m szerokości i 75 cm wysokości z materiału podobnego do tego, z którego wykonano ramki brzegowe. Cienki drut rozciągnięty w poprzek tej ramki na wysokości 25 cm nad stołem podtrzymuje przezroczystą siatkę (wykonaną z materiału podobnego do tego, z którego robi się siatki do włosów). Siatka ta nie będzie pochłaniać więcej niż 10% światła przechodzącego przez nią — a prawdopodobnie dużo mniej.

● Na szczycie ramki środkowej znajduje się urządzenie zagrywające. Podtrzymuje ono piłeczkę w pełni widoczną dla obu robotów. Środek piłeczki znajduje się na wysokości 62,5 cm nad poziomem stołu, dokładnie ponad jego środkiem. Urządzenie to skonstruowane jest z drutu o grubości nie przekraczającej 1 mm, a po zagraniu piłeczki usuwa się całkowicie ponad poziom ramki. Piłeczka jest zagrywana w kierunku robota mającego prawo do zagrywki i odbija się raz, zanim przejdzie przez jego ramkę brzegową.

● Oświetlenie zapewniają lampy tungstenowskie, zamontowane na tyczkach w rogach kwadratu 4×4 m. Środek kwadratu pokrywa się ze środkiem stołu. Poziom oświetlenia będzie wynosić ok. 10 wg miernika Westona, co odpowiada naświetleniu 1/60 sek przy przesłonie 5,6 dla filmu o czułości 100 jednostek ASA.

● Na umieszczenie robota przeznacza się miejsce spoczynkowe o powierzchni 1 m², za krawędzią końca stołu. Żadna część robota nie może dotykać ramki brzegowej ani przekraczać jej płaszczyzny. Robot nie może rozciągać się dalej niż 1 m w bok licząc od centrum ramki, tj. łącznie 2 m. Gniazdko zasilania będzie znajdować się na krawędzi miejsca spoczynkowego. W Europie zasilanie wynosi 220 V przy 50 Hz i bezpieczniku 5 A, a w USA 110 V przy 60 Hz i bezpieczniku 10 A.

● Rozmiary rakietki muszą zawierać się w kole o średnicy 12,4 cm. Rakietka może poruszać piłeczką tylko przez jednokrotne odbicie o jej powierzchnię. Niedozwolone jest chwytanie piłeczki, wydmuchiwanie, odpychanie elektrostatyczne ani żadna ich kombinacja. Powierzchnia rakietki może być zakrzywiona, ale podwójne uderzenie będzie powodować utratę punktu.

● Części robota widoczne dla przeciwnika muszą być czarne, wyłącznie z absorbcją podczerwieni w rejonie pasma fal o długości 1 mikrona. To wymaganie można spełnić przez pomalowanie robota czarną emulsją. Na żądanie przeciwnika (w pierwszym roku mało prawdopodobne) rakietka musi posiadać jaskrawą czerwoną lampkę LED (diodę świecącą) zamocowaną w środku oraz zieloną zamocowaną 5 cm od niej.

● Z wyjątkiem lampek opisanych w poprzednim i następnym punkcie, robot nie może emitować światła w kierunku przeciwnika. W celu wykrycia zbliżania się piłeczki do rakietki można użyć promieniowania krzyżowego. Światło rozpraszane w kierunku przeciwnika może pochodzić jedynie od samej piłeczki, a emitowane światło nie może być przesadnie jaskrawe.

● Transmisja ultradźwiękowa jest dozwolona tylko w czasie zbliżania się piłeczki do rakietki i musi ustać z chwilą kontaktu piłeczki z rakietką. Jeśli taka transmisja jest stosowana, to obok ramki środkowej musi być zamontowana jaskrawa, czerwona dioda świecąca, sterowana tak, aby była widoczna dla sędziego oraz dla (zamontowanej również na kablu) fotokomórki przeciwnika. Dioda ta musi świecić się podczas wysyłania ultradźwięków.

● Roboty dysponują czasem 15 sek. na zsynchronizowanie swoich systemów wizyjnych na piłeczce przed zagrywką. Pożądane byłoby, choć nie jest wymagane, aby roboty sygnalizowały swoją gotowość (dźwiękiem albo głosem) tak, aby ten czas można było skrócić. Zagrywa się 5 razy w każdym kierunku, a liczenie punktów odbywa się zgodnie z przepisami tenisa stołowego. Zawodnicy mogą zamieniać stoły między setami, ale muszą tego dokonać w ciągu 5 minut. Mecze rozgrywa się co najmniej jako trzysetowe, a liczba setów jest ustalana zależnie od liczby zgłoszonych robotów.

● Poprawne odbicie polega na tym, że piłka odbija się dokładnie raz od stołu przeciwnika, zanim przejdzie przez jego ramkę brzegową. Piłeczka może ocierać się o ramkę brzegową i środkową, a także o siatkę.

● Jeśli broniący odbije 20 piłeczek pod rząd, wygrywa punkt.

● Jury może zdyskwalifikować robota, jeśli zagraża on bezpieczeństwu, albo nałożyć karę za naruszenie reguł sportowych.

● Wszystkie wymiary podano z tolerancją do 2%.

Oczywiste jest, że roboty powinny być łatwo przenośne i możliwie zabawne Nie powinny też być przesadnie hałaśliwe.

Zgłoszenia i dalsze informacje

Tegoroczny konkurs odbędzie się w dniach 15—18 września w Wenecji. Zapytania, zgłoszenia i wszelką korespondencję należy kierować do stałego organizatora, którym jest John Billingsley, pod następujący adres:

Dr John Billingsley
Department of Electrical and Electronic Engineering
Portsmouth Polytechnic
Anglesea Road
Portsmouth PO1 3DJ
Wielka Brytania

Organizatorzy nie pobierają żadnych opłat za udział w konkursie, natomiast nagrody są dość interesujące — w ostatnich latach były to przeważnie rozmaite systemy mikrokomputerowe, literatura, itp., nie licząc gwarantowanego programu telewizyjnego z udziałem zwycięzcy.

RYSZARD K. KOTT

*) Artykuł był pisany w 1985 r.

Zmierzch pomysłu Gutenberga

Co miesiąc w każdej dziedzinie wiedzy ukazują się setki czasopism specjalistycznych prezentujących dokonania naukowe i techniczne, dziesiątki zbiorów referatów konferencyjnych oraz książki naukowe. Opracowania te pochłaniają wiele ton papieru oraz wymagają dużego wysiłku w procesie przygotowania ich do druku.

Cechą wyróżniającą je w stosunku do gazet i tygodników jest fakt, że żyją dłużej — do ich treści czytelnicy często wracają po kilku miesiącach, a nawet latach, poszukując przez odwołania bibliograficzne informacji związanych z aktualnie prowadzonymi pracami. Nie ma co ukrywać, że nawet najbardziej pracowici naukowcy i inżynierowie mają coraz większe trudności w pełnym ogarnięciu całości literatury, nawet w bardzo wąskim tematycznie zakresie.

Firma Lipra-1 zaproponowała ostatnio nową metodę gromadzenia publikacji z wykorzystaniem techniki informatycznej. Zamiast drukowania, na przykład 500-stronicowego czasopisma Byte proponuje się wytworzenie kostki pamięci ROM o odpowiedniej liczbie bitów informacji. Na początek firma przygotowała „box journal" (czasopismo w pudełku) o pojemności 1024 K bitów w trzech wersjach:

- FAchar-64 — umożliwiający zapisanie w postaci znakowej tekstu odpowiadającego 64 stronom maszynopisu (ok. 2 K znaków na stronie),
- FAgraph-8 — umożliwiający zapisanie w postaci graficznej tekstu oraz rysunków odpowiadających 8 stronom maszynopisu (ok. 16 K punktów na stronie),
- FAcolor-2 — umożliwiający zapisanie w 16 kolorach rysunku formatu A4.

W każdym z tych przypadków box journal jest wielkości pudełka od zapałek. Pudełko to jest, zdaniem firmy, elementem niezniszczalnym, z wyprowadzonym na zewnątrz sprzęgiem 48-bitowym. W obecnych warunkach istnieją techniczne możliwości wielkonakładowego (przepraszam — wielkoseryjnego) automatycznego wytwarzania takich pudełek.

Oczywiście, naturalne wydaje się dołączenie takiego pudełka jako zewnętrznej pamięci do komputera osobistego. Należy jednak pamiętać, że nie każdy jeszcze dysponuje komputerem, a ponadto trudno z nim zasiąść wygodnie w fotelu jak to można zrobić przeglądając gazetę. Jednakże firma wpadła na nowy pomysł, proponując dostarczenie każdemu zainteresowanemu czytadła (ang. unreader) w postaci urządzenia będącego skrzyżowaniem płaskiego ekranu z prostym mikroprocesorem — zasilanym z baterii. Czytadło to, po włączeniu do niego pudełka, umożliwia przeczytanie treści box journal, strona po stronie, wyświetlanej na ekranie. Dodatkowo możliwe jest szybkie przejrzenie zawartości pudełka — tak jak to często robimy z wieloma czasopismami. Tak więc pozostajemy prawie przy naszych przyzwyczajeniach, mając kontakt z nową techniką (nie ma tylko potem możliwości zamiany makulatury na papier).

Jest to dopiero początek rozwoju tego pomysłu, zastępującego z powodzeniem pracę papierni i drukarni. Firma LIPRA pracuje nad kolejnymi wersjami pudełek o większej pojemności. Równocześnie proponuje przystawkę do komputerów osobistych w celu wykorzystania pudełka jako zewnętrznej pamięci ROM równoważnej segmentowemu plikowi sekwencyjnemu (to już żadna nowość). Nowością zaś jest projekt biblioteki komputerowej złożonej z modularnych zestawów szyn umożliwiających dołączenie wielu pudełek. Moduły te są sprzężone z komputerem, który przez dodatkowe programowe i sprzętowe wyposażenie umożliwia wyszukiwanie informacji z tej własnej lokalnej biblioteki. Oczywiście nic nie stoi na przeszkodzie tworzeniu tego rodzaju bibliotek centralnych, dostępnych przez telefon z własnego komputera.

Zadbano też o możliwość samodzielnego przygotowania własnego tekstu w postaci pudełka. Wersja EFAchar-32 umożliwia zapisanie 32 stron maszynopisu przy użyciu programatora pamięci dołączonego do komputera zamiast (lub obok) drukarki. Do zredagowania tekstu można wykorzystać program redagujący. W tym momencie można zadać pytanie o ochronę przed kopiowaniem tekstów. Stąd też w pudełku znajduje się układ opóźniający szybkie przeglądanie wszystkich stron. Strona tekstu pobrana z pudełka blokuje szybkie pobranie następnej strony. Z kolei funkcja szybkiego przeglądania udostępnia z zasady tylko cześć strony. Moim zdaniem jest to jednak rozwiązanie połowiczne. Dalej trzeba liczyć na uczciwość potencjalnego kopisty.

Pozostaje jeszcze kwestia kosztów całego przedsięwzięcia i co ważniejsze, przekonanie wielu wydawców o konieczności zmiany techniki. Istotna sprawą jest też standaryzacja zasad odczytu informacji oraz standardu sprzęgu pudełka. Tworzona bowiem elektronicznie biblioteka musi być jednolita technologicznie w celu ułatwienia jej przeszukiwania przy wyszukiwaniu żądanych informacji.

Według ulotki reklamowej firmy, obecnie koszt wytwarzania box journal — nie licząc honorariów autorskich — nie przekracza 8—10 dolarów. Standardowe czytadło tekstu kosztuje ok. 50 dolarów. Zdaniem firmy koszty te, w przeciwieństwie do stale rosnących cen papieru i druku będą stale malały, zgodnie z tendencjami w mikroelektronice. Do rozstrzygnięcia pozostaje istotny na Zachodzie problem znacznego udziału opłat za reklamy w kosztach wydawnictw. W każdym razie, po pomyśle z gazetą telewizyjną, wprowadzenie czasopism elektronicznych jest szansą na ochronę lasów oraz na obejście problemów poligrafii.

Opracował

WACŁAW ISZKOWSKI

na podstawie firmowej ulotki reklamowej

Firma LIPRA-1, a właściwie 1 APRIL, nie istnieje. Do wymyślenia box journal, czyli pudełek z treścią informacyjną oraz czytadeł skłoniły mnie trudności z wydawaniem INFORMATYKI. Dla przykładu, artykuł ten został napisany 14 września 1985 roku, złożony w redakcji 18 września 1985 i wprowadzony do numeru Nr 1/86, który musiał być oddany do drukarni do 30 listopada 1985 roku. Numer wydrukowano dopiero .... — datę tę trudno mi przewidzieć pisząc te słowa. W każdym razie mam nadzieję, że czytelnicy pierwszego kwietnia 1986 roku już będą mogli go przeczytać.

Przepraszając zawiedzionych — możemy się zgodzić, że pomysł ten w realizacji nie przekracza aktualnych możliwości technologicznych i jest całkowicie realny. Może więc u nas w kraju, zamiast budować nowy Kwidzyń i czekać na rozwój poligrafii — zacząć od razu od box journal.

PS. W czasopiśmie Parallel Computing (Vol. 1, No 2, December 1984) znalazłem reklamę „A new electronic journal project", czyli zapowiedzi nowego czasopisma Computer-Compacts, które oprócz typowej wersji papierowej będzie dostępne sześć razy szybciej (?) przez sieć teleinformatyczną.

Pojawiły się też już optyczne dyski o pojemności 6 G bajtów, umożliwiające przechowywanie w postaci graficznej na przykład całej encyklopedii Britannica. Na razie jest możliwy tylko odczyt kolejnych stron. W każdym razie są to kroki we właściwym kierunku. (WI)

Quattro nowy komputer osobisty ICL

Fascynacja rozwojem amerykańskiej technologii mikrokomputerowej odsunęła nasze tradycyjne zainteresowania osiągnięciami technologii brytyjskiej, która na początku lat sześćdziesiątych ukształtowała, a następnie w decydującym stopniu przyspieszyła rozwój polskiego przemysłu komputerowego, a w konsekwencji polskiej informatyki.

W ostatnich latach zaobserwować można wspólne działania nielicznych już producentów europejskich (ICL, Siemens, Bull i Philips), jednoczących wysiłki, aby — podobnie jak w dziedzinie produkcji samolotów pasażerskich — stawiać czoła supremacji amerykańskiej oraz konkurencji japońskiej w produkcji komputerów. O sprawach tych pisaliśmy w INFORMATYCE wielokrotnie na podstawie doniesień prasy zagranicznej. Aktualna pozycja Wielkiej Brytanii na światowym rynku komputerowym oraz podejmowane przez rząd tego kraju działania mogą być w pewnej mierze inspiracją dla naszych wysiłków zmierzających do częściowego choćby odzyskania przez nas dawnej, całkiem niezłej, pozycji w europejskiej informatyce. Będzie to podejście znacznie bardziej racjonalne niż popadanie w kompleks szybko zwiększającego się zapóźnienia.

Z uwagi na wspomnianą tradycję 20-letniej już współpracy z firmą ICL, a także znaczny — w ciągu ostatnich dwóch lat — wzrost dostaw wyrobów tej firmy na rynek polski, warto bliżej zapoznać się z charakterystyką techniczną i użytkową najnowszego jej wyrobu, wprowadzonego do sprzedaży dopiero w drugiej połowie ub. roku. Jest nim komputer osobisty PC Quattro, zastępujący poprzednią rodzinę komputerów osobistych ICL PC.

PC Quattro to również rodzina profesjonalnych komputerów osobistych, obejmująca obecnie trzy modele: 19, 39 i 49. W porównaniu do swego poprzednika, Quattro charakteryzuje się 2 1/2 raza większą szybkością przetwarzania oraz możliwościami znacznej rozbudowy pamięci na dyskach sztywnych typu Winchester (do 60 MB). Podobnie jak poprzednik jest on ukierunkowany na pracę wielostanowiskową w zestawach do czterech niezależnych ekranowych stanowisk pracy [1]), korzystających ze wspólnych zasobów rozbudowanej konfiguracji jednego z tych stanowisk. Rozwiązanie takie, zapewniające bardziej efektywne wykorzystanie pamięci operacyjnej oraz wszystkich rodzajów urządzeń peryferyjnych, jest niewątpliwie rozwiązaniem bardziej ekonomicznym pod względem zarówno inwestycyjnym, jak i eksploatacyjnym. Przypomina ono tradycyjną koncepcję systemu wielodostępnego, ale niewspółmiernie mniej kosztowną. Chociaż Quattro dzięki przyjęciu filozofii systemu wielostanowiskowego i wyposażeniu go w niezbędne środki sprzętowe i programowe, kwalifikuje się do górnej granicy kategorii komputerów osobistych, to jednak fakt ten nie wyklucza jego stosowania jako typowego komputera osobistego do obsługi pojedynczego stanowiska pracy (model 19).

Quattro został oparty na 16-bitowym mikroprocesorze Intel 8086. Charakteryzuje się częstotliwością zegara 8 MHz, czterema poziomami przerwań oraz czterema kanałami dostępu do pamięci, co pozwala realizować równocześnie do 16 różnych zadań. Pamięć RAM może mieć, w zależności od modelu, standardową pojemność od 256 do 512 KB z możliwością jej rozbudowy modułami po 256 KB we wszystkich modelach do łączonej pojemności 1 MB (1024 KB). W obudowie jednostki centralnej mieszczą się jeden (w modelach 39 i 49) lub dwa (w modelu 19) napędy dyskietek o średnicy 5,25 cala z obustronnym zapisem podwójnej gęstości, co pozwala na jednej dyskietce zarejestrować 782 KB sformatowanych informacji (ok. 1 MB informacji niesformatowanych) oraz przesyłać je z prędkością 250 Kb/s. W modelach 39 oraz 49 znajduje się również wbudowany dysk typu Winchester, o pojemności 10 (model 39) lub 20 MB (model 49) oraz prędkości przesyłania 5 Mb/s. Oprócz tego do każdego z modeli dołączyć można dodatkowe moduły pamięci dyskowej typu Winchester 10 lub 20 MB do maksymalnej łącznej pojemności 60 MB.

Jednostka centralna Quattro ma 6 łączy (portów) asynchronicznych przeznaczonych do przyłączania monitorów ekranowych oraz urządzeń peryferyjnych i przesyłania informacji z szybkością od 50 do 19 200 bodów, jedno łącze synchroniczne o analogicznej szybkości transmisji przeznaczone do połączeń komunikacyjnych, a także jedno łącze asynchroniczne dostosowane do łączenia w sieci lokalne i przesyłania z szybkością do 512 Kb/s.

Quattro może być wyposażony w monitory ekranowe monochromatyczne, wyświetlające zielone znaki na czarnym tle lub odwrotnie, albo kolorowe, zapewniające jednoczesne wyświetlanie ośmiu różnych z palety 64 barw. Oba typy monitorów umożliwiają wyświetlenie 24 linii po 20, 80 lub 132 znaków w jedenastu wzorach znaków alfanumerycznych, a także 80 różnych znaków semigraficznych. Dzięki dużej rozdzielczości (640×400 punktów) zapewniają one możliwość uzyskania pełnej grafiki (w monochromatycznym opcjonalnie, w kolorowym standardowo). Na analogicznych warunkach dostarczana jest do obu monitorów myszka. Monitory można instalować w różnych odległościach od jednostki centralnej, przy czym z pełną szybkością transmisji (19 200 bodów) mogą pracować dwa monitory ustawione w odległości do 45 m od jednostki centralnej. Zwiększenie odległości do 1440 m (maksymalna długość) powoduje zmniejszenie szybkości transmisji do 300 bodów.

Wolnostojąca klawiatura została zaprojektowana pod kątem maksymalnej wygody i możliwości dostosowania do potrzeb użytkownika, w tym również dostosowania do ponad 10 różnych alfabetów narodowych (poprzez wymianę klawiszy). Oprócz standardowej klawiatury alfanumerycznej w układzie standardowym (QWERTY) istnieje wydzielone pole dodatkowych klawiszy numerycznych, pole 22 klawiszy funkcyjnych, umożliwiających stosowanie 85 najczęściej używanych instrukcji i procedur programowych, a także wydzielone pole klawiszy do obsługi programów przetwarzania tekstów.

W zależności od potrzeb wydawniczych użytkownik ma możliwość wyboru różnych rodzajów drukarek: mozaikowej, termicznej lub korespondencyjnej.

Drukarka mozaikowa może działać w trybie normalnym (ang. draft quality) z szybkością 160 zn./s oraz korespondencyjnym (ang. letter quality) z szybkością 33 zn./s. Drukowanie przebiega przy dwukierunkowym przesuwie wałka, przy szerokości wiersza od 136 znaków (normalna gęstość) do 233 znaków (maksymalne zagęszczenie znaków). Istnieje możliwość definiowania dodatkowych znaków graficznych przez użytkownika oraz precyzyjnej prezentacji graficznej (matryca 9×9 igieł), a także użycia papieru w postaci perforowanej taśmy ciągłej lub oddzielnych arkuszy.

Drukarka termiczna zapewnia druk normalny z szybkością 80 zn./s na papierze zwykłym dobrej jakości, termicznym lub przezroczystej folii, w tym również druk wielobarwny, a także precyzyjną prezentację grafiki przy rozdzielczości 144×144 punktów na cal, zarówno na oddzielnych arkuszach, jak i taśmie ciągłej. Istnieje również możliwość drukowania znaków w różnych wersjach językowych oraz uzyskania pisma jakości korespondencyjnej. Druk jednokierunkowy, przy szerokości do 80 znaków w wierszu.

Wreszcie drukarka korespondencyjna zapewnia druk najwyższej jakości w oparciu o wymienne głowice rozetkowe, a także możliwość regulowania gęstości zapisu.

Do Quattro przyłączyć można również ploter, umożliwiający wyprowadzenie w formacie (A3 lub A4) informacji graficznych oraz tekstowych w sześciu kolorach na papierze albo przezroczystej folii.

Ostatnim z możliwych urządzeń peryferyjnych jest nowy typ pamięci masowej na kasetowej taśmie magnetycznej o szerokości 1/4 cala. Pamięć ta ma nominalną pojemność 42,5 MB i jest przeznaczona do okresowego składowania danych zapisanych na dyskach typu Winchester.

[1]) Stąd zapewne nazwa Quattro, adaptowana z pierwowzoru, jakim jest nazwa słynnego modelu samochodu rajdowego firmy Audi Quattro, z niezależnym napędem na wszystkie cztery koła.

Wspomnieć należy również o możliwości zastosowania mikroprocesora Intel 8087 znacznie zwiększającego szybkość obliczeń numerycznych oraz grafiki.

Podstawą oprogramowania Quattro jest najnowszy produkt firmy Digital Research Corporation CCP/M — 86 (Concurrent CP/M) wielodostępny, wielozadaniowy system operacyjny dla mikrokomputerów 16-bitowych. Wraz z systemem operacyjnym dostarczane są podstawowe programy usługowe, interpreter języka BASIC (PBASIC) oraz asembler 8085/8086 wraz z programem do interakcyjnego uruchamiania programów. Oprócz tego dostarczane jest bogate oprogramowanie narzędziowe i aplikacyjne, obejmujące kilkaset pozycji, stanowiące dziś obowiązujący standard dla 16-bitowych komputerów osobistych.

WŁADYSŁAW KLEPACZ

BASIC mikroinformatyczny

Rozwój i postęp uwarunkowany jest efektywnością gromadzenia i przetwarzania informacji.

Wiosną br. w centralnym urzędzie państwowym odpowiadającym przed Historią za wychowanie i nauczanie przyszłości narodu — w Ministerstwie Oświaty i Wychowania — odbyło się robocze spotkanie na temat rewolucji mikrokomputerowej w szkolnictwie. Trochę armat już jest, kanonierzy (właściwie samoucy) — też są, choć trzeba ich liczyć na sztuki. Brakuje tylko amunicji, tzn. podręczników. Znajomy, który był obecny na tym spotkaniu, wyszedł przerażony perspektywą totalnej bezwładności aparatu biurokratycznego. W każdym razie konstruktywnym wynikiem owego spotkania było dojście do wniosku, że krocząc znanymi ścieżkami nauczanie mikroinformatyki w polskich szkołach można podjąć już za ... kilkanaście lat. Tak więc amunicją byłby podręcznik do nauczania mikroinformatyki historycznej.

Powyższe wprowadzenie oraz motto zaczerpnięte z recenzowanego poradnika szkoleniowego[1]) wydają się niezbędne do zrozumienia klimatu, w jakim powstało to dziełko, wydane brązowym drukiem na zielonym papierze. Jest to chyba jedyny, jak dotąd, tak kolorowy poradnik, ale ponoć to nie zamierzony efekt zakładu poligraficznego, w którym po prostu zabrakło innego papieru. Na tle kilkuset (!) podobnego typu pozycji, jakie mieliśmy możność podziwiać na tegorocznych Targach Książki w Warszawie, jest to jednak wydawnictwo ubogie, nieożywione graficznie i kolorystycznie, z wyraźnymi śladami pośpiechu wydawniczego. Ale jest. Co prawda tylko dla klientów zielonogórskiego oferenta mikrokomputerów ZX81 i SPECTRUM — ba, zaledwie po jednym egzemplarzu na każdy zakupiony zestaw. Ale to i tak dobrze, bo jeszcze nie tak dawno użytkownik dostawał w naszym kraju wyłącznie materiały niemieckie lub angielskie. Materiały te, spolszczone na siłę przez domorosłych entuzjastów, niechybnie miały wpływ — zgoła niekorzystny — na wyobrażenia twórców programów nauczania szkolnego o mikroinformatyce.

Wracając do poradnika, to miał on uzupełnić broszury opisujące podstawy programowania konkretnych typów mikrokomputerowych — i jako właśnie taki miał być adresowany specjalnie do osób dopiero wkraczających w świat maszyn cyfrowych. Nasuwa się tu przewrotna myśl, że należałoby sprawdzić czy idea była słuszna, poczynając od specjalistów Ministerstwa Oświaty i Wychowania. Omawiany poradnik, po prostu z braku innego, znajdzie się przecież w rękach nauczycieli forsujących mikroinformatykę w swym środowisku szkolnym. Czy są oni przygotowani do takiej pionierskiej pracy?

[1]) Adam Jeske, Roman Stryjski: Programowanie mikrokomputerów w języku BASIC. Poradnik szkoleniowy, s. 147, Zielona Góra 1985 (Nakładem zagranicznego przedsiębiorstwa produkcyjno-usługowego APINA)

Być może jestem pod silnym wrażeniem ostatnich dyskusji na temat wprowadzania mikrokomputerów do szkolnictwa. Profesjonaliści różnych branż widzieli już znacznie gorsze instrukcje obsługi technicznej i inną literaturę firmową, toteż z omawianego poradnika powinni być — przynajmniej przez pewien czas — zadowoleni. Ale jak tu nie być przewrażliwionym, gdy kuzyn z Holandii właśnie relacjonuje start mikroinformatyczny swej 6-letniej (!) córeczki: otrzymał ze szkoły spis wyprawki do klasy zerowej, gdzie na pierwszym miejscu figuruje mikrokomputer określonej marki (w tym przypadku akurat COMMODORE, aby wszystkie dzieci miały w klasie jednolity sprzęt). Powstaje następne przewrotne pytanie: czy jednolitość oświaty nie jest przypadkiem dogmatyczną fikcją? Czy w dobie rewolucyjnych przemian cywilizacji technicznej ideałem nie jest właśnie jednolite wyposażenie uczniów w danej klasie konkretnej szkoły? Bez konkursów programowych i całego dalszego balastu. A swoją drogą, ciekawym prywatnych opinii oświatowców o tym, jak powinno się pisać recenzje, bo przecież mam świadomość, że z takimi uwagami, mimo iż kiedyś pracowałem nawet w MOiW, wkraczam ostatecznie przecież na cudze podwórko.

Gdyby omawiany poradnik wykorzystać na zajęciach z młodzieżą, z pewnością pojawiłyby się różne uwagi, jak: wytykanie błędów maszynowych i miejscami może dziwnego stylu. W sumie jednak atrakcyjność tematu jest ogromnym kredytem i w rękach dobrego instruktora „Programowanie komputerów w języku BASIC" może wielu młodym osobom pomóc w przezwyciężeniu zacofania mikroinformatycznego. Pod warunkiem, że pod ręką będzie stosowny sprzęt. Mam jednak poważne obawy co do zajęć dla dorosłych.

Omawiany poradnik nie jest — z założenia zresztą — samouczkiem. Z punktu widzenia osoby, która jeszcze nie wkroczyła w świat mikrokomputerów, występują w układzie i treści rozdziałów pewne skoki pojęciowe. Szczegółowe uwagi byłyby tu może nie na miejscu, ale posłużmy się charakterystycznym przykładem. Otóż w rozdziale 4, traktującym o elementach języka BASIC, wyrażeniom logicznym poświęcono tylko $3^1/_2$ strony, podając wiele definicji i trochę przykładów. Wyraźnie brakuje tutaj komentarza, że z braku znaków $\neq$, $\leqslant$, $\geqslant$ w kodzie opisywanego mikrokomputera trzeba używać symboli zastępczych. Brakuje również odsyłacza do podrozdziału 8.2, traktującego o „sztuczkach programowych", gdzie wykorzystuje się wartości logiczne wyrażeń $A = A$ oraz $A < A$ jako nie wymagające osobnego specyfikowania wartości numeryczne 0 oraz 1. Nie tylko osoby ze świadectwem urodzenia wydanym przed utworzeniem PRL mogą tu natrafić na kłopoty. Z kolei w kontekście takich sztuczek nie powinno brakować komentarza o niezalecaniu mieszania typów danych z ogólnoinformatycznego punktu widzenia.

Wydaje mi się, że poradnik zyskałby znacznie na wartości, gdyby uzupełnić go tego typu krótkimi komentarzami oraz odsyłaczami. Uważam zaś za niezbędne dołączenie indeksu w następnym wydaniu. Pożądane też byłoby sporządzenie spisu treści w formie poglądowego schematu blokowego, a także dołączenie wzoru do korespondencji z wydawcą (choćby nt. zauważonych błędów czy różnych sugestii, które by takiej publikacji wyszły na korzyść).

Reasumując: na bezrybiu (bezmikrzu) i robocze wydawnictwo cieszy, byle stało się podstawą do konstruktywnych dyskusji i konstruktywnych decyzji. Wprowadzanie mikroinformatyki do szkół — na początek do klas eksperymentalnych — nie jest proste. Ale bezwzględnie konieczne.

ADAM B. EMPACHER

Systemy informatyczne zarządzania

W Państwowym Wydawnictwie Ekonomicznym ukazała się ostatnio książka „Systemy informatyczne zarządzania" *), będąca pracą zbiorową pod redakcją prof. Tadeusza Wierzbickiego. Jest to jedna z pierwszych prac zapoczątkowujących dorobek naukowy naszej najmłodszej uczelni — Uniwersytetu Pomorza Zachodniego.

Ogólnie tematykę książki można określić jako szeroki obszar problematyki z przecięcia zagadnień zarządzania gospodarką i zastosowań informatyki. Książka dotyczy ważnego i zarazem największego (prawie 70%) obszaru zastosowań informatyki. Mimo tej dominacji w krajowych publikacjach tematyka ta pojawia się niezbyt często.

Przez prawie 30 lat stosowania maszyn cyfrowych w Polsce, zaledwie kilka książek traktuje o informatyce w zarządzaniu. Przy okazji należy przypomnieć, że pionierską pozycją w tej dziedzinie była wydana przed 20 laty książka Władysława Klepacza — „Zastosowanie maszyn matematycznych do automatyzacji zarządzania". Pomijając zakres merytoryczny — zarówno w pierwszej, jak i tej najnowszej publikacji, podobnie określono krąg ich odbiorców, do którego zaliczono kadrę kierowniczą przedsiębiorstw i organizacji gospodarczych oraz projektantów, realizatorów oraz bezpośrednich użytkowników systemów informatycznych zarządzania (SIZ). Omawiana obecnie publikacja rozszerza krąg czytelników o studentów uczelni ekonomicznych, którzy szczególnie dokładnie powinni znać możliwości zastosowania metod i środków informatyki w sferze zarządzania.

Trudno jest zaprezentować tak obszerną pracę i to nie tylko ze względu na jej fizyczną objętość, ale przede wszystkim ze względu na bardzo szeroki zakres tematyczny. Dlatego, odchodząc nieco od tradycyjnej formy recenzji, zamierzam przedstawić w pierwszym rzędzie istotę tej publikacji oraz to, co stanowi jej novum.

Książka, która jest dziełem dziesięcioosobowego zespołu autorskiego, ma ten istotny, rzadko spotykany w publikacjach z zakresu informatyki lub zarządzania gospodarką walor, iż zamieszczono w niej zarówno rozważania teoretyczne (część pierwsza — „Systemy informatyczne i ich wykorzystanie w zarządzaniu"), jak i wywód praktyczny (część druga — „Problemy realizacji SIZ").

Pierwszą część (8 rozdziałów) można ogólnie podzielić na tematykę interesującą głównie przyszłych użytkowników oraz osoby pragnące poszerzyć wiedzę o zastosowaniu informatyki zarówno w przedsiębiorstwie, jak i w gospodarce narodowej, oraz na tematykę adresowaną przede wszystkim do osób tworzących systemy informatyczne zarządzania.

Obie kategorie odbiorców, choć bardziej przydatne będzie to dla informatyków, zainteresują mechanizmy funkcjonowania reformowanej gospodarki socjalistycznej, co przedstawiono jako swoiste kompendium wiedzy na temat reformy gospodarczej. Chciałbym tu zwrócić uwagę na syntetyczny, a zarazem bardzo klarowny rysunek (s. 24) ujmujący strukturę wyniku finansowego przedsiębiorstwa, a więc działań, które winny być podstawą oceny działalności gospodarczej.

Na tak zarysowanym podłożu procesów gospodarczych, po przypomnieniu znanego stwierdzenia, że istotną pomocą we wdrażaniu reformy może być zastosowanie informatyki, zarówno na szczeblu mikro- jak i makroekonomicznym, Autorzy przystępują do prezentacji szerokiej problematyki zastosowań, którą objęto m.in.: systemy informacyjne w gospodarce narodowej (można tu np. prześledzić — podany w formie graficznej — obieg informacji w zarządzaniu dyrektywnym oraz po wprowadzeniu reformy), kierunki i rodzaje zastosowań informatyki w sferze zarządzania, warunki efektywnego stosowania informatyki w przedsiębiorstwie, zastosowania makroekonomiczne (organizacje gospodarcze, regiony, systemy resortowe).

W części pierwszej Autorzy poświęcili kilka podrozdziałów na rozważania o istocie informacji, podając np. charakterystykę barier informacyjnych i zadania, jakie spełnia informacja w zarządzaniu, a zwłaszcza w procesie informacyjno-decyzyjnym. W tym zakresie ograniczono się jednak tylko do technicznego, instrukcyjnego potraktowania tych ważkich zagadnień. Brak jest próby przedstawienia ogólnego znaczenia informacji, a przede wszystkim jej społecznego charakteru.

Druga część książki poświęcona jest problemom realizacji systemów informatycznych. Cechą charakterystyczną tej części jest bardzo szerokie ujęcie problemów, wykraczające poza ramy samego projektowania systemów informatycznych. Szeroko podejmowany proces realizacji SIZ obejmuje bowiem badanie potrzeb, projektowanie wstępne i szczegółowe, uruchomianie systemu, wdrażanie pilotowe systemu, rozpowszechnianie systemu, eksploatację oraz kontrolę i doskonalenie systemu.

W ciekawym modelu całego procesu realizacji SIZ (s. 197) umieszczono centralnie „układ weryfikacji i oceny". Jego centralne położenie świadczy o ważności funkcji tego układu i związanych z nim działań, które decydują o efektywności wdrażania oraz jakości rozwiązania systemu. Układ ten, a zwłaszcza oddzielne traktowanie poszczególnych mechanizmów oceny (ocena kolejno uzyskiwanych rezultatów) — można uznać jako kolejne novum pracy.

Uwagę zwraca przejrzyście i szeroko ujęta identyfikacja potrzeb komputeryzacji. A więc faza powstawania systemu, gdzie najintensywniej z zespołem projektowym winni współpracować bezpośredni użytkownicy. Przydatna dla obu stron jest tu ciekawa propozycja metody opisu zadań (rozdział dziesiąty), w której po kolejnych krokach dochodzi się — w sposób graficzny (s. 225 i 226) — do identyfikacji zadań. Otrzymuje się dwudzielny zbiór zadań, które będą rozwiązywane w ramach przyszłego systemu. Bardzo istotne jest, wyraźne wyspecyfikowanie również zadań, które znajdą się poza obszarem działania systemu informatycznego i będą realizowane innymi technikami.

Z dalszych sześciu rozdziałów chciałbym jeszcze zwrócić uwagę na problemy organizacyjne i integracyjne SIZ. Autorzy podają zarówno ogólne zasady organizacyjnego funkcjonowania takiego systemu, jak też zalecenia szczegółowe, do których m.in. zaliczono uporządkowanie procesów decyzyjnych (każdy użytkownik wie, że nie jest to czynność łatwa), zapewnienie sprawnego przepływu informacji, jednoznaczne określenie źródeł informacji i jej wiarygodność, określenie kompetencji dostępu do zbiorów informacyjnych oraz, co jest szczególnie istotne, zapewnienie aktywnego uczestnictwa i poparcia naczelnego kierownictwa komputeryzowanego przedsiębiorstwa.

Szkoda, że w tym kontekście nie przedstawiono problematyki barier psychologicznych środowiska, w którym powstaje system. Doświadczenie uczy, że nieuwzględnianie tego czynnika, a raczej nieprzeciwdziałanie takim barierom, zazwyczaj bardzo ujemnie wpływa na efektywność systemu.

Problemy integracji (ostatni, szesnasty rozdział) dotyczą przede wszystkim funkcji użytkowych („biała plama zastosowań informatyki" — s. 340), danych oraz informacji technologicznej i technicznej (operacje i urządzenia). Ta ciekawie podana problematyka SIZ podsumowuje tę godną polecenia książkę.

Szkoda więc, że tak przydatna, szczególnie dla praktyki książka, została wydana tylko w 3800 egzemplarzy i dlatego sądzę, że jej pobyt na półkach księgarskich będzie na pewno bardzo krótki.

ANDRZEJ SOKOŁOWSKI

*) Systemy informatyczne zarządzania. PWE, Warszawa 1985, s. 350.

A jednak po polsku!

Ostatnio byłem świadkiem gorącej dyskusji między przedstawicielami firm komputerowych z USA i Europy na temat pisowni jednego ze słów kluczowych języka programowania. Sprawa dotyczyła słowa COLOR po amerykańsku i COLOUR po angielsku. Okazuje się więc, że wbrew propagowanym u nas opiniom — istotne jest dla Amerykanina czy Anglika, aby tekst programu był zgodny z mową potoczną. Podobnie Francuzi stwierdzają, że będą w całości tłumaczyć normy języków programowania na język francuski — łącznie ze słowami kluczowymi.

W Polsce zagadnieniem tym zajęło się ostatnio Polskie Towarzystwo Informatyczne. W stanowisku Prezydium PTI (Biuletyn PTI nr 7—8, 1985, Informatyka nr 11'1985) w sprawie wprowadzenia informatyki do szkół znalazło się też stwierdzenie, że polskie komputery powinny być wyposażone w polski alfabet — na klawiaturze, ekranie, drukarkach oraz w oprogramowaniu. Na seminarium zespołu PTI (1985.10.09, Warszawa) podczas dyskusji tego problemu prezes PTI, prof. W. M. Turski stwierdził w swoim wystąpieniu, że jesteśmy narodem, który przez wiele lat jedyne co miał ze swej narodowości, to język — musimy więc o tym pamiętać. Ilustracją tej wypowiedzi było zdanie: ZADANIE KATA NA LACE.

Niestety, poglądy wielu informatyków przypominają mi znane z nauki o historii literatury polskiej poglądy otoczenia Imć Miłokaja Reja z Nagłowic. Musiał on wreszcie zawołać, że Polacy nie gęsi też swój język mają. Otoczenie zaś wolałb gadać i pisać po łacinie, bo wtedy pospólstwo z trudnością mogło dostąpić zrozumienia, o czym mowa. Ilustracją skutków pobłażliwości i lekceważenia języka może być mój ulubiony przykład z systemu DOS RW dla SM4. System ten wyposażony w terminal z alfabetem angielskim i rosyjskim zgłasza gotowość słowem КОММАНД lub też wypisuje słowo EPPOP, albo nie reaguje na popularną nazwę programu PIP (wpisaną jako rir).

Zanim jednak podsumujemy dotychczasowe dyskusje na temat spolszczenia języków programowania — ustalmy, że dotyczy to tylko systemów i języków bezpośrednio użytkowanych przez ogół nieprofesjonalistów. Informatykom pozostawiamy konieczność nauczenia się języka ojczystego komputerów — języka angielskiego. Możemy się jedynie cieszyć, że to nie Arabowie albo Japończycy skonstruowali pierwszy komputer. (por. W. Iszkowski, Nasz drugi język, Informatyka 9'1984).

Obecny stan wyposażenia komputerów charakteryzuje się całkowitym brakiem przystosowania ich do języka polskiego. Brak klawiatury z polskimi literami: ą,ć,ę,ł,ń,ó,ś,ź,ż oraz ekranów z rastami tych znaków. Także wszystkie drukarki, mogące już drukować prawie wszystko — nie są wyposażone w polski zestaw znaków. Również standardowe oprogramowanie komputerów nie uwzględnia alfabetu polskiego. Pomijam tutaj próby częściowego definiowania znaków jako dodatkowych symboli specjalnych — gdyż dość trudno jest je zainstalować nawet w edytorze tekstów takim, jak Wordstar. Konstruowanie znaków przez nakładanie dwóch znaków na siebie uważam za pomysł chybiony.

Podstawowym warunkiem rozwoju informatyki w Polsce jest wyposażenie komputerów w zestaw polskich znaków. Do rozstrzygnięcia pozostaje umieszczenie tych znaków w zestawie kodu ASCII oraz równoważnego mu kodu ISO. Należy również rozważyć najlepszy układ klawiszy na klawiaturze. Dotychczasowy system nie jest ergonomiczny. Oczywiście, również drukarki powinny być wyposażone w taki zestaw polskich znaków. Co więcej wszyscy krajowi producenci powinni stosować ten sam układ znaków.

Bezdyskusyjne jest dopuszczenie akceptacji polskich liter w komentarzach języka, w identyfikatorach obiektów, w napisach oraz w komunikatach o błędach. Do rozstrzygnięcia pozostaje problem porównywania leksykograficznego tekstów. Rozwiązanie to jest bowiem zależne od ułożenia znaków w zestawie kodów. Zaakceptowanie polskich tekstów przez translatory języków programowania nie powinno nastręczać kłopotów — oczywiście tylko wtedy, gdy nie są to produkty kopiowane w postaci binarnej.

Takie wykorzystanie języka polskiego pozwoli już na usunięcie wielu terminów żargonowych. Młodzi użytkownicy komputerów, co prawda, szybko przyswajają sobie nie znane słowa — ale są to słowa obce, nie znane, i dlatego często są źle odmieniane oraz wymawiane. Nie chcę już tutaj przytaczać przykładów, które ostatnio słyszałem w radiu czy telewizji.

Najbardziej kontrowersyjnym problemem jest wykorzystanie polskich słów kluczowych języka. Inaczej mówiąc, dyskutowane jest pełne przetłumaczenie języka. Dla wielu pomysł ten jest obrazoburczy i przytaczają oni wtedy następujące argumenty:
— język polski nie nadaje się do konstruowania prostych, krótkich i poprawnych gramatycznie zdań
— stosując język polski odcinamy możliwość akceptacji oprogramowania zachodniego oraz możliwość sprzedaży oprogramowania wytwarzanego u nas (sic!)
— uczenie programowania po polsku utrudnia późniejsze nauczenie się programowania w innych językach programowania, których słowa kluczowe pochodzą z angielskiego.

Obecnie już czas, aby wykazać całkowitą bezpodstawność tych opinii. Osobiście uważam, że im szybciej informatyka nie będzie się kojarzyć z językami obcymi, tym łatwiej zostanie zaakceptowana. Na marginesie, nie powinna ona też kojarzyć się z żółwiem rysującym na ekranie kwadrat, co usilnie ostatnio próbuje przekazać telewizja. Uważam, że kolejne nowoczesne translatory języków z równym powodzeniem będą akceptowały różne narodowe alfabety i zestawy słów kluczowych wraz z ich odmianą przez przypadki i osoby. Oczywiście wersję polskojęzyczną trzeba zaimplementować w Polsce!

Kolejny argument obalił prof. W. M. Turski podczas dyskusji na wspomnianym spotkaniu, zadając pytanie: Ile to programów napisanych przez Jasia Kowalskiego (przepraszając wszystkich Janów Kowalskich) ma szanse sprzedaży w innych krajach. I jeszcze o ostatnim argumencie. Ile jest w nim prawdy, mogą powiedzieć wszyscy ci, którzy po nauce Fortranu, Algolu i Pascala próbowali z trudnością poznać Simulę, Lisp czy Prolog.

W obecnej chwili mamy już spolszczone wersje języków Basic i Logo, działające na mikrokomputerach Spectrum. Obie mają jeszcze nie najlepiej dobrane niektóre polskie słowa kluczowe, ale każda droga zaczyna się od wybojów. Efektem spotkania zespołu PTI było utworzenie grupy roboczej, której zadaniem ma być opracowanie i implementacja polskiej (a nie spolszczonej) wersji języka Logo. PTI wzięło na siebie trud częściowego sfinansowania tych prac. Od siebie proponuję, aby zamiast żółwia użyć nieco bardziej żwawego i swojskiego zwierzaka, na przykład chomika lub zająca, istnieje też propozycja, aby był to kurczak.

WACŁAW ISZKOWSKI

■ Od 1 marca br. obowiązuje w NRD nowa norma państwowa TGL 32521/01-06 „Organizacja produkcji. Formularze" („Fertigungsorganisation, Vordrucke"). Norma jest przeznaczona dla zakładów przemysłowych o procesach wytwórczych polegających na produkcji lub montażu części. Stanowi ona przepracowanie redakcyjne oraz uzupełnienie zawartości dwu dotychczasowych norm: TGL 31490/01-03 oraz TGL 34520/01-02. W nowej normie zawarto m. in. wzory wydruków komputerowych związanych z organizacją produkcji, takich jak: przewodniki technologiczne, listy części, karty zleceń, karty zarobkowe, kwity materiałowe itp. Instytucją odpowiedzialną za treść normy jest Centralne Biuro Dokumentacji Źródłowej przy Głównym Urzędzie Statystycznym NRD (Staatliche Verwaltung für Statistik, Zentralstelle für Primärdokumentation, 1026 Berlin, Hans-Beimler. Str. 70—72). Istotną pomocą we właściwym stosowaniu w przedsiębiorstwach znormalizowanych wzorów dokumentów źródłowych jest opublikowana ostatnio przez wydawnictwo Verlag Technik broszura „Rationelle Fertigungsorganisation". (WK)

*

■ O popularności mikrokomputerów MACINTOSH (oraz olbrzymim już kręgu użytkowników tego sprzętu) świadczy fakt, że w lipcu br. zaczął ukazywać się w Wielkiej Brytanii nowy miesięcznik pod nazwą MacVser. Czasopismo to, reklamowane przez wydawcę jako jedyny tego rodzaju periodyk specjalistyczny, zawierać będzie publikacje znanych z miesięcznika PCW (Personal Computer World) publicystów, takich jak Dick Pountain, David Tebbutt oraz Guy Kewney. W informacjach, jakie do nas dotąd dotarły, brak jest niestety ceny nowego czasopisma. (WK)

*

■ Ministrowie państw członkowskich jednej z agend ONZ Organizacji Współpracy i Rozwoju Ekonomicznego OECD (Organization for Economic Cooperation and Development) — na posiedzeniu w dniu 11 kwietnia 1985 r. wydali oświadczenie w sprawie przepływu danych przez granice państwowe. Oświadczenie to stanowi pierwszą oficjalną międzynarodową inicjatywę mającą na celu rozwiązanie problemów gospodarczych, jakie pojawiły się w wyniku rewolucji informacyjnej. Oświadczenie dotyczy również zagadnień politycznych związanych z międzynarodowym przepływem danych dotyczących handlu, wewnętrznych informacji przedsiębiorstw, zautomatyzowanych usług informacyjnych oraz wymiany danych naukowo-technicznych. Przepływ tych danych w coraz większym stopniu wpływa zarówno na gospodarkę poszczególnych krajów, jak i międzynarodową wymianę towarów i usług. Wydanie wspomnianego dokumentu świadczy o tym, że rządy krajów członkowskich OECD akceptują pogląd o konieczności popierania dostępu do danych oraz informacji i popierają wspólne działania zmierzające do rozwiązania istniejących problemów, a także zminimalizowania przeszkód w międzynarodowym przepływie danych. W tym celu uzgodniono podjęcie intensywnych prac badawczych w tej dziedzinie. (WK)

*

■ Działania rządu RFN w dziedzinie mikroelektroniki mające na celu wzmocnienie pozycji tego kraju w stosunku do głównych konkurentów na rynku światowym, jakimi są nadal USA i Japonia, są ostatnio wspomagane również przez przemysł prywatny. Tak np. firma SIEMENS przystąpiła w br. do finansowania zainicjowanego w 1983 r. przez rząd projektu E.I.S. (Entwurf Integrierter Schaltkreise — Projekt Układy Scalone). Projekt ten był dotąd finansowany wyłącznie przez federalne Ministerstwo Badań i Technologii i koordynowany przez Niemieckie Towarzystwo Informatyczne GMD (Gesellschaft für Mathematik und Datenverarbeitung). Finansowanie obejmowało 30 katedr i instytutów z 14 wyższych uczelni i dotyczyło głównie kosztów osobowych oraz wyposażenia w aparaturę naukową. Udział firmy SIEMENS, określony kwotą 20 mln marek (ok. 7 mln dol. USA) umożliwi włączenie do prac badawczych 12 dalszych placówek badawczych wyższych uczelni, nie tylko przez finansowanie tam kosztów osobowych, ale głównie przez wyposażenie zainteresowanych uczelni w komputerowe systemy projektowania VENUS, które wielokrotnie zmniejszają pracochłonność i skracają czas opracowania koncepcji i symulowania działania układów scalonych oraz przygotowania danych niezbędnych do podjęcia produkcji przemysłowej. Firma zapewnia również pełną konserwację systemu VENUS, przeszkolenie jego użytkowników oraz produkowanie układów zaprojektowanych przez naukowców lub studentów.

Oznacza to uzyskanie dla celów badawczych i dydaktycznych najnowocześniejszej aparatury oraz udostępnienie warunków szybkich wdrożeń przemysłowych. Należy podkreślić, że w takich rozmiarach przemysł prywatny wystąpił po raz pierwszy jako sponsor prac badawczo-rozwojowych w dziedzinie mikroelektroniki. Podobnie jak w przypadku dotacji rządowych, firma SIEMENS zawarła odpowiednie porozumienia z GMD, jako instytucją koordynującą cały cykl prac badawczo-wdrożeniowych. Porozumienie to formułuje również współodpowiedzialność rządu i przemysłu w tym, aby metody oraz narzędzia projektowania i testowania układów o dużym (LSI) i bardzo dużym stopniu scalenia (VLSI) zostały opanowane wyłącznie przez narodowe zespoły specjalistów. Stanowić to będzie istotny krok w kierunku zapewnienia krajowi szybkiego osiągnięcia najwyższego poziomu technologii i tym samym zabezpieczenia się przed amerykańsko-japońskim zagrożeniem. (WK)

*

■ Z okazji sześćdziesięciolecia proklamowania Mongolskiej Republiki Ludowej w jej stolicy Ułan-Bator dokonano otwarcia krajowego centrum informacji naukowo-technicznej. Otwarcie to połączono z uruchomieniem transmisji danych z Moskwy poprzez łączność satelitarną. Fakt ten stanowi dalszy krok w kierunku realizacji uchwały RWPG w sprawie stworzenia międzypaństwowej sieci informacyjnej krajów członkowskich.

Wspomniane centrum zapewni mongolskim specjalistom natychmiastowy bezpośredni dostęp do radzieckich baz danych z informacjami dotyczącymi prac naukowych, publikacji książkowych i zawartości czasopism naukowo-technicznych. Podane eksperymenty w zakresie zdalnego dostępu do informacji przeprowadzono z Kubą oraz Wietnamem. Wyposażenie sprzętowe mongolskich terminali pochodzi z ZSRR, NRD i CSRS. W ciągu br. przewiduje się uruchomienie połączeń informacyjnych Ułan-Bator z dalszymi europejskimi krajami RWPG. (WK)

*

■ W pięć miesięcy po wmurowaniu kamienia węgielnego nowego zakładu produkcji układów scalonych firmy SIEMENS w Regensburgu (RFN), ukończono tam w stanie surowym dwie hale produkcyjne o wymiarach 40 × 108 m. Częścią centralną tych obiektów będzie tzw. wydział produkcji czystej o powierzchni ok. 3800 m². Obowiązywać tu będzie 10 klasa czystości powietrza (ponżej 10 drobinek pyłu o średnicy 0,5 μm w stopie³, tzn. ok. 27 l powietrza). Zapewnienie tych warunków produkcji wymagać będzie przefiltrowania w ciągu godziny ok. 4 mln m³ powietrza. W nowym zakładzie w 1987 r. ruszy produkcja seryjna nowoczesnych układów scalonych VLSI, a zwłaszcza pamięci dynamicznej o pojemności 1 M bitów. Układ ten o powierzchni ok. 50 mm² zawierać będzie ok. 2 mln elementów. Warto przypomnieć, że w br. SIEMENS uruchomił zakład o podobnym charakterze w Villach (Austria), który produkuje już pamięci o pojemności 256 K bitów. (WK)

*

■ W laboratoriach firmy SIEMENS w Monachium skonstruowano nowy czujnik optyczny do bardzo dokładnego pomiaru odległości pomiędzy głowicą zapisu-odczytu a powierzchnią dysku. Wymagała tego opracowania ostatnio przez tę firmę pamięć na dysku kompaktowym MEGAFILE, gdzie nominalny odstęp głowicy od powierzchni dysku wynosi zaledwie 0,2 μm. Wspomniany czujnik mierzy w zakresie do 0,8 μm z dokładnością ±0,05 μm.

Zastosowano w nim lasery w obszarze podczerwieni oraz fal widzialnych. Powstające w warstwach granicznych drgania rezonansowe układu głowica—dysk mogą być rejestrowane do częstotliwości 20 kHz. Do czynności pomiaru wysokości „lotu" głowicy dysk magnetyczny jest zastępowany tarczą kwarcową, przepuszczalną dla promieni podczerwonych. Jak wiadomo, głowice zapisu-odczytu w pamięciach dyskowych „pływają" na bardzo cienkiej poduszce powietrznej, przy czym właściwe ustawienie wysokości „pływania" stanowi decydujące kryterium dla uzyskania maksymalnej gęstości zapisu. (WK)

Pyle I.: Pakiet do specyfikacji programów w Adzie

INFORMATYKA 1986, nr 1, s. 1

Charakterystyka metody oraz pakietu służącego do specyfikacji programów w języku Ada. Omówiono konstrukcję pakietu oraz ilustrowany przykładem sposób jego działania.

Пыле И.: Пакет для спецификации программ на языке Ada

INFORMATYKA 1986, № 1, стр. 1

Характеристика метода и пакета предназначенного для спецификации программ на языке Ada. Обсуждено структуру пакета, а его действие проиллюстрировано примером.

Zakręcki J.: Abstrakcje w programowaniu (1)

INFORMATYKA 1986, nr 1, s. 4

Definicja abstrakcji w programowaniu oraz omówienie ewolucji narzędzi umożliwiających jej stosowanie. Podano przykłady użycia abstrakcji w językach Fortran i Ada.

Закрецки И.: Абстракты в программированию (1)

INFORMATYKA 1986, № 1, стр. 2

Определение абстрактов в программированию и обсуждение эволюции инструментов, благодаря которым возможно их применение. Приведены примеры использования абстрактов в языках Фортран и Ada.

Abramowicz W.: Modularny system wyszukiwawczy

INFORMATYKA 1986, nr 1, s. 6

Charakterystyka systemu wyszukiwawczego MIRS opracowanego na Politechnice Federalnej w Zurychu. Omówiono konfigurację użytego sprzętu, konstrukcję i funkcje systemu oraz sposób przygotowania danych i wyszukiwania informacji.

Абрамович В.: Модулярная поисковая система

INFORMATYKA 1986, № 1, стр. 6

Характеристика поисковой системы разработанной политехническим институтом в Цурихе.
Обсуждено структуру и функции системы, конфигурацию оборудования, способы подготовки и поиска данных.

Bielecki J.: Błędy implementacji w kompilatorze języka C firmy Supersoft

INFORMATYKA 1986, nr 1, s. 10

Omówienie niektórych błędów w kompilatorze języka C firmy Supersoft, występujących podczas kompilowania nietypowych konstrukcji programowych.

Белецки И.: Ошибки реализации в компьляторе языка Ц фирмы Supersoft

INFORMATYKA 1986, № 1, стр. 10

Обсуждено некоторые ошибки компилятора языка Ц разработанного фирмой Supersoft выступающие во время компиляции некоторых программ.

Sikora M.: Modułowy system mikroprocesorowy kompatybilny z systemem Mikroster

INFORMATYKA 1986, nr 1, s. 12

Charakterystyka konstrukcji i sposobu działania modułowego systemu mikroprocesorowego do kontroli sterowania i rejestracji danych w procesach technologicznych oraz badaniach naukowych. Podano przykłady zastosowań systemu.

Сикора М.: Модульная микропроцессорная система совместима с системой Mikroster

INFORMATYKA 1986, № 1, стр. 12

Характеристика конструкции и работы микропроцессорной системы осуществляющей функции управления технологическими процессами и сбораданных по научно-исследовательским работам. Приведено пример использования системы.

Nosowski W., Szomański B.: Pakiet symulacyjny DESA

INFORMATYKA 1985, nr 11—12, s. 14

Charakterystyka pakietu programów DESA, opracowanego w Instytucie Organizacji i Zarządzania Politechniki Warszawskiej z przeznaczeniem do komputerowego wspomagania modelowania systemów produkcyjnych. Omówiono budowę i możliwości zastosowań pakietu, a także zakres jego dotychczasowej eksploatacji.

Носовски В., Шомански Б.: Симуляционный пакет

INFORMATYKA 1985, № 11—12, стр. 14

Характеристика программ пакета DESA разработанного Институтом Организации и Управления Варшавского Политехнического Института предназначенного для вспомагательного моделирования производственных процессов. Обсуждено структуру и возможности использования пакета и накопленный опыт его применения.

Tuziemski R.: Język Ratfor dla minikomputera Mera 400

INFORMATYKA 1986, nr 1, s. 16

Charakterystyka najważniejszych konstrukcji języka Ratfor zaimplementowanych na minikomputerze Mera 400. Podano przykład programu, uzasadniający przewagę Ratforu nad Fortranem.

Туземски Р.: Язык Ратфор для миникомпьютера Мера 400

INFORMATYKA 1986, № 1, стр. 16

Характеристика главных элементов язык аРатфор реализованного для миникомпьютера Мера 400. Приведено пример программы иллюстрирующий превосходство языка Ратфор по сравнении с языком Фортран.

Sukiennik J.: Rozrachunek gospodarczy zakładowego ośrodka informatyki

INFORMATYKA 1985, nr 11—12, s. 18

Przykład rozwiązań oraz efekty organizacyjno-ekonomiczne wprowadzenia wewnętrznego rozrachunku gospodarczego w zakładowym ośrodku informatyki.

Сукенник Е.: Хозрасчет в вычислительном центре предприятия

INFORMATYKA 1985, № 11—12, стр. 18

Представлено пример решения и организационно-экономические эффекты применения внутрицехвого расчета в вычислительном центре предприятия.

Pyle I.: Ein Paket für Spezifikation von Ada-Programmen

INFORMATYKA 1986, Nr. 1, s. 1

Eine Charakteristik der Methode und des Pakets für Spezifikation von Ada-Programmen. Es wurde die Struktur des Pakets und seine Arbeitsweise, illustriert mit einem Beispiel, besprochen.

Pyle I.: A package for specification of Ada programs

INFORMATYKA 1986, No. 1, p. 1

Characteristics of the method and package for specification of Ada programs. Structure of the package and its operation, illustrated with an example, are discussed.

Zakręcki J.: Abstraktionen in Programmierung (1)

INFORMATYKA 1986, Nr. 1, s. 4

Eine Definition der Abstraktion in Programmierung, sowie eine Besprechung der Hilfsmittel, die ihre Anwendung ermöglichen. Es wurden Beispiele von Abstraktionanwendung in Fortran und Ada angegeben.

Zakręcki J.: Abstractions in programming (1)

INFORMATYKA 1986, No. 1, p. 4

Definition of abstraction in programming and discussion of tools, which make its application possible. Some examples of abstraction use in Fortran and Ada languages are presented.

Abramowicz W.: Ein modulares Recherchesystem

INFORMATYKA 1986, Nr. 1, s. 6

Eine Charakteristik des MIRS-Recherchesystems, das an der Eidgenössischen Technischen Hochschule in Zürich erarbeitet wurde. Es wurden Hardwarekonfiguration, Struktur und Funktionen des Systems, sowie Methoden von Datenvorbereitung und Informationsrecherche besprochen.

Abramowicz W.: A modular information retrieval system

INFORMATYKA 1986, No. 1, p. 6

Characteristics of the MIRS information retrieval system, which is elaborated on the Federal Technical University in Zürich. Configuration of the hardware, structure and functions of the system, as well as methods of data preparation and information retrieval are discussed.

Bielecki J.: Implementationsfehler im C-Sprache Komilierer von der Firma Supersoft

INFORMATYKA 1986, Nr. 1, s. 10

Eine Besprechung von manchen, in C-Sprache Kompilierer von der Firma Supersoft eintretenden Fehlern, die während der Kompilation von nicht typischen Programmstrukturen erscheinen.

Bielecki J.: Implementation errors in C language Supersoft's compiler

INFORMATYKA 1986, No. 1, p. 10

Discussion of some errors in C language compiler, designed by Supersoft company, which occure when compiling non-typical program structures.

Sikora M.: Ein modulares und Mikroster-kompatibles Mikroprozessorsystem

INFORMATYKA 1986, Nr. 1, s. 12

Eine Charakteristik von Struktur und Arbeitsweise eines modularen Mikroprozessorsystems für Datenkontrolle, -Steuerung und -Registrierung in industriellen Prozessen und wissenschaftlicher Forschung. Es wurden Anwendungsbeispiele des Systems angegeben.

Sikora M.: A modular Mikroster compatible microprocessor system

INFORMATYKA 1986, No. 1, p. 12

Characteristics of the structure and operation method of a modular microprocessor system for data checking, controlling and registrating in industrial processes and scientific research. The system's application examples are presented.

Nosowski W., Szomański B.: DESA — a program package for simulation

INFORMATYKA 1985, No. 11—12, p. 14

Characteristics of the DESA program package, elaborated in the Organization and Administration Institute of the Warsaw Technical University for computer assisted modelling of manufacturig processes. Structure and application possibilities of the package, as well as the scope of its using are presented.

Nosowski W., Szomański B.: DESA — ein Programmpaket für Simulieraufgaben

INFORMATYKA 1985, Nr. 11—12, S. 14

Eine Charakteristik von DESA-Programmpaket, der im Institut für Organisation und Verwaltung der Warschauer Technischen Universität für rechnergeschtützte Modellierung von Produktionssystemen erarbeitet wurde. Es werden die Struktur und Anwendungsmöglichkeiten des Pakets, sowie seine bisherige Ausnutzung besprochen.

Tuziemski R.: Ratfor-Sprache für Mera 400-Kleinrechner

INFORMATYKA 1986, Nr. 1, s. 16

Eine Charakteristik der wichtigsten Ratfor-Strukturen, die auf Mera 400-Kleinrechner implementiert wurden. Es wurde Beispiel eines Programmes, das Ratfors Überlegenheit über Fortran beweisst, besprochen.

Tuziemski R.: Ratfor language for Mera 400 minicomputer

INFORMATYKA 1986, No. 1, p. 16

Characteristics of the most important Ratfor language constructions, implemented on Mera 400 minicomputer. The example of a program, which confirms Ratfor superiority over Fortran is presented.

Sukiennik J.: Economic calculation of a factory computing center

INFORMATYKA 1985, No. 11—12, p. 18

An example of solutions and organizational-economic results of the introduction of internal economic calculations in a factory computing center.

Sukiennik J.: Wirtschaftsabrechnung in einem Betriebsrechenzentrum

INFORMATYKA 1985, Nr. 11—12, S. 18

Ein Beispiel von Lösungen und organisatorisch-ökonomische Effekte der inneren Wirtschaftsabrechnung in einem Betriebsrechenzentrum.

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