maszyny matematyczne

zastosowania

w gospodarce

technice

i nauce

9
1970

SPIS TREŚCI

WYDAWNICTWA CZASOPISM TECHNICZNYCH NOT
Warszawa
Czackiego 3/5

Redakcja: Warszawa, ul. Emilii Plater 20 m. 15, tel. 21-13-91. Zastępca redaktora naczelnego tel. 28-37-29

Zakład Kolportażu WCT NOT, Warszawa, ul. Mazowiecka 12

Zakł. Graf. „Tamka". Z. 2. Zam. 552 Papier powlekany V kl. 80 g. Obj. 3 ark. druk. Nakład 2800. K-43

Cena egzemplarza zł 8.— **INDEKS 36707** **Prenumerata roczna zł 96.—**

maszyny matematyczne

zastosowania w gospodarce, technice i nauce

Nr 9

MIESIĘCZNIK

1970

ROK VI

Wrzesień

Organ Pełnomocnika Rządu do Spraw Elektronicznej Techniki Obliczeniowej i Polskiego Komitetu Automatycznego Przetwarzania Informacji Naczelnej Organizacji Technicznej

JANINA STĘPOWSKA
Instytut Maszyn Matematycznych
Warszawa

681.322.06

Janina Stępowska ur. w 1945 r. ukończyła studia matematyczne na Wydziale Matematyki i Fizyki Uniwersytetu Warszawskiego w 1968 r. W tym samym roku podjęła pracę w Instytucie Maszyn Matematycznych w Warszawie, w Zakładzie Metod Programowania, gdzie pracuje obecnie. Zajmowała się głównie badaniem składni ALGOLu 68 i językami do przetwarzania struktur listowych.

ALGOL 68 – próba prezentacji

W artykule dokonano próby przedstawienia najistotniejszych cech ALGOLu 68. Omówiono podstawowe typy obiektów, deklaracje wielkości stałych i zmiennych, metody tworzenia nowych typów, operacje, zdania i procedury. Przedstawiono także zalety ALGOLu 68, jak uniwersalność i elastyczność, czyniące go przydatnym do wielu zastosowań.

Wkrótce po ukazaniu się Raportu o ALGOLu 68 [1], w czasopiśmie ALGOL-Bulletin ukazał się artykuł C. H. Lindsey'a [3] będący pierwszym nieformalnym opisem tego języka. O tym, jak bardzo taki opis był potrzebny, świadczy tytuł artykułu: „ALGOL 68 z mniejszą ilością łez". Później w ślad za nowymi wersjami Raportu powstawały nowe, zmodyfikowane wersje artykułu Lindsey'a.

Niniejszy artykuł jest próbą przedstawienia polskiemu czytelnikowi najistotniejszych cech ALGOLu 68. Oparty jest on na ostatniej pracy Lindsey'a wydanej w roku 1969.

Główną cechą ALGOLu 68 jest jego rozszerzalność. Języki takie, jak ALGOL 60, FORTRAN, PL/I itd. mają ograniczoną i ściśle określoną gramatyką języka liczbę typów i operacji na tych typach; do określenia nowych działań programista ma do dyspozycji jedynie procedury. Natomiast ALGOL 68 daje nieograniczone wręcz możliwości w tej dziedzinie. Oprócz procedur w dawnym rozumieniu język dopuszcza deklarowanie zarówno nowych struktur danych, jak i operacji na tych strukturach. To właśnie mamy na myśli mówiąc o rozszerzalności ALGOLu 68. ALGOL 68 jest językiem uniwersalnym, który w równym stopniu może służyć obliczeniom numerycznym, przetwarzaniu danych i list, jak też i innym działaniom.

Nowa jest również metoda opisu ALGOLu 68, zapewniająca opis zarówno składni, jak i semantyki. Na marginesie trzeba stwierdzić, że metoda ta ze względu na jej złożoność budzi wiele kontrowersji. Autorzy języka — w celu uproszczenia pisania i odczytywania programów — wprowadzili pewną liczbę nowych symboli podstawowych. W tym samym celu nie ustalili oni konkretnej reprezentacji ALGOLu 68. Każda reprezentacja, która przyporządkowuje wzajemnie jednoznacznie każdemu symbolowi języka dowolny znak jest dopuszczalna. Dla napisania Raportu autorzy ustalili kilka równoważnych reprezentacji. Na przykład symbole **begin** i **end** można w każdym kontekście zastąpić odpowiednio symbolami nawiasów (i); podobnie symbole **if** i **fi** (symbol zamykający zdanie warunkowe **if**) można odpowiednio zastąpić przez symbole (i), natomiast **then** i **else** symbolem |.

Podstawowe typy obiektów

W ALGOLu 68 rozróżniamy 5 podstawowych typów obiektów. Są to: **real, int, bool, char, format.**

Dalej omówimy cztery pierwsze typy. Wartości typu **format** używane są tylko w problemach związanych z wejściem—wyjściem i w dalszym ciągu artykułu nie będziemy się nimi zajmować.

Zmienne typu **int** i **real** mogą przyjmować w zasadzie nieskończenie wiele wartości odpowiednio naturalnych lub rzeczywistych. W praktyce liczba tych wartości jest ograniczona konkretną realizacją; za pośrednictwem wyróżnionych zmiennych maxint, maxreal (odpowiednio typu **int** i **real**) podana jest informacja o zbiorze tych wartości. Zmienne te reprezentują największe liczby danego typu w każdej konkretnej realizacji. Forma liczb typu **int** i **real** jest podobna do formy stosowanej w ALGOLu 60.

Zmienne typu **bool** mogą przyjmować tylko dwie wartości: **true** i **false.** Typ **char** jest przewidziany dla danych tekstowych. Rodzaj i liczba zmiennych typu **char** zależy od zestawu znaków dostępnych na danej maszynie. Każdej zmiennej typu **char** odpowiada w sposób wzajemnie jednoznaczny liczba typu **int.** Odpowiedniość ta może być różna dla różnych reali-

zacji. Liczbę naturalną odpowiadającą danemu znakowi otrzymujemy za pomocą jednoargumentowego operatora **abs**, zastosowanego do tego znaku. Na przykład dla pewnej reprezentacji może być **abs** a = 33.

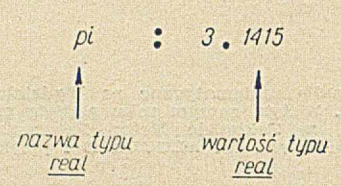
Stałe i zmienne

ALGOL 68, podobnie jak ALGOL 60, dopuszcza dwa rodzaje wielkości: stałe i zmienne.

Inaczej natomiast wyglądają deklaracje tych wielkości w obu językach. W ALGOLu 60 wystąpienie stałej w dowolnym miejscu programu jest jej deklaracją; natomiast ALGOL 68 pozwala na deklarowanie stałych, podobnie jak zmiennych, tzn. nadawanie im nazw, zarezerwowanych tylko dla tych stałych, np.:

real *pi* = 3.1415;

Identyfikator *pi* nie reprezentuje zmiennej; reprezentuje wielkość typu **real**, której stałą wartością jest liczba 3.1415.

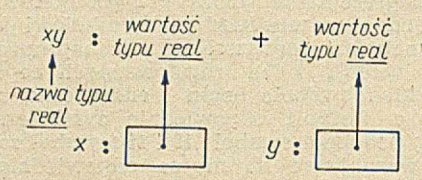


W trakcie wykonywania programu, wartości *pi* nie można zmienić za pomocą instrukcji podstawienia, ponieważ prawą stroną instrukcji podstawienia może być tylko zmienna.

Innym przykładem deklaracji stałej jest:

real $xy = x + y$;

W wyniku wystąpienia w programie takiej deklaracji, w pamięci maszyny zostanie zarezerwowane miejsce o nazwie xy, w którym to miejscu będzie przechowywana aktualna wartość $x + y$.

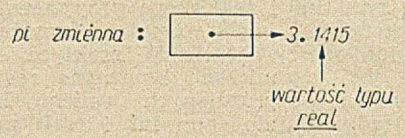


Wartości identyfikatora xy również nie można zmienić przez podstawienie.

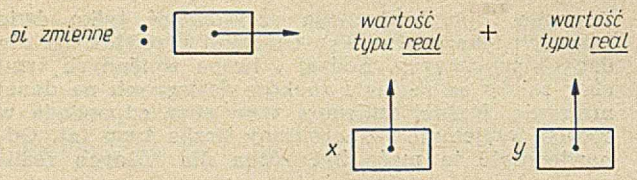
Rozpatrzmy teraz następującą deklarację:

real *pi zmienna:* = 3.1415;

Różni się ona od poprzedniej użyciem znaku := zamiast =. Jest to deklaracja zmiennej *pi zmienna* o początkowej wartości 3.1415.

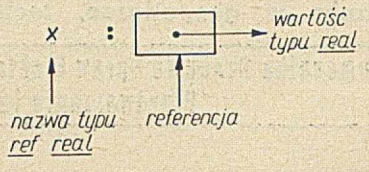


Wartość jej może być zmieniana za pomocą instrukcji podstawienia, np. *pi zmienna* := xy.



Nową wartością zmiennej *pi zmienna* jest wartość stałej xy.

Zastanówmy się, jakie konsekwencje ma użycie w programie deklaracji **real** x. Utworzony identyfikator x posiada swoją nazwę — x. Nazwa identyfikatora zadeklarowanego jako **real** x posiada tzw. referencję (na rysunku przedstawioną jako prostokąt) do wartości typu **real**; natomiast wartością zmiennej o identyfikatorze zadeklarowanym jako **real** może być dowolna liczba typu **real**.



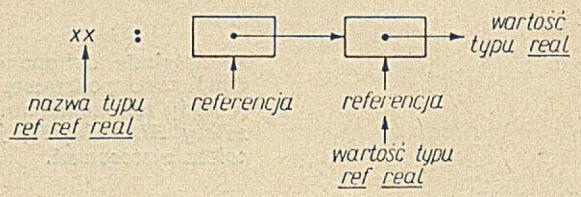
Rozróżnienie, jakie czynimy mówiąc o nazwie identyfikatora, zmiennej, jego referencji i wartości zmiennej jest analogiczne do rozróżnienia pomiędzy nazwą (symboliczną) komórki, jej adresem i zawartością.

W momencie deklaracji takiej jak **real** x tworzone są więc dwie wartości: stała i zmienna

— stała (nazwa identyfikatora) typu **ref real** posiadająca referencję do

— zmiennej wartości typu **real**.

ALGOL 68 pozwala również na deklarowanie identyfikatorów, których wartościami mogą być nazwy, tzn. obiekty typu **ref real** np. **ref real** xx.



W tym przypadku utworzoną w pamięci wartością zmienną jest nazwa; jest to nazwa wartości typu **real**. W wyniku podstawienia $xx := x$ aktualną wartością identyfikatora xx jest nazwa x.

Tworzenie nowych typów

Jedną z ciekawszych własności ALGOLu 68 jest możliwość tworzenia nieograniczonej liczby nowych typów. Poprzednio np. poznaliśmy typ **ref real**. Analogicznie można utworzyć zmienne typu **ref ref real**; będą one mogły przyjmować wartości, które są nazwami nazw zmiennych typu **real**, tzn, są typu **ref real**. Kontynuując to postępowanie otrzymamy nieskończoną liczbę typów : **ref ref ref real**, **ref ref ref ref real** itd. Na tej samej zasadzie możemy otrzymać nieskończoną liczbę typów z typu **int** lub dowolnego innego typu.

W stosunku do zmiennych o wartościach liczbowych możliwa jest deklaracja precyzji. W każdej konkretnej realizacji istnieją ograniczenia dotyczące największej liczby naturalnej i rzeczywistej. Za pomocą symbolu **long** deklarujemy zmienną o podwójnej precyzji, np. **long real**, potrójnej precyzji, np. **long long real** itd. Wyróżnione zmienne typu **int**: *intlengths* i *reallengths* określają maksymalną precyzję, jaką dopuszcza dana realizacja dla liczb typu **int** i **real**. Inną nieograniczoną grupę typów można otrzymać za pomocą symboli [ ], np. przez dopisanie [ ] do **int** lub **real** otrzymamy typ [ ]**int** lub [ ]**real**. Typy te odpowiadają znanym nam już z innych języków tablicom, np. [1 : n]**real** x jest deklaracją wektora n liczb rzeczywistych : $x[1],..., x[n]$.

Możemy też zadeklarować tablicę dwuwymiarową, trójwymiarową itd.

Na przykład [1 : *n*, 1 : *m*] **int** *y*, [1 : *x* + *t*, *m* : *n*, 0 : 5] **real** *z*.

Będą one odpowiednio typów : [,] **int** i [, ,] **real**. W ALGOLu 60 tablica zadeklarowana w danym bloku miała w nim ustalone przez aktualną wartość granice i na tej podstawie translator rezerwował dla niej odpowiednią liczbę miejsc w pamięci. Podobnie było w wyżej cytowanych przykładach.

Tymczasem ALGOL 68 stwarza nową możliwość deklarowania tablic również o zmiennych granicach. Granice takie oznaczamy symbolem **flex**. Szczególnie pożyteczne są one w przypadku tekstów.

Rozważmy np. deklarację:

[1 : **flex**] **char** *s*;

W powyższej deklaracji tylko dolna granica jest stała. Przy każdym podstawieniu wartości na zmienną *s* górna granica przyjmuje nową wartość:

s := "górna granica staje się 26";
s := "górna granica staje się teraz 32";

Innym symbolem określającym granice tablicy jest **either**.

Deklaracja:

[*h* : **either**] **real** *x* ;

oznacza, że górna granica jest albo zmienna (**flex**) albo stała. Z każdej tablicy, składającej się z więcej niż jednego elementu, można wyodrębnić jej podzbiory. Rozważmy np. deklarację:

[*h* : *k*]**ref int** *wek*;
wek[*i*] jest zmienną typu **ref int**
wek[*i* : *j*] jest zmienną typu [ ] **ref int**.

Inny przykład:

[*m* : *n*, *h* : *k*]**real** *mat*;
mat[*i*, *j*] jest zmienną typu **real**
mat[*i*,] jest zmienną typu [ ]**real**
mat[*i* : *j*, *p* : *q*] jest zmienną typu [,]**real**.

W związku z poprzednio omówionymi możliwościami tworzenia nowych typów w ALGOLu 68 istnieje też możliwość nadawania im nazw. Służy do tego symbol **mode**. Tak wprowadzone nowe typy można używać na równi ze zdefiniowanymi w języku, np.

mode napis = [1 : **flex**] **char**;

i analogicznie:

mode tablica = [1 : *n*, 1 : *n*] **real**;

Można teraz deklarować zmienne typu napis i tablica:

napis *s*; (deklaracja równoważna: [1 : *n*]**char** *s*)
tablica *A*; (deklaracja równoważna: [1 : *n*]**real** *A*).

Poznane dotychczas symbole: **ref**, **long** i [ ] służyły do tworzenia dowolnej liczby nowych typów z uprzednio zdefiniowanych. Nieco innym celom służy symbol **struct**. Obiekty typu **struct** powstają ze złożenia wcześniej określonych struktur danych. Za przykład struktury niech posłuży nam definicja typu **compl**:

mode compl = **struct**(**real** *re*, *im*);

Typ **compl** jest strukturą składającą się z dwóch pól o nazwach *re* i *im*, reprezentujących liczby rzeczywiste. Od tego momentu możemy używać deklaracji zmiennych typu **compl** na równi z innymi. Możemy np. zadeklarować strukturę *z* pisząc **compl** *z*. Pola dowolnej struktury można otrzymać za pomocą operatora **of**. Zarówno *re* **of** *z*, jak i *im* **of** *z* mogą przyjmować dowolne wartości rzeczywiste. Należy zwrócić uwagę, że *re* i *im* nie są identyfikatorami. Napis **real** *re*, *im* nie generuje żadnych wartości; spełnia on podobną rolę, jak zbiór parametrów formalnych w deklaracji procedury. Symbole *re* i *im* są nazwami pól struktury *z*.

Rozważmy teraz dwie struktury:

(1) **struct**(**real** *re*, *im*);
(2) **struct**(**real** *re* 1, *im* 1);

Napisy (1) oraz (2) opisują dwie różne struktury. Ilustruje to poniższy przykład:

struct(**real** *re*, *im*) *z* 1
struct(**real** *re*1, *im*1) *z*2, *z*3;
*re*1 **of** *z*2 := 3.14;
*im*1 **of** *z*2 := 0.0;

Instrukcja *z*3 := *z*2 powoduje przypisanie polom struktury *z*3 wartości odpowiednich pól struktury *z*2; natomiast przypisanie *z*1 := *z*2 jest nieprawidłowe, ponieważ identyfikatory po obu stronach symbolu := są różnych typów. Pola wewnątrz struktury mogą być dowolnego typu z wyjątkiem struktury właśnie zdefiniowanej. Wszystkie inne typy, z innymi strukturami włącznie, a także z referencjami do aktualnie definiowanej struktury są dopuszczalne, np.:

mode przykład = **struct**(**ref** [1 : *n*] **ref bool** *p*, **struct** (**real** *i*, **int** *j*, **ref char** *k*) *q*, **ref przykład** *r*);

Struktury mogą być utworzone z pól o różnych typach:

mode książka = **struct** (**napis** *tekst*, **int** *indeks*);
książka *Kubuś Puchatek*;

Wartością pola „*tekst* **of** *Kubuś Puchatek*" może być dowolny obiekt typu **napis** i podobnie — wartością pola „*indeks* **of** *Kubuś Puchatek*" może być dowolna liczba naturalna.

Poza sposobami dotychczas wymienionymi można określać nowe typy za pośrednictwem symbolu **union**; łączy on kilka typów w jeden.

Na przykład zmienne typu **napint** określonego w następujący sposób:

mode napint = **union**(**napis**, **int**);

mogą przyjmować wartości zarówno typu **napis**, jak i **int**.

Zastanówmy się, co w praktyce oznacza zadeklarowanie zmiennej typu **napint**:

napint *rok*;

Identyfikator *rok* jest nazwą wartości typu **napis** lub **int**.

W związku z tym poprawne są przypisania:

rok : = "1970"; (przypisany jest napis)
rok : = "tysiąc dziewięćset siedemdziesiąt" (przypisany jest napis)

rok : = 1970; (przypisana jest liczba całkowita).

Przypisanie wartości zmiennym, zadeklarowanym za pomocą symbolu **union**, odbywa się po uprzednim sprawdzeniu zgodności typów zmiennych po obu stronach symbolu przypisania. Służy do tego symbol ::= np.:

napis *tekst*;
tekst ::= *rok*;

Instrukcja ta jest wykonywana w następujący sposób:

1. Jeżeli prawa strona symbolu ::= jest typu **napis**, to wartość tego podstawienia jest **true**, a ponadto na zmienną *tekst* zostanie podstawiona wartość zmiennej *rok*.

2. Jeżeli prawa strona symbolu ::= jest typu **int**, to wartość tego podstawienia jest **false**, zaś wartość zmiennej *tekst* nie ulega zmianie.

Operacje, operatory i formuły

Ważną własnością ALGOLu 68 jest możliwość deklarowania nowych operatorów za pomocą operatorów już znanych lub przedefiniowania operatorów. Jako przykład posłużą nam definicje operatorów logicznych:

op $\neg$ = (**bool** *a*) **bool** : **if** *a* **then false else true fi**;

op $\vee$ = (**bool** *a*, *b*) **bool** : **if** *a* **then true else** *b* **fi**;

op $\wedge$ = (**bool** *a*, *b*) **bool** : **if** *a* **then** *b* **else false fi**;

op = = (**bool** *a*, *b*) **bool** : $(a \wedge b) \vee (\neg a \vee \neg b)$,

op $\neq$ = (**bool** *a*, *b*) **bool** : $\neg(a = b)$;

op abs = (**bool** *a*) **int** : **if** *a* **then** 1 **else** 0 **fi**;

Zasadnicza definicja operatora (napis między dwukropkiem a średnikiem) aktywizuje się w momencie wystąpienia tego operatora wewnątrz formuły. Dla tego samego operatora można używać różnych symboli; wszystkie one będą miały to samo znaczenie, np.:

op $\vee$ = (**bool** *a*, *b*) **bool** : **if** *a* **then true else** *b* **fi**;

op or = (**bool** *a*, *b*) **bool** : **if** *a* **then true else** *b* **fi**;

Oczywiście formuły $(a \vee b)$ i (*a* **or** *b*) mają tę samą wartość.

Istnieje subtelna, ale bardzo ważna różnica między występowaniem operatorów w formułach i wołaniem procedur. Wołanie procedury odbywa się za pomocą identyfikatora procedury niezależnego od parametrów aktualnych; natomiast interpretacja operatora w formule jest uzależniona od typu argumentów. W związku z tym ten sam symbol może być użyty w innej deklaracji tego samego bloku, a nawet w tym samym wyrażeniu, np.:

op = = (**bool** *a*, *b*) **bool** : $(a \wedge b) \vee (\neg a \wedge \neg b)$;

op = = (**compl** *a*, *b*) **bool** *re*: **of** *a* = *re* **of** *b* $\wedge$ *im* **of** *a* = = *im* **of** *b*;

Każdy operator posiada swój priorytet, wyrażający się liczbą naturalną od 1 do 10 i określający kolejność wykonywania operacji w trakcie wyliczania wartości formuły. Operatory jednoargumentowe posiadają zawsze wyższy priorytet niż wszystkie operatory dwuargumentowe. Za pomocą symbolu **priority** można określić priorytet każdego zdefiniowanego operatora, np. **priority** $\wedge$ = 3.

Zdania

W ALGOLu 68 instrukcje w programach można wykonywać sekwencyjnie i równolegle. Programista może również — w zależności od potrzeby — kierować kolejnością rozpoczęcia wykonywania poszczególnych instrukcji równoległych. Sprawę tę pozostawiamy tutaj bez dalszych wyjaśnień, a zajmiemy się jedynie instrukcjami wykonywanymi sekwencyjnie.

ALGOL 68 wprowadza zamiast instrukcji, wyrażeń i bloków nową jednostkę syntaktyczną — zdanie. W dalszym ciągu artykułu będziemy się nadal posługiwać określeniem instrukcji i wyrażenia ze względu na tradycję i nawyki.

Zdaniem jest więc zarówno instrukcja $x := a + b$, jak i wyrażenie $a + b$. W a r t o ś c i ą i n s t r u k c j i p o d s t a w i e n i a j e s t w a r t o ś ć w y r a ż e n i a s t o j ą c e g o p o p r a w e j s t r o n i e s y m b o l u p r z y p i s a n i a, w tym wypadku będzie to wartość wyrażenia $a + b$; wartością wyrażenia $a + b$ jest suma wartości identyfikatorów *a* i *b*. W związku z przypisaniem wartości instrukcji podstawienia w ALGOLu 68 poprawne jest np. takie zdanie: $a := x + (t := y \times z)$;

Zdaniem jest również ciąg instrukcji i wyrażeń oddzielony średnikami i ujęty w nawiasy **begin** i **end** lub (i) zdanie takie będziemy nazywać złożonym. Wartością zdania złożonego jest wartość ostatniego zdania poprzedniego **end** lub).

Rozważmy przykłady:

(1) **begin**

$x := a + b$;

x

end

(2) **begin**

$a := 5$;

$b := 7$;

$x := a + b$

end

Wartością zdania (1) jest wartość wyrażenia x; wartością zdania (2) — wartość instrukcji $x := a + b$. Spośród wszystkich instrukcji wartość posiada tylko instrukcja podstawienia. Wynika stąd, że jedynie takie zdanie posiada wartość, którego ostatnim zdaniem składowym jest wyrażenie lub instrukcja podstawienia.

Oprócz instrukcji i wyrażeń zdanie złożone może zawierać również deklaracje. Zdanie takie, o ile posiada wartość, może występować w wyrażeniu po prawej stronie instrukcji podstawienia, np.

$a := x +$ (**real** *a*, *b*, *y*;

$a := 5$; $b := 7$;

$y := a + b$);

lub

$a := (x + y)$/**begin real** x; $x := a + b$; x **end**;

Przez zastąpienie wyrażenia x standardową instrukcją wyjścia — *print*(x) otrzymujemy zdanie złożone nie posiadające wartości:

begin real x;

$x := a + b$;

print(x)

end

gdyż instrukcja **print** (x), która jest ostatnim zdaniem powyższego zdania złożonego, nie posiada wartości. Zdań złożonych używa się głównie w celu utworzenia zmiennych lokalnych dla danego zdania, zdefiniowania pewnego działania lub w celu określenia jakiejś pomocniczej wartości. Może ono być również użyte w celu wskazania kolejności wykonywania operacji w formułach.

Oprócz zdań złożonych w programach mogą również występować zdania warunkowe i zdania wyboru.

Zdania warunkowe mają postać:

if *b* **then** *z*1 **else** *z*2 **fi**

Zamiast symboli **if, then, else, fi** można używać symboli (, | , | ,). Na przykład przytoczone zdanie można przepisać w postaci (b | z1 | z2).

Zdanie warunkowe wykonuje się w następujący sposób:

Jeżeli zdania boolowskie *b* ma wartość **true**, to należy wykonać i podać wartość (o ile ją posiada) zdania złożonego *z*1, w przeciwnym przypadku — wykonać i podać wartość (o ile ją posiada) zdania złożonego *z*2. Jeżeli *z*1 i *z*2 są zdaniami posiadającymi wartość, to zdanie warunkowe może występować po prawej stronie instrukcji podstawienia, np.:

$x := (i < j \mid a + b \mid a - b)$

Jeżeli ponadto wartości zdań *z*1 i *z*2 mogą być interpretowane jako posiadające typ **ref**, to zdanie warun-

kowe może występować również po lewej stronie instrukcji podstawienia, np.:

begin real x, y;

$(j < i \mid y \mid x)$:= 3,1415

end

Jest to równoważne zdaniu ($j<i \mid x$:= 3,1415 $\mid y$:= 3,1415). Oprócz znanych już symboli **then** i **else**, w zdaniu warunkowym mogą występować symbole **thef** i **elsf**, będące skrótami pary symboli **then if** i **else if**.

Usprawnieniem przełącznika (switch) ALGOLu 60 jest w ALGOLu 68 zdanie wyboru:

case n **in** i_1, i_2 ..., i_k **out** z **esac**

Liczba naturalna n określa, czy ma być wykonana instrukcja pierwsza i_1, druga i_2, ... czy też k-ta i_k. Jeżeli n jest większa od liczby instrukcji, to wykonywane jest zdanie z.

Skoki i powtórzenia

Instrukcja skoku w ALGOLu 68 jest bodajże jedyną, której znaczenie nie uległo żadnej zmianie w porównaniu z ALGOLem 60. Jako przykład może nam posłużyć poniższe zdanie:

begin real x := 0; **int** n := 1;

E: $x := x + x^n$;

$n := n + 1$;

if $n \leqslant 100$ **then goto** E **fi**

end

Zamiast **goto** E można również pisać E. Zdanie to wykonuje się tak samo jak w ALGOLu 6.

W każdym zdaniu w dowolnym miejscu może wystąpić instrukcja **goto exit** (lub prościej **exit**). Powoduje ona opuszczenie aktualnie czynnego zdania złożonego. Taki skok w danym zdaniu złożonym może być tylko jeden, np.:

begin real x := 0; **int** n := 1;

if $j>0$ **then goto** E **fi**;

0.1; **exit**;

E: $x := x + x^n$;

$n := n + 1$;

if $n \leqslant 100$ **then goto** E **fi**

end

W tym przypadku wartością zdania złożonego gdy $j \leqslant 0$ jest 1.0, ponieważ jest ono ostatnim zdaniem poprzedzającym **end**.

Wielokrotnego wykonania jakiegoś ciągu instrukcji można dokonać stosując zdanie powtórzeniowe (pętlę):

begin real x := 0; **int** n := 100;

to n **do** $x := x + x^n$

end

Zwrot **to** n **do** jest szczególnym przypadkiem pełnego zdania powtórzeniowego:

for n **from** $w1$ **by** $w2$ **to** $w3$ **while** b **do** S

gdzie:

— n jest dowolnym identyfikatorem typu **int**; wystąpienie tego identyfikatora jest jego deklaracją, której zakresem jest zdanie **for**,

— $w1$, $w2$, $w3$ są wyrażeniami o wartości typu **int**,

— b jest wyrażeniem o wartości typu **bool**,

— S jest zdaniem.

W poszczególnych przypadkach każda z części zdania powtórzeniowego z wyjątkiem zdania S może być opuszczona. Sposób interpretowania takich skrótów jest określony następującymi regułami:

Jeżeli n nie jest używane w zdaniu **for**, to **for** n można opuścić. Jeżeli **from** $w1$ jest opuszczone, to wykonywane jest **from** 1.

Jeżeli **by** $w2$ jest opuszczone, to wykonywane jest **by** 1.

Jeżeli **to** $w3$ jest opuszczone, to wykonywane jest **to** ∞.

Jeżeli **while** b jest opuszczone, to wykonywane jest **while true**.

Zdanie **do** S wykonuje się nieskończenie wiele razy, S wykonuje się raz.

Procedury

Tradycyjną już metodą definiowania nowych działań są procedury. Również ALGOL 68 dysponuje wszystkimi dotychczas znanymi rodzajami procedur. Określa je postać deklaratora [1]) procedury. Przykładami deklaratorów są:

proc	— procedura bez parametrów
proc real	— procedura funkcyjna bez parametrów, dająca wartość typu **real**
proc(real, int, ref char)	— procedura 3-parametrowa
proc(real, int, ref char)bool	— procedura 3-parametrowa dająca wartość typu **bool**.

Dobrze znane z ALGOLu 60 procedury bez parametrów wzbogaciły się o możliwości, które daje ALGOL 68. Rozważmy następujące dwa przykłady:

proc E = : **begin real** x := 7, y := 15.2, z;

print(z := $(x + y)/2$ — *sqrt* $(x \times y)$)

end

proc $E1$:= : **begin**

goto *END*;

END: **end**;

W pierwszym przykładzie identyfikator E jest nazwą stałej (procedury); treść procedury E nie może być zmieniona. Natomiast w drugim przykładzie treść może być zmieniona przez podstawienie np. $E1$:= E. Zdanie **begin goto** *END*; *END* : **end**; jest początkową wartością zmiennej $E1$. Zauważmy, że podobnie jak obiekty typu **real** czy **int**, obiekt typu **proc** może być stałą lub zmienną; jego wartością jest treść procedury.

Zadeklarowaną procedurę można użyć w zdaniu:

begin if $j > 0$ **then** E **else** 0 **fi end**;

Ciekawą własnością ALGOLu 68 jest możliwość nadawania typu **proc** dowolnym zdaniom:

proc p;

$p := x := 3.1415$;

Zadeklarowana powyżej procedura p jest procedurą bezparametrową; zmienna p jest typu **proc**. Oznacza to, że na p można podstawić tylko wartości typu **proc**. W zdaniu p := x := 3.1415 ani x, ani 3.1415 nie jest typu **proc**, a pomimo to podstawienie jest prawidłowe. Wykonuje się ono w następujący sposób : instrukcja x := 3.1415 nie jest wykonywana, tzn. wartość x nie zmienia się, ale instrukcja ta będzie potraktowana jako ciało procedury i zostanie podstawiona na p. Inaczej mówiąc, zdanie x := 3.1415 otrzyma typ **proc** i zostanie przypisane zmiennej p.

Treścią procedury może być nie tylko zdanie określające czynność, lecz również zdanie określające wartość. W tym drugim przypadku procedura nosi nazwę funkcyjnej. W deklaracji takiej procedury występuje także określenie typu wartości treści procedury, np.:

proc real Ez = **real** : **begin real** x := 7, y:= 15.2, z;

z : = $(x+y)/2$—*sqrt* $(x \times y)$

end;

---

[1]) Część deklaracji określająca typ deklarowanej zmiennej.

Napis **real** ustala, że wymaganym typem wartości ciała procedury jest **real**; typem identyfikatora *E2* jest **proc real** — nowy typ otrzymany z **real**.

Przykładem deklaracji procedury z parametrami jest:

proc(real, real)real *D* =

(**real** *a*, *b*)**real**: ((*a* + *b*)/2 — *sqrt*(*a* × *b*));

Typem identyfikatora *D* jest **proc(real, real)real**.

Poznajmy bliżej poszczególne części deklaracji procedury:

1. Część **real** *a*, *b* określa parametry formalne *a* i *b* typu **real**. Parametry aktualne podstawiane w miejsce formalnych muszą być również typu **real**. Są jednak możliwe odstępstwa od tej reguły, np. jeśli zostanie podstawiony parametr typu **int**, to będzie on automatycznie zamieniony na typ **real**.

2. Część **real**: ustala, że wymaganym typem wartości treści procedury jest **real**.

3. Podczas wywołania procedury zdaniem *a*+*D*(*x*,*y*)/3 treść przyjmie następującą postać:

begin real *a*, *b*;

a := *x*; *b* := *y*;

(*a* + *b*)/2 — *sqrt*(*a* × *b*)

end

a całe zadanie stanie się równoważne zdaniu:

a + **begin real** *a*, *b*; *a* := *x*; *b* := *y*; (*a* + *b*)/2 — *sqrt* (*a* × *b*) **end**/3

Może się również zdarzyć, że typ parametru aktualnego nie jest ustalony w momencie deklaracji procedury:

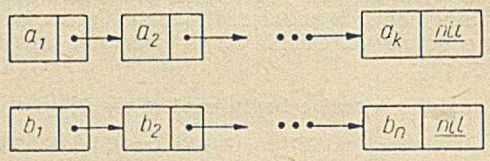
```
proc intreal = (union(int, real) inre):
                begin real x := 0, p;
                  if p ::= inre
                  then (to p do x := x + x^p)
                  else inre
                end;
```

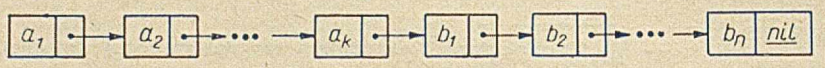
Parametrem aktualnym procedury *intreal* może być zarówno wartość typu **int**, jak i **real**. Wykonanie ciała procedury w trakcie jej wywołania będzie uzależnione od typu parametru aktualnego. Parametrami procedury mogą być obiekty dowolnego typu. W szczególności mogą być procedurami, np.:

```
proc p = (int k, proc (int)real par)real:
            begin real sum := 0;
              for j to k do sum := sum+par (j);
              sum
            end
```

W trakcie wywołania procedury parametr aktualny podstawiony w miejsce parametru formalnego musi być typu **proc (int) real**, np. *x* := *p* (100, (**int** *i*) **real**: 1/*i*);

Jeżeli parametrem formalnym jest procedura bezparametrowa, to parametrem aktualnym może być zdanie, które w trakcie realizacji otrzyma typ **proc**. Na zakończenie podamy jeszcze przykład elementarnej procedury pozwalającej dwie struktury listowe[2]) połączyć w jedną.





Użyty tu symbol :≠: oznacza operator Boole'owski, który daje wartość **true** wtedy i tylko wtedy, gdy wiąże on bądź różne nazwy, bądź nazwy o różnych typach. Symbol **nil** jest stałym symbolem dla nazwy nie posiadającej referencji do żadnej wartości. Podana procedura jest określona w sposób rekursywny.

```
begin
  mode atom = [1 : flex]char;
  mode lista = struct(atom car, ref lista cdr);
  proc(lista, lista)lista połącz =
  (lista x, lista y)lista: (cdr of x :≠: nil |
                              ((cdr of x, połącz(cdr of x,y)) |
                              ((car of x, y));
  .
  .
  .
end
```

Uwagi końcowe

ALGOL 68 w porównaniu z innymi językami programowania jest znacznie wszechstronniejszy. Świadczy o tym m. in. to, że jego poszczególne podzbiory mogą służyć m. in. do obliczeń numerycznych, przetwarzania list lub innych bardziej złożonych struktur danych oraz do przetwarzania napisów. Jeśli istniejące typy i operacje okażą się niewystarczające, istnieją również możliwości definiowania nowych. Programista ma więc do swojej dyspozycji środki umożliwiające mu tworzenie języka według własnych potrzeb. Ponadto może on opisywać ograniczenia wynikające z użycia konkretnej maszyny.

Wśród innych cech, które świadczą najlepiej o nowoczesności ALGOLu 68, wymienić trzeba również te własności języka, które pozwalają na znaczne skracanie i upraszczanie programów. Warto również zwrócić uwagę na jednakowe potraktowanie wszystkich typów, przez co uzyskano możliwość lepszego wykorzystania tworzonych przez programistę obiektów, takich jak np. tablica nazw procedur, która może być w trakcie programu „zapełniana" zgodnie z potrzebami. Dodać jeszcze trzeba, że ALGOL 68 posiada rozbudowany mechanizm wejścia—wyjścia, czym nie dysponował wcześniejszy ALGOL 60.

Opisując ALGOL 68 nie wyczerpaliśmy wszystkich zagadnień. Osobnego wyjaśnienia wymagają m. in. problemy związane z wejściem i wyjściem, równoległym wykonywaniem zadań i działaniami na plikach danych. Zainteresowanych odsyłamy do poszczególnych pozycji przytoczonej bibliografii.

ALGOL 68 nie został jeszcze wprawdzie zrealizowany dla żadnej maszyny, co utrudnia ocenę jego pełnej przydatności, nie ulega jednak wątpliwości, że wartości, które reprezentuje w sensie teoretycznym, a zwłaszcza takie, jak uniwersalność i elastyczność, stawiają go w rzędzie języków przyszłości.

---

2) Por. np. Jowita Koncewicz — Języki do przetwarzania struktur listowych, Maszyny Matematyczne, nr 5/1970.

BIBLIOGRAFIA

[1] A. van Wijngarden, B. J. Maillaux, I. E. L. Peck, C. H. Koster, Report on The Algorithmic Language ALGOL 68. Mathematic Centrum, Amsterdam 1969

[2] S. G. van der Meulen, O. H. Lindsey, Informal Introduction to ALGOL 68, Mathematische Centrum, Amsterdam 1969

[3] C. H. Linsey, ALGOL 68 with Fewer Tears, ALGOL Bulletin

JAN MAŁUSZYŃSKI
Instytut Maszyn Matematycznych
Warszawa

681.32206:801.56

Mgr inż. Jan Małuszyński jest absolwentem Wydziału Automatyki i Maszyn Matematycznych Moskiewskiego Instytutu Energetycznego. W 1968 roku ukończył Studium Zaoczne Matematyki Uniwersytetu Warszawskiego. Od 1967 roku pracuje w Zakładzie Metod Programowania Instytutu Maszyn Matematycznych, gdzie zajmuje się zagadnieniami składni języków programowania.

Opis składni języków programowania

Najbardziej rozpowszechnioną formalizację opisu składni języków programowania jest notacja Backusa (BNF). Notacja BNF jest pewnym sposobem zapisu gramatyk bezkontekstowych, zdefiniowanych przez Chomsky'ego. Aparat gramatyk bezkontekstowych jest na ogół zbyt słaby dla wyrażenia niektórych cech jezyka programowania. W artykule opisano niektóre próby utworzenia nowego aparatu do syntetycznego opisu składni języków programowania, w szczególności ALGOLu 68.

Twórcy języka programowania stoją zawsze przed trudnym zadaniem, jakim jest udostępnienie tego języka szerszemu ogółowi.

Przede wszystkim muszą oni określić, jakie napisy uważa się za dozwolone konstrukcje językowe. Następny problem polega na ścisłym sprecyzowaniu znaczenia tych konstrukcji; chodzi tu o określenie efektów wykonania programu zapisanego w tym języku. Wymaga to na ogół dokładnego określenia abstrakcyjnej maszyny używanej do wykonywania programu. Autorzy języka muszą ponadto dać użytkownikowi pewne praktyczne wskazówki, dotyczące stosowania zwrotów języka. Te trzy aspekty opisu języka nazywane są *składnią*, *semantyką* i *pragmatyką*. Są one ściśle ze sobą powiązane i trudno wyznaczyć dokładnie granice pomiędzy nimi.

W tym artykule poruszymy niektóre zagadnienia związane z metodami opisu składni języków programowania.

Programista korzystający z danego języka programowania oczekuje rozstrzygającej wszystkie wątpliwości formalnej recepty, która mówiłaby, jak tworzyć dozwolone zwroty i jak z tych zwrotów budować program. Muszą tu jednak istnieć dość szerokie możliwości wyboru, gdyż zwroty dobiera się w zależności od konkretnego rozwiązywanego zadania. W czasie tłumaczenia programu, maszyna sprawdza, czy użyto dozwolonych zwrotów i czy struktura programu jest poprawna. W tym celu korzysta ona z algorytmu rozbioru, określonego w czasie tworzenia translatora na podstawie opisu składni. Zadania programisty i maszyny są więc w tym przypadku odwrotne. Nic dziwnego, że opis składni przeznaczony dla maszyny różni się zazwyczaj od opisu składni przeznaczonego dla człowieka. Opis składni, który określa, jak uzyskiwać dozwolone zwroty i jak budować z nich programy, nazwiemy opisem syntetycznym, opis algorytmu rozbioru — analitycznym opisem składni. Ten artykuł omawia niektóre metody syntetycznego opisu składni.

Problem formalnego opisu składni języków programowania nie został jeszcze w pełni rozwiązany m. in. dlatego, że niektóre cechy programu związane wyłącznie z jego postacią nie dają się opisać w istniejących formalizmach.

Na przykład występujący w ALGOLu 60 problem deklaracji — postulat jednoznacznego i niesprzecznego deklarowania używanych zmiennych — nie jest objęty formalnym opisem składni tego języka. Korzystając tylko z formalnego opisu składni można więc napisać dozwolony formalnie program, który będzie niepoprawny z powodu naruszenia pewnych reguł, nie dających się wyrazić za pomocą przyjętych metod.

W związku z szybkim rozwojem języków programowania, koniecznością staje się automatyzacja konstrukcji translatorów. Podstawą wszelkich badań w tej dziedzinie jest właśnie formalizacja opisu składni. Duża liczba publikacji na ten temat świadczy o jego aktualności. Można przypuszczać, że nowe metody opisu składni języków programowania pozwolą zmniejszyć liczbę cech programu nie objętych formalnym opisem.

Notacja BNF i klasyfikacja Chomsky'ego

Najbardziej rozpowszechnionym, stosowanym od dawna formalizmem opisu składni języków programowania, jest notacja Backusa (BNF), zastosowana m. in. do opisu składni ALGOLu 60 [1].

Skończony zbiór reguł określa, w jaki sposób można tworzyć dozwolone zwroty. Każda reguła jest napisem, w którym występuje symbol ::= oddzielający lewą i prawą stronę reguły. Lewa strona każdej reguły nazywa się *zmienną metajęzykową*. Zbiór wszystkich lewych stron tworzy *alfabet zmiennych*. Każda zmienna metajęzykowa ma postać napisu ujętego w specjalne nawiasy ⟨ ⟩. Po prawej stronie reguły oprócz zmiennych metajęzykowych może wielokrotnie występować symbol pomocniczy | oraz inne symbole nie zawarte w nawiasach. Te ostatnie określają *alfabet terminalny*. Weźmy jako przykład:

⟨*cyfra*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

⟨*liczba całkowita*⟩ ::= ⟨*cyfra*⟩ | ⟨*liczba całkowita*⟩ ⟨*cyfra*⟩

Powyższe reguły określają alfabet zmiennych metajęzykowych V = {⟨cyfra⟩, ⟨liczba całkowita⟩} oraz alfabet terminalny T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Ogólnie, prawa strona reguły ma postać ciągu składników oddzielonych symbolami |. Każdy z tych składników może zawierać zarówno symbole z alfabetu zmiennych, jak i z alfabetu terminalnego. Dozwolone wyrażenie opisanego w ten sposób języka — to każde słowo zawierające wyłącznie symbole z alfabetu terminalnego, które można otrzymać z jednej wyróżnionej zmiennej metajęzykowej przez kolejne zastępowanie pojawiających się zmiennych metajęzykowych dowolnymi składnikami z odpowiadających im reguł. Przypuśćmy, że w powyższym przykładzie zmienną wyróżnioną jest ⟨*liczba całkowita*⟩. Wówczas słowo 10 jest dozwolonym wyrażeniem określonego przez podane reguły języka, ponieważ zmienną ⟨*liczba całkowita*⟩ możemy za pomocą reguł przekształcić w następujący sposób:

⟨*liczba całkowita*⟩ ==> ⟨*liczba całkowita*⟩ ⟨*cyfra*⟩ ==>

⟨*cyfra*⟩ ⟨*cyfra*⟩ ==> 1⟨*cyfra*⟩ ==> 10

Łatwo zauważyć, że kolejność zastępowania zmiennych nie wpływa na ostatecznie uzyskiwane słowo języka. Słowo to zależy od wyboru składników reguł przy zastępowaniu zmiennych metajęzykowych występujących w pośrednich słowach.

W celu zmniejszenia liczby reguł i uproszczenia zapisu BNF stosuje się czasem pewne konwencje skrótowe, np. reguły BNF w postaci:

1. $V ::= S_1\ T_1\ S_2 \mid S_1\ T_2\ S_2 \mid \ldots \mid S_1\ T_n\ S_2 \mid \mid$
2. $V ::= S_1\ S_2 \mid S_1\ T_1\ S_2 \mid S_1\ T_2\ S_2 \mid \ldots \mid S_1\ T_n\ S_2$
3. $V ::= U \mid VU$

gdzie $T_1, T_2, \ldots T_n$ oznaczają dowolne niepuste[1]) słowa, a S_1, S_2 oraz U — dowolne słowa, można zapisywać w postaci:

1. $V ::= S_1\ \{T_1 \mid T_2 \mid \ldots \mid T_n\}\ S_2$
2. $V ::= S_1\ [T_1 \mid \ldots \mid T_n]\ S_2$
3. $V ::= U :::$

Zapis postaci

$V ::= S_1\ \{T_2 : T_1\} :::\ S_2$

oznacza

$V ::= S_1\ T_1\ [\{T_2\ T_1\} :::\ S_2$

Stosując konwencję 3 możemy drugą regułę z poprzedniego przykładu zapisać w następujący sposób:

⟨*liczba całkowita*⟩ ::= ⟨*cyfra*⟩ :::

Teoretyczną podstawą do badań opisów, podobnych do notacji BNF, stanowi praca Chomsky'ego [2]. Według definicji podanej w tej pracy, gramatykę określa się, podając alfabet zmiennych z wyróżnionym symbolem początkowym, alfabet terminalny oraz zbiór produkcji. Produkcje określają, w jaki sposób można przekształcać wyróżniony symbol nieterminalny i otrzymane z niego słowa. Każdą produkcję stanowią dwa słowa połączone symbolem →, który oznacza „można przepisać jako". Ogólna postać produkcji jest następująca:

$\alpha A \beta \rightarrow \alpha \beta \omega;$

gdzie α, β, ω są słowami zbudowanymi z dowolnych symboli należących do alfabetu zmiennych lub alfabetu terminalnego, A jest symbolem z alfabetu zmiennych.

Chomsky sklasyfikował gramatyki według dodatkowych ograniczeń spełnianych przez postać produkcji. Określił on w ten sposób cztery klasy gramatyk i odpowiednio języków, ponumerowane kolejno od 0 do 3.

[1]) Tzn. zawierające co najmniej jeden symbol z alfabetu terminalnego lub alfabetu zmiennych metajęzykowych.

Tak więc formułuje się następujące ograniczenia:

- dla gramatyk typu 0 nie wprowadza się dodatkowych ograniczeń na postać produkcji,
- dla gramatyk typu 1, zwanych *gramatykami kontekstowymi*, słowo ω w żadnej produkcji nie może być słowem pustym,
- dla gramatyk typu 2, zwanych *gramatykami bezkontekstowymi*, wszystkie produkcje są postaci $A \rightarrow \omega$,
- dla gramatyk typu 3, zwanych *gramatykami regularnymi*, produkcje są postaci $A \rightarrow aB$ lub $A \rightarrow a$, gdzie A, B są symbolami z alfabetu zmiennych a — symbolem z alfabetu terminalnego.

Klasa języków, które można opisać w notacji BNF, jest klasą języków bezkontekstowych. Każda reguła notacji BNF określa tyle produkcji gramatyki bezkontekstowej, ile składników zawiera jej prawa strona. Lewe strony tych produkcji są identyczne z lewą stroną reguły BNF, prawą stronę każdej produkcji stanowi jeden ze składników.

Jeżeli w alfabecie zmiennych, określonym przez zbiór reguł z poprzedniego przykładu, symbolem wyróżnionym jest ⟨*liczba całkowita*⟩, to równoważna gramatyka bezkontekstowa składa się z następujących elementów:

- alfabet zmiennych: ⟨*liczba całkowita*⟩, ⟨*cyfra*⟩
- alfabet terminalny: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- zbiór produkcji:

⟨*liczba całkowita*⟩ → ⟨*cyfra*⟩

⟨*liczba całkowita*⟩ → ⟨*liczba całkowita*⟩ ⟨*cyfra*⟩

⟨*cyfra*⟩ → 0
⟨*cyfra*⟩ → 1
⟨*cyfra*⟩ → 2
⟨*cyfra*⟩ → 3
⟨*cyfra*⟩ → 4
⟨*cyfra*⟩ → 5
⟨*cyfra*⟩ → 6
⟨*cyfra*⟩ → 7
⟨*cyfra*⟩ → 8
⟨*cyfra*⟩ → 9

Cechą istotną gramatyki bezkontekstowej jest to, że zmienną metajęzykową w przekształcanym słowie można zawsze zastąpić prawą stroną odpowiedniej produkcji bez względu na to, jakie inne symbole występują w tym słowie. Nie ma więc kontroli nad całością słowa, wszystkie przekształcenia dokonywane są lokalnie i niezależnie. Widać stąd, że formalizm ten nie nadaje się do opisywania integralnych cech, takich np. jak żądanie, by używane w programie zmienne były uprzednio zadeklarowane.

Próby utworzenia nowego aparatu syntetycznego opisu składni

Aparat gramatyk bezkontekstowych jest zbyt mało precyzyjny dla wyrażenia niektórych cech języka. Z kolei zarówno języki typu 1 bez dodatkowych ograniczeń, jak i języki typu 0 wymagają dość złożonych algorytmów rozbioru, przy czym dla języków typu 0 w ogólnym przypadku nie istnieją algorytmy wykrywania błędów syntaktycznych [2]. Z tego powodu nowe formalizmy pojawiające się w literaturze służą zazwyczaj do opisywania pewnych szczególnych klas języków typu 1. Omówimy teraz niektóre z tych propozycji.

Jak już wspominaliśmy, trudność w zastosowaniu gramatyk bezkontekstowych polega zwykle nie na tym, że nie możemy za pomocą gramatyki bezkontekstowej otrzymać napisów o żądanych własnościach, lecz na tym, że nie da się sformułować ograniczeń, które pozwoliłyby tworzyć wyłącznie napisy o żądanych cechach. Istnieją próby formalnego sformułowania takich ograniczeń. W trakcie przekształcania słów

za pomocą produkcji gramatyki bezkontekstowej mamy zwykle do wyboru kilka słów, którymi możemy zastąpić każdą eliminowaną zmienną metajęzykową. Są to prawe strony wszystkich tych produkcji, w których dany symbol występuje po lewej stronie. Można sformułować pewne kryteria ograniczające ten wybór w zależności od poprzednio dokonanych wyborów. Otrzymuje się w ten sposób ograniczony język, który nie musi być językiem bezkontekstowym. Kryteria te formułuje się różnie w różnych pracach. Weźmy jako przykład następujące sformułowanie [3]. Każda z produkcji gramatyki bezkontekstowej opatrzona jest pewną wyróżniającą tę produkcję etykietą. Ciąg przekształceń dokonanych za pomocą tej gramatyki można opisać za pomocą słowa kontrolnego — ciągu etykiet kolejno stosowanych produkcji. Wystarczy teraz narzucić ograniczenie na postać słowa kontrolnego, aby uzyskać język różny od języka określonego przez daną gramatykę.

Weźmy jako kolejny przykład gramatykę bezkontekstową

$$G = (\{X, Y, Z\}, \{x, y, z\}, X, F)$$

(podane zostały kolejno: alfabet zmiennych, alfabet terminalny, symbol wyróżniony i nazwa zbioru produkcji).

Zbiór produkcji F podamy wraz z etykietami:

$$f_1{:}X \to XYZ \quad f_2{:}X \to xX \quad f_3{:}Y \to yY \quad f_4{:}Z \to zZ$$

$$f_5{:}X \to x \quad f_6{:}Y \to y \quad f_7{:}Z \to z$$

Narzucając ograniczenie, że słowo kontrolne ma postać:

$$f_1 (f_2 f_3 f_4)^i f_5 f_6 f_7$$

gdzie $i = 0, 1, 2...$ oznacza liczbę powtórzeń „podsłowa" ujętego w nawiasy, otrzymujemy język, w którym słowa są postaci $x^{i+1}\ y^{i+1}\ z^{i+1}$. Z teorii wiadomo, że język ten nie jest językiem bezkontekstowym. Żądaną postać słowa kontrolnego można łatwo opisać za pomocą prostej gramatyki typu 3. Mamy więc tu przypadek opisu języka za pomocą dwóch prostych gramatyk, bezkontekstowej i regularnej.

Autorzy metody *makrogramatyk* [4] rozwinęli formalizm gramatyk bezkontekstowych w nieco inny sposób. Wydaje się, że najistotniejszą cechą ich pomysłu jest umożliwienie równoczesnego tworzenia różnych fragmentów słowa języka i elastycznego operowania tymi fragmentami. Korzysta się w tym celu ze specjalnego *alfabetu parametrów formalnych*. Symbole z tego alfabetu występują tylko w produkcjach makrogramatyki i symbolizują takie fragmenty. Rolę zmiennych metajęzykowych pełnią tzw. *symbole funkcyjne*, z których jeden jest wyróżniony jako *symbol początkowy*. Każdy symbol funkcyjny ma pewną przyporządkowaną stałą nieujemną liczbę parametrów, które zawsze występują ujęte w nawiasy i oddzielane przecinkami wraz z każdym wystąpieniem tego symbolu. Te parametry — to różne napisy, stała jest tylko ich liczba. Wyróżniony symbol funkcyjny ma liczbę parametrów równą zeru. Produkcje makrogramatyki, podobnie jak produkcje gramatyki bezkontekstowej, służą do przekształcania napisów. Lewą stronę każdej produkcji stanowi symbol funkcyjny, występujące wraz z nim parametry są symbolami z alfabetu parametrów formalnych.

Prawa strona produkcji jest napisem, który może zawierać symbole funkcyjne, symbole terminalne oraz te symbole z alfabetu parametrów formalnych, które wystąpiły po lewej stronie. Postać tego napisu podlega ściśle określonym regułom, których nie będziemy przytaczać.

Podamy przykładową makrogramatykę:

Zbiór produkcji: $S \to F\ (a, b, c)$

$$F\ (x, y, z) \to F\ (xa, yb, zc)$$

$$F\ (x, y, z) \to xyz$$

Alfabet symboli funkcyjnych: S, F

Symbol wyróżniony: S

Alfabet symboli terminalnych: a, b, c

Alfabet parametrów formalnych: x, y, z

Język opisywany przez makrogramatykę określa się tak samo, jak w przypadku gramatyki bezkontekstowej. Jest to zbiór tych słów zbudowanych z symboli terminalnych, które można uzyskać z wyróżnionego symbolu, stosując produkcje makrogramatyki. Natomiast sam sposób stosowania produkcji jest nieco inny. Jeśli w przekształcanym napisie wybrany został symbol funkcyjny, który ma być zastąpiony, to postępuje się w następujący sposób:

- wybiera się dowolną produkcję, w której ten symbol występuje po lewej stronie,
- ustala się na podstawie kolejności występowania odpowiedniość między parametrami symbolu funkcyjnego w przekształcalnym napisie i parametrami formalnymi lewej strony wybranej produkcji,
- parametry formalne występujące po prawej stronie wybranej produkcji zastępuje się odpowiednimi parametrami „aktualnymi", występującymi w przekształcanym napisie,
- usuwa się z przekształcanego napisu symbol funkcyjny wraz z parametrami i wstawia się na jego miejsce przekształconą prawą stronę produkcji.

Korzystając z opisanych zasad przekształcania wyprowadzimy słowo *aaabbbccc* należące do języka opisanego przez makrogramatykę z poprzedniego przykładu:

$$S \Longrightarrow F\ (a, b, c) \Longrightarrow F\ (aa, bb, cc) \Longrightarrow F\ (aaa, bbb, ccc)$$
$$\Longrightarrow aaabbbccc$$

Łatwo zauważyć, że ta makrogramatyka opisuje język, którego słowa są postaci $a^i b^i c^i$ $i = 1, 2...$, a więc język, który nie jest językiem bezkontekstowym. Słowa tego języka tworzy się łącząc niezależnie utworzone fragmenty o jednakowej liczbie symboli. Na ogół przy przekształcaniu napisów, przestrzega się pewnych reguł ograniczających dowolność wyboru zastępowanych symboli funkcyjnych. W pracy [4] zdefiniowano kilka odmian makrogramatyk różniących się pod tym względem. W niektórych z nich dopuszcza się m. in stosowanie nawiasów kwadratowych w celu ograniczenia wyboru zastępowanych symboli funkcyjnych.

Posługując się tym aparatem, autorzy zdefiniowali przykładowy prosty język programowania, w którym problem deklaracji został rozwiązany za pomocą składni. Nie udało się jednak wykluczyć możliwości sprzecznych deklaracji. Różne odmiany makrogramatyk służą do opisywania różnych klas języków. Algorytmy rozbioru nie zostały podane w źródłowej pracy [4], wiadomo jednak, że istnieją.

Szczególną klasę stanowią makrogramatyki, w których wszystkie symbole funkcyjne są zeroparametrowe — są to po prostu gramatyki bezkontekstowe.

Omówimy teraz pokrótce aparat użyty do opisu składni języka ALGOL 68 [5].

Zapiszemy na wstępie kilka wybranych reguł z dobrze znanej składni ALGOLu 60:

⟨*identifier list*⟩ ::= ⟨*identifier*⟩ | ⟨*identifier list*⟩ ⟨*identifier*⟩

⟨*formal parameter list*⟩ ::= ⟨*formal parameter*⟩ |

⟨*formal parameter list*⟩ ⟨*parameter delimiter*⟩ ⟨*formal parameter*⟩

⟨*actual parameter list*⟩ ::= ⟨*actual parameter*⟩ | ⟨*actual parameter list*⟩ ⟨*parameter delimiter*⟩ ⟨*actual parameter*⟩

⟨*bound pair list*⟩ ::= ⟨*bound pair*⟩ | ⟨*bound pair list*⟩, ⟨*bound pair*⟩

⟨*for list*⟩ ::= ⟨*for list element*⟩ | ⟨*for list*⟩, ⟨*for list element*⟩

⟨*type list*⟩ ::= ⟨*simple variable*⟩ | ⟨*type list*⟩, ⟨*simple variable*⟩

⟨*array list*⟩ ::= ⟨*array segment*⟩ | ⟨*array list*⟩, ⟨*array segment*⟩

⟨*switch list*⟩ ::= ⟨*designational expression*⟩ | ⟨*switch list*⟩, ⟨*designational expression*⟩

⟨*subscript list*⟩ ::= ⟨*subscript expression*⟩ | ⟨*subscript list*⟩, ⟨*subscript expression*⟩

⟨*parameter delimiter*⟩ ::=, |)⟨*letter string*⟩(

Przepiszemy te reguły, wprowadzając następujące zmiany:

- symbol ::= zastępujemy dwukropkiem, symbol | średnikiem;
- wszystkie symbole terminalne zastępujemy umownymi znakami tych symboli; od zmiennych metajęzykowych odróżnia je to, że nazwy te muszą kończyć się słowem „symbol" (uniezależniamy się w ten sposób od konkretnej reprezentacji);
- opuszczamy nawiasy wyróżniające zmienne metajęzykowe;
- sąsiadujące symbole w składnikach reguł oddzielamy przecinkami;
- na końcu każdej reguły umieszczamy kropkę.

Otrzymujemy w ten sposób następujące reguły:

identifier list : identifier; identifier list, comma symbol, identifier list.

formal parameter list : formal parameter; formal parameter list, parameter delimiter, formal parameter.
actual parameter list : actual parameter; actual parameter list, parameter delimiter, actual parameter.

bound pair list : bound pair; bound pair list, comma symbol, bound pair.

for list : for list element; for list, comma symbol, for list element.

type list : simple variable; type list, comma symbol, simple variable.

array list : array segment; array list, comma symbol, array segment.

switch list : designational expression; switch list, comma symbol, designational expression.

subscript list : subscript expression; subscript list, comma symbol, subscript expression.

parameter delimiter : comma symbol; right bracket symbol, letter string, left bracket symbol.

Można łatwo zauważyć, że wszystkie te reguły, z wyjątkiem ostatniej, zostały utworzone według jednego schematu:

META *list* : META *list element*; META *list*, META *list delimiter*, META *list element*.

Aparat użyty do opisu składni ALGOLu 68 pozwala korzystać z takich schematów w celu definiowania całych grup reguł, a nawet nieskończonych zbiorów reguł. W tym celu pisane wielkimi literami słowa, zwane *metanotacjami*, opisuje się oddzielnie za pomocą tzw. *metareguł*. Postać metareguł różni się od postaci przekształconych reguł z przykładu 6 tym, że zmienną metajęzykową jest każda metanotacja, a każde słowo zapisane małymi literami może być uważane za symbol terminalny. W związku z tym przecinki są zbędne i nie występują w metaregułach. Zbiór wszystkich metareguł może opisywać różne języki bezkontekstowe, w zależności od tego, która metanotacja zostanie wyróżniona. Język określony przez wyróżnienie pewnej metanotacji będziemy nazywać *rozwinięciem* tej metanotacji.

Podamy jako przykład zbiór metareguł opisujących metanotację META

META : ALFA; *for; type; array; switch; subscript.*

ALFA : *identifier; formal parameter; actual parameter; bound pair.*

Rozwinięciem metanotacji ALFA jest zbiór składników drugiej z podanych metareguł. Rozwinięcie metanotacji META zawiera prócz tego wszystkie składniki terminalne występujące w pierwszej metaregule. Schematy i metareguły służą do tworzenia właściwych reguł opisujących składnię języka programowania. W tym celu schematy przekształca się tak, aby prawa strona każdego z przekształconych schematów nie zawierała średnika. Podobnie przekształcaliśmy reguły BNF na odpowiadające im produkcje gramatyki bezkontekstowej. Nasz przykładowy schemat można zastąpić dwoma schematami:

META *list* : META *list element*.

META *list* : META *list*, META *list delimiter*, META *list element*.

Eliminując w każdym z tych nowych schematów metanotacje w ten sposób, że każde wystąpienie pewnej metanotacji zastępujemy ustalonym elementem, dowolnie wybranym z jej rozwinięcia — otrzymujemy właściwe reguły. Tak np. z pierwszego z powyższych schematów można m. in. otrzymać regułę *identifier list : identifier*. Natomiast napis *identifier list : bound pair*. nie jest regułą, gdyż powstał przez zastąpienie w jednym schemacie dwóch wystąpień tej samej metanotacji różnymi elementami z jej rozwinięcia.

Jeżeli rozwinięcie każdej metanotacji zawiera tylko skończoną liczbę elementów (tak, jak w naszym przykładzie), to ze schematów można otrzymać tylko skończoną liczbę różnych reguł, a więc jest to wówczas aparat opisujący, jak tworzyć reguły podobne do reguł BNF.

W ogólnym przypadku, gdy rozwinięcia metanotacji są nieskończone, można utworzyć nieskończenie wiele różnych reguł i nieskończenie wiele zmiennych metajęzykowych. W opisie składni ALGOLu 68 korzysta się z tej możliwości w celu ścisłego opisu niektórych nowych cech języka.

Prócz schematów i metareguł, opis składni może zawierać niektóre reguły podane jawnie.

Podamy teraz zbiór schematów i reguł, z którego można otrzymać za pomocą metareguł z ostatniego przykładu wszystkie przekształcone reguły z przykładu poprzedniego. Sprawdzenie tego faktu pozostawiamy Czytelnikowi.

META *list* : META *list element*; META *list*, META *list delimiter*, META *list element*.

ALFA *list element* : ALFA.

META *list delimiter* : *comma symbol.*

type list element : simple variable.

array list element : array segment.

switch list element : designational expression.

formal parameter list delimiter : parameter delimiter.

actual parameter list delimiter : parameter delimiter.

parameter delimiter : right bracket symbol, letter string, left bracket symbol.

Większość jawnie podanych reguł służy tylko do przemianowania zmiennych metajęzykowych, gdyż różnorodna terminologia ALGOLu 60 nie jest przystosowana do takiej metody opisu.

Reguły uzyskane ze schematów podanych w opisie składni ALGOLu 68 za pomocą metareguł oraz reguły podane w jawnej postaci służą do tworzenia programów w języku ALGOLu 68 w ten sam sposób, jak reguły BNF służyły do tworzenia programów w języku ALGOL 60. Jednak utworzone w ten sposób programy ALGOLu 68 są niezależne od konkretnej reprezentacji symboli terminalnych, gdyż zawierają tylko umowne nazwy tych symboli. Programy te mogą być semantycznie niepoprawne, gdyż niektóre cechy, podobnie jak w ALGOLu 60, nie zostały objęte for-

malnym opisem składni. Dotyczy to m. in. problemu zgodności deklaracji. Język opisany przez reguły składni nazywa się *ścisłym — strict language*. Do programu zapisanego w tym języku można dołączać komentarze oraz zastępować niektóre zwroty pewnymi skrótami. Te przekształcenia nie zmieniają znaczenia programu i są podyktowane względami pragmatycznymi. Język, który powstaje w ten sposób, nosi nazwę *rozwiniętego — extended language*.

Metoda opisu składni tego języka nie jest sformalizowana w takim stopniu, jak opis języka *strict language* dlatego, mówiąc o składni ALGOLu 68 mamy na myśli właśnie *strict language*.

Udowodniono [6], że korzystając z metody użytej do opisu składni ALGOLu 68 można opisać każdy język typu 0. Wynika stąd, że nie istnieje ogólny algorytm wykrywania błędów syntaktycznych, nadający się dla wszystkich języków, które mogą być w ten sposób opisane. Okazuje się jednak, że składnia języka ALGOL 68 podlega pewnym ograniczeniom, które umożliwiają znalezienie takiego algorytmu. Ograniczenia te nie zostały jawnie podane w dokumencie źródłowym. Wydaje się, że warunkiem szerszego zastosowania tej metody opisu jest sformułowanie takich ograniczeń.

✽

Nie wspomnieliśmy w tym artykule o ciekawej metodzie [7] zastosowanej do opisu składni języków PL-1 oraz ALGOL 60. Jej zaleta polega na tym, że aparat użyty do opisu składni jest pomocny również przy opisywaniu semantyki. Jest to więc jednolity system opisu składni i semantyki i omawianie w oderwaniu metody opisu samej składni nie wydaje się celowe.

Nie omawialiśmy również formalizmów [8], [9], które — jak się zdaje — zostały specjalnie utworzone w celu opisu problemu deklaracji w językach programowania; nie wiadomo jednak, czy nadają się one do opisu innych istotnych cech.

Wszystkie metody, o których była mowa, wzorują się w pewnym stopniu na gramatykach Chomsky'ego, przeznaczonych w założeniu do badania własności języków naturalnych. Z kolei osiągnięcia w dziedzinie formalnego opisu składni języków programowania znajdują zastosowanie w innych działach informatyki. Dlatego wydaje się celowe udostępnienie osiągniętych rezultatów szerszemu ogółowi informatyków.

BIBLIOGRAFIA

[1] Naur P. (ed.), Revised report on the algorithmic language ALGOL 60. Comm. ACM 6 1963, pp. 1—17

[2] Chomsky N., On certain formal properties of grammaire., Information and Control., v. 2. nr 2, 1969

[3] Salomaa Y., Probabilistic and weighted grammars., Information and Control., v. 15, nr 6, 1969

[4] Kuno S., Mathematical linguistics and automatic translation. Report nr NSF-22., Comput. Lab. of Harvard Univ. 1/68

[5] Van Wijngaarden A. (ed.), Report on the algorithmic language ALGOL 68., Numerische Mathematik B. 14, Heft 2, 1969

[6] Sintzoff M., Existence of van Wijngaarden's syntax for every recursively enumeralble set., Ann. Soc. Sc. de Bruxelles, 2, 1967

[7] Lucas P., Lauer P., Stigleitner H., Method and notation for the formal definition of programming languages. Technical Rep. TR 25.087, IBM Lab. Vienna 1968

[8] Greibach S.A., HOPOROFT J.E., Scattered context grammars. IFIP Congress, North Holland 1968

[9] Whitney G. E., The generation and recognition properties of table languages, IFIP Congress. North Holland 1968.

WITOLD CICHOMSKI
EUGENIUSZ DUDEK

681.325.5.21.021

Kalkulatory elektroniczne — budowa, działanie i zastosowanie

Opisano typową organizację wewnętrzną kalkulatorów elektronicznych oraz sposób wykonywania na nich operacji arytmetycznych, funkcyjnych, wektorowych i pomocniczych.

Konieczność przetwarzania dużej ilości danych w ograniczonym czasie przy jednoczesnym zapewnieniu dużej dokładności oraz poprawności wyników wymaga automatyzacji i mechanizacji prac obliczeniowych. Zagadnienie to można podzielić na trzy kierunki:

- tworzenie nowych metod przetwarzania danych,
- konstruowanie urządzeń do automatycznego przetwarzania danych,
- wdrażanie istniejących i opracowanych już metod i urządzeń do prowadzenia konkretnych prac.

Problem automatyzacji i mechanizacji przetwarzania danych rozwijał się historycznie w dwóch, niezależnych aż do ostatnich lat, kierunkach. Jeden (wcześniejszy), związany głównie z pracami administracyjnymi i prostymi obliczeniami arytmetycznymi, wykorzystywał początkowo mechaniczne, następnie elektromechaniczne i ostatnio urządzenia elektroniczne typu biurowego. Drugi, utworzony do wykonywania złożonych naukowo-technicznych obliczeń numerycznych, rozwinął się gwałtownie na bazie elektronicznych maszyn cyfrowych. Oczywiście stosowanie maszyn cyfrowych zapewnia dużo większą moc przetwarzania w porównaniu z mniejszymi urządzeniami liczącymi i mogłoby wydawać się, że powinno je wyeliminować z użycia. Stosowanie jednak maszyn cyfrowych wymaga rozwiązania szeregu dodatkowych problemów, jak zapewnienie ich konserwacji i obsługi przez odpowiednio wykwalifikowany personel, przygotowanie rozwiązywanego problemu w postaci dostępnej dla maszyny cyfrowej (zaprogramowanie zadania w odpowiednim języku). Poza tym duże są koszty maszyn i ich instalacji.

Wykonywanie drobnych prac obliczeniowych na bazie dużych ośrodków, utrzymujących zespoły wykwalifikowanego personelu, jest problematyczne, ponieważ wyniki potrzebne są „na bieżąco" (chociaż ostatnio i to uzyskiwane jest przez tzw. systemy timersharingu). Dla takich potrzeb konstruowane są specjalne urządzenia, które cechuje przede wszystkim prostota obsługi, duża niezawodność działania, standardowe zasilanie i niewielki pobór mocy (możliwość instalowania w dowolnych warunkach), odporność na wpływy zewnętrzne (temperatura, warunki klimatyczne), niewielkie rozmiary i dość duża szybkość pracy.

Największe korzyści przy spełnieniu tych warunków uzyskuje się stosując urządzenia elektroniczne budowane na elementach półprzewodnikowych i ferromagnetycznych. Urządzenia takie nazywane są arytmometrami lub kalkulatorami elektronicznymi bądź kalkulatorami. Organizacja wewnętrzna tych kalkulatorów różni się znacznie od dawnych modeli, uwzględniając większe możliwości operacyjne. Ogólnie można wyróżnić dwie zasadnicze możliwości, tj. automatyczne i programowe wykonywanie operacji.

Pierwsza właściwość jest charakterystyczna dla początkowych rozwiązań, które umożliwiały wykonanie jedynie prostych operacji arytmetycznych (dodawanie, odejmowanie, mnożenie, dzielenie) oraz przesłań między rejestrami. Wynikało to z tego, że układ sekwencyjny, sterujący automatycznym wykonaniem operacji (mnożenie realizowane jest przez ciąg dodawań i przesunięć, dzielenie przez ciąg odejmowań, porównań, dodawań i przesunięć) stawał się bardzo złożony i kłopotliwy w realizacji dla bardziej złożonych operacji (np. funkcje elementarne). Ponieważ jednak w szeregu zastosowań inżynieryjno-technicznych dysponowanie np. funkcjami elementarnymi jest bardzo istotne, do rozwiązania tego problemu wprowadzono techniki zapożyczone z maszyn cyfrowych (wykonywanie operacji w wyniku sterowania programowego z zachowaniem typowej struktury podprogramów). Programowe wykonanie operacji jest powszechnie stosowane w nowych rozwiązaniach kalkulatorów. Pozwala to na wprowadzenie szeregu operacji typowych dla maszyn cyfrowych i wykonywanych w porównywalnym czasie, przy zachowaniu pozostałych cech kalkulatorów (głównie prostoty obsługi, niewielkich wymiarów, standardowego zasilania).

Wśród kalkulatorów o programowym wykonywaniu operacji wyróżnić można dwa typy urządzeń:

- kalkulatory, które umożliwiają wykonywanie operacji jedynie z klawiatury (przez operatora),
- kalkulatory wykonujące ponadto automatycznie sekwencje operacji wprowadzanych do kalkulatora z nośników pośrednich (najczęściej z kart magnetycznych). Kalkulator ma wówczas także możliwość wyprowadzania ciągu operacji z klawiatury na kartę magnetyczną. Umożliwia to opracowanie i wykonanie typowych programów obliczeń, często dosyć złożonych, bez konieczności wprowadzania poszczególnych operacji. Dzięki temu kalkulator może pracować z szybkością porównywalną z szybkością maszyny cyfrowej. Powszechnie uważa się, że rozwiązanie takie (kalkulatory tego typu są produkowane przez kilka firm) zastępuje małe maszyny cyfrowe.

Organizacja wewnętrzna kalkulatorów

Typowy schemat blokowy kalkulatora elektronicznego pokazany jest na rys. 1.

Każda operacja wykonywana z klawiatury realizowana jest przez szereg operacji elementarnych urządzenia, sterowanych przez odpowiednie układy logiczne. Klawiatura schematycznie oznaczona jest przez *WPR* (układy wprowadzania) i obejmuje zarówno klawisze cyfrowe, jak i operacyjne. Klawiatura współpracuje z układami sterowania *US*, zadaniem których jest zidentyfikowanie wprowadzanych cyfr i zainicjowanie odpowiednich działań oraz rozpoczęcie odpowiedniej sekwencji operacji elementarnych (w przypadku naciśnięcia klawisza operacyjnego). Sekwencje operacji podstawowych realizowane są przez układy *P* i *UA* przy współpracy z układami sterowania *US*. Realizacja tych sekwencji zależna jest od typu sterowania (automatyczne czy programowe wykonywanie operacji). Podczas automatycznego wykonywania operacji naciśnięcie klawisza operacyjnego spowodować musi ustawienie układu *P* w stanie początkowym dla danej operacji. Odbywa się to przez odpowiednie układy kodujące bloku *US* i bloku *UA*. Dalsze operacje elementarne są już automatycznie wytwarzane w pętli *P*, *US*, *UA*, aż do zakończenia sekwencji operacji podstawowych. W przypadku pro-

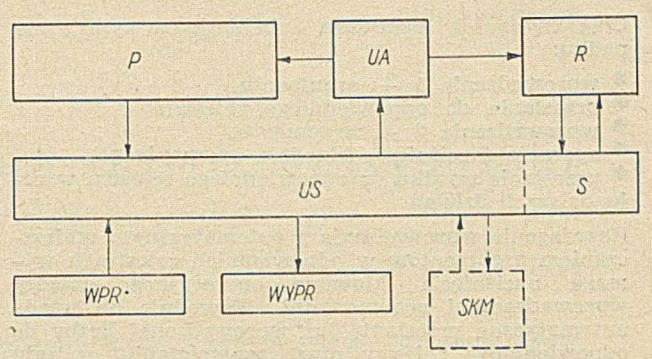
gramowego wykonywania operacji z klawiatury każda z nich wykonywana jest przez odpowiedni program, umieszczony w pamięci *P*. Pamięć ta musi spełniać szczególne wymagania, jest pamięcią bierną (tylko odczytywaną). Wykonanie takiej pamięci jest znacznie łatwiejsze technologicznie i bardziej proste konstrukcyjnie od pamięci aktywnej (odczytywanej i zapisywanej). Dlatego możliwe jest zrealizowanie takiej pamięci (nawet o dosyć dużej pojemności) w urządzeniu typu biurowego. Współpraca z taką pamięcią (nazywaną pamięcią stałą) polega na wyznaczeniu adresu odczytywanego miejsca pamięci oraz wykonywania cyklu odczytu pamięci. Zadania te spełniają układy *US* i *UA*. Naciśnięcie klawisza operacyjnego powoduje wprowadzenie przez układy kodujące bloku *US* do układów adresowania *UA* adresu początkowego programu tej operacji. Podczas cyklu odczytywania pamięci przesyłana jest do układów sterowania *US* pierwsza operacja elementarna programu, która w pętli *US*, *UA* wyznacza adres następnej operacji, a jednocześnie steruje pozostałymi układami w celu wykonania koniecznych działań. Zakończenie programu sekwencji operacji elementarnych) równoznaczne jest z wykonaniem operacji i umożliwia wprowadzanie dalszych informacji.

Dane początkowe, wyniki pośrednie i końcowe przechowywane są w bloku rejestrów *R*. Blok ten zazwyczaj obejmuje kilka do kilkunastu rejestrów (rejestry operacyjne) o długości zależnej od dokładności urządzenia (zasadniczo kilkanaście cyfr dziesiętnych). Operacje wykonywane na zawartościach tych rejestrów przeprowadzane są w układach sumatora *S*, a wyniki zapamiętywane w rejestrach *R*. Ze względu na dziesiętny zapis informacji zewnętrznej zasadniczo stosuje się dziesiętno-dwójkowe kodowanie informacji. Przetwarzanie słów (liczb) wykonywane jest wtedy szeregowo lub szeregowo-równolegle (tetradami), zależnie od konstrukcyjnych rozwiązań układów.

Wyniki obliczeń są zobrazowywane przez wskaźniki świetlne (neonowe lampy cyfrowe bądź specjalne lampy oscyloskopowe). Można wtedy jednocześnie wyświetlać zawartość kilku rejestrów operacyjnych. Często istnieje możliwość dołączania urządzeń drukujących do trwałej rejestracji wyników lub też opracowane urządzenia wykonuje się w kilku wariantach z wyświetlaniem lub drukowaniem wyników. Układy wyświetlania schematycznie zaznaczono przez blok *WYPR*.

Na schemacie pokazano również możliwość dołączenia układu wprowadzania informacji z nośników pośrednich *SKM*, z możliwością jej przesyłania w dwu kierunkach i wprowadzania do kalkulatora oraz wyprowadzania na nośnik pośredni. Podczas automatycznego wykonywania ciągów operacji wprowadzanych przez *SKM*, informacja o kolejnych operacjach podawana jest w *SKM* początkowo do bloku rejestrów *R* (gdyż jest to jedyny układ pamięciowy o możliwości zapisywania informacji), a następnie pobierana, analizowana i wykonywana w układach *US*, *UA* i *P*. Blok rejestrów jest wtedy odpowiednio rozbudowany. Wszystkie układy logiczne (oprócz bloku *WYPR*), wykonywane są na elementach półprzewodnikowych i magnetycznych. Bloki *US* i *UA* wykonywane są więc w technice tranzystorowo-diodowej oraz tranzystorowo-ferromagnetycznej. Blok *R* ma zasadniczo postać pamięci na rdzeniach magnetycznych o organizacji zależnej od przyjętej realizacji przetwarzania (szeregowa lub szeregowo-równoległa). Natomiast blok pamięci stałej *P* wykonywany jest jako tak zwana pamięć „zaszyta" na rdzeniach magnetycznych bądź na cienkich warstwach magnetycznych z odpowiednimi układami wybierania.

W kalkulatorach wykorzystuje się zarówno zapis stałoprzecinkowy, jak i zmiennoprzecinkowy. Położenie przecinka dziesiętnego ustawiane jest automatycznie. W zapisie stałoprzecinkowym (zasadniczo na zewnątrz odpowiednie przełączniki na klawiaturze) można ustalać liczbę miejsc za przecinkiem. Dokładność wyników oraz zakres wyników pośrednich i końcowych wymaga reprezentowania kilkunastu cyfr



Rys. 1. Schemat blokowy kalkulatora elektronicznego

dziesiętnych (12 do 16). W niektórych rozwiązaniach zakres ten jest rozszerzany (aż do 24 cyfr dziesiętnych) lub zawężany (do 8 cyfr). Przekroczenie zakresu (dla ustalonej liczby cyfr za przecinkiem) sygnalizowane jest przez specjalny świetlny wskaźnik nadmiaru. Najczęściej powoduje to przerwanie dalszego działania urządzenia.

W zapisie zmiennoprzecinkowym każda liczba przedstawiona jest jako para liczb (mantysa oraz wykładnik dziesiętny). Mantysa zawsze jest z zakresu 0,1 do 1,0 lub 1,0 do 10,0, a wykładnik wskazuje rzeczywiste położenie przecinka (na prawo lub na lewo) od tak znormalizowanej mantysy.

Ponieważ wykładnik zasadniczo ma zakres rzędu od —100 do +100, uzyskuje się zakres dynamiczny liczb rzędu 10^{200}. Przekracza to zakresy dynamiczne najczęściej spotykanych maszyn cyfrowych i umożliwia obliczenia zarówno na liczbach bardzo dużych, jak i bardzo małych. Ponieważ odczytywanie zapisu zmiennoprzecinkowego jest trochę kłopotliwe, najczęściej istnieje możliwość automatycznego przekształcania wyników na postać stałoprzecinkową, dla której można ustalać liczbę cyfr za przecinkiem. Jeżeli liczba zmiennoprzecinkowa przekracza zakres ustalonej postaci stałoprzecinkowej, automatycznie wyświetlana jest w postaci zmiennoprzecinkowej. Podczas zapisu zmiennoprzecinkowego liczba cyfr mantysy wynika jedynie z koniecznej dokładności obliczeń i wynosi 9 lub 10 cyfr dziesiętnych. Zakres liczb zmiennoprzecinkowych i efektywna dokładność obliczeń jest jednak znacznie większa niż w tym samym co do długości słowa zapisie stałoprzecinkowym. Wyniki obliczeń, doprowadzane do postaci ustalonej zewnętrznie, są automatycznie zaokrąglane w celu zwiększenia dokładności. W niektórych rozwiązaniach, przeznaczonych do specjalnych zastosowań, istnieje ponadto możliwość wprowadzania automatycznego pewnych stałych.

Operacje arytmetyczne

Należą do nich działania podstawowe (dodawanie, odejmowanie, mnożenie i dzielenie) oraz wykonywane na niektórych urządzeniach operacje obliczania procentów (wariant mnożenia) i obliczania arytmetycznego pierwiastka kwadratowego. Argumenty operacji (wszystkie te operacje oprócz pierwiastkowania są dwuargumentowe) wprowadzane są do ustalonych dwóch rejestrów operacyjnych. Wynik zawsze umieszczany jest w ustalonym rejestrze, którego zawartość jest automatycznie wyświetlana.

Umieszczanie argumentów operacji w odpowiednich rejestrach może być automatyczne lub ręczne. Podczas ręcznego wprowadzania liczby do ustalonego rejestru (na ogół tego samego, z którego wyświetlane są wyniki) przesyłana jest ona do wybranego rejestru po naciśnięciu odpowiedniego klawisza sterującego, umieszczonego na pulpicie. Podobnie przesyła się wyniki do rejestrów pamięci, które nie są wykorzystywane podczas wykonywania działań i przeznaczone są jako pamięć przejściowa.

Wprowadzenie drugiego argumentu do rejestru wprowadzania i naciśnięcie klawisza operacji powoduje wykonanie operacji na wprowadzonych argumentach.

Ciąg czynności operatora obejmuje w tym przypadku:

- wprowadzenie 1 — argumentu,
- przesłanie do odpowiedniego rejestru,
- wprowadzenie 2 — argumentu,
- operacja (naciśnięcie klawisza operacyjnego),
- przesłanie wyniku do odpowiedniego rejestru w celu dalszych działań.

Rozwiązanie wprowadzania z automatycznym umieszczaniem argumentów w odpowiednich rejestrach wymaga naciśnięcia klawisza operacyjnego między wprowadzanymi argumentami. Powoduje on wtedy automatyczne przesłanie już wprowadzonej liczby do odpowiedniego rejestru oraz „zapamiętanie" rodzaju operacji. Sama operacja natomiast jest wykonywana po wprowadzeniu drugiego argumentu i naciśnięciu specjalnego klawisza (najczęściej znak równości). Ciąg czynności operatora ma wówczas postać:

- wprowadzenie 1 — argumentu,
- wprowadzenie operacji (ale nie wykonanie),
- wprowadzenie 2 — argumentu,
- wprowadzenie znaku równości (wykonanie operacji).

Na szczególną uwagę zasługuje rozwiązanie zastosowane np. w kalkulatorach typu TMK (prod. polskiej), gdzie dzięki specjalnym układom sekwencyjnym możliwe jest wykonywanie ciągów operacji kolejno następujących po sobie w sposób analogiczny do zapisu tych operacji, co znacznie ułatwia obsługę urządzenia; np. ciąg obliczeń:

$// \; a+b \; / \cdot c - d \; / : e$

realizowany jest w wyniku wykonania takich operacji, jak:

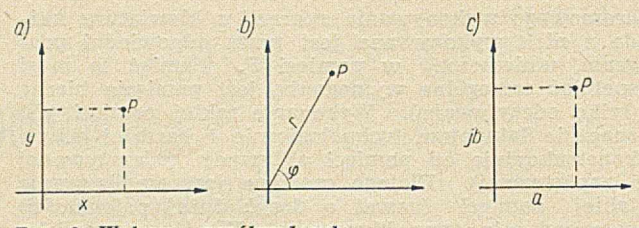
- wprowadzenie a,
- operator dodawania,
- wprowadzenie b,
- operator mnożenia (wykonanie dodawania),
- wprowadzenie c,
- operator odejmowania (wykonanie mnożenia),
- wprowadzenie d,
- operator dzielenia (wykonanie odejmowania),
- wprowadzenie e,
- operator znaku równości; powoduje on wyświetlenie wyniku po zakończeniu obliczenia.

Operacje funkcyjne

Należą do nich funkcje trygonometryczne (sin, cos, tg), odwrotne funkcje trygonometryczne (arcsin, arccos, arctg), funkcje hiperboliczne (sinh, cosh, tgh), odwrotne funkcje hiperboliczne (arsinh, arcosh, artgh), funkcje logarytmiczne i wykładnicze (lg, ln, exp), funkcje pierwiastkowe (pierwiastki dowolnych stopni całkowitych).

Funkcje te, oprócz ostatniej, są jednoargumentowe i wykonywane na zawartości ustalonego rejestru operacyjnego (najczęściej rejestru wprowadzania). Nie we wszystkich urządzeniach realizowany jest pełny zestaw tych funkcji, gdyż są one powiązane wzajemnie ustalonymi relacjami, a zbiór operacji dostępny bezpośrednio z klawiatury uzależniony jest przede wszystkim od przeznaczenia urządzenia. Przy operacjach trygonometrycznych i odwrotnych do trygonometrycznych istnieje możliwość wykonywania obliczeń dla wartości wyrażonych w stopniach lub radianach. Wtedy wybór realizacji także zależny jest od zastosowań. Niektóre urządzenia mają możliwość operowania obydwoma wielkościami (stopniami i radianami), wybieranymi przez operatora za pomocą odpowiedniego przełącznika umieszczonego na pulpicie urządzenia. Możliwość wyznaczania funkcji trygonometrycznych z dowolnych wartości kątów i obliczania odwrotnych funkcji trygonometrycznych pozwala na określenie wartości kątów, co jest szczególnie wygodne podczas obliczeń wykonywanych w radianach.

Operacje logarytmiczne i wykładnicze pozwalają ponadto na obliczanie dowolnych potęg dowolnych wartości, oczywiście mających sens arytmetycznie.



Rys. 2. Wykresy współrzędnych

Operacje wektorowe

Są one przewidziane do wykonywania obliczeń zasadniczo w przestrzeni dwuwymiarowej (na płaszczyźnie) oraz do operowania liczbami zespolonymi. Operacje te wykonywane są równocześnie na argumentach dwuskładnikowych, np. dla operacji jednoargumentowych na zawartościach dwóch rejestrów, a dla operacji dwuargumentowych na zawartościach czterech rejestrów.

Każdy punkt w przestrzeni dwuwymiarowej jest najczęściej współrzędnymi prostokątnymi x i y (rys. 2 *a*) lub współrzędnymi biegunowymi r i φ (rys. 2 *b*). Współrzędne te potrzebne są do wykonywania różnych operacji na tych samych danych. Pierwszym problemem jest więc konwersja między współrzędnymi prostokątnymi i biegunowymi w obydwie strony. Realizowanie konwersji według bezpośrednich zależności:

$$x = r\cos\varphi \qquad r = \sqrt{x^2+y^2}$$

$$y = r\sin\varphi \qquad y = \operatorname{arc\,tg}\frac{y}{x}$$

jest kłopotliwe i czasochłonne. Dlatego wprowadza się konwersje jako operacje automatycznie wykonywane z klawiatury. Posiadanie automatycznych konwersji jest szczególnie korzystne podczas obliczeń wykonywanych na liczbach zespolonych. Wtedy współrzędne prostokątne przypisuje się odpowiednio części rzeczywistej i urojonej liczby (rys. 2 *c*), a współrzędne biegunowe odpowiednio modułowi i argumentowi liczby zespolonej. Dodawanie i odejmowanie liczb zespolonych wykonywane jest we współrzędnych prostokątnych, a mnożenie i dzielenie — we współrzędnych biegunowych. Podczas wykonywania obliczeń konieczna jest więc częsta konwersja współrzędnych.

Ponadto dodawanie i odejmowanie wykonywane jest na obydwu składowych (części rzeczywistej i urojonej) automatycznie. Podczas mnożenia i dzielenia operacje na modułach (mnożenie lub dzielenie) i argumentach (dodawanie lub odejmowanie) są inne i na ogół, wykonywane w wyniku dodatkowych operacji (ln oraz exp), wprowadzane ręcznie.

Możliwość automatycznego dodawania i odejmowania argumentów dwuskładnikowcyh pozwala ponadto na wykonywanie innych działań na wektorach dwuwymiarowych oraz na działania akumulacyjne (np. obliczanie iloczynu skalarnego wektorów wielowymiarowych, wyznaczanie wyznaczników macierzy itd.). Możliwość operowania na argumentach dwuskładnikowych wiąże się na ogół z potrzebą wyświetlania obydwu składników. Do tego celu wykorzystywane są zasadniczo specjalne lampy oscyloskopowe.

Operacje pomocnicze

Oprócz wyżej wymienionych, do dyspozycji operatora są zarówno operacje funkcyjne, jak i sterujące. Z operacji funkcyjnych można wymienić obliczanie wartości bezwzględnej liczby w ustalonym rejestrze, obliczanie części całkowitej liczb, zliczanie wykonywanych operacji bądź przeliczanych pozycji (do zastosowań administracyjnych). Pomocnicze operacje sterujące obejmują zerowanie rejestrów, przesyłania między rejestrami, wprowadzanie stałych współczynników do ustalonych rejestrów, pewne operacje na rejestrach pamięci (na ogół tylko dodawanie i odejmowanie).

S. KABLESZKOW
Sofia
Bułgaria

681.32206:795.41

Pewien program do rozwiązywania problemów brydżowych

Podano zarys metody i opisano program rozwiązujący problemy brydżowe przy użyciu języka symulującego SIMBOL. Program został uruchomiony na maszynie ATLAS w Manchester University.

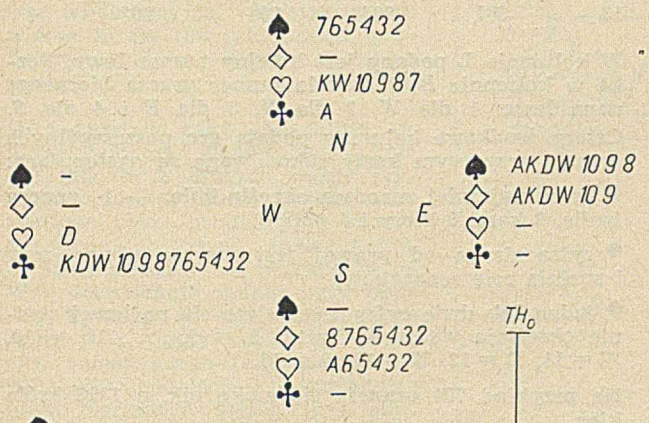
Problemy brydżowe, o których będzie mowa, są pewnym uproszczeniem rzeczywistej gry, ponieważ podaje się w nich rozkład wszystkich czterech rąk. W problemach takich przyjmuje się zwykle, że rozgrywającym jest gracz *S* (rys. 1). Musi on wziąć pewną ilość lew po zadanym pierwszym wyjściu gracza *W*; *N* jest dziadkiem.

Jako założenia początkowe problemu podaje się zatem: układ czterech rąk, kolor atutowy (względnie bez atu), ilość lew, jaką należy wziąć oraz pierwszy wist. Często układ kart w problemie odpowiada sytuacji, jaka powstaje po zagraniu kilku pierwszych lew. Wówczas pierwsze wyjście może należeć do dowolnego gracza.

Problemy tego rodzaju rozwiązuje się w celu poznania specjalnych sposobów rozgrywki.

Zarys metody

Wszystkie możliwości przebiegu rozgrywki można dobrze przedstawić za pomocą drzewa pokazanego na rys. 2.



Rys. 1. SN grają siedem kierów (wielki szlem)

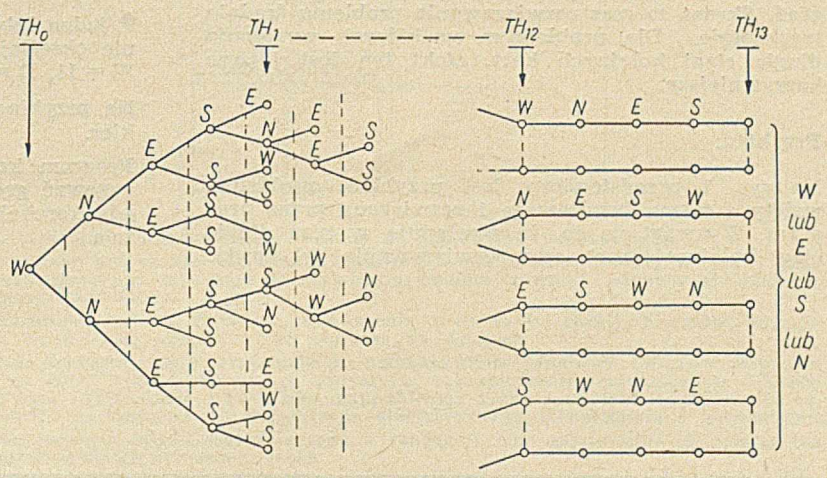
Każdy wierzchołek reprezentuje gracza, na którego w danym momencie przypada kolej zagrania karty, a krawędzie wychodzące z tego wierzchołka reprezentują możliwe zagrania tego gracza. L_i określa, kto wziął lewę w *i* okrążeniu, z tym, że L_0 oznacza kto wistuje (np. jeśli trzecią lewę bierze *N*, to $L_3 = N$). Jeśli rozpatrujemy wierzchołek reprezentujący gracza, który wziął lewę w *i* okrążeniu, to z wierzchołka tego wychodzi 13-*i* krawędzi. W zależności od układu liczba ta, dla pozostałych wierzchołków odpowiadających *i* lewie, waha się od 1 do 13. Maksymalna ilość dróg w takim drzewie wynosi $(13)^4$. W rzeczywistej grze, zależnie od doświadczenia graczy, mniejsza lub większa ilość możliwych zagrań jest eliminowana niejako podświadomie. Można rozróżnić kilka rodzajów tej eliminacji.

Pierwszy z nich zachodzi wtedy, gdy gracz ma kilka kolejnych kart w jednym kolorze. Jest wtedy obojętne, w którą z nich zagra, rozpatruje więc tylko sens zagrania w jedną z nich.

Drugi ma miejsce, gdy utożsamiamy ze sobą pewną ilość nie różniących się istotnie dróg w drzewie rozgrywki. Prosty przykład: aby wziąć wymaganą ilość lew *NS* muszą uzyskać trzy lewy w atu, mając A K D. Gracz *E* ma w tym kolorze W 7 2. Kolejność, w jakiej *E* będzie dorzucał swoje karty do zagranych A, K i D nie ma znaczenia, można więc utożsamić ze sobą w tym przypadku sześć różnych dróg.

Trzeci rodzaj eliminacji polega na tym, że gracz nie mając możliwości wzięcia aktualnie rozgrywanej lewy dorzuca do niej najniższą kartę. Na przykład: *W* wistuje w asa pik, *E* ma w pikach K 2. *E* najczęściej dorzuca 2.

Takie sposoby redukowania liczby rozpatrywanych wariantów stosuje się w czasie rozgrywki podświado-



Rys. 2. Wszystkie możliwości przebiegu rozgrywki przedstawione za pomoca drzewa

mie. Podczas nauki uczymy się ich świadomie, aby potem stosować je mechanicznie.

Z drugiej strony rutynowany gracz zawsze zwraca uwagę na sytuacje tego rodzaju. W powyższym przykładzie, przy grze w bez atu w asa atakuje się z drugiego, silnego koloru. Dobry gracz na miejscu *E* powinien wtedy dodać króla.

Program rozwiązujący problemy brydżowe napisano przy użyciu języka SIMBOL. Jest to język symulacyjny, wprowadzony przez ICL.

Program został uruchomiony na maszynie ATLAS w Manchester University. Po translacji zajmuje on 22 bloki pamięci (1 blok = 512 czterdziestoośmiobitowych słów) i jest przechowywany na taśmie magnetycznej. W zależności od rozkładu kart, dany problem jest rozwiązywany w czasie 1—15 minut, po czym, w formie łatwej do zrozumienia, drukowany jest zbiór możliwych rozwiązań.

Metoda użyta w programie opiera się na tzw. korekcji retrospektywnej. Program rozpoczyna szukanie rozwiązania próbując pierwszą możliwą drogę w drzewie reprezentującym wszystkie rozgrywki.

Jeśli w pewnym momencie liczba lew wziętych przez *WE* jest większa niż pozwalają na to warunki początkowe, znaczy to, że obrany sposób gry jest zły i program zaczyna korygować dotychczasową rozgrywkę. Pierwszym zmienianym jest ostatnie zagranie gracza pary *NS*. Karta, w którą zagrał poprzednio, jest odpowiednio markowana jako zła, a zostaje zagrana następna możliwa. Jeśli zmiana nie przyniesie pozytywnego rezultatu, to znaczy para *WE* w dalszym ciągu wygrywa wpadkową lewę, program ponawia próbę zagrania innej karty. Gdy wszystkie karty rozpatrywanego gracza są już zamarkowane jako złe, znaczy to, że nie może on wziąć tej lewy i program powtarza tą samą procedurę dla drugiego gracza pary *NS*.

Jeśli w końcu okazuje się, że żaden z nich nie może wziąć tej lewy, to rozgrywkę cofa się o jedna rundę, powtarzając znowu to samo postępowanie. Gra jest cofana tak długo, aż program znajdzie sposób rozgrywki, pozwalający nie oddać wpadkowej lewy.

Tak znaleziona rozgrywka nie jest jeszcze prawidłowym rozwiązaniem. Jest ona tylko ciągiem zagrań dającym wygraną przy danej grze pary *WE*. Z uwagi na to, że podczas korekcji gry pary *NS* gra *WE* nie była zmieniana, program zaczyna z kolei korygować ich grę, postępując tak samo jak przy poprawianiu gry *NS*. Skoro tylko zostanie znalezione zagranie, dzięki któremu *WE* biorą lewę (tzn. obkładają grę bez jednej), program powraca do korygowania gry *NS*. Postępując w ten sposób program nie może pominąć prawidłowego rozwiązania problemu. Przyjęta metoda skraca czas potrzebny na znalezienie rozwiązania w porównaniu z czasem potrzebnym na badanie wszelkich możliwych sposób rozgrywki. Omawiany program realizuje ponadto dwa pierwsze z omawianych powyżej sposobów eliminowania pewnych zagrań. Skraca to czas rozwiązywania problemu średnio trzykrotnie. Dla problemów, w których występują długie ciągi kolejnych kart, efekt ten jest jeszcze korzystniejszy.

Przykład:

Na rys. 1 przedstawiony jest przykład problemu, wzięty z pisma brydżowego i rozwiązany przez program. Z uwagi na to, że występują w nim długie ciągi kolejnych kart, czas rozwiązywania był bardzo krótki — około 1 min. i 20 s. *NS* grają siedem karo. *W* wistuje w 2 trefle.

Łatwo jest wziąć dwanaście z trzynastu wymaganych lew — jedenaście lew na kara i dwunastą na asa treflowego. Trzynastą można wyrobić w kierach lub pikach. Nie ma trudności, jeśli do asa trefl *E* dołoży kiera, gdyż wtedy można wyrobić przebitkami blotkę w tym kolorze. Pamiętać jedynie należy, aby pierwszą lewę karową wziął *S*. Jeśli *E* dołoży pika, *N* powinien zagrać w drugiej lewie króla karo, stawiając gracza *E* w sytuacji przymusowej. Gdy *E* do kara dołoży pika, *S* dokłada blotkę, a następnie wyrabia przebitkami blotkę pikową. Jeśli natomiast *E* dorzuci kiera, to króla karo należy zabić asem i wyrobić blotkę kierową. W tabelce przedstawiono rozwiązania tego problemu znalezione przez maszynę. Wybrano tylko dwa, istotnie różniące się warianty.

Wariant I

L	*W(L)*	*N(L)*	*E(L)*	*S(L)*	*B(L)*
1	201	1401	1404	803	2
2	1202	1302	1304	602	2
3	1301	704	1204	1402	4
4	1201	1102	1403	703	2
5	1101	604	1104	502	4
6	1001	1002	1303	603	2
7	901	504	1004	402	4
8	801	902	1203	503	2
9	701	404	904	302	4
10	601	802	1103	403	2
11	501	304	804	202	4
12	401	702	1003	303	2
13	301	204	903	203	2

Wariant II

1	201	1401	1404	803	2
2	1202	1302	1403	1402	4
3	1301	1102	1303	703	2
4	1201	704	1304	602	4
5	1101	1002	1203	603	2
6	1001	604	1204	502	4
7	901	902	1103	503	2
8	801	504	1104	402	4
9	701	802	1003	403	2
10	601	404	1004	302	4
11	501	702	903	303	2
12	401	304	904	202	4
13	301	204	804	203	4

W kolumnie *L* podany jest kolejny numer lewy, liczba w kolumnie *B(L)* określa numer gracza biorącego daną lewę: 1 dla *W*, 2 dla *N*, 3 dla *E* i 4 dla *S*. Cztery środkowe kolumny podają grę poszczególnych graczy, przy czym karty zakodowane są następująco:

- cyfra najmniej znacząca określa kolor — 1 oznacza trefle, 2 kara, 3 kiery i 4 piki,
- cyfra druga od prawej jest zawsze równa zeru i spełnia rolę separatora
- jedna lub dwie cyfry znajdujące się po lewej stronie określają wysokość karty: 2 = 2, ..., 10 = 10, W = 11, D = 12, K = 13, A = 14.

Na przykład 704 oznacza siódemkę pik, a 1303 króla kier.

Program kończy swoje działanie, gdy usiłując skorygować grę musi cofnąć się do rundy zerowej lub gdy korekcja wymaga zmiany karty pierwszego wyjścia.

Opracował M. GŁOWACKI

TELEDACJA W PRZEMYŚLE OBUWNICZYM[1]

W ostatnich latach wiele wysiłku poświęca się udoskonalaniu metod przesyłania informacji z miejsca ich powstawania do centrum obliczeniowego celem dalszego ich opracowania przez komputer. Metody stosowane dotąd niejednokrotnie nie gwarantują szybkiego uzyskania wyników przetwarzania, będących bazą do podejmowania prawidłowych decyzji. W wielu przypadkach czas staje się elementem krytycznym, szczególnie w warunkach walki konkurencyjnej, z jaką muszą się liczyć producenci.

Wyrazem dążenia do jak najściślejszego powiązania użytkowników systemu APD z informacjami w sferze produkcji niech będzie przykład zastosowania komputera GE-120, współpracującego z siecią teledatorów we francuskich zakładach obuwia damskiego CHARLES JOURDAN. Dystrybucją wyrobów tej firmy zarówno na terenie Francji, jak i całego świata zajmują się Charles Jourdan, Christian Dior, Seducta. Dzienna produkcja jednego z kompleksu zakładów, znajdującego się w Roman, wynosi ponad 4000 par. Ponadto w ramach tego kompleksu produkcją obuwia zajmują się zakłady SORIC w Turnon oraz DANAUD w Annonay. Roczna produkcja tych przedsiębiorstw wyraża się liczbą 1 250 000 par obuwia.

Charakterystyczne jest to, że nie jest to produkcja masowa czy nawet wielkoseryjna. Występuje tu wielka liczba różnych modeli, produkowanych w bardzo krótkich seriach.

Dwa razy w roku prezentuje się nową kolekcję zawierającą około 600 modeli, z których każdy wykonany jest w dużej liczbie wariantów. Seria produkcyjna każdego wariantu wynosi maksymalnie 12 par i z reguły nie przekracza 10. Wyprodukowanie jednej pary obuwia wymaga aż 150 różnorodnych operacji (z reguły około 100 operacji), wobec czego przy wymienionych 600 modelach z uwzględnieniem wariantów liczba tych operacji wzrasta do rzędu 500 000 i wymaga codziennego wystawienia około 40 000 kart pracy.

Sytuacja ta już stwarzała sporo problemów natury organizacyjnej, a jeśli weźmiemy pod uwagę szereg innych zagadnień, występujących w przedsiębiorstwie CHARLES JOURDAN, jak:

- rejestracje i przekazywanie otrzymanych zamówień w ciągu dnia,
- przygotowywanie i ciągła kontrola wyrobów w procesie produkcji w zakresie każdego zamówionego modelu przez poszczególnych klientów,
- określenie zarobku dla 800 pracowników za wykonane operacje produkcyjne, tj. obliczenie około 40 000 kart pracy dziennie,
- umożliwienie bezpośredniej informacji klientów o terminach realizacji ich zamówień,
- uzyskiwanie bieżących danych statystycznych (stany zapasów, rozliczenie z klientami, ilości wyprodukowanych par obuwia w poszczególnych modelach itp.).

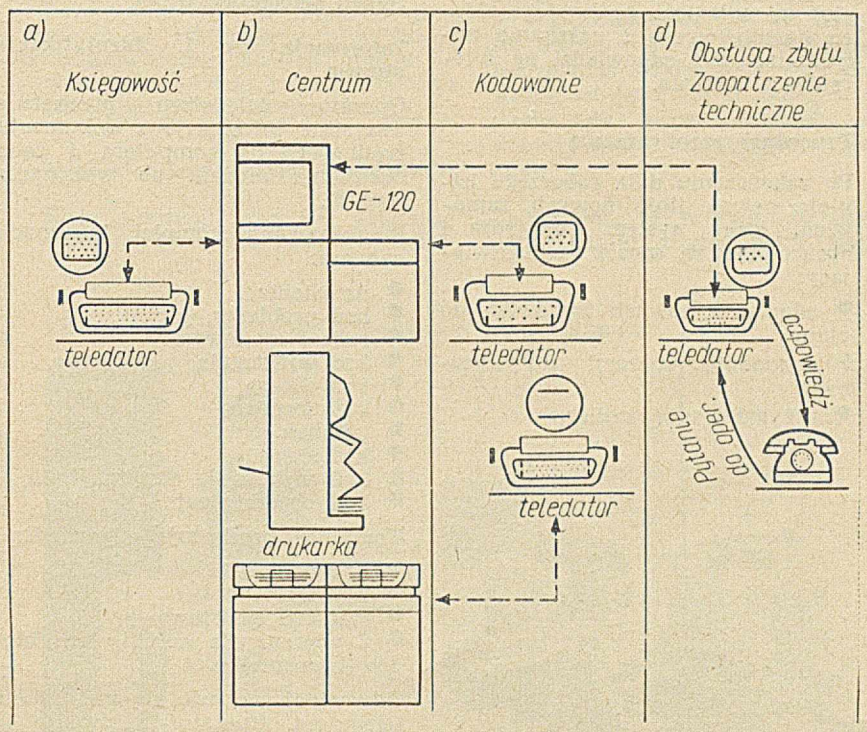
W tej sytuacji łatwo zrozumiemy, że należało znaleźć rozwiązanie gwarantujące szybką i pewną realizację powyższych funkcji. Zbudowano zatem system APD działający na bieżąco (real time), wykorzystując zalety, jakimi odznacza się współpraca komputera z teledatorami.

Schemat na rys. 1 ukazuje organizację tego systemu i drogi przepływu informacji.

Sprzęt i podstawowe zbiory informacji

Opisywany system realizuje się na komputerze GE-120, którego pamięć operacyjna posiada pojemność 24 K; w zestaw komputera wchodzą: dwie jednostki dyskowe, czytnik dokumentów, drukarka wierszowa oraz teledatory (rys. 2). Na dyskach magnetycznych utrzymuje się podstawowe zbiory kartotekowe zawierające:

- katalog wyrobów,
- rejestr zamówień,
- plany produkcji i realizacja zleceń,



Rys. 1. System bieżącego informowania na stanowisku teledacyjnym

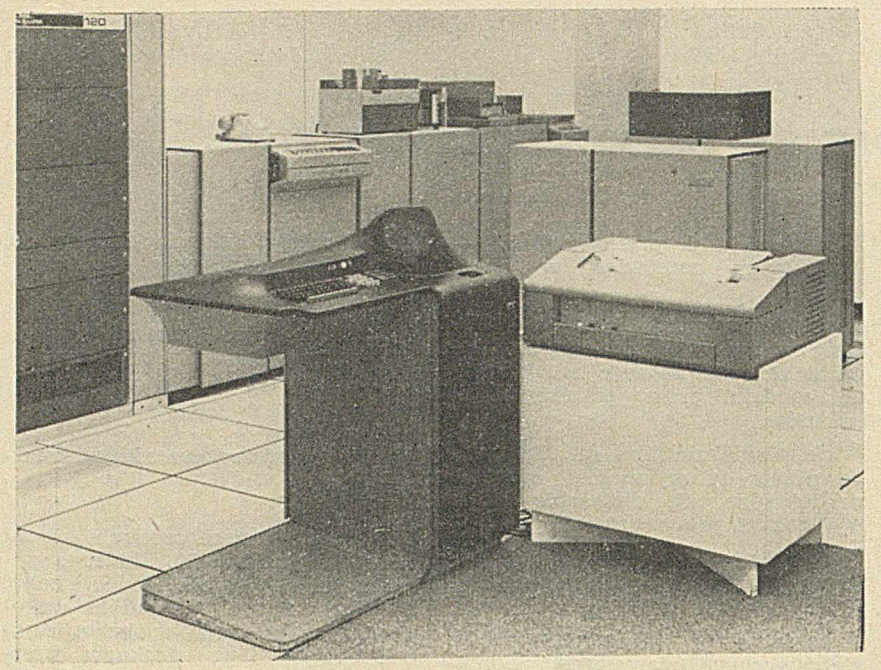
a) operator przekazuje rozliczenia i rachunki wystawione w danym dniu i dane inwentaryzacyjne w czasie rzeczywistym; udziela również odpowiedzi odnośnie do stanu rachunków

b) kartoteki: katalog wyrobów, zamówienia, planowania, różnic, klientów, dostawców, księgowości ogólnej — są zapisane na dyskach

c) oba teledatory są przeznaczone do przekazywania zamówień. Wartość zamówienia na dzień bieżący łącznie ze statystyką, zabezpieczeniem technicznym i materiałowym. Teledatory 1-2-3-4 obsługują również różne kartoteki odnośnie do odpowiedzi na pytania o charakterze statystycznym, technicznym i zaopatrzenia

d) telefoniczne odpowiedzi klientowi o realizacji jego zamówienia w całości lub w poszczególnych numerach zleceń

[1]) Opracowano na podstawie *La Télé — Gestion an Service de l'Industrie de la Chaussure (Bull-GE).*



Rys. 2. Ogólny wygląd zestawu GE-120 wraz z teledatorem w postaci ekranu i drukarki (z prawej)

- kartotekę klientów,
- kartotekę dostawców,
- księgowość,
- inne (płace, tabele podatkowe, katalog oprzyrządowania itp.).

Cztery teledatory umożliwiają dostęp na odległość do każdego zbioru kartotekowego i uzyskanie natychmiastowej odpowiedzi na interesujące pytania.

Procedury przetwarzania

Po zakończeniu dnia roboczego istnieje pewna ilość nowych zamówień, które system rejestruje i opracowuje w sposób umożliwiający:

- włączenie nowych zamówień do planu,
- dokonanie operacji statystycznych,
- przygotowanie rozliczeń.

Rys. 3. Współpraca z komputerem przez teledator

Uruchomienie nowego zamówienia inicjuje się wypełnieniem specjalnego dokumentu źródłowego, zawierającego podstawowe dane o kliencie oraz charakterystykę zamawianego obuwia. Następnie dalsze operacje podejmuje system w relacji teledator/GE-120.

Informacja nr 1. Identyfikacja obuwia

Operator teledatora przekazuje wszystkie informacje z dokumentu źródłowego do komputera i żąda określenia modelu do wykonania (rys. 3).

Przekazywany komplet informacji zawiera:

- nr klienta,
- kod problemu,
- rodzaj opakowania,
- kod wykonania,
- kod obcasa,
- kod formy,
- gatunek,
- kod skóry,
- kod wykonania wnętrza buta,
- kod ornamentacji.

Komputer przekazuje odpowiedź:

- nazwa modelu,
- nr formy,
- materiał podstawowy,
- 2 wiersze dla korekty ew. błędów transmisji.

Odpowiedź ukazuje się na ekranie teledatora. Operator kontroluje ją i zatwierdza. W ten sposób następuje zarejestrowanie zamówienia.

Informacja nr 2. Identyfikacja klienta i wielkości zamówienia

Etap ten precyzuje zamówienie pod względem ilościowym w ramach modelu, z uwzględnieniem wszystkich klientów zamawiających ten model.

Operator teledatora przekazuje następujące informacje:

- ilości według rozmiarów,
- termin,
- tęgość obuwia,
- kod klienta,
- ilość całkowita zamówienia.

Po kontroli otrzymanych informacji GE-120 rozlicza zamówienia na warianty (maks. 12 par) i nadaje numery zleceń dla każdego wariantu oraz współpracując ze wszystkimi kartotekami przekazuje jednocześnie odpowiedź zawierającą poniższe dane:

- nazwa klienta,
- cena jednostkowa,
- numer zlecenia pierwszego i ostatniego wariantu,
- jeden wiersz dla korekty błędów transmisji.

W czasie przekazywania powyższych informacji uaktualnia się kartoteki danymi wynikającymi z nowego zamówienia, niezbędnymi do ułożenia harmonogramu produkcji.

System, dzięki możliwości współpracy z teledatorami w układzie pytanie/odpowiedź, może być wykorzystany do bieżącego informowania klientów o stanie ich zamówień.

A oto przykład takiego wykorzystania:

- Klient: Kiedy będę mógł odebrać model obuwia produkowany na zlecenie nr x?
- Operator: podaje nr zlecenia poprzez teledator do centrum.
- GE-120: redaguje odpowiedź w postaci wydruku zawierającego:
  - — nr zlecenia,
  - — nazwę modelu,
  - — nazwę formy,
  - — rodzaj i kolor skóry,
  - — nr klienta,
  - — termin wykonania,
  - — ilość par.
- Klient: Jakie jest zaawansowanie wszystkich moich zamówień?
- Operator: podaje nr klienta poprzez teledator do centrum.
- GE-120: formułuje odpowiedź zawierającą:
  - — zamówione ilości obuwia,
  - — ilość w uruchomieniu,
  - — ilość w toku produkcji,
  - — ilość przekazaną do magazynu.
  - — ilość wysłaną.
- Klient: Jak zaawansowane są moje zamówienia według rodzaju skóry, formy i modelu?
- Operator: przekazuje nr klienta poprzez teledator do centrum.
- GE-120: drukuje odpowiedź.

W ten sposób system umożliwia uzyskanie w dowolnej chwili wszystkich informacji o produkowanych aktualnie wyrobach. Ponadto system prowadzi na bieżąco rozliczenia finansowe i materiałowe. Średni czas opracowania szczegółowej odpowiedzi na postawione pytanie wynosi 3 minuty. Poza podstawowymi dziedzinami system wy-

konuje szereg funkcji dodatkowych, wśród których warto wymienić chociażby niektóre:

- potwierdzenie przyjętych zamówień,
- fakturowanie,
- rozliczanie klientów i dostawców,
- prowadzenie księgowości,
- obliczanie płac personelu,
- kontrola i ewidencja produkcji,
- sterowanie zapasami magazynowymi,
- inwentaryzacja ciągła,
- analizy ekonomiczne.

Wydaje się, że przyjęto rozwiązanie systemu APD w omawianych warunkach jest optymalne i wysoce nowoczesne. Pełna ocena na podstawie powyższych informacji nie jest, niestety, możliwa. Można jedynie sądzić, że ekonomicznie system jest opłacalny i w pełni zadowala użytkowników.

Opracował STEFAN KWIATEK
ZOWAR

REPRODUKOWANIE WYDRUKÓW KOMPUTEROWYCH[1]

681.327.5411:681.621:77293

W miarę wzrostu popularności informacji przetwarzanej przez komputer coraz większej wagi nabiera problem reprodukowania kopii wydruków. Artykuł przedstawia ocenę szeregu metod, które służą takiej właśnie reprodukcji.

Sprawa produkowania kopii z wydruków komputerowych jest dla wielu firm bardzo zasadniczym problemem. Opisane poniżej badania wykonane zostały na specjalne zlecenie koncernu przemysłu ciężkiego. Przedmiotem badań było przeanalizowanie i porównanie kilku różnych procesów, które umożliwiają wykonywanie kopii z wydruków komputerowych. Metody te zostały porównane pod względem czasochłonności i kosztów.

Problem znalazł się w centrum uwagi, kiedy koncern skomputeryzował proces drukowania kosztorysów i cenników. Zdarzało się, że firma potrzebowała 25 kopii całego cennika. Pięć do dziesięciu kopii wysyłano do użytkowników i te właśnie kopie musiały być szczególnie wysokiej jakości. Reszta kopii pozostawała do użytku wewnętrznego. Aby uzyskać pożądaną jakość kopii, stosowano metodę drukowania każdej strony cennika na kliszach papierowych, które używane były następnie jako matryce dla drukarki offsetowej. Stosując tę konwencjonalną metodę nawet kopie używane na terenie firmy drukowano offsetowo. Przy uwzględnieniu czasu i kosztów produkowania kopii zwiększenie ich ilości z 10 do 25 jest niczym w porównaniu z czasem i kosztami, których wymaga przygotowanie matrycy.

W przyszłości cały cennik wraz z kosztorysem i specyfikacją techniczną otrzymywać się będzie w formie wydruku, po przetworzeniu na komputerze odpowiedniego programu. I tu od razu nasuwa się pytanie: jak powinien wyglądać wydruk z drukarki komputera?

W celu znalezienia najlepszej metody reprodukowania oryginału zbadano różne możliwości. Firma wypróbowała praktycznie wszystkie spośród opisanych poniżej metod. Przy każdej metodzie odpowiednio pobrano próbki, co umożliwiło porównanie tych metod zarówno pod względem czasu i kosztów, jak i pod względem jakości otrzymywanych kopii.

Cenniki liczą na ogół 20—40 stron. Aby zapewnić porównywalność poszczególnych metod, przyjęto, że cennik składa się z 30 stron. Zrobiono również założenie, że potrzeba 20 kopii takich 30-stronicowych cenników.

Ponieważ pierwszym krokiem w każdej z metod jest uzyskanie oryginału na komputerze, przy analizowaniu poszczególnych metod uwzględniono czas i koszt takiego przebiegu na maszynie.

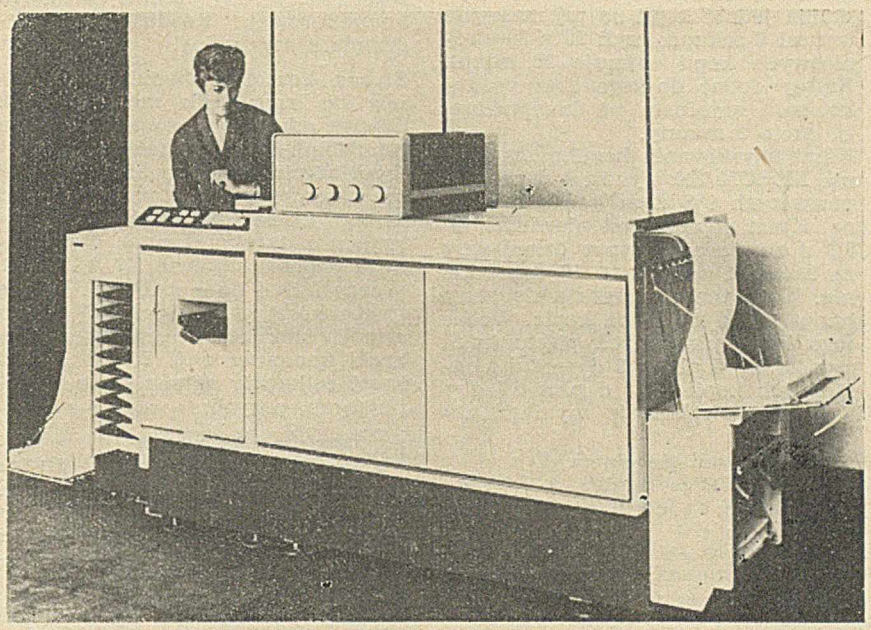
Drukarka komputerowa (metoda 1)

Przy założeniu, że na drukarkę komputerową przypada 5 warstw nieograniczonego co do długości papieru, możliwe jest uzyskanie w czasie przebiegu maszyny jednego oryginału i 4 czytelnych kopii. Chcąc otrzymać więcej kopii należy odpowiednią ilość razy powtórzyć przebieg drukowania na komputerze. Nie znaczy to, że musi być powtórzony cały program, w wyniku którego wydrukowany został cennik. W czasie pierwotnego przebiegu cennik zostaje zapisany na taśmie lub dysku i stąd następnie drukowany jest pośrednio lub bezpośrednio w przypadku, gdy większa ilość kopii okaże sę potrzebna.

Jest sprawą polityki danego przedsiębiorstwa, czy tylko pierwszy egzemplarz wydruku wysłany będzie do użytkowników, czy też wysyłać się będzie również pierwszą kopię. Decyzja ta uzależniona jest oczywiście od tego, ilu przebiegów potrzeba do wyprodukowania niezbędnej ilości egzemplarzy wysyłkowych.

Zakładając, że w celu uzyskania odpowiedniej liczby kopii należy

CEP — powielacz wydruków komputerowych Rank XEROX



1) Źródło: Fürst Wolfgang — Reproducing Computer Printout, *Data Systems*, 1970, marzec.

czterokrotnie powtarzać przebieg, otrzymuje się czas drukowania:

$4 \times 30 \times 6 = 720$ sekund = 12 minut,

gdzie 6 sekund stanowi przeciętny czas drukowania jednej strony na drukarce o szybkości 600 wierszy na minutę. Program, w wyniku którego powstaje cennik, wymaga komputera średniej wielkości. Koszt godziny pracy takiego komputera jest rzędu £ 30. Wobec tego koszt wyprodukowania 20 kopii wszystkch 30 stron wynosi

$$\frac{30 \times 12}{60} = £6\ 8\ s$$

Należy uwzględnić jeszcze koszt papieru, który sięga £ 20 za 1000 pięciostronicowych (1 oryginał plus 4 kopie) zestawów. Za 20 zestawów wyniesie on zatem

$$\frac{20 \times 120}{1000} = £2\ 8\ s$$

W ten sposób koszt całkowity wynosi £ 8 8 s, a stąd koszt wyprodukowania jednej kopii — 3,4 d. Metoda 1 jest bardzo elastyczna: zmniejszenie liczby kopii znajduje natychmiast odzwierciedlenie w skróceniu czasu i obniżeniu kosztu. Należy wspomnieć, że istnieje szybka i niedroga metoda produkowania „niebieskich kopii". Polega ona na zastosowanu półprzeroczystego papieru (specjalny papier, który wygląda tak jak normalny, ale może być używany jako matryca na „niebieskim" powielaczu) na drukarce komputerowej. Papier ten wkłada się w miejsce pierwszej strony wydruku. Taki oryginał umożliwia szybkie powielenie dowolnej ilości kopii dla użytku wewnętrznego.

Kopiarki (metoda 2)
XEROX 720 (metoda 2a)

W wyniku pierwszego przebiegu komputera uzyskuje się oryginał, z którego następnie robi się kopie na maszynie XEROX 720. Czas uzyskania jednej kopii na tej maszynie wynosi 8 sekund, czyli 20 × 30-stronicowych kopii zajmuje 80 minut. Należy dodać do tego czas produkowania oryginału na komputerze, tj. około 3 minuty.

Koszt uzyskiwania kopii na maszynie XEROX 720 wynosi 4 d za pięć pierwszych i 2 d za każdą następną. Koszt papieru jest niewielki, bo 0,3 d za stronę. Czynsz dzierżawny za maszynę wynosi £ 10 miesięcznie. Koszt ten amortyzuje się przez narzut 0,3 d na każdą kopię. Wobec tego koszt pierwszych pięciu kopii wynosi 3,7 d plus 0,3 d = 4 d, a następnych kopii — 1,7 d plus 0,3 d = 2 d. Koszt całkowity wyniesie zatem:

3 minuty czasu komputera o koszcie £ 30 za godzinę	£ 1 10 s
600 stron po 0,3 d	£ 1 5 s
150 kopii po 4 d plus 450 kopii po 2 d	£ 6 5 s
razem	£ 9 0 s

Średni koszt kopii wynosi 3,6 d.

Należy uważać, aby bęben kopiarki XEROX był dobrze oczyszczony, ponieważ kopie mają tendencję do wykazywania szarego tła. Metoda ta wymaga więcej czasu niż metoda 1, ale przeciętna jakość produkowanych kopii jest zdecydowanie lepsza od kopii wychodzących bezpośrednio z drukarki komputerowej. Koszty są tego samego rzędu, co przy metodzie 1. Warto jednak wspomnieć, że przeciętny koszt kopii zmienia się w zależności od ilości kopii. Na przykład, gdyby potrzebowano tylko pięciu kopii cennika, średni koszt kopii wzrósłby z 3,6 d do 6,7 d, czyli prawie o 100%. Oczywiście koszt obniżyłby się, gdyby potrzebowano więcej niż 20 kopii.

XEROX 3600 (metoda 2b)

Procedura otrzymywania kopii na kopiarce XEROX 3600, która jest zmodernizowaną wersją nie sprzedawanej już dziś kopiarki XEROX 2400 przypomina procedurę stosowaną na maszynie XEROX 720. Nowocześniejszy XEROX 3600 różni się od XEROX 720 szybkością i ceną. Również jakość kopii otrzymywanych na kopiarce XEROX 3600 jest dużo lepsza. Obydwie maszyny są w pełni zautomatyzowane, ale gdy na XEROX 720 można z jednego oryginału otrzymać jednorazowo 24 kopie, na modelu 3600 można ich otrzymać 499. Bardzo wygodnym urządzeniem dodatkowym przy maszynie 3600 jest sorter. Jeżeli potrzeba 20 kopii książki, każda kopia spada do jednej z dwudziestu kieszeni i w ten sposób składa się automatycznie 20 kopii książki. Sorter taki można wydzierżawić za opłatą £ 30 miesięcznie, a dodatkowe banki dziesięciokieszeniowe po £ 10 miesięcznie.

XEROX 3600 jest trzy razy szybszy od maszyny 720. Przy średniej szybkości 2,5 sekundy na kopię, uzyskanie 20 kopii 30-stronicowego cennika zajmuje 25 minut. Dodać do tego należy trzy minuty drukarki komputerowej, produkującej egzemplarz oryginalny.

Strona kosztów również przedstawia się inaczej dla maszyny 3600. Czynsz miesięczny wynosi £ 10, a cena kopii 3,7 d za każdą z pierwszych pięciu kopii oryginału i 0,75 d za każdą następną, począwszy od szóstej. Obciążając każdą kopię narzutem z tytułu czynszu dzierżawnego podnosi się cenę 3,7 d do 4 d, a cenę 0,75 d do 1 d. Dodać jeszcze trzeba koszt papieru, czyli 0,3 d za stronę, oraz koszt przebiegu drukarki komputerowej. W ten sposób koszt całkowity składa się z następujących pozycji:

3 minuty czasu komputera o koszcie £ 30 za godzinę	£ 1 10 s
600 stron po 0,3 d	£ 1 5 s
150 kopii po 4 d i 450 kopii po 1 d	£ 4 7 s
razem	£ 7 2 s

Przeciętny koszt kopii wynosi 2,85 d.

Produkowane tą metodą kopie są tak wysokiej jakości, że można je wysyłać do klientów. Przeciętny koszt kopii jest zdecydowanie niski. Zależy jednakże od liczby produkowanych kopii. Jeżeli liczba ta zmniejsza się np. z 20 do 5, to koszt jednej kopii wzrasta z 2,85 d do 6,7 d.

Druk offsetowy (metoda 3)

Druk offsetowy jest bardzo odpowiedni w tych wypadkach, gdy potrzebne są kopie szczególnie wysokiej jakości. Cechą charakterystyczną druku offsetowego jest to, że każda kopia produkowana jest drogą wprowadzenia papieru pomiędzy dwa wałki, z których dolny wiedzie formę offsetową, a górny rozprowadza farbę drukarską. Rodzaje maszyn używane do takiego drukowania to Gestetner, A. B. Dick, Rotaprint lub podobne drukarki offsetowe, o cenach kształtujących się w granicach £ 500—5000, zależnie od stopnia automatyzacji, a stąd i szybkości.

Są trzy możliwości wykonywania matryc drukarskich przy systemie komputerowego drukowania cenników. Można je produkować metodą na wpół ręczną lub za pomocą specjalnej maszyny produkującej klisze fotograficzne i można zapisywać kliszę bezpośrednio na drukarce komputerowej.

Zapis kliszy na drukarce komputerowej (metoda 3a)

W przypadku zastosowania metody zapisywania kliszy bezpośrednio na drukarce komputerowej, egzemplarz oryginalny produkowany przez komputer musi być zapisany na półprzezroczystym papierze, czyli takim, o jakim była mowa wyżej. Następnie za pomocą powielacza wielokolorowego półprzezroczysta kopia zostaje przeniesiona na specjalną kliszę metaliczną. Przed wejściem na drukarkę offsetową klisza ta musi być wywołana i ręcznie oczyszczona. Jest to dość żmudny proces, trwający 10—15 minut, co powoduje, że czas całkowity produkcji kliszy wzrasta do 20 minut. Jeżeli doda się do tego tyle sekund, ile wynosi liczba kopii (jedna sekunda jest bowiem czasem drukowania jednej kopii), czas całkowity otrzymania 20 × 30-stronicowych kopii cennika wyniesie 10 godzin i 10 minut.

Cena jednej kliszy specjalnej dla powielacza wielokolorowego wynosi 3 s, a jeżeli doda się do tego wywołanie kliszy, cena wzrasta do 3,6 s. Koszt drukowania na drukarce offsetowej średniej wielkości można przyjąć za 0,5 d.

Koszt całkowity kształtuje się zatem następująco:

3 minuty czasu komputera o koszcie £ 30 za godzinę	£ 1 10 s
30 klisz po 3 s 6 d	£ 5 5 s
600 kopii offsetowych po 0,5 d	£ 1 5 s
razem	£ 8 0 s

Średni koszt kopii wynosi 3,2 d.

Produkowanie za pomocą tej metody kopii odpowiedniej jakości wymaga trochę doświadczenia i praktyki. Półprzezroczysty papier komputerowy musi być starannie dobrany i wskazane jest, przed rozpoczęciem stosowania tej metody, zrobienie kilku próbnych przebiegów. Przy tej metodzie koszt jednej kopii również uzależniony jest od ogólnej ilości kopii, jaką chce się uzyskać z jednej formy offsetowej. Jeżeli liczba kopii spada do pięciu, średni koszt kopii podnosi się do 11,3 d i proces staje się nieekonomiczny zarówno z punktu widzenia czasu, jak i kosztów.

TABLICA

Metoda	Średni koszt kopii		Czas min	Ocena jakości
	przy podstawie 20 kopii	przy podstawie 5 kopii		
1	3,4 d	3,4 d	12	różna
2a	3,6 d	6,7 d	83	zadowalająca
2b	2,8 d	6,7 d	28	dobra
3a	3,2 d	11,3 d	610	dobra
3b	1,7 d	5,3 d	150	bardzo dobra
3c	1,3 d	4,7 d	28	bardzo dobra

Automatyczne przygotowanie matryc (metoda 3b)

Automatyzacja procesu produkowania matryc w omawianej metodzie wymaga zastosowania specjalnej maszyny, której użytkowanie kosztuje £ 32 miesięcznie. Maszyna fotografuje oryginał zdjęty z komputera i reprodukuje go na normalnej kliszy papierowej. Klisze takie dostarczane są przez producentów drukarek offsetowych w cenie około 6 d za sztukę.

Otrzymanie jednej matrycy na maszynie produkującej klisze zabiera około 5 minut, a otrzymanie jednej kopii z takiej formy na drukarce offsetowej — 1 sekundę. Cały proces produkowania 20 × 30-stronicowych kopii trwa zatem 2,5 godziny. Koszt całkowity składa się z kosztu czasu komputera produkującego egzemplarz pierwotny, kosztu wyprodukowania matryc na maszynie do produkcji klisz i kosztu drukowania na drukarce offsetowej. A więc:

3 minuty czasu komputera o koszcie £ 30 za godzinę	£ 1 10 s
wyprodukowanie 30 klisz po 1 s sztuka	£ 1 10 s
600 kopii offsetowych po 0,5 d	£ 1 5 s
razem	£ 4 5 s

Średni koszt kopii wynosi 1,7 d. Kopie otrzymane tą metodą są doskonałej jakości.

Średni koszt kopii jest zachęcający i niski w porównaniu z innymi metodami. Należy nadmienić, że koszt otrzymania formy na maszynie do produkcji klisz został skalkulowany przy założeniu, że produkuje się 300 klisz tygodniowo. Jeżeli zmniejszyć tę podstawę do bardziej prawdopodobnej ilości 100 klisz tygodniowo, to średni koszt kopii podniesie się do 2,3 d. Ale nawet ta cena jest konkurencyjna w stosunku do cen związanych z przedstawionymi dotąd metodami.

Zmiana taśmy (metoda 3c)

W chwili obecnej coraz większą popularność zyskuje metoda opisana poniżej. Zmieniając taśmę na drukarce komputerowej można drukować bezpośrednio na kliszach papierowych nieograniczonej długości, które to klisze mogą być następnie użyte jako formy na drukarce offsetowej. Zmiana taśmy jest kwestią sekund, a większość drukarek nie wymaga oprócz tego żadnych innych zmian. Klisze papierowe mogą być zawczasu zaopatrzone w nagłówek i rubryki stanowiące z góry zaplanowany układ tekstu.

Czas i koszty przedstawiają się następująco: wydrukowanie 30 stron na kliszach papierowych wymaga jednego przebiegu maszyny, czyli 3 minut, rozdzielenie i przygotowanie klisz do powielenia — około 15 minut, wydrukowanie kopii na drukarce offsetowej — około 10 minut. W ten sposób uzyskanie kopii zabiera 28 minut.

Na koszt składają się następujące pozycje:

3 minuty czasu komputera o koszcie £ 30 za godzinę	£ 1 10 s
30 klisz po £ 20 za 1000 sztuk	12 s
koszt amortyzacji specjalnej taśmy	10 s
600 kopii offsetowych po 0,5 d	£ 1 5 s
razem	£ 3 17 s

Średni koszt kopii wynosi 1,3 d. Metoda ta, stosowana przez wielu użytkowników komputerów, daje bardzo dobre rezultaty. Uzyskiwane kopie są doskonałej jakości. Metoda nie wymaga żadnej specjalnej maszyny do produkcji klisz, która to maszyna pociąga dodatkowy koszt miesięczny w wysokości £ 32. Średni koszt kopii jest wobec tego niezależny od ilości produkowanych matryc. W ramach naszego porównania metoda ta wykazuje osiągnięcie najniższego kosztu kopii.

Z drugiej jednak strony, jeżeli liczba kopii każdej strony spada z 20 do np. 5, przeciętny koszt uzyskania kopii zwiększa się z 1,3 d do 4,7 d. Zamieszczone poniżej zestawienie pozwala porównać tę metodę z innymi metodami przedstawionymi w artykule. Zestawienie obejmuje wszystkie sześć metod, które poddano badaniom. Przedstawione w nim zostały główne cechy każdej metody. Średni koszt kopii podany jest dwukrotnie: raz dla podstawy, którą jest uzyskanie 20 kopii oryginału, i drugi raz, gdy celem jest uzyskanie 5 kopii. Ocena jakości kopii, zamieszczona w ostatniej kolumnie tabeli, jest oczywiście dość subiektywna. Przy określaniu jakości brane były pod uwagę takie cechy kopii, jak kontrastowość druku w stosunku do białego papieru, czytelność i tło.

Opracowała **EWA ZAWISZA**
ZOWAR

2) Wszystkie wartości pieniężne w niniejszym artykule podane są w angielskich jednostkach monetarnych: £ — funty, s — szylingi, d — pensy.

CZY PRZYSZŁOŚĆ NALEŻY DO MINIKOMPUTERÓW

W Stanach Zjednoczonych AP...

Konferencja EPD, która z ramienia Stowarzyszenia Managerów Amerykańskich (American Management Association) obradowała niedawno w Nowym Jorku, wysunęła przypuszczenie, że w r. 1975 na każde 10 komputerów — 6 będzie minikomputerami. Minikomputery nie tylko zaczynają przejmować funkcje wykonywane dotąd wyłącznie przez ludzi z pomocą łączy telekomunikacyjnych, lecz są stosowane ze względów ekonomicznych jako uzupełnienie wielkich i kosztownych komputerów do przetwarzania danych.

Na przykład komputer IBM 360/95, używany w Ośrodku Lotów Kosmicznych Goddard w Greenbelt w stanie Maryland w celu kontroli satelity pogody TIROS-M, otrzymał niedawno minikomputer Varian 620/i, który stał się ogniwem pośrednim pomiędzy wspomnianym

komputerem z danymi napływającymi z TIROS-M.

Minikomputery stają się buforem dla informacji po to, aby do wielkich maszyn cyfrowych docierały tylko te informacje, które są dla nich ważne. Procedura ta pozwala na dokonanie znacznych oszczędności.

A oto przykład użycia minikomputerów, celem przekazywania danych na odległość geograficzną, jako czynnika nadzoru i kontroli. Koncern naftowy Getty Oil, którego siedziba znajduje się w Coaling w Kaliforni, stosuje minikomputery do kontroli rurociągu doprowadzającego ropę naftową do rafinerii położonych nad zatoką San Francisco.

Podobnie też w celu nadzoru koncern energetyczny w Minnesota stosuje system kontrolny oparty na „telepamięci". Koszty tego systemu wynoszą 85 000 dolarów, lecz kontrola oparta na zasadzie „telepamięci" wyeliminowała w praktyce wszelkie błędy, których mógł się dopuścić pracownik obsługujący maszynę. Używany tu minikomputer — to Varian 620 z pamięcią o pojemności 8192 słów 16-bitowych. Za pomocą systemu „telepamięci" kontroluje on przeszło 1050 funkcji.

W Europie...

W marcu 1970 r. wyprodukowano w Mediolanie pierwszy w Europie „kieszonkowy komputer" — IBM 3. Został on natychmiast dostarczony firmie w Oggiono, produkującej silniki i maszyny przędzalnicze. Przed tym „noworodkiem", którego zastosowanie jest bardzo szerokie, otwierają się duże możliwości.

IBM 3 jest owocem zbiorowego wysiłku siedmiu laboratoriów badań naukowych (warto zaznaczyć przy okazji, że firma IBM utrzymuje 30 ośrodków badań naukowych na całym świecie): Fishkill, San Jose, Rochester oraz Research Triangle Park w St. Zjednoczonych AP, Hursley w W. Brytanii, Boeblingen w NRF i Corbeil-Essone we Francji.

„Kieszonkowe komputery" IBM 3 produkowane są również od niedawna w USA na Florydzie, a w roku przyszłym produkcję ich podejmie Japonia w Fujisawa. Oczekuje się, że wkrótce opanują one rynek światowy.

Poszczególne elementy nowego systemu mogły zostać wyprodukowane jedynie dzięki zorganizowanej na szeroką skalę kooperacji i współpracy międzynarodowej. Wszystkie zainteresowane fabryki powiązane są ze sobą systemem „teleprocessing" przez 24 godziny na dobę. Dyski produkowane są w W. Brytanii w Havant, rejestrator danych — w Greenock. Francuska fabryka w Corbeil produkuje mikropłaty i moduły monolityczne MST (Monolithic Systems Technology); należy zaznaczyć, że właśnie w komputerze IBM 3 znalazły one jedno z pierwszych zastosowań przemysłowych.

IBM 3 może przetwarzać informacje z 96-kolumnowych kart perforowanych, mniejszych od kart do gry, jak również bezpośrednio z dysków magnetycznych. Będzie mógł też zostać przystosowany do innych zespołów, stając się w ten sposób elementem peryferyjnym bądź marginalnym wielkiego komputera.

Źródła: *Electronic Weekly*, 8.04.1970
Le Figaro, 19.03.1970

K. P.

jest jedynym krajem świata kapitalistycznego, mającym rozwinięty własny przemysł komputerów. Szeroka baza technologiczna w elektronice i w „hardware" daje możliwość rywalizacji ze Stanami Zjednoczonymi AP w niedalekiej przyszłości. Już obecnie Japonia skutecznie rywalizuje z Wielką Brytanią na wschodnioeuropejskim rynku komputerów.

Niedawno towarzystwo Tokyo Shibaura Electric sprzedało 800 kalkulatorów elektronicznych do Związku Radzieckiego. Japońskie Stowarzyszenie Rozwoju Przemysłu Elektronicznego zapowiedziało też ostatnio, że Węgry zamierzają w tym roku wysłać misję do Japonii w celu przestudiowania różnych aspektów japońskiego przemysłu maszyn cyfrowych.

Rynek komputerów w Japonii

(udziały procentowe — grudzień 1969 r.)

Nazwa koncernu japońskiego		Z jakim koncernem jest związany
IBM Japonia	— 35%	IBM
Hitachi	— 13,8	RCA
Fujitsu	— 13,6	z żadnym
NEC	— 11,5	Honeywell
Univac	— 10,2	Univac
Toshiba	— 3,7	GE
NCR Japonia	— 3,2	NCR
Oki	— 3,2	Univac
Inne wraz z Mitshubishi	— 5,8	

Źródło: *Financial Times* 16.03.1970 (specj. dodatek na temat Japonii), *Electronics Weekly* 4 i 25.03 oraz 15.04.1970, *The Japan Econ. Journal* 17.03.1970

K. P.

JESZCZE O JAPONII...

Japoński przemysł komputerów jest chyba najszybciej rozwijającym się przemysłem w tym kraju, którego gwałtowna ekspansja przemysłowa jest przedmiotem fascynacji całego świata. Obecnie już eksploatuje się tam ponad 5000 komputerów, z tego zaledwie około jedna czwarta importowanych, reszta zaś krajowej produkcji. Przewiduje się, że w ciągu najbliższych dwóch lat produkcja komputerów wzrośnie o około 50%.

Japońska produkcja komputerów koncentruje sie głównie na małych komputerach. Tak np. w ciagu od marca do końca września 1969 r. japońscy producenci sprzedali 663 jednostki, ogólnej wartości przeszło 50 milionów funtów bryt. Oznacza to wzrost o 18,7% w stosunku do tego samego okresu 1968 r.; przewiduje się jednak znacznie większy postęp w br.

Japońskie Ministerstwo Międzynarodowego Handlu i Przemysłu wszelkimi sposobami popiera rozwój przemysłu komputerów, mając na celu doścignięcie USA w 1980 r. Tak np. stworzono specjalny koncern do spraw rozwoju przetwarzania danych, o budżecie przeszło 200 000 funtów. Rząd pomógł również w stworzeniu kartelu, który bierze udział w produkcji urządzeń peryferyjnych. W 1966 r. rozpoczęto pracę nad budową wielkich komputerów. Wkrótce ma być ukończony pierwszy wielki komputer produkcji japońskiej o dużej szybkości.

Japonia — prócz Stanów Zjednoczonych AP i Wielkiej Brytanii —

PRETO informuje...

EFEKTY STOSOWANIA KOMPUTERÓW

Przytaczamy poniżej niektóre przykłady efektów zastosowań od kilku lat maszyn matematycznych w Zakładach Elektronicznej Techniki Obliczeniowej podległych Pełnomocnikowi Rządu do Spraw Elektronicznej Techniki Obliczeniowej.

Komputer na usługach telekomunikacji

Zakład Elektronicznej Techniki Obliczeniowej w Gdyni realizuje od 1 stycznia 1967 r. system SARP (automatyczne rozliczanie i fakturowanie należności za usługi telekomunikacyjne). System został uruchomiony w niespełna rok. Aktualne efekty ekonomiczne w skali roku wynoszą 720 tysięcy złotych i sprowadzają się głównie do oszczędności 30 etatów.

System jest realizowany na EMC ICT-Serii 1900.

Sykopp — 1

Zakład Elektronicznej Techniki Obliczeniowej we Wrocławiu opracował na maszynę Mińsk-22 System Kontroli i Planowania Produkcji (SYNKOPP-1). System ten obejmuje następujące zagadnienia:

- techniczne przygotowanie produkcji,
- planowanie produkcji,
- rachunek kosztów.

Wykonany jest on dla dziewięciu zakładów Dolnego Śląska.

Niektóre dolnośląskie przedsiębiorstwa eksploatują system od kilku lat.

W wyniku zastosowania tego systemu daje się stwierdzić następujące efekty we wszystkich dziewięciu zakładach. W fazie wprowadzania systemu:

- uporządkowano dokumentację technologiczną i konstrukcyjną pod względem kompletności, jak i rzetelności zawartych w niej informacji (normy czasowe i materiałowe). Z tym łączy się uporządkowanie spraw symboliki:

— stanowisk roboczych,
— operacji,
— wydziałów, oddziałów,
— jednostek miar,
— części wyrobów,
— indeksów materiałowych.

W fazie eksploatacji systemu

- nastąpiło znaczne zmniejszenie okresu czasu niezbędnego na uzyskanie informacji, w szczególności w zakresie obliczeń dotyczących normatywów jednostkowych, planowania operatywnego, planów zużycia materiałowego,
- uzyskano pewność w odniesieniu do rzetelności informacji,
- usprawniono system zarządzania,
- nastąpiło zmniejszenie zatrudnienia w komórkach planowania operatywnego oraz w komórkach planowania zaopatrzenia materiałowego,
- uzyskano precyzyjne określenie pracochłonności produkcji w oddziałach, co pozwala panować lepiej nad polityką zatrudnienia tych oddziałów.

Na ogół jest to związane z wygospodarowaniem rezerw w tej dziedzinie.

Dla ilustracji podamy tylko rachunek ekonomiczny stosowania systemu dla zakładów „PAFAL" za rok 1968.

Koszty systemu:		
Koszty obliczeń	zł	513 000
Koszty komórki EPD	zł	113 000
	zł	626 000
Oszczędności:		
Oszczędności na robociźnie w związku z prawidłowym wyliczeniem pracochłonności w oparciu o bazę aktualnych normatywów rob/godz.	zł	12 180 000
Poprawa rytmiczności produkcji ze względu na zmniejszenie ilości godzin przestoju (poprawa planowania) uniknięto zatrudnienia 10 osób	zł	70 000
	zł	310 000
		12 860 000
poniesione nakłady		473 000
efekt	zł	12 387 000

Od kilku miesięcy SYKOPP jest adaptowany przez Zakład Elektronicznej Techniki Obliczeniowej dla przedsiębiorstw województwa bydgoskiego.

Pakiet obliczeń produkcyjnych FSC Starachowice

Zastosowanie EMC IBM-1440 przez Zakład Elektronicznej Techniki Obliczeniowej ZOWAR dla obliczeń produkcyjnych w Fabryce Samochodów Ciężarowych w Starachowicach przynosi w skali roku efekty ekonomiczne w wysokości ponad 27 milionów złotych.

Efekty te dzielą się na:

oszczędności na funduszu płac	400 tys. zł
obniżka kosztów produkcji wynikająca z lepszego wykorzystania maszyn i urządzeń	6 000 tys. zł
zmniejszenie produkcji w toku	2 000 tys. zł
zmniejszenie stanu zapasów materiałowych	18 000 tys. zł
poprawa rytmiczności produkcji	1 000 tys. zł

Przytoczyliśmy tylko kilka przykładów efektów gospodarczych stosowania maszyn matematycznych w sieci Zakładów Elektronicznej Techniki Obliczeniowej. Do sprawy tej będziemy powracać.

SPRAWOZDAWCZOŚĆ

Na mocy Zarządzenia nr 10/70 Pełnomocnika Rządu do Spraw Elektronicznej Techniki Obliczeniowej z 23 marca 1970 r. została opublikowana „INSTRUKCJA nr 1/70 w sprawie sprawozdawczości statystycznej ośrodków obliczeniowych wyposażonych w elektroniczne maszyny cyfrowe, ośrodków maszyn analitycznych i stacji przygotowania danych".

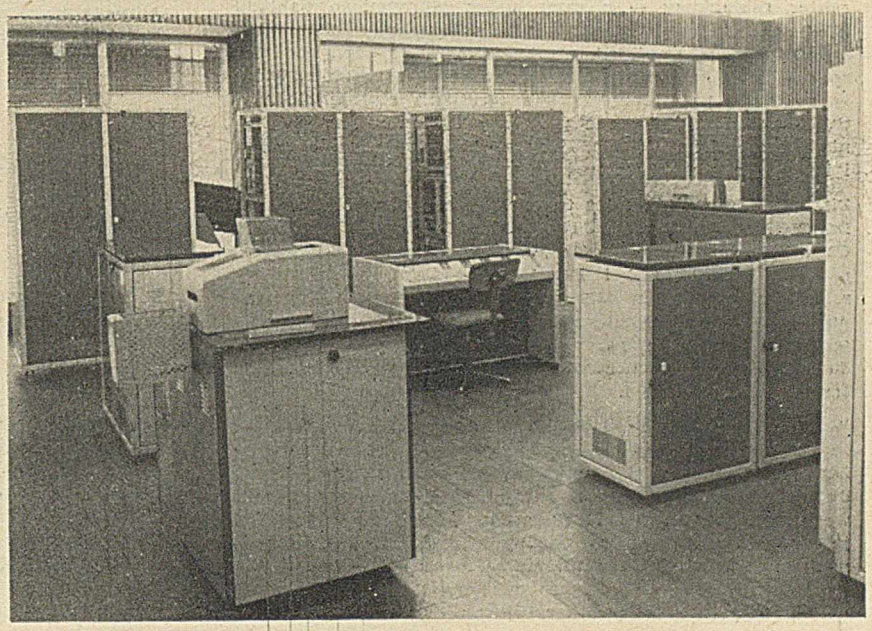
Publikacja zawiera obok wspomnianej instrukcji szereg bardzo cennych informacji na temat ośrodków obliczeniowych, wykazy zainstalowania tam maszyn i urządzeń obliczeniowych na koniec 1969 r. oraz aktualne adresy tych ośrodków.

Instrukcja nr 1/70 zawiera między innymi dane o:

- jednostkach sprawozdawczych,
- terminach, rozdzielnikach i formularzach,
- zasadach sporządzania sprawozdawczości z ośrodków i stacji nowo organizowanych,
- dzienniku pracy maszyny,
- książce zleceń,
- ogólnych zasadach wypełniania sprawozdań.

JÓZEF ŚNIECIŃSKI
Biuro PRETO

Jedna z dwóch elektronowych maszyn cyfrowych MIŃSK 32 przygotowana do montażu w Zakładzie Elektronicznej Techniki Obliczeniowej — Katowice



WIADOMOŚCI PKAPI

KOMUNIKAT

W dniach 22—23 października 1970 r. w Rzeszowie, w Domu Technika, ul. Kopernika 1, odbędzie się konferencja naukowo-techniczna na temat „Ekonomiczno-organizacyjne efekty zastosowania API".

Ogólny program konferencji przewiduje wysłuchanie referatu generalnego i przeprowadzenie dyskusji wraz z opracowaniem wniosków. Pozostałe referaty zostaną zamieszczone w materiałach konferencyjnych.

Adres Sekretariatu Komitetu Organizacyjnego: Rzeszów, ul. Kopernika 1, tel. 328-91, wew. 27.

Liczba uczestników ograniczona.

Z KRAJU

KONFERENCJA W GRUDZIĄDZU

W dniu 22 maja 1970 r. w Grudziądzu odbyła się — zorganizowana przez Polskie Towarzystwo Ekonomiczne, Oddział Wojewódzki w Bydgoszczy, Sekcja d.s. ETO — konferencja ekonomistów przemysłu i budownictwa woj. bydgoskiego pn.: „ELEKTRONICZNA TECHNIKA OBLICZENIOWA INSTRUMENTEM NOWOCZESNEGO ZARZĄDZANIA PRZEDSIĘBIORSTWEM". Referaty wygłosili naukowcy i praktycy, m. in.: doc. dr habil. T. Wierzbicki — Rachunek kosztów w warunkach nowoczesnej techniki obliczeniowej, dr inż. A. Targowski — Kierunki stosowania komputerów w zarządzaniu, mgr inż. K. Szumlas — Przewidywany rozwój zastosowań elektronicznej techniki obliczeniowej w woj. bydgoskim w latach 1971—1975 na tle oceny stanu dotychczasowego.

Z ramienia Sekcji koreferat do referatu mgr inż. Szumlasa wygłosił mgr A. Okniński przedstawiając problemy związane z wyborem modelu przygotowania i wdrożenia w przedsiębiorstwie systemu elektronicznego przetwarzania danych w warunkach występujących w woj. bydgoskim. Dyskutanci poruszali szereg problemów technicznych i organizacyjnych. Do ciekawszych wypowiedzi zaliczyć należy głos mgr Mrozowieckiego z Zakładów Azotowych we Włocławku. Przedstawił on doświadczenia z zakresu zastosowania elektronicznej maszyny do sterowania procesami produkcyjnymi.

Powołana przez konferencję komisja wniosków opracowała postulaty, które zostały przekazane władzom wojewódzkim. Dotyczyły one m. in.: kształcenia kadr na Uniwersytecie im. Mikołaja Kopernika w Toruniu i Wyższej Szkole Inżynierskiej w Bydgoszczy; rozwinięcia sieci placówek ZETO; udziału ekonomistów poprzez Sekcję ETO w informowaniu kierownictw przedsiębiorstw o najnowszych osiągnięciach w zastosowaniu elektronicznej techniki obliczeniowej do zarządzania przedsiębiorstwem.

ADAM OKNIŃSKI

PRZEGLĄD WYDAWNICTW

COMPUTER DECISIONS

Znana firma wydawnicza HAYDEN PUBLISHING COMPANY w Nowym Jorku uruchomiła we wrześniu 1969 roku nowe czasopismo, popularyzujące problemy komputeryzacji gospodarki. Czasopismo to ma wprawdzie niewątpliwie charakter reklamowy — o czym prócz olbrzymiego nakładu 90 500 egz. i znacznego procentu egzemplarzy okazowych świadczy masa, skądinąd, atrakcyjnych i dowcipnych ogłoszeń — jednakże każdy numer zawiera 6 artykułów, które w możliwie przystępny sposób (grafika!) starają się wytłumaczyć skomplikowane problemy Systemów Informowania Kierownictwa (MIS), języka APL, symultacji rynku i innych zagadnień zastosowaniowych.

Firma HAYDEN kilka lat temu wydała słynny STANDARD DICTIONARY OF COMPUTERS AND INFORMATION PROCESSING (WEIK, 196) oraz popularne podręczniki fortranowskie HAAGa (196 i 196). Nowy, blisko 100-stronicowy miesięcznik (w roku 1969 ukazały się jednak tylko trzy numery — we wrześniu, listopadzie i grudniu), wydawany techniką wielokolorową, świadczy o odkryciu przez wydawcę luki w dotychczasowej działalności wydawniczej na terenie amerykańskim.

W słowie wstępnym od Redakcji znajdujemy wyjaśnienia, że obecnie nie są znane jeszcze wszystkie potencjalne możliwości zastosowań komputerów, które to maszyny znajdują się dopiero w początkowym stadium swego burzliwego rozwoju; ale dla ludzi podejmujących decyzje ważne jest zdanie sobie sprawy choćby z części tych potencjalnych możliwości, jakie aktualnie są rozeznane — bez względu na to, czy są oni ekspertami komputerowymi, czy też nie; dlatego też miesięcznik celuje na wszystkich potencjalnych użytkowników komputerów, szczególny nacisk kładąc na jasność treści i bogate wyjaśnienia graficzne. Trzy pierwsze przejrzane przez nas numery nie podważyły sensu tych zapowiedzi, zwłaszcza iż wielki format (prawie 30 × 40 cm) i wysokiej klasy papier satynowany, za który płacą ogłoszeniodawcy, stwarzają istotnie nie spotykane gdzie indziej możliwości graficzne.

Adres wydawcy: 850 THIRD AVENUE, NEW YORK 10022; warunki płatnej prenumeraty nieznane.

A. B. E.

BIBLIOGRAFIA KSIĄŻEK POLSKICH

z dziedziny maszyn matematycznych i licząco-analitycznych

● **Maszyny matematyczne w geodezji** — GAŹDZICKI J. Wydanie 2 uzupełnione. Wyd. Politechniki Warszawskiej, Warszawa, 1969, ss. 249, cena zł 16 (skrypt)

Wyd. 1 ukazało się w 1965 r. i zawierało 190 stron. W wydaniu 2 pozostawiono bez zmian osiem początkowych rozdziałów. Wprowadzono również nowe rozdziały. Rozwój techniki obliczeniowej i klasyfikacja urządzeń matematycznych. Urządzenia analogowe. Maszyny licząco-analityczne. Pozycyjne systemy liczenia. Zasady budowy i działania EMC. Podstawowe wiadomości o maszynie UMC 1. System W20. Elementy programowania. Podstawowe wiadomości o maszynie GEO 1. System programów podstawowych. Wia-

(c.d. na III str okł.)

(dokończenie ze str. 24)

domości uzupełniające o programowaniu. Wybrane algorytmy obliczeń geodezyjnych. Elektroniczna technika obliczeniowa w geodezji. Charakterystyka niektórych EMC zainstalowanych w Polsce.

Skrypt obejmuje wykłady prowadzone przez autora na 2 i 5 roku studiów Wydziału Geodezji i Kartografii Politechniki Warszawskiej. Zapoznaje w sposób encyklopedyczny z podstawami maszyn liczących i ich zastosowaniami w geodezji.

● **Mechanizacja obliczeń geodezyjnych** — BYCHAWSKI T. Państw. Przedsięb. Wyd. Kartograficznych, Warszawa, 1969, ss. 189, cena zł 18,50

Łączność z maszyną liczącą. Maszyny statystyczne. Organizacja pracy maszyn statystycznych. Arytmetyka EMC. Programowanie obliczeń. Współpraca ze stacją maszyn matematycznych. Książka wprowadzająca w zagadnienie budowy, działanie i zastosowanie maszyn liczących, w geodezji przeznaczona jest dla geodetów.

● **Niektóre aspekty zastosowania maszyn cyfrowych do interpretacji materiałów geofizyki wiertniczej** — KALETA A. i inni. Wyd. Śląsk, Katowice 1969, ss. 25. Praca Instytutu Naftowego

Omówiono algorytmy i programy wykonane w Zakładzie Maszyn Matematycznych Instytutu Naftowego, dotyczące automatycznej interpretacji parametrów geofizyki wiertniczej przy użyciu EMC „ODRA 1013", a w szczególności wyznaczania: litologii przewiercanych warstw i współczynnika nasycenia wodą skał ropno-gazowych, oporności właściwych warstw wg metodyki Guyoda oraz metodyki Komarowa itp. Praca przeznaczona dla geologów i inżynierów górniczych.

● **Elektroniczne maszyny cyfrowe w handlu angielskim** — JERCZYŃSKA M. Wyd. Instytutu Handlu Wewnętrznego. Warszawa 1969, ss. 154, cena zł 12. Opracowania i materiały 66

Część ogólna: Kierunki zastosowań EMC w handlu angielskim. Warunki rozszerzenia zastosowań w skali krajowej. Wpływ zastosowań EMC na systemy kierowania i zarządzania, organizację przedsiębiorstw oraz zatrudnienia. Część szczegółowa. Założenia organizacji systemu EPD w domach towarowych, dla sklepów detalicznych z odzieżą w handlu wysyłkowym. Załączniki — wzory dokumentu i tabulogramy. W opracowaniu przedstawiono wyniki studiów i badań przeprowadzanych w czasie 3-miesięcznego pobytu na stypendium ONZ w Anglii. Praca przeznaczona dla projektantów systemów EPD.

● **Zasady działania maszyn cyfrowych i programowania** — ORMICKI A. Wyd. Akademii Górniczo-Hutniczej w Krakowie, Kraków 1969, ss. 159, cena zł 10,50 (skrypt)

Systemy liczbowe i kody. Struktura maszyny i rozkazy. Zasady programowania. Schematy logiczne zespołów maszyny. Schematy elektroniczne niektórych zespołów maszyny.

Skrypt napisany jest na podstawie wykładów prowadzonych na Wydziale Elektrotechniki Górniczej i Hutniczej Wieczorowego Studium Inżynierskiego AGH dla specjalności Automatyka. Skrypt nie służy do przygotowania specjalistów z zakresu EMC. Zawiera podstawowe wiadomości, które powinien posiadać każdy inżynier, aby porozumieć się ze specjalistą EMC.

● STRESS: **Podręcznik użytkowania.** Problemowo-zorientowany język programowania EMC do rozwiązywania zagadnień z zakresu konstrukcji inżynierskich. Tłum. wyd. ang. z r. 1964. Wyd. ETOPROJEKT, Warszawa 1969, ss. 79. Zeszyty problemowe ETO, nr 7

Część podstawowa. Objaśnienie instrukcji. Stosowanie STRESSu. Opis systemu STRESS. Dodatki: A — Streszczenie metody liniowej analizy konstrukcji. B — Język STRESS — IBM. Zadaniem podręcznika było zapoznanie czytelników (inżynierów budowlanych, programistów) na przykładzie języka STRESS z nowymi kierunkami programowania EMC, jakimi są języki problemowo-zorientowane, ułatwiające stosowanie tych maszyn w zwykłej praktyce inżynierskiej (tu do obliczania płaskich i przestrzennych konstrukcji prętowych)

● **System automatycznego przetwarzania informacji w zakładach wytwórczych prefabrykatów żelbetowych** — MASŁOWSKI M. Wyd. Instytutu Organizacji i Mechanizacji Budownictwa, Warszawa 1969, ss. 97. Zeszyty problemów rozwoju budownictwa. Zeszyt nr 7, cena zł 16

Opracowanie podaje ogólne informacje organizacyjne dla Zakładu Obliczeniowego, który podejmie się obsługi Systemu Automatycznego Przetwarzania Informacji (SAPI). Praca została napisana w oparciu o doświadczenia Zakładu Obliczeniowego Zjednoczenia Budownictwa Rolniczego, który prowadził obliczenia na EMC ODRA-1003 w ZETO Szczecin.

Podano: założenia systemu EPD, zakres oddziaływania i wykorzystania systemu, efekty wdrożeniowe — ekonomika systemu, organizacja wdrożenia, baza normatywów, organigram czynności. Celem opracowania jest zaznajomienie przyszłych użytkowników z problematyką wdrożenia SAPI.

● **System kontroli przebiegu prac związanych z opracowaniem programów dla systemu EMC 4—50 oraz dokumentacja sprawozdawczo-statystyczna** — WOJTAN G., MACIELIŃSKI A. — Oprac. wyd. ang. Wyd. Hutniczego Przedsiębiorstwa Maszynowych Obliczeń Analitycznych, Katowice 1969, nlb.

Prowadzenie statystyki z przebiegu prac związanych z oprogramowaniem EMC. Cele systemu kontroli oraz dokumentacji sprawozdawczo-statystycznej. Główne elementy składowe tego systemu. Lista ocen projektu programu EMC. Harmonogram programowania. Karta kontrolna programu. Plan i realizacja prób programu. Analiza błędów. Raport przebiegu prac projektowych nad programem EMC i systemem EPD. Syntetyczny raport przebiegu prac projektowych nad EPD. Tok postępowania przy wprowadzaniu zmian do opracowanego projektu SEPD. Dodatek: formularze dokumentacji systemu kontroli przebiegu prac oraz dokumentacji sprawozdawczo-statystycznej oraz instrukcje ich wypełniania. Materiały szkoleniowe dla programistów.

● **Sformalizowana metoda projektowania programów aktualizacji w przypadku przetwarzania sekwencyjnego** — MACIELIŃSKI A. — Oprac. wyd. ang. Wyd. Ośrodka Badań Ekonomicznych i Organizacji Hutnictwa, Hutniczego Przedsiębiorstwa Maszynowych Obliczeń Analitycznych, Katowice 1969, ss. 16.

Przebiegi aktualizacji zbiorów w ujęciu przetwarzania sekwencyjnego dotyczą w zasadzie zbiorów zapisanych na taśmach magnetycznych. Omówiono przebiegi „scalenia" i „dobierania" jako szczególne przypadki przebiegu aktualizacji oraz projektowanie przebiegów aktualizacji na EMC ICL systemu 4—50. W schematach blokowych posłużono się instrukcjami w języku COBOL.
Opracowanie przeznaczone jest dla osób zaznajomionych z podstawowymi przebiegami EMC.

● **Założenia do systemu automatycznego przetwarzania informacji w agendach technicznego przygotowania produkcji** — BYSIEK S., LASEK M., MILEWSKI L. i inni. Wyd. Centralnego Resortowego Ośrodka Przetwarzania Informacji, Warszawa 1969, ss. 105. Biblioteka opracowań Centralnego Resortowego Ośrodka Informacji. Wybrane zagadnienia z zakresu przetwarzania informacji. Zeszyt nr 10.

Część wstępna: ogólna charakterystyka przedsiębiorstwa i wyjściowego systemu przetwarzania informacji, omówienie założeń do SAPI w Zakładach Elektrotechniki Motoryzacyjnej ZELMOT w Warszawie. Założenia do SAPI w agendach TPP i produkcji: etapy projektowania i realizacja, założenia projektowe SAPI w agendach TPP i produkcji, założenia do emisji dokumentacji warsztatowej przy zastosowaniu automatów piszących OPTIMA-528, wymagania projektowanego systemu. Realizacja prac przygotowawczych do wprowadzenia SAPI w Zakładach ZELMOT. Praca przeznaczona dla projektantów systemów przetwarzania informacji.

opracował **J. Klamborowski**

Stowarzyszenie Elektryków Polskich

rozwija współpracę z przemysłem

3 lipca br. odbyło się kolegium Zjednoczenia Przemysłu Automatyki i Aparatury Pomiarowej MERA, które poświęciło wiele uwagi szczegółom rozwijającej się współpracy ze Stowarzyszeniem Elektryków Polskich. Stało się ono okazją do uroczystego podpisania porozumienia o współpracy pomiędzy Zjednoczeniem i SEP, stanowiącego formalne podsumowanie dotychczasowych form współdziałania.

W uroczystości podpisania porozumienia ze strony SEP wzięli udział: prezes Stowarzyszenia mgr inż. T. Dryzek, wiceprezes mgr inż. H. Gąsowski, sekretarz generalny mgr inż. J. Sypniewska, zastępca sekretarza generalnego inż. D. Malicki, przedstawiciel Sekcji Elektroniki i Telekomunikacji, członek honorowy SEP mgr inż. S. Ignatowicz, zastępca przewodniczącego Centralnej Komisji Automatyki i Pomiarów dr inż. A. Sowiński, przewodniczący Komisji Elektronicznego Przetwarzania Informacji mgr inż. A. Bossowski oraz przedstawiciele zainteresowanych czasopism NOT.

Na dokumencie podpis złożyli: prezes SEP mgr inż. T. Dryzek oraz dyrektor Zjednoczenia mgr inż. J. Huk.

Porozumienie precyzuje węzłowe zagadnienia, które będą przedmiotem współdziałania zarówno na szczeblu organów centralnych SEP i kierownictwa Zjednoczenia, jak i ogniw terenowych w podległych Zjednoczeniu zakładach.

Porozumienie dotyczy w szczególności:

— szkolenia i doskonalenia kadry inżynieryjno-technicznej (sympozja, konferencje, narady, kursy, odczyty),

— udział Stowarzyszenia w opracowywaniu i realizacji planów postępu technicznego oraz rozwijania kultury technicznej w zakładach,

— rozwijania ruchu wynalazczego i racjonalizacji,

— współdziałania w zakresie współpracy naukowo-technicznej z zagranicą, wymiany doświadczeń i informacji naukowo-technicznej,

— zakresu kompetencji i rozwoju działalności kół zakładowych SEP w podległych Zjednoczeniu zakładach i przedsiębiorstwach oraz ich przystępowaniu do SEP w charakterze członków zbiorowych.

W czasie interesującej dyskusji, która rozwinęła się na tle przemówień prezesa T. Dryzka i dyrektora J. Huka, podkreślono konieczność położenia nacisku na rozwój działalności kół zakładowych jako społecznego czynnika opiniującego techniczne problemy zakładu, oddziałującego na kształtowanie się etyki zawodowej inżynierów i techników oraz koleżeńskiego klimatu dobrej współpracy kadry technicznej w zakładzie, przy skoncentrowaniu uwagi na węzłowych problemach technicznych i gospodarczych.

Bardzo pokaźną grupę zagadnień, interesujących środowiska Zjednoczenia MERA, prowadzi Komisja Elektronicznego Przetwarzania Informacji SEP. W najbliższym okresie Komisja organizuje następujące konferencje naukowo-techniczne:

— software podstawowy polskich EMC,

— o metodach przetwarzania informacji niezformalizowanej.

Komisja podejmuje również takie problemy, jak:

— stymulowanie rozwoju klubów użytkowników maszyn ODRA i ZAM,

— inicjatywa w zakresie społecznego opiniowania i oceny sprzętu i systemu informacji w kraju,

— rozwój regionalnych środowisk związanych z elektroniczną techniką obliczeniową,

— organizowanie odczytów specjalistów zagranicznych.

W swej pracy Komisja będzie współpracowała z PK API.

Dyrektor Warszawskich Zakładów Urządzeń Informatyki poruszył w dyskusji bardzo istotne zagadnienia: prawidłowej oceny celowości stosowania maszyn matematycznych w poszczególnych przypadkach, przyzwyczajenia młodych inżynierów do logiki pracy tych maszyn, która powinna być rozumiana nie tylko przez specjalistów zajmujących się programowaniem, lecz również przez szeroki wachlarz użytkowników. Na tym tle istnieje konieczność podjęcia przez SEP akcji popularyzacyjnej i szkoleniowej w omawianym zakresie.

Dyskusja wykazała również, że zawarte porozumienie będzie stanowiło konkretną platformę do dalszego rozwoju działalności organów Zarządu Głównego SEP, Centralnej Komisji Automatyki i Pomiarów oraz Komisji Elektronicznego Przetwarzania Informacji w środowiskach przemysłu automatyki i aparatury pomiarowej.

Ustalono, że po pewnym czasie odbędzie się spotkanie Prezydium Zarządu Głównego SEP z kierownictwem Zjednoczenia w celu podsumowania wyników realizacji porozumienia i wytyczenia kierunków dalszego współdziałania.

DARIUSZ MALICKI
Zastępca Sekretarza Generalnego SEP