11 1979

informatyka

WYDAWNICTWO NOT SIGMA

W NUMERZE

Strona

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
ul. Świętokrzyska 14a
00-950 Warszawa
skrytka pocztowa 1004

Redakcja: 00-041 Warszawa, ul Jasna 14/16, pokój 331, tel. 27-71-40 lub centrala 26-32-61 w. 285, dyżury redakcji 10.00—13.00

Zakł. Graf. „Tamka". Zam. 452. Papier druk. sat. V kl. 70 g. A1. Obj. 5 ark. druk. Nakład 7350. C-94.

Cena 1 egzemplarza zł 25.— INDEKS 36124 Prenumerata roczna zł 300.—

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 11
MIESIĘCZNIK
1979
ROK XIV
listopad

ORGAN KOMITETU INFORMATYKI, MINISTERSTWA NAUKI, SZKOLNICTWA WYŻSZEGO I TECHNIKI ORAZ KOMITETU NAUKOWO-TECHNICZNEGO NOT DS. INFORMATYKI

ARTUR DUBIELEWICZ, JAN MAGOTT
Instytut Cybernetyki Technicznej
Politechniki Wrocławskiej

Sprzętowa realizacja systemów operacyjnych

Istniejące systemy operacyjne (SO) realizowane są prawie w całości w sposób programowy, niekiedy tylko pewne ich funkcje realizowane są sprzętowo. Zwykle są to takie funkcje, które w zasadniczy sposób wpływają na efektywność działania systemu komputerowego. Zdaniem niektórych autorów [3, 4] uzyskanie maksymalnej efektywności systemu komputerowego w znacznej mierze zależy od prawidłowego podziału systemu operacyjnego na części: sprzętową i programową.

Stały rozwój, doskonalenie produkcji, znaczne zmniejszenie wymiarów i kosztów wytwarzania elementów i układów elektronicznych umożliwiają ich zastosowanie do sprzętowych realizacji systemów operacyjnych. Dotychczasowe realizacje wykazują jednak, że nieopłacalne jest ich wykonanie sprzętowe w całości. Wzrost efektywności działania systemu komputerowego dla pewnych funkcji systemu operacyjnego jest zbyt mały w porównaniu z kosztami ich realizacji.

Sprzętowe realizacje systemów operacyjnych można podzielić na realizacje układowe (których przykładem jest SO systemu komputerowego SYMBOL [6]) i realizacje mikroprogramowe (np. SO VENUS [5] i SO BASIC [7]). Z realizacją mikroprogramową wielu autorów wiąże nadzieje na szerokie jej zastosowanie [1, 2]. Podstawową zaletą tego typu realizacji w przypadku pamięci stałej, wielokrotnie zapisywanej, jest możliwość budowania różnych funkcjonalnie systemów operacyjnych dla tej samej maszyny cyfrowej.

W niniejszym artykule przedstawiono system operacyjny SYMBOL jako przykład realizacji układowej oraz systemy operacyjne systemów VENUS i BASIC jako przykłady realizacji mikroprogramowych. W dalszej części zawarte jest oszacowanie złożoności realizacji układowej dużego systemu operacyjnego typu GEORGE 3 dla maszyn cyfrowych serii ICL 1900 i ODRA 1300. Następnie oceniono wady i zalety sprzętowej realizacji systemu operacyjnego.

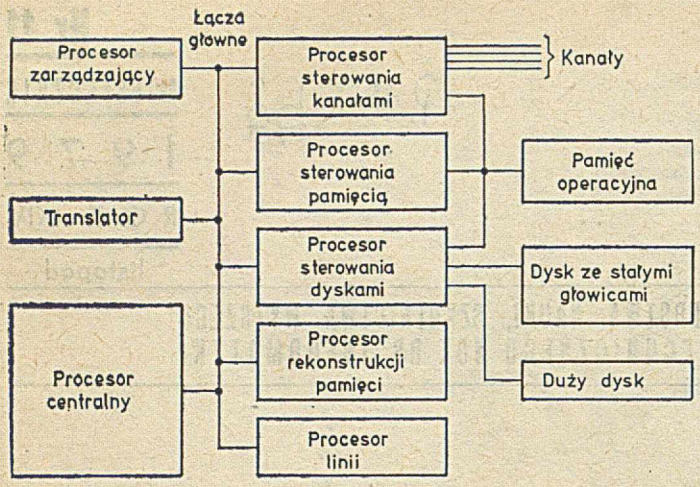
SYSTEM SYMBOL

Przykładem systemu komputerowego, w którym wiele działań wykonywanych jest bez angażowania oprogramowania, jest system SYMBOL [6]. Jest to system eksperymentalny, przeznaczony do badania możliwości przeniesienia funkcji systemu operacyjnego wykonywanych programowo na sprzęt.

SYMBOL jest systemem wielodostępnym ogólnego przeznaczenia. Zawiera on osiem wyspecjalizowanych procesorów. Schemat blokowy tego systemu przedstawiony jest na rys. 1.

Mgr inż. Artur DUBIELEWICZ ukończył Wydział Elektroniki Politechniki Wrocławskiej w 1976 r. W tym samym roku został przyjęty na studia doktoranckie w Instytucie Cybernetyki Technicznej Politechniki Wrocławskiej. Specjalizuje się w teorii systemów operacyjnych.

Dr inż. Jan MAGOTT ukończył Wydział Elektroniki Politechniki Wrocławskiej w 1975 r. W 1978 r. uzyskał tytuł doktora. Pracuje w Instytucie Cybernetyki Technicznej Politechniki Wrocławskiej na stanowisku adiunkta. Interesuje się teorią automatów i systemami operacyjnymi

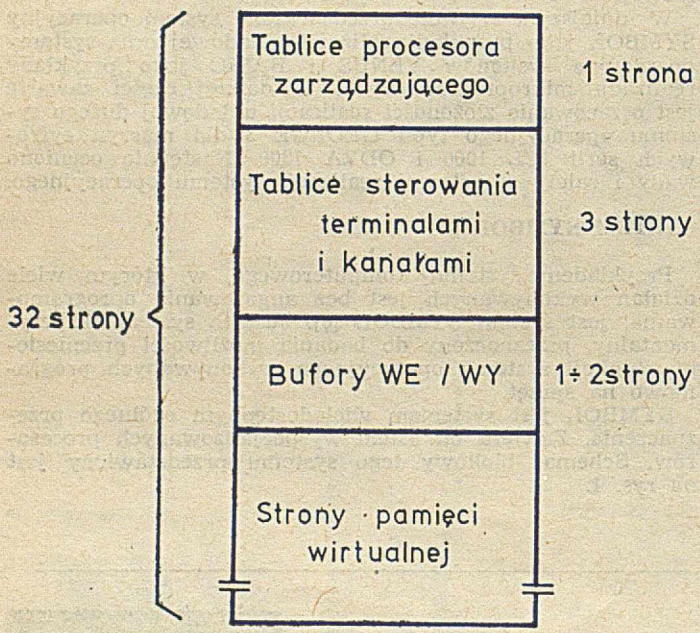


Rys. 1. Schemat blokowy systemu SYMBOL

Pamięć systemu jest pamięcią trójpoziomową. Pamięć operacyjna ma pojemność 8 K słów 64-bitowych i jest pamięcią ferrytową o cyklu 2,5 μs. Jest ona podzielona na 32 strony — po 256 słów na stronie. Szybka pamięć pomocnicza zawiera 500 stron znajdujących się na małym dysku ze stałymi głowicami. Pamięć masowa zawiera 50 000 stron znajdujących się na dużym dysku. Podstawowa konfiguracja systemu zawiera 18 000 podwójnych mikroukładów tranzystorowych.

Podział informacji w pamięci operacyjnej

Szczególnie istotna jest strona zarezerwowana dla procesora zarządzającego. Strona ta zawiera tablice wywołań oprogramowania, słowa sterujące kolejkami, słowa sterujące podziałem pamięci i tablice słów zarezerwowanych dla translatora.



Rys. 2. Podział informacji w pamięci operacyjnej

Podział pamięci

Podział realizowany jest sprzętowo. Podziałem tym steruje procesor sterowania pamięcią. Zamiast zwykłych operacji typu „czytaj", „pisz", pozostałe procesory mają dostęp do zbioru 15 operacji, które umożliwiają kontakt z pamięcią. Procesor sterowania pamięcią steruje również pamięcią asocjacyjną zastosowaną w organizacji pamięci wirtualnej. Procesor rekonstrukcji pamięci reorganizuje zwolnione obszary pamięci na potrzeby nowego użytkownika.

Kompilacja

Translator tłumaczy program napisany w języku wysokiego poziomu SYMBOL na program wynikowy. Translator realizuje kompilację dzięki zorientowaniu języka SYMBOL na translację sprzętową.

Wprowadzanie i wyprowadzanie zadań

Wprowadzanie i wyprowadzanie zadań realizowane są z zastosowaniem procesora linii i procesora sterowania kanałami. Procesor linii redaguje tekst. Wprowadzenie i wyprowadzanie zadań oraz redagowanie tekstu wykonywane są bez angażowania procesora centralnego.

Sterowanie pracą z podziałem czasu

Procesor zarządzający steruje przejściami między stanami pracy systemu. Podstawowymi stanami realizacji zadania użytkownika są: ładowanie, kompilacja, wykonanie, wprowadzanie i wyprowadzanie. Ponieważ system jest systemem wielodostępnym, dodatkowymi stanami są stany terminali.

Wszystkie przejścia między stanami są realizowane (przynajmniej częściowo) za pomocą algorytmów odwzorowanych w sprzęcie. Jedna z ważniejszych zmian stanu realizacji zadania, a mianowicie przejście ze stanu ładowania do stanu kompilacji, wykonywana jest bez stosowania oprogramowania.

W skład procesora zarządzającego wchodzi procesor sterujący kolejkami do wszystkich procesorów pracujących z podziałem czasu. Procesor zarządzający steruje przesyłaniem informacji niezbędnych dla łączności między algorytmami odwzorowywanymi w sprzęcie i procedurami oprogramowania.

Wnioski

Według autorów systemu SYMBOL pewne aspekty pracy sprzętu tego systemu są wielkim osiągnięciem w porównaniu z pracą procesorów sterowanych za pomocą programów. Sprzętową realizację podziału i sterowania pamięcią uważają oni za dużą zaletę przedstawionego systemu w porównaniu z klasycznymi systemami cyfrowymi. W pewnych krytycznych przypadkach sterowania pamięcią i jej podziału oraz w zakresie podziału zasobów systemu uzyskana efektywność jest 10—100 razy większa od programowej. Wzrost efektywności pracy z dużymi zbiorami jest znacznie mniejszy.

SYSTEM VENUS

System operacyjny VENUS [5] jest interakcyjnym wieloprogramowym systemem konwersacyjnym zbudowanym dla małej mikroprogramowej maszyny INTERDATA 3. Głównym celem projektantów systemu VENUS było pokazanie, że odpowiednia architektura maszyny cyfrowej może znacznie ułatwić projektowanie systemu oprogramowania.

Mikroprogramowana maszyna INTERDATA 3 umożliwia projektantom systemu uzyskanie charakterystyki maszyny najlepszej dla założonych zastosowań.

Mikropamięć INTERDATA 3 zawiera 2 000 instrukcji. Pamięć operacyjna nie może przekraczać 64 Kbajtów. Pamięciami zewnętrznymi są dwie jednostki pamięci taśmowej i jeden półmegabajtowy dysk. Urządzenia te są podłączone do procesora poprzez kanały sprzętowe. Natomiast urządzenia wejścia/wyjścia, takie jak czytnik kart, drukarka wierszowa i terminale, współpracują z procesorem poprzez mikroprogram stymulujący pracę kanału.

System operacyjny VENUS ma strukturę poziomą. W strukturze tego systemu występują następujące poziomy: 1) procesorów wirtualnych, 2) pamięci wirtualnej, 3) konsol wirtualnych, 4) wirtualnych urządzeń wejścia/wyjścia, 5) programów użytkowników.

System operacyjny VENUS może realizować maksymalnie 16 procesów, które komunikują się między sobą poprzez semafory. Ponadto semafory wykorzystywane są przez procesory do synchronizacji z urządzeniami wejścia/wyjścia i do przesyłania komunikatów poprzez kolejki.

Pamięć systemu jest podzielona na segmenty zawierające 256 stron. Strona składa się z 256 bajtów.

W celu uzyskania znacznej szybkości sprawdzania, czy strona znajduje się w pamięci operacyjnej, funkcja ta realizowana jest za pomocą mikroprogramu. Jeżeli wymagana strona nie zostanie znaleziona, to mikroprogram wywołuje program poszukujący tej strony w pamięci zewnętrznej.

Sterowanie procesami, sterowanie pamięcią wirtualną i symulacja kanału dla urządzeń wejścia/wyjścia realizowane są mikroprogramowo i charakteryzują się dużą szybkością działania. Pewne funkcje systemu operacyjnego — ze względu na wymaganą ich elastyczność — zrealizowane są w sposób programowy. Przykładami funkcji zrealizowanych programowo są: szeregowanie zadań, wyszukiwanie stron w pamięci zewnętrznej, sterowanie kolejkami, przyporządkowywanie zewnętrznym nazwom segmentów nazw wewnętrznych.

SYSTEM BASIC

System komputerowy BASIC [7] zbudowany jest na bazie minikomputera WANG 2200 A. System ten służy do wykonywania programów napisanych w języku konwersacyjnym BASIC. Jednostka centralna zawiera pamięć operacyjną o pojemności 4 Kbajtów. Pamięć ta może być rozbudowana do pojemności 32 Kbajtów przez wmontowanie do tej samej obudowy dodatkowych bloków pamięci po 4 K. Znaczna część systemu operacyjnego i translator języka BASIC są wbudowane do osobnej części jednostki centralnej w postaci gotowych układów mikroprogramowych. W pamięci operacyjnej systemu znajduje się jedynie program zarządzający zbiorami w pamięci zewnętrznej, zajmujący około 700 bajtów. Dzięki temu prawie cała pamięć operacyjna jest dostępna dla użytkownika.

OSZACOWANIE ZŁOŻONOŚCI AUTOMATU ODWZOROWUJĄCEGO SO

W celu oszacowania złożoności automatu wykonującego funkcje systemu operacyjnego należałoby rozpatrzyć system operacyjny o organizacji i języku wewnętrznym ukierunkowanym na realizację sprzętową. Autorzy nie znaleźli w literaturze informacji na temat dużego systemu operacyjnego zrealizowanego w taki sposób. System operacyjny SYMBOL jest przykładem niedużego systemu.

Próba oszacowania złożoności automatu odwzorowującego duży system operacyjny jest kontrowersyjna ze względu na konieczność przyjęcia założeń w procesie szacowania. Próba taka zostanie przeprowadzona dla systemu operacyjnego GEORGE 3. Zakłada się, że automat odwzorowujący wymieniony system operacyjny generuje takie same instrukcje, jak instrukcje wchodzące w skład systemu operacyjnego. System operacyjny GEORGE 3 w wersji języka wewnętrznego zajmuje ok. 600 K słów pamięci. Przyjmuje się, że automat odwzorowujący analizowany system operacyjny jest automatem typu Moore'a. Literami alfabetu wyjściowego tego automatu są instrukcje w języku wewnętrznym, każda instrukcja wchodzącą w skład systemu operacyjnego generowana jest w innym stanie. Z powyższego wynika, że rozważany automat musi mieć co najmniej 600 K stanów. Instrukcja języka wewnętrznego jest ciągiem 24 bitów. Stąd rozważany automat musi mieć 24 układy kombinacyjne dla generacji bitów słowa maszynowego. Argumentami funkcji realizowanych za pomocą tych układów są zmienne związane ze stanami przerzutników, za pomocą których zakodowany jest stan automatu, natomiast wartościami — pozycje słowa maszynowego. Złożoność automatu zależy od złożoności układów kombinacyjnych dla wymienionych wyżej 24 wyjść i układów kombinacyjnych generujących sygnały pobudzające przerzutniki. We współczesnych systemach cyfrowych odpowiednikiem zmiany stanu rozważanego automatu jest zmiana stanu licznika rozkazów. Stąd złożoność układów kombinacyjnych generujących sygnały pobudzające przerzutniki jest podobna do złożoności układu służącego do wyznaczania adresów kolejno wykonywanych instrukcji, czyli nie jest duża. Czynnikiem decydującym o złożoności rozważanego automatu jest zatem złożoność układów kombinacyjnych dla 24 wyjść.

Aparatem formalnym przydatnym w procesie syntezy układów kombinacyjnych jest algebra Boole'a. Stosując algebrę Boole'a autorzy uzyskali przybliżoną ocenę liczby bramek analizowanego automatu, równą 300 tys. Liczba bramek w jednostce centralnej ODRA 1305 jest w przybliżeniu dziesięć razy mniejsza. Ponieważ jednostka centralna ma wyjątkowo duży wpływ na całkowity koszt systemu komputerowego, bardzo kosztowna jest sprzętowa realizacja dużego systemu operacyjnego.

*
* *

Na tle powyższych rozważań nasuwają się następujące wnioski:

- Brak uzasadnienia dla realizacji całego systemu operacyjnego w sposób sprzętowy

Ze względu na złożoność automatu odwzorowującego duży system operacyjny, realizacja całego systemu operacyjnego w sposób sprzętowy jest bardzo kosztowna.

System operacyjny powinien być łatwo modyfikowalny. Często jest on rozbudowywany, a pewne algorytmy są zastępowane nowymi algorytmami. Możliwości zmian w oprogramowaniu są oczywiście znacznie większe niż w przypadku sprzętu.

W fazie konstruowania sprzętowego systemu operacyjnego należałoby rozwiązać również wiele problemów niezawodnościowych.

Proces uruchamiania urządzenia wykonującego funkcje dużego systemu operacyjnego oraz proces diagnozowania tak dużego urządzenia byłyby bardzo złożone.

- Sprzętowe sterowanie pamięcią

Dzięki zastosowaniu pamięci buforowej lub asocjacyjnej można uniknąć bardzo wolnego programowego przeglądania tablic stron. W przypadku zastosowania szybkiej pamięci buforowej w pamięci tej przechowywana jest tablica stron. Rozwiązanie to ze względu na znaczny koszt pamięci buforowej jest bardzo drogie. W przypadku zastosowania pamięci asocjacyjnej w rejestrach tej pamięci przechowywane są zwykle informacje o aktywnych stronach. Rozwiązanie to ze względu na istnienie rejestrów zawierających informacje o stronach i wielu układów szybkiego porównywania również jest bardzo kosztowne.

Sprzętowe sterowanie pamięcią zapewnia:

— znaczną szybkość sprawdzania, czy wymagana strona znajduje się w pamięci operacyjnej
— znaczną szybkość sprowadzania żądanej strony do pamięci operacyjnej
— nieangażowanie procesora długimi przeszukiwaniami, dzięki czemu wzrasta przepustowość systemu cyfrowego.

- Sprzętowe sterowanie wieloprogramowością

Sprzętowe sterowanie wieloprogramowością ma następujące zalety:

— szybkie przełączanie programów, dzięki czemu przedział wartości dla doboru odpowiednich kwantów czasu może być większy
— szybkie wykrywanie sytuacji impasowych, dzięki zastosowaniu układów wykrywających takie stany [8].

- Brak uzasadnienia dla realizacji sprzętowej sterowania dużymi strukturami danych w pamięciach zewnętrznych

Brak tego uzasadnienia wynika ze zbyt małego wzrostu efektywności w zakresie wyszukiwania informacji w porównaniu z kosztami wymaganego sprzętu.

LITERATURA:

[1] Bruce S., Shriver S.: Microprogramming and numerical analysis. IEEE Trans. on Comp. C-20, No 7/1971
[2] Buliej C.: Mikroprogrammirowaniye. Izdatielstwo „Mir", Moskwa 1973
[3] Głuszkow B. M. i in.: Woprosy razwitiya struktur CWM w swiazi s sistiemami ikhmatiematiczieskowo obiespieczieniya, Kibiernietika No 10/1967
[4] Katzan H. jr.: Computer Organization and the System/370. Van Nostrand Reinhold Co., New York 1971
[5] Liskov B.: The Design of the Venus Operating System. Comm. of ACM, vol. 15, No 3 (March) 1972
[6] Smith W. R., and al.: SYMBOL — A large Experimental System Exploring Major Hardware Replacement of Software. Spring Joint Computer Conf., 1971
[7] Wolpe J., Wolpe M.: Podręcznik systemu BASIC. Warszawa 1973
[8] Nezu K.: Deadlock Detector. Trans. Inst. Electron. and Commun. Eng. JPN. Sect. E, vol. E-60, No 10 (October) 1977

RYSZARD GILECKI, WOJCIECH SAWICKI
Centrum Informatyki Energetyki i Energii Atomowej
WŁODZIMIERZ TABACZYŃSKI
Państwowy Zakład Higieny
Warszawa

Program formatujący teksty

Program DOKUMENT służy do formatowania tekstów. Napisany został w języku PASCAL. Pierwszą jego wersję uruchomiono na maszynie CDC CYBER 72 w 1977 r., drugą — w lipcu br. na maszynie R-32. Zadaniem programu jest przekształcenie podanego przez użytkownika tekstu wejściowego na tekst wynikowy w formie książki, przy czym zasady formatowania określa użytkownik.

W wielu ośrodkach, niezależnie od siebie, opracowano podobne programy na własne potrzeby, np. [2, 3]. Ze względu na lokalne przeznaczenie programy te na ogół charakteryzują się stosunkowo niewielkim zestawem funkcji. Celowe więc wydawało się uruchomienie programu stanowiącego uogólnienie dotychczasowych, a jednocześnie łatwego w eksploatacji.

FUNKCJE PROGRAMU

Sposób przekształcenia tekstu wejściowego na tekst wynikowy jest określany przez użytkownika za pomocą specjalnych dyrektyw (rozkazów i dyspozycji), realizujących następujące funkcje:

- ustalenie sposobu przetwarzania tekstu wejściowego, np. wybór trybu wypełniania lub kopiowania
- określenie kształtu formatowanej strony (szerokości i wysokości zadrukowanego pola, podział na kolumny itp.)
- opis tekstu wejściowego, np. zdefiniowanie zbioru znaków przestankowych
- podjęcie odpowiedniej jednorazowej akcji: przejście do nowego wiersza, akapitu, kolumny lub strony, pozostawienie miejsca na rysunek
- określenie postaci spisu treści, interpretacja fragmentów tekstu jako tytułów rozdziałów
- uznanie fragmentu tekstu za odnośnik i umieszczenie go u dołu strony
- określenie postaci indeksu i umieszczenie w nim żądanych słów
- odwołanie do pojęć zdefiniowanych w innym miejscu tekstu, np. (str. 15).

Wyróżniono dwa rodzaje dyrektyw: rozkazy i dyspozycje. Rozkaz powoduje zmianę wartości parametru sterującego formatowaniem (np. wielkości odstępów między słowami) bądź specyfikuje czynność, która ma być wykonana jednorazowo w momencie napotkania rozkazu (np. przejście do nowej strony). Dyspozycja natomiast określa specjalne przeznaczenie niektórych fragmentów tekstu (np. tytuł rozdziału, odnośnik). Duża liczba dyrektyw (44 rozkazy i 6 dyspozycji) oraz przyjęcie zasady, że każdy z parametrów sterujących formatowaniem tekstu może być zmieniony w dowolnym momencie, zapewniają użytkownikowi bogate możliwości przekształcania tekstu.

Każdy z parametrów sterujących formatowaniem tekstu ma przypisaną wartość domyślną, odpowiadającą oczekiwaniom większości użytkowników. Najprostszy sposób działania programu polega zatem na sformatowaniu tekstu wejściowego zgodnie z wartościami domyślnymi parametrów. Wyklucza to jednak korzystanie z wielu ciekawych własności programu DOKUMENT, jak np. tworzenie spisu treści, indeksu, umieszczanie nagłówków na stronach tekstu wynikowego itd.

Standardowo tekst wynikowy drukowany jest na drukarce wierszowej. W celu uzyskania większej liczby egzemplarzy wyników bez ponownego wykonywania programu można zapisać tekst wynikowy w zbiorze dyskowym lub taśmowym, a następnie wielokrotnie wydrukować jego zawartość.

Program DOKUMENT dokładnie kontroluje poprawność danych wejściowych. Badana jest przede wszystkim poprawność syntaktyczna i semantyczna dyrektyw. W przypadku napotkania błędu syntaktycznego następujący po nim fragment tekstu wejściowego aż do ogranicznika odcinka dyrektyw (patrz opis języka wejściowego) jest ignorowany. W przypadku błędu semantycznego (np. wartość pewnego parametru sterującego formatowaniem koliduje z wartościami innych, wcześniej określonych parametrów) dyrektywa jest ignorowana lub parametrowi przypisana zostaje wartość domyślna. Wystąpienie każdego błędu powoduje ponadto wygenerowanie odpowiedniego komunikatu diagnostycznego.

Kopia tekstu wejściowego wraz z komunikatami diagnostycznymi i dodatkowymi informacjami o przebiegu pracy programu (np. listami aktualnych wartości parametrów) może być zapisana do zbioru wyjściowego, różnego od zbioru zawierającego tekst wynikowy.

Ustalenie sposobu przetwarzania tekstu wejściowego

Program DOKUMENT przewiduje dwa podstawowe tryby pracy: wypełnianie i kopiowanie. Tryb wypełniania (przyjmowany domyślnie) oznacza, że kolejne wiersze tekstu wynikowego (książki) będą wypełniane zgodnie z aktualnie obowiązującymi wartościami parametrów justowania, odstępów między słowami, wcięć pierwszych wierszy akapitów itp. Tryb kopiowania natomiast oznacza, że kolejne wiersze tekstu wynikowego mają być wierną kopią rekordów tekstu wejściowego. Umożliwia to zamieszczanie w tekście np. fragmentów poezji czy tekstów źródłowych programów.

Użytkownik może ustalić, że formatowany tekst ma być justowany, tzn. ostatnie słowo w każdym wierszu (z wyjątkiem ostatniego wiersza akapitu) ma być wyrównywane do prawego marginesu, a wszystkie słowa wiersza równomiernie rozsunięte.

Mgr Ryszard GILECKI ukończył studia w 1978 roku na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Od 1977 r. pracuje w Centrum Informatyki Energetyki i Energii Atomowej, w Pracowni Oprogramowania Podstawowego.

Mgr Wojciech SAWICKI ukończył studia w 1978 roku na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Od 1977 r. pracuje w Centrum Informatyki Energetyki i Energii Atomowej, w Pracowni Oprogramowania Podstawowego.

Do grupy dyrektyw ustalających sposób przetwarzania tekstu wejściowego należą również rozkazy specyfikujące wielkości odstępów między znakami w słowie („rozstrzelony" druk), słowami w wierszu i wierszami na stronie tekstu wynikowego, a także wielkości wcięć pierwszych linii akapitów.

Określenie kształtu formatowanej strony

Program DOKUMENT pozwala na swobodny wybór wielkości marginesów na stronie, a dzięki temu na ustalenie wysokości i szerokości druku. Jedną z ciekawszych własności programu jest możliwość drukowania tekstu wynikowego w kolumnach, przy czym określenie liczby kolumn, ich szerokości i odstępów między nimi należy do użytkownika.

Formatowane strony mogą zawierać dodatkowe wiersze u góry i u dołu każdej strony, nazywane dalej nagłówkami. Składają się one z trzech pól, umieszczonych odpowiednio w lewym rogu, na środku i w prawym rogu strony, przy czym każde z tych pól może być puste. Istnieje ponadto możliwość określania różnych nagłówków dla stron o numerach parzystych i nieparzystych. Poszczególne pola nagłówków mogą zawierać dowolny tekst lub numer aktualnego rozdziału albo strony.

Opis tekstu wejściowego

Ciągi rozkazów i dyspozycji ujmowane są w specjalne nawiasy ograniczające. Domyślnie są nimi dwa rzadko używane znaki, przy czym można je zmienić za pomocą odpowiednich rozkazów.

Wśród znaków tworzących tekst wejściowy szczególną rolę odgrywają znaki przestankowe. W trybie wypełniania są one dołączane bez odstępów do poprzedzającego słowa, niezależnie od liczby spacji oddzielających je od tego słowa w tekście wejściowym. Zbiór znaków traktowanych przez program jako przestankowe może składać się z dowolnych znaków, nie tylko takich, które są uznawane za znaki przestankowe z punktu widzenia interpunkcji. Zbiór ten można zdefiniować za pomocą specjalnego rozkazu (można oczywiście skorzystać z domyślnego zestawu, zawierającego znaki ogólnie przyjęte za przestankowe).

Zarówno znaki przestankowe, jak i ograniczniki odcinków rozkazów i dyspozycji kończą słowa tekstu wejściowego. Podobnie terminatorem słowa jest spacja (chyba, że po niej wystąpi znak przestankowy — jest on wówczas dołączony do słowa). Nie jest natomiast terminatorem słowa koniec rekordu tekstu wejściowego. Domyślnie przyjmuje się, że rekord tekstu wejściowego zawiera 80 znaków; wielkość tę można jednak zmienić. Ma to szczególne znaczenie w przypadku, gdy tekst wejściowy zapisany jest w pewnym zbiorze utworzonym za pomocą programu typu UPDATE, wykorzystującego przeważnie ostatnie pola rekordu (kolumny karty) na identyfikację.

Tytuły rozdziałów

Program DOKUMENT pozwala na tworzenie spisu treści oraz na automatyczne numerowanie rozdziałów. Dopuszczalna jest trójpoziomowa numeracja rozdziałów. W tekście wejściowym wystarczy określić poziom danego rozdziału oraz podać jego tytuł.

Mgr Włodzimierz TABACZYŃSKI ukończył studia w 1978 roku na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Od 1977 r. pracuje w Państwowym Zakładzie Higieny, w Zakładzie Statystyki Medycznej.

W sformatowanym tekście tytuł zostanie wyodrębniony w sposób domyślny lub określony przez użytkownika (przykładowo można zażądać, aby wszystkie rozdziały poziomu pierwszego rozpoczynały się od nowej strony). Tytuły wszystkich rozdziałów wraz z odpowiadającymi im numerami stron umieszczane są w spisie treści.

Odnośniki

Określone fragmenty tekstu wejściowego mogą stanowić odnośniki. Każdy taki fragment umieszczany jest w dolnej części strony i oznaczany numerem, który wystąpi też w miejscu odwołania do odnośnika. W przypadku, gdy cały odnośnik nie mieści się na danej stronie, jego końcowa część zostaje przeniesiona do następnej strony.

Indeks

Możliwe jest umieszczenie dowolnego słowa lub sekwencji słów w indeksie. Indeks, składający się ze wszystkich haseł, posortowanych w porządku alfabetycznym, zostanie wydrukowany na końcu tekstu wynikowego. Obok każdego hasła wypisane będą numery stron lub rozdziałów, w których napotkano dyspozycję nakazującą przesłanie tego hasła do indeksu.

Odwołania

Ciekawą własnością programu DOKUMENT jest generowanie odwołań do pojęć zdefiniowanych w innym miejscu tekstu w postaci np. (str. 7) lub (par. 2). Można to osiągnąć za pomocą dyspozycji definiującej odpowiednie etykiety w miejscach, do których i z których ma nastąpić odwołanie, przy czym odwołanie to może być do strony lub do rozdziału.

SPOSÓB OPRACOWANIA PROGRAMU DOKUMENT

Program DOKUMENT tworzony był przy użyciu metodologii programowania strukturalnego [1, 4]. W sposób szczególnie rygorystyczny przestrzegano zasad projektowania zstępującego. Wydaje się, że warto poświęcić kilka słów na opis tego procesu, jak i na płynące z niego wnioski.

Język wejściowy i budowa programu

Po sprecyzowaniu funkcji programu opracowano język wejściowy i opisano jego składnię przy użyciu notacji BNF, rozszerzonej o konstrukcję iteracyjną. Poniżej przedstawiono podstawową część opisu składni języka wejściowego:

1. <tekst wejściowy> ::={<odcinek>}
2. <odcinek>::= <odcinek tekstu>|<odcinek rozkazów>|<odcinek dyspozycji>
3. <odcinek tekstu>::={<znak tekstu>}
4. <odcinek rozkazów>::= <ogranicznik odcinka rozkazów> <ciąg rozkazów> <ogranicznik odcinka rozkazów>
5. <odcinek dyspozycji> ::=<ogranicznik odcinka dyspozycji> <ciąg dyspozycji> <ogranicznik odcinka dyspozycji>
6. <ciąg rozkazów> ::= <rozkaz> {,<rozkaz>}
7. <ciąg dyspozycji> ::= <dyspozycja> {,<dyspozycja>}
8. <rozkaz> ::= <rozkaz 1> |.....| <rozkaz m>
9. <dyspozycja> ::= <dyspozycja 1> |.....| <dyspozycja n>

Zastosowanie notacji BNF do opisu języka wiąże się w sposób naturalny z regułami projektowania zstępującego. Kolejne kroki prowadzące od symbolu początkowego do symboli terminalnych odpowiadają kolejnym etapom projektowania. Na najwyższym poziomie hierarchii (program główny) następuje podział na trzy jednostki funkcjonalne (procesor tekstu, procesor rozkazów i procesor dyspozycji), odpowiadające trzem wartościom zmiennej metajęzykowej <odcinek>. W następnym kroku dokonywany jest podział na jednostki realizujące:

— przetwarzanie w trybie wypełniania lub kopiowania (procesor tekstu)
— poszczególne rozkazy (procesor rozkazów)
— poszczególne dyspozycje (procesor dyspozycji)

Każda tak określona jednostka funkcjonalna programu została zapisana w PASCALU w postaci procedury lub elementu listy wyboru instrukcji case.

Zalety przyjętej metody

Przestrzeganie przyjętej metody tworzenia programu istotnie ułatwiło znajdowanie błędów w fazie testowania, a także uprościło modyfikowanie i rozszerzanie programu. Rozszerzanie języka wejściowego i działającego programu o nowe elementy wykonywane było kilkakrotnie i nie przedstawiało żadnych trudności. Przyjęta struktura modularna programu, jak również konsekwentne nadawanie znaczących nazw zmiennym używanym w programie, umieszczenie dużej liczby komentarzy i rygorystyczne akapitowanie tekstu programu sprzyjały czytelności i sprawiły, że sam tekst programu jest w dużej mierze jego dokumentacją.

Odstępstwa od struktury modularnej

Mimo systematycznego sposobu projektowania naszkicowana struktura modularna programu nie jest wolna od pewnych odstępstw. Zostały one jednak wprowadzone świadomie i były logiczną konsekwencją zamierzonych możliwości programu. Najistotniejsze odstępstwa dotyczą współdziałania między procesorem dyspozycji a procesorem tekstu. W trybie przetwarzania dyspozycji następują bowiem odwołania do procesora tekstu, ponieważ tytuły rozdziałów i teksty odnośników formatowane są zgodnie z takimi samymi regułami jak tekst właściwy. Warto wspomnieć, iż zapewnienie poprawności działania tych właśnie elementów programu przedstawiało największą trudność. Należy sądzić, że było to spowodowane wspomnianymi odstępstwami od struktury modularnej.

Język implementacji

Dla implementacji wybrano język PASCAL, szczególnie zalecany przez teoretyków programowania strukturalnego [5]. PASCAL dzięki swojemu repertuarowi konstrukcji oraz bogactwu struktur danych ułatwia czytelne, zwięzłe i zdyscyplinowane programowanie. W szczególności przetwarzanie tekstów jest w tym języku wygodne dzięki przeznaczonemu do tego celu zespołowi obiektów (typy standardowe char i text, funkcje standardowe ord i chr).

Program DOKUMENT wykorzystuje rozmaite typy i metody strukturalizacji danych, operując przede wszystkim na obiektach o wartościach znakowych, a także plikach i rekordach o różnorodnej budowie. Posługiwanie się takimi strukturami danych jest w PASCALU naturalne i czytelne. Inne powszechnie używane języki programowania nie dają takich możliwości — albo nie dysponują w ogóle wieloma typami danych (FORTRAN, ALGOL) albo operują nimi w sposób daleko mniej czytelny (COBOL, PL/I).

Jeszcze jednym argumentem przemawiającym za stosowaniem PASCALA jest łatwa przenaszalność programów. Kompilatory PASCALA istnieją dziś w oprogramowaniu większości znanych typów komputerów (w szczególności posiadają je maszyny dominujące w Polsce: IBM/RIAD, ICL/ODRA, CDC). Przenoszenie programów napisanych w PASCALU między wymienionymi typami komputerów nie przedstawia praktycznie żadnych trudności. Program DOKUMENT został przeniesiony, jak zaznaczono we wstępie, z maszyny CDC CYBER 72 na R-32. Jedyne zmiany w tekście programu konieczne przy przenoszeniu wiązały się z innym sposobem traktowania upakowanych struktur danych oraz nieco zmienioną reprezentacją znaków.

Wymagania sprzętowe

Wykonanie programu DOKUMENT na maszynie R-32 pracującej pod kontrolą systemu operacyjnego OS/MFT wymaga około 70 KB pamięci operacyjnej oraz jednej jednostki pamięci dyskowej typu IBM 2314 (EC 5061) lub IBM 2311 (EC 5052). Dla danych o wielkości 100 stron tekstu wynikowego konieczne jest przydzielenie 10 cylindrów dysku IBM 2314 lub 30 cylindrów dysku IBM 2311. Zamiast jednostki dyskowej można wykorzystać jednostki pamięci taśmowej — konieczne są wówczas 4 jednostki. Prędkość przetwarzania tekstu na maszynie R-32 wynosi 300—400 znaków wejściowych na 1 sekundę pracy jednostki centralnej.

LITERATURA:

[1] Dahl O. J., Dijkstra E. W., Hoare C. A. R.: Structured Programming. Academic Press London 1972

[2] Kaczmarska E.: Automatyczne łamanie tekstów wydawniczych. Sprawozdania Instytutu Informatyki UW nr 61, Warszawa 1977

[3] Tesler L.: PUB — The Document Compiler. Stanford Artificial Intelligence Project, Operating Note 70, September 1972

[4] Turski W. M.: Metodologia programowania. WNT Warszawa 1978

[5] Wirth N.: Algorithms + Data Structures = Programs. Prentice-Hall 1976

ZENON MITAL
Instytut Informatyki
Politechniki Gdańskiej

Jednym z tematów prac nad realizacją systemów oprogramowania czasu rzeczywistego, prowadzonych od kilku lat w Zespole Oprogramowania Podstawowego Maszyn Cyfrowych (Instytut Informatyki Politechniki Gdańskiej), jest niezawodność oprogramowania. W poniższym artykule Autor wykorzystuje doświadczenia uzyskane w trakcie tych prac, wzbogacone przeglądem odnośnej tematyki w literaturze światowej.

Niezawodność oprogramowania

Pojęcie niezawodności zawiera się w sformułowaniu, że niezawodny system komputerowy [1]) to taki system, którego zachowanie jest zawsze zgodne z wymaganiami określonymi przy wyznaczaniu jego funkcji [1, 2]. I w tym właśnie aspekcie będą poniżej omawiane zagadnienia niezawodności oprogramowania.

Z niezawodnością wiąże się ściśle pojęcie błędu systemu. Pojęcie to można rozpatrywać w kategoriach przyczynowych lub skutkowych. W pierwszym przypadku pod pojęciem błędu rozumie się przyczynę niezamierzonych zmian transformacji funkcjonalnych realizowanych przez dany element systemu. W drugim przypadku pojęcie błędu obejmuje niepoprawne wartości sygnałów logicznych występujących w dowolnych elementach systemu komputerowego. Należy podkreślić, że oba pojęcia błędu często stosowane są zamiennie.

W zależności od tego, w jakim etapie rozwoju systemu tkwią źródła błędów, można je sklasyfikować w dwóch następujących grupach:

1) błędy powstałe podczas projektowania i produkcji systemu komputerowego (błędna implementacja pierwotnych wymagań)

2) błędy powstałe podczas eksploatacji systemu w wyniku wpływów wewnętrznych lub zewnętrznych (np. starzenie się elementów).

Błędy oprogramowania należą do pierwszej grupy, natomiast błędy sprzętu — do obu grup.

Idealnym spełnieniem wymogu niezawodności jest stworzenie i użytkowanie systemu wolnego od błędów. W praktyce stosuje się łagodniejsze kryteria jeżeli tylko
— błędy nie powodują (lub powodują tylko częściowe) zubożenia funkcji systemu
— konsekwencje błędów nie powodują przerwy w eksploatacji systemu.

Aby maksymalnie wyeliminować błędy oprogramowania lub przynajmniej łagodzić ich skutki (a także skutki defektów sprzętu), stosuje się najrozmaitsze zabiegi, które można zaliczyć do następujących kategorii działań:

- zapobieganie powstawaniu błędów
- wykrywanie i usuwanie błędów
- odtwarzanie stanu, jaki istniał przed wystąpieniem błędów.

ZAPOBIEGANIE POWSTAWANIU BŁĘDÓW

Środki zapobiegające powstawaniu błędów powinny być stosowane na wszystkich etapach rozwoju systemu. Istota ich polega na zapewnieniu zgodności rzeczywistych intencji z opisem wymagań wyrażanym w różnych językach na różnych poziomach rozwoju systemu. Sprzyja to zapewnieniu zgodności ostatecznej implementacji z pierwotnymi wymaganiami.

Już na etapie wstępnym należy liczyć się z powstawaniem błędów — wymagania formułowane przez specjalistów różnych dyscyplin i wyrażane mniej lub bardziej precyzyjnie w języku potocznym lub w językach specjalistycznych muszą być przetransformowane na postać formalnych wymagań do projektu systemu. Obserwuje się rozwój środków wspomagających te zadania — głównie języków formalnego wyrażania wymagań.

Kolejnym czynnikiem rzutującym na niezawodność oprogramowania jest metodologia projektowania. Stworzono odpowiednie metody, z których najważniejsze opierają się na koncepcji programowania strukturalnego.

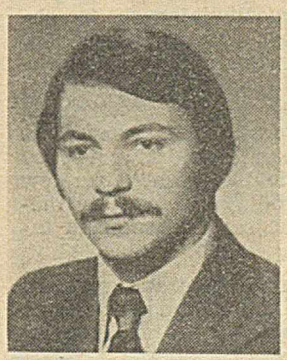
Koncepcja programowania strukturalnego stosowana w językach programowania zbliża zachowanie dynamiczne programu z jego statycznym obrazem. Zwiększa to czytelność programu, ułatwiając tym samym niezbędną weryfikację wymagań. Ponadto struktura takiego programu jest dogodniejsza do testowania.

Istotne jest również wykorzystanie problemowych języków programowania. Pozwalają one na wyrażanie intencji programisty bardziej jawnie i przejrzyście, w formie zbliżonej do języka specjalistycznego. Dzięki temu można uzyskać lepszą komunikację zamierzeń i celów programu, a specjaliści formułujący pierwotne wymagania mogą uczestniczyć w kolejnych stadiach rozwoju systemu oprogramowania.

Rozważając wpływ języka programowania na poprawność programu, w pierwszym rzędzie należy wymienić wprowadzanie zasad wykluczających popełnianie przez programistę przypadkowych błędów mechanicznych (eliminacja konstrukcji językowych podatnych na popełnianie błędów) oraz stosowanie środków pomocnych podczas uruchamiania i testowania programów [5].

Częstym powodem błędnego działania programu jest to, że narusza on specyfikacje języka, w którym jest pisany, a system tego języka nie wykrywa takich naruszeń. Stąd nieokreślone wyniki pewnych operacji. Można temu zapobiec, jeżeli każdy warunek zakazany przez specyfikację języka będzie odpowiednio wcześnie wykryty.

[1]) Należy pamiętać, że na niezawodność systemu komputerowego składa się odpowiednie działanie zarówno sprzętu, jak i oprogramowania



Mgr inż. Zenon MITAL w 1972 r. ukończył studia na Wydziale Elektroniki Politechniki Gdańskiej. Pracuje w Instytucie Informatyki PG. Specjalizuje się w problematyce oprogramowania komputerowych systemów czasu rzeczywistego, szczególnie w zastosowaniu do sterowania procesami.

Obok bezwzględnych zalet wyżej wymienionych języków wysuwa się pod ich adresem wiele zastrzeżeń dotyczących głównie wydajności programowania. Wymaganie wielu dodatkowych deklaracji oraz instrukcji o charakterze kontrolnym jest uciążliwe i niewygodne. Nie zawsze uzyskiwany wzrost niezawodności usprawiedliwia tę uciążliwość powodującą wzrost czasochłonności kodowania programu. Ponadto nie zawsze można zaakceptować znaczny wzrost zapotrzebowania na pamięć i czas maszyn wynikający ze stosowania wspomnianych języków.

WYKRYWANIE I USUWANIE BŁĘDÓW

Wśród metod i środków ułatwiających uzyskanie systemu oprogramowania o dużej pewności działania ważne miejsce zajmują testowanie i narzędzia uruchamiania programów. Często podkreśla się, że zadania te pochłaniają większość wysiłków ponoszonych w procesach tworzenia i implementacji oprogramowania.

Szeroko pojmowane testowanie ma pokazać, że program i wymagania funkcjonalne korespondują ze sobą. Nie wystarczy tu troska o poprawność przepływu informacji, ale należy również ocenić, w jakim stopniu poziom opisu wyraża rzeczywiste zamiary [5]. Testowanie odbywa się drogą różnego typu operacji przeznaczonych do identyfikacji błędów i dostarczania dowodów, że oprogramowanie spełnia określone wymagania. Testowanie może odbywać się na etapie kompilacji programów lub podczas ich próbnego działania. Wyczerpujące testowanie złożonych systemów oprogramowania jest przedsięwzięciem trudnym i nie zawsze wykonalnym z uwagi na dużą liczbę ścieżek przepływu sterowania. Aby to zadanie ułatwić, system dzieli się na podsystemy, które można testować niezależnie.

Ze względu na wagę zagadnienia, jego złożoność i czasochłonność, obserwuje się duże zainteresowanie rozwojem programowych narzędzi testowania, zwiększających efektywność, umożliwiających częściową automatyzację. Prace dotyczą języków programowania wzbogacanych narzędziami obsługi testowania i uruchamiania, jak również konstrukcji systemów automatyzacji testowania.

Typowy proces testowania zawiera cztery główne podprocesy [3]:

1) wybór ścieżek testowych
2) generacja wejść testowych
3) ocena przebiegu testu
4) ocena wyjść testowych.

Omawiane w literaturze strategie testowania oparte są najczęściej na zasadzie „poziom po poziomie". Polega ona na tym, że testowanie składa się z kilku etapów związanych z różnymi poziomami abstrakcji programu. Ścieżki sterowania (ścieżki testowe) mogą się zatem odnosić tylko do określonego poziomu abstrakcji. Określając zbiór wymaganych ścieżek testowych, generuje się dane testowe z nastawieniem na wykonanie tych ścieżek oraz ocenia poprawność danych wyjściowych. W zależności od dostępnej informacji stosuje się różne (deterministyczne lub stochastyczne) schematy generacji danych wejściowych: generację bezpośrednią lub postępowanie iteracyjne. Kryteria oceny danych wyjściowych pozostają w ścisłym związku z metodą generacji danych wejściowych oraz zależą od ilości informacji dostępnej z opisu programu. Kryteria te mogą zmieniać się od bezpośredniej oceny wartości oczekiwanego wyjścia do prostych własności typu sum kontrolnych czy funkcjonalnych lub statystycznych zależności między wartościami wyjścia itp.

W każdym ze schematów testowania występuje potrzeba obserwacji zachowania się programu podczas przebiegów testowych. Jednym ze sposobów umożliwiających tę obserwację jest użycie tzw. monitorów programowych [3]. Stanowią je segmenty programowe, włączone i realizowane wewnątrz programu testowanego (pod kontrolą programu zarządzającego realizację testu), które umożliwiają identyfikację ścieżek przepływu sterowania, zatrzymanie wykonywania programu, jeśli sterowanie zejdzie z obrębu ścieżki testowej, wczesne wykrywanie błędnych warunków dla lokalizacji błędu blisko jego źródła itp. C. Ramamoorthy [3] podaje algorytmy optymalnego rozmieszczenia takich monitorów.

L. Stucki [4] opisuje inny przykład organizacji systemu wspomagającego testowanie. System ten zbiera i analizuje dane w dwóch głównych płaszczyznach:

— profilu syntaktycznego programu źródłowego
— aktualnego przedstawienia statystyk programu w odniesieniu do różnych zbiorów danych testujących; statystyki te dostarczają interesujących danych o realizacji instrukcji źródłowych.

Dodatkowe informacje o zakresach danych rozszerzają możliwości poszukiwań technik automatycznej generacji danych testowych. Perspektywicznym celem tych badań jest eliminacja redundancji testów i zaprojektowanie procedury prowadzącej do uzyskania minimalnego zbioru przypadków testujących.

O ile funkcją testowania jest pokazanie ewentualnych błędów, to głównym celem stosowania narzędzi uruchomieniowych jest lokalizacja przyczyn błędów oraz ich usunięcie. Służy temu ukazywanie wyników działania programów oraz ich korygowanie, osiągane dzięki takim zabiegom, jak:

- przesuwanie, wstawianie punktów przerwań oraz nowych instrukcji
- przeglądanie wybranych obszarów programów
- inicjowanie lub przenoszenie sterowania do innego punktu
- realizacja śladów retrospektywnych
- symulacja przerwań i operacji we/wy
- kontrola czasu.

Należy wspomnieć, że liczne mechanizmy wykorzystywane w procesie detekcji błędów i uruchamiania programów realizowane są sprzętowo, np. protekcja pamięci, sygnalizacja nadmiarów itd.

ODTWARZANIE

Mimo przedsięwzięć prewencyjnych oraz wysiłków ponoszonych przy testowaniu i uruchamianiu złożonych systemów oprogramowania zawsze mogą powstać jakieś błędy. Konieczne jest dysponowanie odpowiednimi mechanizmami odtwarzania stanu programów, jakie istniały przed wykryciem błędów. Mechanizmy te powinny być określane we wczesnych etapach rozwoju systemu i zawierać się w ogólnej filozofii systemu. Nie należy ich traktować jako doraźnie tworzonych elementów dodatkowych.

Mechanizmy te powinny być stosowane na różnych poziomach oprogramowania, m.in. dla zabezpieczenia się przed niepowodzeniem odtwarzania na określonym poziomie. Z drugiej strony wprowadzenie wielu poziomów odtwarzania zwiększa złożoność systemu oprogramowania, a więc zmniejsza jego niezawodność.

Istotnym elementem w procesie odtwarzania jest czas — różny dla różnych funkcji systemu. Bardzo pożądany jest zatem wybór odpowiednich mechanizmów odtwarzania odnoszących się do określonej funkcji.

Istnieją dwie fundamentalne przesłanki decydujące o sposobie podejścia do odtwarzania [1, 2]:

- wewnątrz każdego systemu występuje specyficzna hierarchia determinująca, które jego składniki i w jakim porządku muszą być poddane działaniom odtwarzania
- w tym samym czasie nie potrzeba odtwarzać wszystkich składników systemu.

W wielu systemach, zwłaszcza czasu rzeczywistego, wymagane jest szybkie odtwarzanie po wystąpieniu błędu (defekcie sprzętu lub błędzie oprogramowania), który nie zawsze można natychmiast skorygować. Jedną z dróg jest stosowanie restartów. Zabezpieczany w określonych momentach wolny od błędów stan systemu po usunięciu przyczyny błędu umożliwia powtórzenie procesu obliczeniowego od określonego miejsca, a nie od samego początku programu. Zbyt częste zabezpieczanie stanu systemu w celu zapewnienia szybkiego odtwarzania pociąga jednak za sobą znaczny wzrost czasu realizacji programu. Istnieje zatem problem optymalnego wyboru punktów restartowych. Stosowane są różne kryteria, np. kryterium minimalizacji średniego czasu zabezpieczeń stanów, kryterium minimalizacji czasu realizacji programu itp. [6].

Rozmiar informacji niezbędnej do zabezpieczenia i umożliwiającej restart zależy od miejsca w programie. Od rozmiaru tej informacji i rodzaju stosowanej pamięci zależy również czas zabezpieczania.

Odrębne podejście do zagadnień odtwarzania w przypadku błędów oprogramowania prezentuje B. Randell [7], proponując strukturę programu wzbogacanego o dodatkowe mechanizmy umożliwiające detekcję błędów i odtwarzanie.

Poszczególne bloki programu, zwane blokami odtwarzania, zawierają blok zasadniczy, test akceptujący oraz pewną liczbę bloków alternatywnych. Blok zasadniczy jest blokiem pierwotnym, realizującym wymagane operacje. Test akceptujący jest przeprowadzany na wyjściu bloku zasadniczego dla oceny poprawności jego wykonania. Bloki alternatywne w różny sposób realizują te same funkcje co blok zasadniczy, a poprawność ich wykonania sprawdzana jest powtórzeniem testu akceptującego. W przypadku niepomyślnego wyniku testu akceptującego po wykonaniu bloku zasadniczego funkcje spełniane przez ten blok są przetwarzane przez kolejne realizacje bloków alternatywnych, aż do pozytywnego wyniku testu akceptującego. Jeżeli nie uzyska się takiego wyniku dla żadnego bloku alternatywnego w danym bloku odtwarzania, to kolejny etap analogicznego działania będzie przebiegał w nadrzędnym bloku odtwarzania, zawierającym zakwestionowany blok.

Prezentowana metoda zapewnia protekcję niższych poziomów odtwarzania w przypadku ich niepowodzenia. Jest to więc bardzo skuteczny, wielopoziomowy model postępowania. Jedynym mankamentem tej metody jest duża (szczególnie w sensie zajętości pamięci) redundancja oprogramowania.

* * *

Z praktycznym stosowaniem omówionych metod i środków zmierzających do poprawy niezawodności systemu oprogramowania związane są określone nakłady. Rozważając te nakłady w kategorii kosztów ponoszonych dla uzyskania określonych funkcji i cech systemu, należy każdorazowo dokonać bilansu opłacalności i ocenić, czy ponoszone koszty nie przewyższają przewidywanych korzyści.

LITERATURA:

[1] T. Gibbons: Integrity and Recovery in Computer Systems. NCC Publications, Manchester, Hayden Book Company Inc., Rochelle Park, New Jersey 1976

[2] Computer Systems Reliability. C. Bunyan-editor, Infotech Information Ltd., Berkshire 1974

[3] C. Ramamoorthy: Optimal Placement of Software Monitors Aiding Systematic Testing. IEEE Transactions on Software Engineering, Nr 4/1975

[4] L. Stucki: A prototype Automatic Program Testing Tool. Fall Joint Computer Conference, 1972

[5] Program Test Methods. Prentice-Hall Inc., Englewood Cliffs 1973

[6] K. Chandy, C. Ramamoorthy: Rollback and Recovery Strategies for Computer Programs. IEEE Transactions on Computers, Nr 6 1972

[7] B. Randell: System Structure for Software Fault Tolerance, IEEE Transactions on Software Engineering, Nr 2/1975

LIDIA BOCIAN, GRZEGORZ JUCHNIKOWSKI

Instytut Informacji Naukowej, Technicznej i Ekonomicznej
Warszawa

Metoda sieciowa GERT

W wielu dziedzinach ludzkiej działalności zachodzi konieczność badania własności pewnych systemów (proces produkcyjny, grupa społeczna, system komunikacyjny, system obiegu informacji itp.). Celem takich badań może być również optymalizacja działania systemu, sprawdzenie niezawodności, określenie stanów krytycznych, czy też przewidywanie zachowania systemu w przyszłości. Aby ten cel osiągnąć, tworzy się abstrakcyjny model systemu na potrzeby jednego aktualnie badanego systemu lub też opracowuje się ogólne metody pozwalające na modelowanie różnych systemów według tych samych reguł. Wśród takich ogólnych metod coraz większą popularność zyskują metody graficzne, często wspomagane programami komputerowymi. Należy do nich także metoda sieciowa GERT *(Graphical Evaluation and Review Technique)*.

PODSTAWOWE POJĘCIA

Językiem opisu modelu tworzonego za pomocą metody graficznej jest sieć wraz ze zbiorem parametrów przyporządkowanych jej elementom. Przez **sieć** rozumiemy graf skierowany, zbudowany z **węzłów** odpowiadających operacjom logicznym oraz **ścieżek** reprezentujących akcję modelowanego systemu. Podstawowym parametrem, jaki wiąże się ze ścieżką, jest prawdopodobieństwo zaistnienia akcji przez nią reprezentowanej **(aktywacji)**. Ponadto często przypisuje się ścieżce parametry dodatkowe: czas akcji, zysk, strata itp. Wielkości te są zwykle zmiennymi losowymi.

Samą sieć można sobie wyobrazić jako pewną strukturę statyczną, przez którą przepływają elementy dynamiczne — **impulsy**, poruszające się wzdłuż ścieżek, od węzła do

Mgr Lidia BOCIAN ukończyła studia na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego (1978 r.). Pracuje w Zakładzie Struktur Sieci Informatycznych w Instytucie Informacji Naukowej, Technicznej i Ekonomicznej. Zajmuje się adaptacją systemu informacyjno-wyszukiwawczego ISIS na potrzeby SINTO i rozbudową systemu TEKLA na użytek Clearinghouse.

Mgr inż. Grzegorz JUCHNIKOWSKI ukończył studia na Wydziale Elektroniki (specjalność: Informatyka) Politechniki Warszawskiej (1979 r.). Pracuje w Zakładzie Struktur Sieci Informatycznych w Instytucie Informacji Naukowej, Technicznej i Ekonomicznej. Zajmował się konstrukcją systemu informacyjno-wyszukiwawczego na potrzeby Clearinghouse, obecnie uczestniczy w tworzeniu systemu relacyjnej bazy danych, opartego na systemie informacyjno-wyszukiwawczym ISIS.

węzła. Na przykład, jeśli pewna sieć jest modelem dworca kolejowego, to węzły jej odpowiadają różnym punktom usługowym (kasy, informacja, przechowalnia bagażu, restauracja itp.) lub decyzjom pasażerów (o oddaniu bagażu do przechowalni, o zakupie biletu itp.), ścieżki wskazują kierunki poruszania się pasażerów po dworcu, zaś impulsy reprezentują samych pasażerów. Dlatego też w dalszym ciągu tej pracy używane będą takie pojęcia, jak: **pobudzenie węzła** dla wyrażenia sytuacji, w której impuls wpłynął właśnie do węzła, **aktywacja ścieżki** — gdy impuls „został wypuszczony" z węzła na ścieżkę.

Będziemy mówić, że **węzeł jest zrealizowany**, jeśli spełniona jest pewna funkcja logiczna pobudzeń węzła przez ścieżki doń wchodzące. Jeśli aktywowane są wszystkie ścieżki, to **węzeł** jest **typu deterministycznego**, jeśli zaś ze ścieżkami tymi związane są prawdopodobieństwa aktywacji mniejsze od 1 i w danym momencie aktywowana jest tylko jedna z nich, to **węzeł** jest **typu probabilistycznego.**

Metoda GERT istnieje w dwóch wersjach: analitycznej (GERT) i symulacyjnej (GERTS). Wersję analityczną można stosować bez użycia komputera, natomiast do badania systemów przy użyciu metody symulacyjnej komputer jest już narzędziem niezbędnym. Program symulacyjny o nazwie GERTS III jest napisany w FORTRANIE przez A. Pritskera i R. Burgessa.

Przedmiotem badań prowadzonych w Instytucie Informacji Naukowej, Technicznej i Ekonomicznej jest metoda symulacyjna. Badania te dotyczą możliwości i celowości stosowania metody GERTS III do modelowania różnego rodzaju systemów. Dlatego ta właśnie metoda jest tematem niniejszego artykułu.

CHARAKTERYSTYKA ELEMENTÓW SIECI GERTS III

Węzły sieci GERTS III charakteryzowane są przez rodzaj wejścia i wyjścia. Wyjście z węzła może być probabilistyczne lub deterministyczne, wejście natomiast specyfikowane jest przez liczbę pobudzeń potrzebnych do realizacji węzła. Rys. 1 przedstawia dwa możliwe typy węzłów.

Rys. 1. Węzły sieci GERTS III: a) probabilistyczny, b) deterministyczny

l — liczba pobudzeń potrzebnych do pierwszej realizacji węzła
k — liczba pobudzeń węzła potrzebnych do każdej następnej realizacji
n — numer węzła w sieci (węzły numeruje się od 2)

Ze ścieżką sieci GERTS III wiąże się prawdopodobieństwo aktywacji oraz czas jej trwania (czas przejścia impulsu przez ścieżkę). Czas ten jest zmienną losową. Typ rozkładu prawdopodobieństwa tej zmiennej wraz ze zbiorem parametrów rozkładu (tymi parametrami są: wartość oczekiwana, wariancja, wartości minimalna i maksymalna zmiennej) są elementami opisu sieci, przygotowywanego jako dane dla programu symulacyjnego. Możliwe są następujące rozkłady zmiennej czasowej: stały, normalny, jednostajny, Erlanga, logarytmiczno-normalny, Poissona, beta, gamma, trójparametrowy beta.

Węzły charakteryzowane są dodatkowo przez funkcję, jaką pełnią w sieci. Istnieją cztery typy węzłów specjalnych: węzły źródłowe, końcowe, statystyczne, pomocnicze.

Specyfikacja węzła jest także częścią danych dla programu symulacyjnego. Węzły źródłowe (symbol takiego węzła przedstawia rys. 2a) są realizowane w chwili zerowej — są to więc węzły wysyłające pierwszy impuls do sieci. Węzły końcowe (symbol na rys. 2b) są węzłami, których realizacja wskazuje moment zakończenia symulacji.

Rys. 2. Węzły specjalne: a) źródłowy, b) końcowy

Dla węzłów statystycznych program GERTS III oblicza statystyczne wielkości czasowe związane z realizacją węzła. Statystyki te mogą być następujące:

F — czas pierwszej realizacji węzła
A — czas wszystkich realizacji węzła
B — czas upływający pomiędzy kolejnymi realizacjami węzła
I — czas potrzebny do przejścia między danym węzłem a najbliższym węzłem pomocniczym
D — czas upływający między pierwszym pobudzeniem węzła a jego realizacją (dla l, $k \neq 1$).

Węzły pomocnicze służą jako punkty odniesienia do liczenia statystyk typu I.

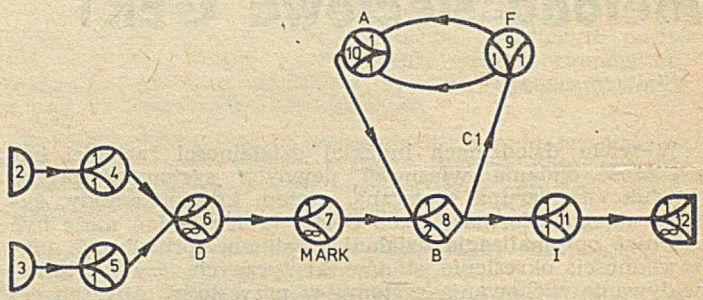
Inną wielkością statystyczną obliczaną przez program GERTS III jest liczba aktywacji ścieżki. Jest to realizowane przez przyporządkowanie ścieżce licznika zwiększającego swą wartość o 1, ilekroć ścieżka jest aktywowana. Program oblicza liczbę przejść przez ścieżkę (aktywacji) do momentu realizacji każdego węzła statystycznego.

Program GERTS III symuluje działanie sieci przez generowanie czasów aktywności ścieżek (zgodnie z zadanymi rozkładami) i ich sumowanie. Jest to więc symulacja realizowana metodą kolejnych zdarzeń.

Dane wejściowe dla programu stanowi opis węzłów, ścieżek, parametrów rozkładu zmiennej czasowej oraz informacje ogólne dotyczące liczby kroków symulacji, nazwy modelu, liczby węzłów w sieci, w tym liczby węzłów źródłowych, końcowych, statystycznych itp.

Program GERTS produkuje wydruk zawierający opis danych, wyniki obliczeń statystycznych, a także — na żądanie — ślad przebiegu symulacji dla wyznaczonych kroków.

Poniższy przykład pozwoli na dokładniejsze wyjaśnienie zasad modelowania sieci i działania programu. Rys. 3 przedstawia pewną sieć GERTS.



Rys. 3. Przykładowa sieć GERTS III

Węzły 2 i 3 są węzłami źródłowymi. Tak więc w chwili zerowej aktywowane będą ścieżki 2→4 i 3→5. Z węzłów 4 i 5 wychodzą ścieżki do węzła 6. Aby węzeł ten był zrealizowany, potrzebne są pobudzenia pochodzące z obu ścieżek doń wchodzących (4→6 i 5→6). Po zrealizowaniu węzła 6 aktywowana jest ścieżka 6→7, pobudzenie węzła 7 jest równoznaczne z jego realizacją, więc dalej aktywowana jest ścieżka 7→8. Węzeł 8 ma wyjście probabilistyczne, a zatem aktywowana będzie tylko jedna ze ścieżek 8→9 albo 8→11. Jeśli impuls przejdzie ścieżką 8→11, to wkrótce zostanie pobudzony węzeł końcowy 12 i symulacja się zakończy. Jeśli zaś aktywowana będzie ścieżka 8→9, to po zrealizowaniu węzła 9 aktywowane będą obie ścieżki łączące go z węzłem 10 (czyli węzeł 10 będzie pobudzony dwukrotnie). Jednak do realizacji 10 wystarczy jedno pobudzenie, a zatem, gdy tylko impuls z jednej ścieżki 9→10 do niego dotrze, zostanie od razu przesłany na ścieżkę 10→8. Ale węzeł 8 do drugiej realizacji potrzebuje dwóch pobudzeń, więc właśnie tutaj impuls „musi czekać" aż przypłynie impuls przebiegający drugą ścieżką 9→10.

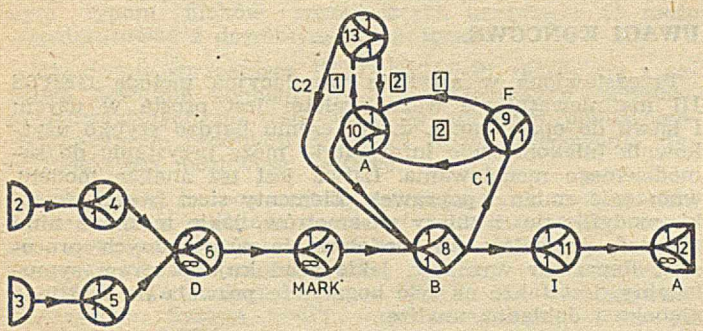
Węzły 6, 8, 9, 10, 11 i 12 są węzłami statystycznymi, węzeł 7 jest węzłem pomocniczym. Licznik został przypisany ścieżce 8→9. Wykorzystane zostały wszystkie możliwe rozkłady zmiennej czasowej oraz wszystkie typy statystyk.

MODYFIKACJA SIECI

Program GERTS III oferuje dodatkowo możliwość dokonania chwilowej modyfikacji struktury sieci. W przypadku spełnienia pewnego warunku jeden węzeł może być zastąpiony innym. Modyfikację taką realizuje się przez przyporządkowanie ścieżce liczby naturalnej — numeru ścieżki — i oznaczenie, że zakończenie aktywacji tej właśnie ścieżki ma spowodować zmianę węzłów. Graficzny symbol modyfikacji przedstawia rys. 4. Pojawienie się w sieci takiego symbolu informuje, że po zakończeniu aktywacji ścieżki oznaczonej numerem 1 węzeł B ma być zastąpiony węzłem A.

Rys. 4. Symbol modyfikacji sieci GERTS III

Poniższy przykład pozwoli dokładniej zapoznać Czytelnika z tym mechanizmem.



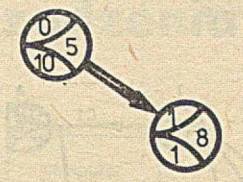
Rys. 5. Sieć GERTS III z modyfikacją

Sieć z rys. 5 ma strukturę bardzo podobną do sieci z rys. 3. Różnią się one paroma tylko szczegółami, w tym właśnie wystąpieniem modyfikacji. Zwróćmy uwagę na ten fragment sieci. Jeśli po zrealizowaniu węzła 8 zostanie aktywowana ścieżka 8→9, to dalej aktywowane są obie ścieżki łączące węzły 9, 10, oznaczone numerami 1 i 2. Czasy przejścia tych ścieżek mogą być różne, jeśli więc aktywacja ścieżki 1 zostanie zakończona wcześniej niż aktywacja ścieżki 2, to węzeł 10 zostanie zastąpiony węzłem 13, tak, że następną aktywowaną ścieżką będzie 13→8. Należy zwrócić uwagę, że modyfikacja działa na wyjściu z węzła; tzn. pobudzenie ze ścieżki 1 jest odbierane przez węzeł 10, a przesyłane dalej z węzła 13. Jeśli natomiast pobudzenie węzła 10 pochodziło ze ścieżki 2, to modyfikacja nie zachodzi i następną aktywowaną ścieżką jest 10→8.

METODA GERTS IIIQ

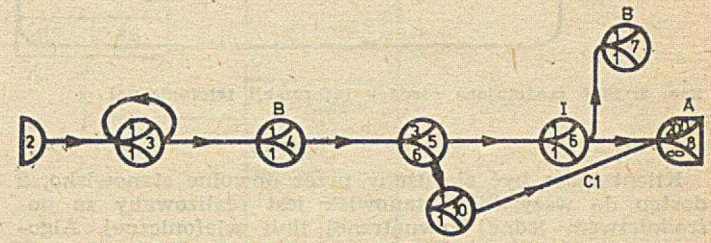
Modelowanie systemu w rodzaju sklepu, stacji obsługi itp. wymaga stworzenia pewnych dodatkowych elementów sieci, pozwalających na przechowywanie impulsów — zatrzymywanie ich w kolejce. Program GERTS IIIQ, będący zmodyfikowaną wersją GERTS III, dopuszcza stosowanie w sieci takich właśnie elementów — węzłów kolejkowych.

Symbol węzła kolejkowego jest taki sam, jak przedstawiony na rys. 1, z tym, że inne są znaczenia liczb k i l; l oznacza początkową długość kolejki, a k — liczbę miejsc w „poczekalni" (maksymalną długość kolejki). Ścieżka wychodząca z węzła kolejkowego nazywa się ścieżką obsługi. W danej chwili może być obsługiwany tylko jeden impuls. Jeżeli ścieżka obsługi jest zajęta, a do węzła napływają nowe impulsy, to są one zatrzymywane i przechowywane w węźle do momentu zwolnienia ścieżki obsługi. Gdy zostaje ona zwolniona, z węzła pobierany jest według zadanego priorytetu kolejny impuls. Jeśli liczba pobudzeń w danym czasie przekracza maksymalną długość kolejki, to impulsy są albo usuwane z systemu, albo przesyłane do węzła wskazanego przez projektanta sieci. Na rys. 6 przedstawiono, jak zaznacza się takie przesłanie w sieci.



Rys. 6. Oznaczenie kierunku usunięcia impulsu

Kolejny przykład ilustruje działanie sieci z węzłami kolejkowymi. Jednocześnie pokazuje on pewną klasę zastosowań metody GERTS IIIQ.



Rys. 7. Sieć modelująca linię produkcyjną

Sieć przedstawiona na rys. 7 jest modelem linii produkcyjnej o jednym stanowisku. Pętla wokół węzła 3 generuje elementy podlegające obróbce na stanowisku, któremu odpowiada węzeł kolejkowy 5. Czas obróbki pojedynczego elementu reprezentuje ścieżka 5→6. W chwili początkowej (rozpoczęcie dnia pracy) trzy elementy już czekają na obróbkę. Maksymalna długość kolejki reprezentuje wydajność stanowiska roboczego. Węzeł 7 jest włączony do sieci w celu wyznaczenia odstępów czasu, w jakich elementy schodzą z linii produkcyjnej (statystyka typu B). Symulacja zostanie zakończona, gdy węzeł końcowy 8 zostanie pobudzony 200 razy.

Dla węzłów kolejkowych są zbierane specjalne statystyki; dla każdego z nich liczona jest liczba impulsów przebywających w kolejce, liczba impulsów usuniętych z węzła w jednostce czasu (z powodu przekroczenia pojemności „poczekalni") oraz prawdopodobieństwo, że ścieżka obsługi jest zajęta. Dla każdej z tych wielkości podawana jest wartość średnia, odchylenie standardowe, minimum i maksimum.

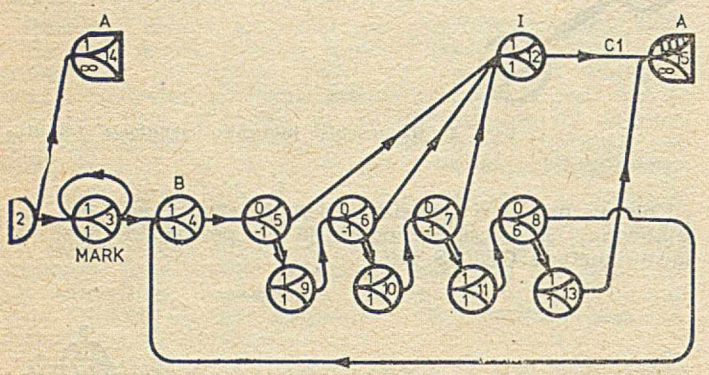
Tak więc, jeśli symulacja sieci z rys. 7 wykaże dużą liczbę impulsów usuniętych do węzła 10, będzie to oznaczało, że wydajność stanowiska jest zbyt mała w stosunku do liczby elementów podlegających obróbce. Na odwrót, niewielkie prawdopodobieństwo zajętości ścieżki obsługi (5→6) będzie wskazywać na niepełne wykorzystanie stanowiska. Inne informacje, jakich może dostarczyć symulacja tej sieci, to: czas wykonania całej pracy, czas spędzony przez pojedynczy element w systemie, częstość napływania impulsów do systemu.

Powstały jeszcze dwie wersje programu symulacyjnego, będące — jak i program GERTS IIIQ — rozszerzeniami bazowego programu GERTS III. Są to: wersja GERTS IIIC *(Costs)*, przeznaczona do zastosowań typu ekonomicznego, pozwalająca na obliczanie statystycznych wielkości kosztów związanych z realizacją modelu, oraz wersja GERTS IIIR *(Resources)*, pozwalająca na uwzględnienie informacji o ograniczeniach i przesunięciach zasobów (koszty, materiały), przeznaczonych na realizację modelu.

Na zakończenie przedstawimy konkretny przykład wykorzystania metody GERTS IIIQ. Będzie to model ośrodka informacji telefonicznej.

MODEL OŚRODKA INFORMACJI TELEFONICZNEJ

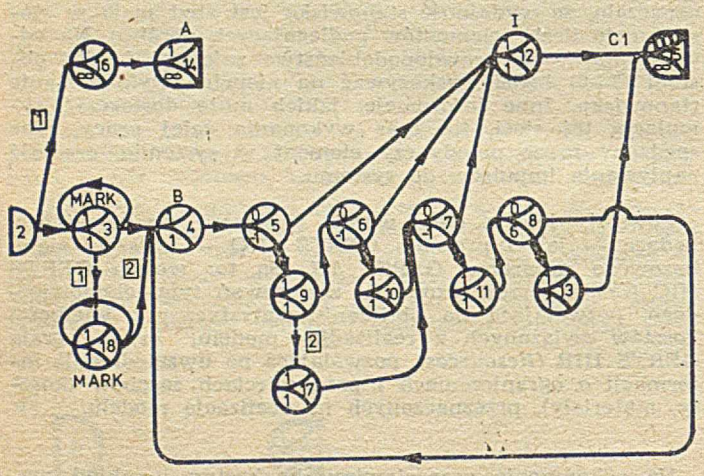
Dla ustalenia obsady takiego ośrodka niezbędne jest określenie optymalnej liczby stanowisk informacyjnych w zależności od liczby zgłaszanych zapytań i czasu koniecznego na udzielenie odpowiedzi. Na rys. 8 przedstawiona jest sieć modelująca obsługę klientów przez ośrodek wyposażony w trzy stanowiska informacyjne.



Rys. 8. Sieć modelująca ośrodek informacji telefonicznej

Klient może być obsłużony przez dowolne stanowisko, a dostęp do wszystkich stanowisk jest realizowany za pośrednictwem jednej zewnętrznej linii telefonicznej. Algorytm obsługi w ośrodku jest następujący: jeśli pierwsze stanowisko jest wolne, to klient otrzymuje z niego odpowiedź, jeśli zaś jest zajęte, klient jest automatycznie łączony z następnym wolnym stanowiskiem. Jeśli zajęte są wszystkie stanowiska, to zgłoszenie klienta jest umieszczane w kolejce, a on sam słyszy sygnał informujący go, że musi czekać na udzielenie odpowiedzi. Jeśli zajęte są wszystkie stanowiska i kolejka czekających klientów osiągnęła maksymalną długość, to następni klienci są informowani o przeciążeniu ośrodka specjalnym sygnałem i zostają rozłączeni.

Odstępom czasu między kolejnymi zgłoszeniami odpowiada czas aktywności ścieżki wokół węzła 3. Trzem stanowiskom informacyjnym odpowiadają kolejkowe węzły 5, 6, 7, przy czym liczba *–1* w dolnej części węzła oznacza, że kolejka właściwie nie występuje, co odpowiada natychmiastowemu przełączeniu klienta do następnego stanowiska informacyjnego. Czas aktywności ścieżek 5→12, 6→12 i 7→12 odpowiada czasowi udzielania odpowiedzi. Kolejkę klientów oczekujących na odpowiedź reprezentuje węzeł 8. Z rysunku ponadto wynika, że klient po odczekaniu w kolejce niekoniecznie musi uzyskać informację, bowiem po wyjściu z tej kolejki jest traktowany dokładnie tak samo, jak klient zgłaszający się do ośrodka z zewnątrz (nie ma różnicy między impulsami pochodzącymi z węzłów 3 i 8).



Rys. 9. Model ośrodka informacji telefonicznej z uwzględnieniem zmiany częstotliwości napływania zapytań

Ścieżkom 12→15 i 13→15 przyporządkowane są liczniki C1 i C2, mierzące odpowiednio liczbę klientów, którzy uzyskali odpowiedź w ośrodku, oraz liczbę klientów, którzy zostali rozłączeni z powodu przekroczenia możliwości ośrodka. Węzeł 3 jest węzłem pomocniczym, a dla węzła 12 zbierana jest statystyka typu I — czas przejścia od węzła 3; odpowiada to czasowi, jaki poświęca klient na uzyskanie informacji. Symulacja zostanie zakończona po 10 godzinach (czasu symulowanego przez program) lub po obsłużeniu tysiąca klientów, czyli po zrealizowaniu węzła końcowego 14 lub 15. Symulacja tego modelu dostarczy informacji na temat średniej liczby klientów czekających na uzyskanie odpowiedzi, pozwoli także stwierdzić, w jakim stopniu wykorzystywane były stanowiska informacyjne.

Wykorzystując oferowany przez program GERTS III mechanizm modyfikacji można badać działanie ośrodka uwzględniając np. zmianę częstotliwości zgłoszeń klientów i zmniejszenie liczby stanowisk. Na rys. 9 przedstawiona jest sieć, w której zaznaczone są te właśnie zmiany.

Do sieci wprowadzone zostały dodatkowe węzły 16, 17 i 18, a ścieżkom 2→16 i 18→4 zostały przyporządkowane numery 1 i 2. Ze ścieżką 1 związany jest pewien stały czas, krótszy od czasu przejścia ścieżki 2→14 z poprzedniego modelu. Po upływie tego czasu nastąpi zmiana częstotliwości zgłoszeń (węzeł 3 zostanie zastąpiony węzłem 18, a ścieżce 18→18 przypisana zostanie zmienna czasowa o innych parametrach rozkładu). Gdy zmiana ta zostanie zauważona w ośrodku (po zakończeniu aktywacji ścieżki 2), nastąpi wyłączenie jednego stanowiska informacyjnego (stanowiska odpowiadającego węzłowi 6).

UWAGI KOŃCOWE

Przedstawiona w artykule symulacyjna metoda GERTS III ma niewątpliwie ważną zaletę: jest prosta w użyciu i łatwa do opanowania, dzięki czemu bardzo szybko użytkownik (niekoniecznie informatyk) może przystąpić do samodzielnego modelowania. Łatwa jest też analiza modelu, wnoszenie zmian i poprawek. Elementy sieci (węzły, ścieżki, modyfikacje) i zbiory parametrów, jakie można z nimi związać, pozwalają na modelowanie różnorodnych procesów. Repertuar wyników, jakie produkuje program symulacyjny, jest także na tyle bogaty, że pozwala na możliwie głęboką i dokładną analizę.

Metoda GERTS III ma również pewne niedostatki. Ujawniają się one na przykład w mechanizmie węzłów kolejkowych. Wydaje się, że z impulsami pobudzającymi te węzły związanych jest zbyt mało atrybutów. Na przykład jeśli modeluje się ruch klientów w sklepie samoobsługowym, to uzasadnione wydaje się wymaganie, aby klienci wchodzący do kolejki po koszyki decydowali się czekać w zależności od własnej cierpliwości i potrzeb — jeśli ktoś jest mało cierpliwy, rezygnuje z czekania w dziesięcioosobowej kolejce, jeśli zaś pewne artykuły są mu bardzo potrzebne, będzie czekać nawet wtedy, gdy kolejka liczy piętnaście osób. Metoda GERTS III nie dostarcza środków pozwalających na modelowanie takich sytuacji. Ograniczeniem wydaje się także liczba algorytmów, według których impulsy mogą być pobierane z kolejki. W programie GERTS IIIQ uwzględnione są tylko dwa algorytmy FIFO *(First-In-First-Out)* i LIFO *(Last-In-First-Out)*.

Pewne zastrzeżenia budzi także sposób, w jaki realizowane są modyfikacje w sieci GERTS III. Trudne do modelowania są sytuacje, w których projektant sieci chce uzyskać modyfikację trwałą. Widać to nawet na przykładzie ośrodka informacji telefonicznej (rys. 9). Poza tym fakt, że jednorazowa aktywacja może wywołać tylko jedną modyfikację, powoduje, że trzeba wprowadzać do modelu wiele dodatkowych węzłów i ścieżek.

Czytelników zainteresowanych metodą GERT informujemy, że programy GERT i GERTS są rozpowszechniane przez Lehigh University w Bethlehem (Pensylwania). Można je otrzymać wysyłając taśmę magnetyczną i 25 dolarów pod następującym adresem:
Lehigh Computing Center attn: Librarian Packard
Laboratory Lehigh University
Bethlehem, Pa 18015.

LITERATURA:

[1] Whitehouse G. E.: Systems analysis and design using network techniques. Prentice-Hall, Inc., Engelwood Cliffs, New Jersey 1973
[2] Feller W.: Wstęp do rachunku prawdopodobieństwa. PWN, Warszawa 1969

MIROSŁAW LIZUREK
Instytut Informatyki PW
Warszawa

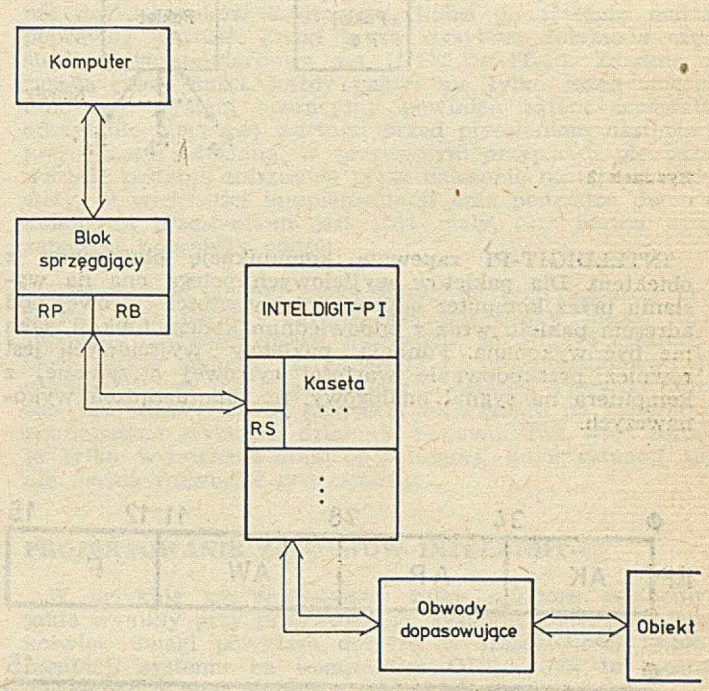
Oprogramowanie kaset urządzenia INTELDIGIT-PI

Rozwój technologii oraz dążenie do efektywnego wykorzystania środków produkcji coraz częściej zmuszają nas do sięgania po nowoczesny sprzęt pomiarowy i regulacyjny. Wymaga on jednak precyzyjnego sterowania, a otrzymane wyniki pomiarów trzeba opracowywać według skomplikowanych algorytmów. Szybkie uzyskanie wyników — i to w postaci przejrzystych tabel czy raportów — umożliwia komputerowy system pracujący w czasie rzeczywistym. System taki na bieżąco zbiera dane o przebiegu produkcji i przetwarza je w celu optymalnego sterowania procesem technologicznym. Ponadto równie ważną jego funkcją jest generacja dokumentów z przebiegu produkcji.

Dla realizacji ww. funkcji komputer musi być sprzężony z punktami pomiarowymi obiektu za pośrednictwem zestawu umożliwiającego przekodowanie napływających informacji. Wymagania takie spełnia urządzenie INTELDIGIT-PI, opracowane w Zakładzie Doświadczalnym Instytutu Automatyki i Pomiarów „MERA-PIAP". Dzięki różnym typom bloków sprzęgających urządzenie to może współpracować z dowolnym komputerem.

Urządzenie sprzęgające INTELDIGIT-PI składa się z zestawu dowolnie wymienialnych funkcjonalnych modułów, przystosowanych do współpracy ze sprzętem pomiarowo-regulacyjnym produkcji krajowej. Nowe opracowania stale zwiększają możliwości zestawu.

Ze względu na powszechne zainteresowanie urządzeniem warto omówić niektóre problemy związane z jego oprogramowaniem. Oto jakie problemy pojawiły się w trakcie opracowywania i uruchamiania Systemu Otrzymywania Raportów Technologicznych — SORT [1] w Cukrowni Krasnystaw. System SORT został zaimplementowany na komputerze ODRA 1325 (wyposażonym w egzekutor przemysłowy EX2P), współpracującym z zestawem INTELDIGIT-PI.



Rysunek 1

RP — rejestr przerwań
RB — rejestr buforowy dla odczytanej wartości
RS — rejestr stanu kasety

URZĄDZENIE INTELDIGIT-PI

Urządzenie INTELDIGIT-PI umożliwia automatyczną rejestrację i sterowanie obiektami przemysłowymi oraz procesami technologicznymi.

INTELDIGIT-PI ma strukturę modułową, przy czym w skład pojedynczego zestawu może wchodzić do ośmiu kaset. Każda z nich zawiera sterownik oraz do 16 pakietów funkcjonalnych. W jednej kasecie mogą znajdować się pakiety we/wy różnych typów.

Komunikacja między komputerem a urządzeniem PI odbywa się poprzez blok sprzęgający. Jego zadanie polega na dopasowaniu sygnałów we/wy komputera konkretnego typu do sygnałów zestawu PI. Struktura zestawu pokazana jest na rysunku 1.

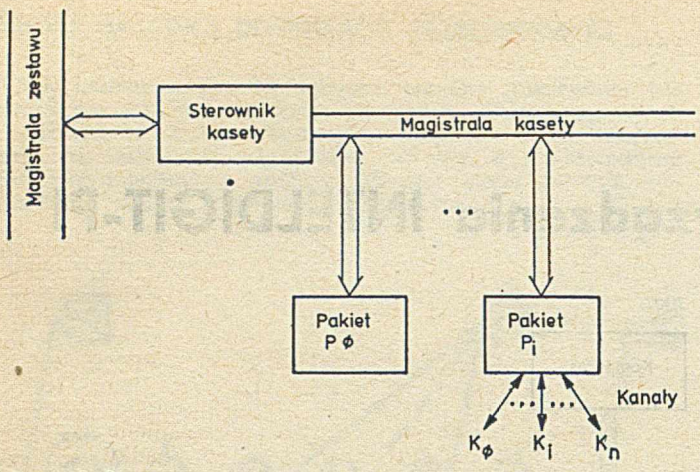
Wśród dostępnego zbioru pakietów można wyróżnić trzy podstawowe typy:

1) pakiety wejściowe — ich zadaniem jest wstępne przekodowanie napływających sygnałów z czujników pomiarowych; ponadto przechowują uzyskaną informację do chwili przekazania jej do komputera
2) pakiety wyjściowe — mają za zadanie odebranie informacji przychodzącej z komputera, a następnie przesłanie jej po przekodowaniu do urządzeń wykonawczych
3) pakiety dopasowujące — ich zadanie polega na dopasowaniu sygnałów urządzenia PI z sygnałami czujników i regulatorów.

Jak wynika ze struktury INTELDIGIT-PI kaseta stanowi pewien moduł wyższego poziomu i jako taka ma w tym zestawie przyporządkowany określony adres. Wykorzystując ten adres, kaseta może komunikować się z komputerem, zgłaszając określonym sygnałem przerwanie. Z kolei komputer może przesyłać pod wskazanym adresem konkretne wartości. Można zauważyć, że kaseta zawierająca tylko obwody (pakiety) dopasowujące nie ma komunikacji z komputerem. W związku z tym jej adres nie ma znaczenia i dlatego jest ona nazywana kasetą nieadresowalną. Na rysunku 2 przedstawiono uproszczony schemat blokowy kasety zestawu INTELDIGIT-PI.

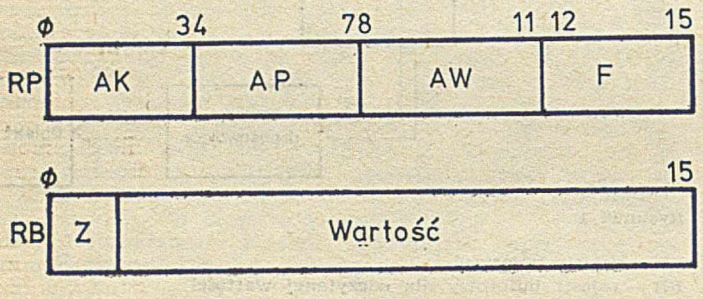
INTELDIGIT-PI jest urządzeniem operującym na słowach 16-bitowych, przy czym format informacji dostępnych w rejestrach: buforowym (RB) i przerwań (RP) pokazuje rysunek 3. Rejestry te znajdują się fizycznie w bloku sprzęgającym [1] i mogą być odczytywane przez komputer. Rejestr przerwań RP przeznaczony jest do przechowywania informacji dotyczącej źródła oraz typu przerwania, natomiast rejestr buforowy RB przechowuje odczytaną wartość, której interpretacja zależy od typu pakietu nadającego (pakietu wejściowego).

Mgr inż. Mirosław LIZUREK ukończył studia w 1976 r. na Wydziale Elektroniki PW. Pracuje w Instytucie Informatyki PW. Zajmuje się m.in. oprogramowaniem podstawowym minikomputera GEO-20. Ściśle współpracuje ze studenckim ruchem naukowym zrzeszonym w Warszawskim Centrum Studenckiego Ruchu Naukowego.



Rysunek 2

INTELDIGIT-PI zapewnia komunikację obustronną z obiektem. Dla pakietów wyjściowych polega ona na wysłaniu przez komputer odpowiedniej wartości cyfrowej pod adresem pakietu wraz z odpowiednim kodem funkcji, jaka ma być wykonana. Funkcją pakietów wyjściowych jest również przekodowanie wartości cyfrowej otrzymanej z komputera na sygnał analogowy, np. dla urządzeń wykonawczych.



Rysunek 3

AK — adres kasety AK∈<Ø, 15>
AP — adres pakietu w kasecie AP∈<Ø, 15>
AW — numer kanału we/wy AW∈<Ø, 15>
F — funkcja (kod przerwania) dla pakietów wejściowych (wyjściowych)
Z — bit znaku

Opracowując system operacyjny działający pod kontrolą programu zarządzającego (np. egzekutor EX2P), programista ma do dyspozycji zestaw ekstrakodów [1], a otrzymywane przerwania są już częściowo przetworzone. W związku z tym można mówić o przerwaniu typu RB, tj. takim, z którego możemy otrzymać bezpośrednio odczytaną wartość (program zarządzający realizuje odczyt potrzebnych rejestrów bloku sprzęgającego i do systemu operacyjnego przekazuje słowo adresowe i wartość w formacie jednego przerwania) [2].

Z punktu widzenia projektanta systemu współpracującego z INTELDIGIT-PI można wyróżnić dwa typy współpracy — synchroniczny i asynchroniczny:

1) synchroniczny — polega na tym, że inicjatywa odczytu pakietów wejściowych należy do systemu operacyjnego
2) asynchroniczny — polega na tym, że system odczytuje stany tylko zgłaszających się pakietów, przy czym zgłoszenie następuje w wyniku wystąpienia określonej sytuacji w kontrolowanym procesie.

W pierwszym przypadku odczyt stanów pakietów następuje przez przesłanie do nich w określonym odstępie czasu odpowiednich kodów funkcji [1].

[1]) Ekstrakody są pewnymi procedurami znajdującymi się w programie zarządzającym, dostępnymi na poziomie programu użytkowego przez rozkazy o wyróżnionych kodach

Drugi przypadek jest bardziej złożony ze względu na to, że inicjatywa należy do procesu i sygnał może nadejść w dowolnym czasie. Zdarzenie to sygnalizowane jest przerwaniem wraz z podaniem adresu kasety, w której znajduje się pakiet żądający obsługi. Fakt zgłoszenia pakietu pamiętany jest w rejestrze stanu kasety. System operacyjny musi odczytywać ten rejestr i dopiero na tej podstawie określić przyczynę zgłoszenia kasety. Najprostszą reakcją na taką sytuację będzie odczyt stanu zgłaszającego się pakietu.

Należy zaznaczyć, że pakiety wyjściowe również przesyłają zgłoszenia, ale ich interpretacja jest odmienna. Zgłoszenie pakietu wyjściowego, po przesłaniu do niego wartości wraz z kodem funkcji inicjującej transmisję z komputera, sygnalizuje zakończenie tej operacji.

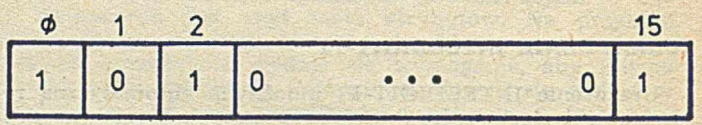
Przejdźmy teraz do omówienia pewnych problemów związanych z oprogramowaniem INTELDIGIT-PI.

ADRESOWANIE PAKIETÓW KASETY

W kasecie INTELDIGIT-PI może znajdować się do 16 pakietów. Pakiety te adresowane są od 0 do 15. Adres kasety jest jej numerem w zestawie. Pełny adres pakietu składa się z numeru kasety w zestawie oraz numeru pakietu w kasecie. Jak z tego wynika, adres kasety i adres zerowego pakietu są identyczne. Ta niejednoznaczność w adresowaniu może sprawić poważne problemy związane z rozpoznaniem źródła otrzymanego przerwania typu RB. Dla zilustrowania powyższego problemu rozpatrzmy następujący przykład.

W zestawie INTELDIGIT-PI, składającym się z pięciu kaset, rozmieszczone są pakiety typu wejściowego i wyjściowego. W trzeciej kasecie znajdują się pakiety wejściowe impulsowe typu PI—05 [3]. Pakiety tego typu zliczają impulsy przychodzące od czujników indukcyjnych. Z chwilą przepełnienia licznika pakiety zgłaszają tę sytuację do sterownika kasety, który generuje odpowiednie przerwanie.

Prześledźmy sytuację, jaka zaistniała po przepełnieniu licznika pakietu o adresie 03/00/00 [2]). Pakiet ten zgłasza swój stan do kasety, co zostaje zapamiętane w rejestrze stanu kasety. Z każdym bitem rejestru związany jest adres pakietu. Po zgłoszeniu pakietu bit w odpowiednim numerze przyjmuje wartość 1 (zgłoszone pakiety na rysunku 4: P0, P2, P15).



Rysunek 4

Po otrzymaniu zgłoszenia kasety 03 system operacyjny inicjuje odczyt jej stanu, w odpowiedzi na co otrzymuje przerwanie typu RB o adresie 03/00/00 i wartości B0 = 1, B1 ... B15 = 0. Po stwierdzeniu, który z pakietów przesłał zgłoszenie, następną operacją jest inicjacja odczytu jego stanu. System operacyjny otrzyma ponownie przerwanie typu RB o adresie 03/00/00, ale będzie ono dotyczyło tym razem stanu pakietu.

Z powyższego przykładu wynika, że aby rozróżnić takie zdarzenie należy pamiętać, jaka operacja została inicjowana wcześniej. Jest to bardzo niewygodne, gdyż wymaga sprawdzania dodatkowych warunków oraz stosowania stosów programowych, co powoduje znaczny wzrost objętości systemu. Zauważmy ponadto, że niesprawność sterowania kasety lub bloku sprzęgającego, polegająca na generacji większej liczby przerwań typu RB, całkowicie rozsynchronizuje proces zbierania informacji.

Pamiętanie stanu wcześniejszych zdarzeń staje się wręcz niemożliwe, jeżeli pewne operacje wykonywane są na różnych poziomach: program zarządzający i system operacyjny. Omówiona sytuacja nie wystąpi przy takim zestawieniu pakietów w kasecie, że na pozycji 00 znajduje się pakiet wyjściowy nie przesyłający wartości RB.

[2]) Przyjęto następującą zasadę oznaczania adresu pakietu: pozycja kasety w zestawie/pozycja pakietu na kasecie/numer kanału

PROBLEMY SYGNALIZACJI ZGŁOSZENIA PAKIETÓW

Zgodnie z zasadą działania kaset zestawu INTELDIGIT-PI, z chwilą zgłoszenia pakietu sterownik kasety wysyła poprzez blok sprzęgający do komputera zgłoszenie o odpowiednim adresie. Jednocześnie zgłoszenie zostaje zapamiętane na odpowiednim bicie rejestru stanu kasety. Od tej chwili kaseta przechodzi do stanu, w którym każde następne zgłoszenie jej pakietu powoduje ustawienie w rejestrze stanu kasety odpowiedniego bitu, natomiast kaseta nie generuje żadnego zgłoszenia.

Zwolnienie kasety, tzn. przejście do stanu generacji zgłoszeń, następuje dopiero po odczytaniu stanów wszystkich oczekujących pakietów.

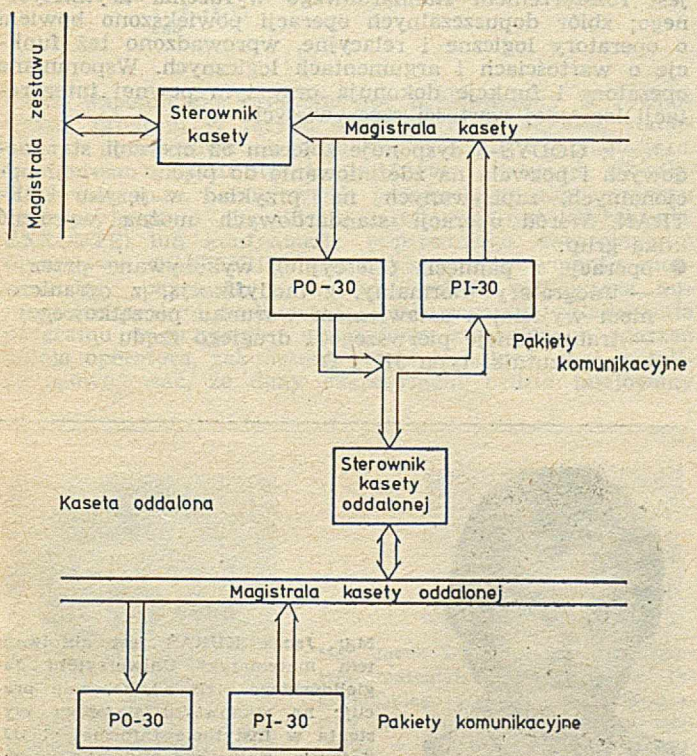
Rozpatrzmy następującą sytuację. System operacyjny odbiera przerwanie o zgłoszeniu jednej z kaset zestawu i w odpowiedzi inicjuje odczyt jej stanu. Następnym etapem jest analiza słowa stanu kasety, która polega na rozpoznaniu zgłoszonych pakietów i inicjacji ich odczytu.

W trakcie analizy stanu zostały zgłoszone inne pakiety, co zostało oczywiście zapamiętane na odpowiednich bitach słowa stanu kasety. Ponieważ operacja odczytu pakietów nie została zakończona, kaseta nie generuje ponownego zgłoszenia, a w konsekwencji nie jest inicjowany ponowny odczyt, co powoduje przepełnienie jej stanu.

Aby powyższy stan wyeliminować, współpraca z INTELDIGIT-PI powinna zapewnić powtarzanie odczytu stanu kasety do chwili otrzymania jego wartości zerowej, czyli do chwili zwolnienia kasety. Z analizy sytuacji wynika, że jakkolwiek problem jest rozwiązywalny programowo, to powoduje ona wydłużenie procedury obsługi. Dodatkową niedogodnością jest tu potrzeba zajmowania łącz transmisyjnych zbędnymi odczytami stanu kasety.

OPROGRAMOWANIE KASET ODDALONYCH

Stosując zestaw INTELDIGIT-PI w zakładach przemysłowych o dużej liczbie punktów pomiarowo-regulacyjnych może się okazać, że liczba pakietów zestawu jest zbyt mała. Rozwiązanie tego problemu jest możliwe przez zastosowanie kaset oddalonych. Kasety te współpracują przez pakiety znakowe typu PI-30 i PO-30 z kasetą stacjonarną.



Rysunek 5

Na rysunku 5 przedstawiono schemat dołączenia kaset oddalonych. Współpraca z kasetą oddaloną polega na przesyłaniu do pakietu PO-30 znaków w kodzie ISO-7, przeznaczonych do sterowania odczytem pakietów i otrzymywania z pakietu PI-30 znaków dających po przetworzeniu stan pakietów kasety oddalonej. Ponieważ INTELDIGIT-PI operuje słowami 16-bitowymi, aby otrzymać pełną informację należy złożyć kolejne znaki w słowo o standardowej długości.

Przejdziemy teraz do sformułowania problemu oprogramowania kaset oddalonych. Operacje odczytu stanu pakietów z kasety oddalonej powodują automatycznie przesłanie dwóch znaków 8-bitowych, które po złożeniu dadzą poprawną wartość. Znaki te są wysyłane kolejno z częstotliwością standardową dla INTELDIGIT-PI. Zgodnie z zasadą pracy kaset, każdy pakiet ma tylko jeden rejestr buforowy. System operacyjny powinien zatem zapewnić odczytanie pierwszej wartości przed przesłaniem następnej przez kasetę oddaloną. W przeciwnym przypadku pierwsza wartość zostanie zniszczona przez nałożenie następnej. Niestety, w większości inmplementacji czas pomiędzy dwoma kolejnymi przesłaniami jest zbyt mały, aby można było zapewnić poprawny odczyt.

Poszczególne odcinki czasu zależą od typu maszyny cyfrowej oraz budowy samego systemu operacyjnego. Aby zapewnić poprawną pracę, należałoby zmniejszyć częstotliwość zegara zestawu INTELDIGIT-PI lub zainstalować linie opóźniające dla samej kasety oddalonej. Pierwszy sposób jest jednak nie do przyjęcia ze względu na generalne zmniejszenie szybkości działania zestawu. Tak więc zostaje tylko wstawienie linii opóźniającej, gdyż sytuacji tej nie można rozwiązać programowo.

PROJEKTOWANIE ZESTAWÓW INTELDIGIT-PI

W artykule zasygnalizowano tylko niektóre problemy, jakie wynikły przy projektowaniu systemu SORT [1]. Jakkolwiek uwagi powyższe dotyczą w szczególności implementacji systemu na komputerze ODRA-1325, to można wyciągnąć z nich wnioski ogólniejsze.

Na podstawie doświadczeń uzyskanych przy oprogramowaniu zestawu INTELDIGIT-PI należy stwierdzić, że ma on wiele zalet (modularność, łatwość kompletowania zestawów, pewna standaryzacja obsługi kaset). Nie sposób pominąć jednak kilku uwag krytycznych, traktując je jako zalecenia dla przyszłych projektantów podobnych systemów. Należy:

- zwrócić szczególną uwagę na fazę projektowania zestawu celem uniknięcia konsekwencji niejednoznaczności adresowania
- przy projektowaniu nowych urządzeń dążyć do uwzględnienia wymagań już istniejącego sprzętu komputerowego, z którym mają one współpracować
- uwzględnić standardy wprowadzone przez firmowe oprogramowanie podstawowe, aby zestawy INTELDIGIT-PI były oprogramowywalne pod kontrolą istniejących programów zarządzających (egzekutorów).

Nieuwzględnienie przy projektowaniu powyższych założeń powoduje nadmierny wzrost kosztów systemów operacyjnych oraz wymaganych przez nich pamięci.

LITERATURA:

[1] Bohonos-Jędraszak K., Lizurek M., Jurkowski J.: Komputerowy system otrzymywania raportów technologicznych. Referat z konferencji „Zastosowanie komputerów w przemyśle", Szczecin 20—21 listopada 1978

[2] Egzekutor EX2P. Elektroniczna maszyna cyfrowa ODRA 1325. ELWRO—Wrocław 1971

[3] Sprzężenie komputerów z elementami automatyki i pomiarów urządzenia INTELDIGIT-PI. „MERA-PIAP" Warszawa 1975

JERZY KRÓL, JACEK KURAŚ, JACEK LEMBAS
Instytut Informatyki Uniwersytetu Jagiellońskiego
Kraków

Język symulacyjny dla

W Instytucie Informatyki Uniwersytetu Jagiellońskiego opracowany został język GODYS-5, przeznaczony do symulacji komputerowej dynamicznych systemów ciągłych ze zdarzeniami dyskretnymi, oraz jego procesor dla komputerów serii ODRA 1300.

Język GODYS-5 nadaje się do symulacji systemów zawierających zarówno elementy ciągłe (takie jak integratory), jak też zmieniające skokowo stan (przerzutniki, komutatory, detektory przekroczeń alarmowych). Podstawą sporządzania programu w tym języku jest zapis modelu matematycznego badanego obiektu w postaci układu równań różniczkowo-logicznych, schematu blokowego lub grafu funkcyjnego. Zasadniczą częścią programu jest opis tego modelu, odwzorowywany w procesie translacji w model cyfrowy. Na modelu można przeprowadzać eksperymenty symulacyjne, związane na przykład ze zmianą wartości parametrów systemu lub symulatora, których wyniki podbudowane odpowiednią teorią pozwalają wnioskować o własnościach obiektu. Wyniki symulacji mogą być prezentowane w postaci przejrzystych wydawnictw (tabel i wykresów sporządzanych na drukarce wierszowej), mogą też być przetwarzane (na przykład przez programy analizy widmowej, statystycznej i inne).

Podjęcie prac nad stworzeniem języka GODYS-5 było wynikiem zapotrzebowania szeregu zespołów badawczych na język symulacyjny o wspomnianym przeznaczeniu. W pracach tych niemałe znaczenie miały doświadczenia autorów uzyskane w trakcie realizacji prostych języków symulacyjnych bazujących na grafach funkcyjnych (GODYS-1, 2, 3 [1, 3]), a w szczególności w trakcie realizacji wzorcowego języka symulacyjnego GODYS-4 [4].

Przystępując do projektu języka GODYS-5 przyjęto następujące założenia:

- język powinien być prosty i zrozumiały, jednocześnie powinien mieć duże możliwości wyrażania pojęć w zakresie przewidywanych zastosowań
- język i jego procesor powinny dać się łatwo modyfikować
- proces translacji powinien być oparty na metodach formalnych
- procesor języka powinien efektywnie pracować w typowej konfiguracji systemu ODRA 1300 z PAO 32 K słów i z pamięciami taśmowymi lub dyskowymi.

Konsekwencją kilkumiesięcznej realizacji projektu było opracowanie języka o strukturze frazowej, łatwego do opanowania i prostego w użytkowaniu, oraz napisanie jego procesora w formie dwóch programów (translator i system wykonawczy). Programy te zostały napisane strukturalnie w językach FORTRAN IV (95%) i PLAN (5%). Pozwala to na stosunkowo łatwą ich modyfikację w niewielkim tylko stopniu zmniejszając szybkość realizacji eksperymentu symulacyjnego.

Program w języku GODYS-5 składa się z dwóch części [5]: 1) opisu modelu symulowanego systemu i 2) sekwencji dyrektyw sterujących eksperymentami przeprowadzanymi na modelu.

W procesie translacji opis modelu jest przetwarzany na równoważny program w języku maszyny abstrakcyjnej SIM/AGH (zdefiniowanej na potrzeby projektu). Maszyna ta implementowana jest przez interpreter, którego pracą sterują wspomniane dyrektywy. Generowane w trakcie eksperymentu symulacyjnego wyniki gromadzone są w pamięci masowej, a następnie opracowane w sposób określony dyrektywami wyjścia (PRINT, PRPLOT, GRAPH, PLOTXY).

Opis modelu składa się z:

- sekcji deklaracji
- właściwego opisu modelu.

W sekcji deklaracji deklarowane są parametry modelu (parametry proste i tablice), którym następnie, w eksperymentach symulacyjnych, mogą być nadawane wartości. Deklarowane są tam również identyfikatory — synonimy stałych używanych w opisie modelu oraz indentyfikatory zmiennych obserwowanych w procesie symulacji.

Na właściwy opis modelu składa się sekwencja instrukcji opisu modelu, ujęta w nawiasy syntaktyczne BEGIN-END. Instrukcje opisu modelu mają postać instrukcji podstawienia, przy czym:

— zmienne występujące po lewej stronie operatora podstawienia muszą być unikalne, tzn. mogą być zdefiniowane tylko raz

— instrukcje opisu modelu nie muszą być wykonywane w kolejności zapisu.

Kolejność wykonywania operacji w programie wynikowym, zwana często sekwencją obliczalną, określana jest przez translator na poziomie języka maszyny SIM/AGH.

Wyrażenie pojawiające się w instrukcji opisu modelu jest rozszerzeniem standardowego wyrażenia arytmetycznego; zbiór dopuszczalnych operacji powiększono bowiem o operatory logiczne i relacyjne, wprowadzono też funkcje o wartościach i argumentach logicznych. Wspomniane operatory i funkcje dokonują przy tym pewnej interpretacji logicznej wartości rzeczywistych.

Język GODYS-5 dysponuje zbiorem 52 operacji standardowych i pozwala na zdefiniowanie do pięciu operacji opcjonalnych, zapisywanych na przykład w języku FORTRAN. Wśród operacji standardowych można wyróżnić kilka grup:

- operacje z pamięcią (inercyjne), wykonywane przez
  — integratory (normalny, z modyfikacją, z ograniczeniem wyjścia, z ustawieniem warunku początkowego)
  — transmitancje pierwszego i drugiego rzędu
  — przerzutniki typu JK i D

Dr inż. Jerzy KRÓL jest absolwentem kierunku „automatyka", specjalność „technika cyfrowa" (AGH-1970), stopień doktora uzyskał na Wydziale Automatyki i Informatyki Politechniki Śląskiej (1974). Od 1975 roku pracuje w Instytucie Informatyki UJ na stanowisku adiunkta, pełniąc obowiązki zastępcy dyrektora Instytutu i kierownika Zakładu Podstaw Informatyki. Specjalizuje się w zakresie metod translacji języków programowania i systemów operacyjnych.

Mgr Jacek KURAŚ jest absolwentem matematyki Uniwersytetu Jagiellońskiego (1974 r.). Obecnie pracuje na stanowisku starszego asystenta w Instytucie Informatyki UJ. Zajmuje się zagadnieniami związanymi z systemami operacyjnymi i procesami współbieżnymi.

komputerów ODRA 1300

— licznik
— histereza typu „luz mechaniczny"
— ekstrapolator zerowego rzędu
— ekstrapolator algebraiczny, tzw. blok izolacyjny, umożliwiający wprowadzenie definiowanych pętli algebraicznych
— generatory liczb losowych
— operacje specjalne (dioda, tyrystor, układ antyrównoległy dioda—tyrystor)

- operacje z historią
— różniczkowanie
— transmitancja typu (As+1)/(Bs+1)
— opóźnienie jedno- i wielokrokowe
— detektor zera
— generatory pulsów i impulsów
— komutatory
- operacje natychmiastowe
— matematyczne, w tym funkcje ciągłe jednej i dwóch zmiennych zadane tablicami
— funkcje specjalne, używane w symulacji systemów dynamicznych (ogranicznik, funkcja charakterystyczna na odcinku domkniętym, funkcja pochylniowa, funkcje typu skoku jednostkowego, kwantyzator, przekaźnik bez histerezy, przełączniki sterowane, funkcja schodkowa zadana tablicą, detektor przekroczeń alarmowych).

Przykładowy program-opis modelu przedstawiono poniżej.

```
GODYS-5        COMPILATION BY #XGD$        V1.0    06/06/78


   0        MODEL BANGBANG
   1        PARAMT X10,X20,A,B,C,D
   2        PREPARE X1,X2,SIGNX2,U
   3        BEGIN
   4        X1= INTEG(X2;X10)
   5        X2= INTEG(U-X1;X20)
   6        SIGNX2 = RELAY(X2;A,B,-A,-B)
   7        U= RELAY (X1+X2- SIGNX2*LOG(ABS(1+X2+SIGNX2)) ;        B
            A,-B,-A,B)
            END
```

Jak wspomniano, program jest przetwarzany w równoważny program w języku maszyny abstrakcyjnej, stanowiący podstawowe wejście do systemu wykonawczego. Pracą tego systemu sterują dyrektywy symulacji. Umożliwiają one m.in. zainicjowanie pojedynczego eksperymentu symulacyjnego o charakterze autonomicznym (dyrektywa EXECUTE) lub kontynuację poprzedniego eksperymentu (dyrektywa CONTINUE). Dla takiego eksperymentu można ustalić parametry symulacji (metodę całkowania numerycznego, krok całkowania itp.), warunki składowania programu w przypadku wystąpienia szczególnych zdarzeń (akcja operatora, zakończenie czasu symulacji, itp.), a także spowodować, że dany eksperyment będzie inicjowany nawet w przypadku, gdy w poprzednim był błąd wykonania. Możliwe jest też bieżące śledzenie wartości zmiennych modelu na określonym odcinku czasu.

Aktualnie system wykonawczy dysponuje sześcioma procedurami całkowania numerycznego, w tym zmiennokrokową i przyspieszania wykładniczego dla całkowania układów równań sztywnych (o skrajnie różnych wartościach własnych).

Wartości parametrów modelu można ustalać dyrektywami DATA. Dyrektywy wyjścia powodują wyprowadzanie w określonej formie (tabela, wykres wartości sporządzony na drukarce wierszowej, płaszczyzna stanu) podzbioru zmiennych obserwowanych, których wartości zarejestrowane zostały w pamięci masowej, w trakcie eksperymentu symulacyjnego. Skalowanie wykresów dokonywane jest automatycznie, z uwzględnieniem optymalnego (w sensie czytelności) doboru współczynnika skali lub też skala ustalana jest na podstawie podanych przez użytkownika wartości granicznych danej zmiennej. Inne dyrektywy pozwalają na składowanie i wznawianie programu i na komunikację systemu wykonawczego (poprzez plik w pamięci masowej) z innym programem, na przykład z programem optymalizacji parametrycznej.

Zasadniczymi elementami translatora i systemu wykonawczego są przetworniki sterowane składnią i bazujące na gramatykach precedensyjnych. Pozwoliło to na uzyskanie przejrzystej struktury programowej systemu oraz dobrej diagnostyki i lokalizacji błędów.

Translator jest dwuprzebiegowy. W pierwszym przebiegu generowany jest lokalnie zoptymalizowany program w języku maszyny abstrakcyjnej. W drugim przebiegu następuje wyznaczenie sekwencji obliczalnej operacji i utworzenie programu wynikowego. Program ten poprzez plik pamięci masowej przekazywany jest do systemu wykonawczego. W systemie tym zainicjowanie eksperymentu symulacyjnego poprzedzone jest intensywną kontrolą poprawności parametrów modelu i symulacji. Pozwoliło to na zminimalizowanie kontroli błędów na etapie interpretacji i tym samym zapewniło jej dużą szybkość.

Kompilator i system wykonawczy zaopatrzone są obficie w środki uruchamiania, takie jak wydruk mapy identyfikatorów, symbolicznej mapy programu wynikowego, śladu wykonania i inne, co pozwala na łatwe wykrycie i lokalizację ewentualnych błędów.

Język GODYS-5 użytkowany jest od wiosny 1978 r. w wielu zespołach — zarówno dla celów badawczych, jak i dydaktycznych (dla tych celów procesor języka rozpowszechniany jest nieodpłatnie). Uzyskane doświadczenia pozwalają sądzić, że język ten jest prostym i efektywnym narzędziem do symulacji szerokiej klasy systemów dynamicznych ciągłych i ciągłych ze zdarzeniami dyskretnymi.

Należy też podkreślić, że analiza wspomnianych doświadczeń wpłynęła na ostateczną formę implementacji języka GODYS-4 [4] i nadała jej szczególnie wysokie walory użytkowe.

Mgr Jacek LEMBAS jest absolwentem matematyki Uniwersytetu Jagiellońskiego (1974 r.). Obecnie pracuje na stanowisku starszego asystenta w Instytucie Informatyki UJ. Zajmuje się problemami związanymi z automatyczną konstrukcją translatorów.

LITERATURA:

[1] Jakubowski R., Król J.: Implementation of the simulation language based on functional graphs. „Podstawy Sterowania" t. 2, 1972
[2] Król J.: Symulacja cyfrowa złożonych systemów elektromechanicznych. Rozprawa doktorska. Politechnika Śląska, Gliwice 1974
[3] Praca zbiorowa: GODYS-3 — język do symulacji złożonych systemów dynamicznych bazujących na grafach funkcyjnych. Dokumentacja Projektowa — „Systemy Sterowania". PETROINFORM, Kraków 1974
[4] Praca zbiorowa: GODYS-4 — język do symulacji ciągłych układów dynamicznych. Cz. I. Raport roboczy II UJ, Kraków 1977
[5] Praca zbiorowa: GODYS-5 — język do symulacji ciągłych ze zdarzeniami dyskretnymi. Raport roboczy II UJ, Kraków 1978

MAREK BILIŃSKI, EDWARD PTAK
Biuro Studiów i Projektów Hutnictwa „Biprostal"
Kraków

Język wyszukiwawczy JWO

System Wyszukiwania Informacji ODYS został zaprojektowany i oprogramowany w Biurze Studiów i Projektów Hutnictwa „Biprostal" w Krakowie w latach 1978—1979, a obecnie jest sukcesywnie wdrażany na komputerze ICL-2903. System bazuje na zbiorach indeksowo-sekwencyjnych zapisywanych na dyskach o pojemności 4,9 mln znaków. Wyszukiwanie informacji odbywa się w oparciu o kartotekę inwersyjną z wykorzystaniem hierarchicznego tezaurusa zarówno w trybie wsadowym, jak i konwersacyjnym.

W systemie ODYS można gromadzić informacje o książkach, artykułach, patentach, normach, literaturze firmowej oraz o projektach BSiPW „Biprostal". Uniwersalny format zapisu danych, wzorowany na adaptowanym formacie MARC II LC, pozwala na dowolne rozszerzenie zakresu gromadzonej informacji o dodatkowe klasy dokumentów oraz łatwą wymianę zbiorów z innymi systemami.

Język wyszukiwawczy JWO został opracowany dla tworzenia opisów dokumentów gromadzonych w zbiorach systemu ODYS oraz dla formułowania pytań umożliwiających dotarcie do żądanego dokumentu.

OPIS I GRAMATYKA JĘZYKA JWO

Język JWO jest językiem o ściśle zdefiniowanej składni. Opisy wyszukiwawcze związane z treścią indeksowanych dokumentów oraz pytania kierowane do systemu budowane są według omówionych niżej reguł z uprzednio określonych i zawartych w tezaurusie deskryptorów. Opisy wyszukiwawcze mogą składać się z jednego lub większej liczby deskryptorów, nazywanych dalej podmiotami, oraz z deskryptorów modyfikujących, tzw. określeń. Deskryptory modyfikujące mają za zadanie bądź to bliższe określenie deskryptorów głównych, bądź też podanie aspektu, w jakim zostały one użyte. Rolę deskryptora w opisie lub pytaniu określa sposób zapisu deskryptora w pytaniu i użyte ograniczniki.

Definicje składniowe języka podano niżej w notacji Backusa-Naura, zaś ich semantyczną interpretację wraz z przykładami podano jako komentarz do tych definicji.

$S = Z \cup Al$ jest słownikiem języka JWO (zbiór Z jest zbiorem zmiennych syntaktycznych, AL jest alfabetem języka JWO).

$$Al = D_1, D_2, \ldots, D_I, P_1, P_2 \ldots P_J, O_1, O_2 \ldots O_K,$$
$$A, B, C, \ldots, X, Y, Z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0$$

gdzie: D_1 — deskryptor z tezaurusa systemu ODYS (jego opis słowny lub numer systemowy)
P_J — deskryptor parametryczny
O_K — ogranicznik.

<ogranicznik> ::= ? | . | <operator> | <nawias> | <modyfikator> | <identyfikator> | <separator>

<operator> ::= <operator relacji> | <operator działania>
<operator relacji> ::= <|=|>
<operator działania> ::= * | I | — | NIE | + | LUB
<nawias> ::= (|) | [|]
<modyfikator> ::= ! | :
<separator> ::= ,
<identyfikator> jest elementem, za pomocą którego można dotrzeć do informacji określonego typu zawartych w opisie dokumentu, np. C1 oznacza pole „autor", OD pole opisu deskryptorowego, J4 pole opisu bibliograficznego itp.

Rozkaz

Najbardziej złożoną jednostką syntaktyczną języka wyszukiwawczego jest rozkaz. Definiuje się go następująco:

1) <rozkaz> ::= <pytanie specjalne> | <pytanie>
2) <pytanie specjalne> ::= ?W <deskryptor> | ?S <deskryptor> | ?C | ?Z | ?0 | ?P | | ?B | ?NNNNN
3) <pytanie> ::= <identyfikator>[<wartość>] | <pytanie>
   <identyfikator>[<wartość>]
4) <wartość> ::= <pytanie deskryptorowe> | <autor> | | <sygnatura> ..... wartość pola określonego przez identyfikator pola wyszukiwawczego.

Przykłady rozkazów:

C1 [MEADOW CH. T.] C2 [ANALIZA SYSTEMÓW INFORMACYJNYCH]
Wyszukana zostanie książka Ch. T. Meadowa pt. „Analiza systemów informacyjnych".
?P
Jest to pytanie specjalne, w którym wymaga się od programu wyszukiwawczego powtórzenia ostatnio zadanego pytania. Wszystkie pytania specjalne omówione są w dalszej części.
OD[(STAL NIERDZEWNA : (EKSPORT: KRAJ RWPG)).]
Wyszukane zostaną dokumenty mówiące o eksporcie stali nierdzewnej do krajów RWPG.

Zasady tworzenia pytania deskryptorowego

Zmienna wartość identyfikowana symbolem OD nosi nazwę pytania deskryptorowego i pozwala sprecyzować treść poszukiwanego dokumentu; konstruuje się ją zgodnie z następującymi definicjami:

1) <pytanie deskryptorowe> ::= <opis> .
2) <opis> ::= <zdanie złożone> | <modyfikacja> | | <opis> <grupa działania> <zdanie złożone>
3) <zdanie złożone> ::= <zdanie> <grupa działania> <zdanie>
4) <grupa działania> ::= , | , <operator działania> ,
5) <zdanie> ::= <opis podstawowy> | (<opis podstawowy>)

Mgr inż. Marek BILIŃSKI ukończył Wydział Elektrotechniki Akademii Górniczo-Hutniczej w Krakowie w 1974 r. Od tego czasu pracuje w Biurze Studiów i Projektów Hutnictwa „Biprostal" w Krakowie (obecnie na stanowisku projektanta w Pracowni ETO). Zajmuje się opracowywaniem i wdrażaniem systemów epd na potrzeby projektowania wspomaganego komputerem.

Mgr inż. Edward PTAK ukończył studia na Wydziale Elektrotechniki Akademii Górniczo-Hutniczej w Krakowie w 1974 r. W tym samym roku rozpoczął pracę w Biurze Studiów i Projektów Hutnictwa „Biprostal" (obecnie na stanowisku projektanta w Pracowni ETO). Zajmuje się opracowywaniem komputerowych systemów wspomagających projektowanie.

6) <opis podstawowy> ::= <deskryptor uogólniony> | <opis podstawowy> <grupa działania> <deskryptor uogólniony>
7) <deskryptor uogólniony> ::= <deskryptor> | <parametr>
8) <parametr> ::= <deskryptor parametryczny> <operator relacji> <liczba dziesiętna bez znaku>
9) <modyfikacja> ::= <modyfikacja!> <zdanie złożone> | (<modyfikacja :> <zdanie złożone>)
10) <modyfikacja!> ::= <deskryptor> !
11) <modyfikacja :> ::= <deskryptor> : | <modyfikacja!> :

Dwa przykłady pytań deskryptorowych:
a) (SUWNICA : UDZWIG > 10.0).
b) (SILNIK ELEKTRYCZNY ! : KONSTRUKCJA, (OBLICZANIE : METODA), PRĄD > 1.0,I, PRĄD <60.0, NAPIĘCIE ZNAMIONOWE = 400.0).

RELACJE LOGICZNE W PYTANIU DESKRYPTOROWYM

Elementy wchodzące w skład pytania deskryptorowego połączone są ze sobą za pomocą tzw. grupy działania. Grupa działania pozwala określić związek, jaki zachodzi pomiędzy jej poprzednikiem i następnikiem w pytaniu deskryptorowym, przy czym związek ten może mieć postać sumy, różnicy względnie iloczynu logicznego.

Iloczyn (część wspólna dwóch terminów) oznaczany jest symbolami „*" lub „I" jako A, *, B lub A,I,B. Obejmuje on obszar przedmiotowy wspólny dla tematów A i B. W pytaniu znaczy to, że zarówno A, jak i B muszą być jednocześnie obecne w tym samym opisie deskryptorowym.

Suma dwóch terminów zapisywana jest przy użyciu symboli „+" lub „LUB" jako A, +, B względnie A, LUB, B i oznacza, że temat o szerszym zakresie określany jest albo przez A, albo przez B, lecz nie ogranicza się tylko do niego. A, LUB, B w pytaniu oznacza, że w opisie deskryptorowym oczekuje się któregokolwiek z tych terminów lub obydwu, lecz nie jest wymagane ich równoczesne wystąpienie.

Dopełnienie terminu, oznaczane symbolem „—" lub „NIE", wskazuje, że dany termin nie powinien znaleźć się w poszukiwanych opisach deskryptorowych. Oznacza to, że jeżeli w pytaniu pojawia się zapis A,—,B lub A, NIE, B, to wymaga się obecności A przy równoczesnej nieobecności B w każdym wyszukanym opisie deskryptorowym. Użycie operatora „NIE" ze względu na określenie kontekstów ma w języku JWO szczególne znaczenie. Możemy bowiem skonstruować pytanie:

OD [KONSTRUKCJA —(KONSTRUKCJA: STAL).]
aby znaleźć dokumenty na temat wszelkich konstrukcji z wyjątkiem konstrukcji stalowych.

RELACJA MODYFIKACJI W PYTANIU DESKRYPTOROWYM

Modyfikacjami nazwano relacje, które umożliwiają określenie wagi danego deskryptora w opisie deskryptorowym oraz pozwalają na rozpoznanie kontekstów, w jakich użyto deskryptorów w opisie czy też pytaniu. Definicję syntaktyczną modyfikacji podano wyżej, obecnie zostanie wyjaśniona ich semantyczna interpretacja.

Modyfikację kontekstowo-określeniową oznaczono w definicjach jako <modyfikacja:>, zaś najlepiej obrazuje ją następujący przykład: (TOR:KOLEJ)

W przykładzie — pierwotnie ze zbioru wszystkich dostępnych dla systemu dokumentów wybrane zostaną tylko te, w opisach których występują deskryptory TOR i KOLEJ, a następnie zbadana będzie relacja „|:". W konsekwencji, w dalszych operacjach, program wyszukiwawczy uwzględni tylko te dokumenty, w których określenie KOLEJ w odniesieniu do podmiotu TOR odpowiada na pytanie jaki? jaka? jakie? w jakim aspekcie?, a zatem relacja (TOR : KOLEJ) odpowiada w języku polskim terminowi: TOR KOLEJOWY.

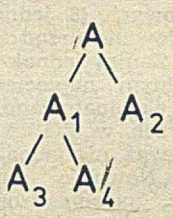
Modyfikację ze względu na temat główny oznaczono jako <modyfikacja!>. Zadaniem jej jest oznaczenie najbardziej ważkich, wiodących w opisie dokumentu tematów, celem uniknięcia błędnej koordynacji pytania z opisem. Dla przykładu rozpatrzmy dokument „A", mówiący o produkcji stali jako o procesie technologicznym i o zastosowanych w tym procesie materiałach.

Jeżeli w celu wyszukania w zbiorze systemu ODYS dokumentów mówiących o wytwarzaniu topników zastosujemy pytanie (TOPNIKI: PRODUKCJA), to najprawdopodobniej wyszukamy między innymi omawiany wyżej dokument „A", uzyskując niewłaściwą odpowiedź. Język JWO pozwala na uniknięcie tej sytuacji przez zastosowanie modyfikacji ze względu na temat główny. Jeżeli postawione wyżej pytanie sformułujemy: (TOPNIKI!: PRODUKCJA), to dokument „A" nie zostanie uwzględniony w odpowiedzi, gdyż w jego opisie deskryptorowym symbolem tematu głównego „!" mogą być oznaczone STAL! czy też PROCES TECHNOLOGICZNY! ale nie TOPNIKI.

PYTANIA SPECJALNE

Pytania specjalne są rozkazami ułatwiającymi użytkownikowi systemu ODYS proces wyszukiwania i nie podlegają one, w przeciwieństwie do pytania, kompilacji przez program wyszukiwawczy.

Pytania specjalne powoduje wykonanie jednej z poniższych funkcji:
?P — wyświetla na monitorze ekranowym ostatnio przetwarzane pytanie
?0 — powtarza ostatnią odpowiedź systemu
?B — powoduje wyświetlenie informacji o błędach kompilacji w przypadku ich zaistnienia w pytaniu wyszukiwawczym
?Z — zapisuje odpowiedź systemu ODYS w specjalnie do tego celu przeznaczonym zbiorze
?C — kasuje z powyższego zbioru zapisane uprzednio odpowiedzi
?nnnnn — wyświetla treść dokumentu o numerze systemowym nnnnn
?WA$_1$ — powoduje ponowne przetworzenie pytania wyszukiwawczego po automatycznym rozszerzeniu go o deskryptory węższe w hierarchii deskryptora A$_1$; np. w tezaurusie zakodowana jest hierarchia:



— pytanie OD [A$_1$, B.] zostanie zamienione na pytanie OD [(A$_1$+A$_3$+A$_4$), B.]
?SA$_1$ — jak wyżej, z tym, że dotyczy deskryptora szerszego. To samo pytanie zostanie zamienione na OD [(A$_1$+ +A), B.]

* * *

Język JWO Systemu Wyszukiwania Informacji ODYS został zaprojektowany w celu stworzenia aparatu umożliwiającego wyszukiwanie dokumentów z możliwie najwierniejszym określeniem zawartej w nich treści.

Zaznaczenie tematu głównego dokumentu i uwzględnienie deskryptorów wraz z aspektem, w jakim zostały one użyte, a także możliwości korzystania z wielkości parametrycznych przy poprawnym indeksowaniu dokumentów powinny pozwolić na uzyskanie kompletnej i dokładnej odpowiedzi.

Ponadto język JWO umożliwia bezpośrednie korzystanie z hierarchicznej budowy tezaurusa i automatyczne poszerzanie pytań wyszukiwawczych o deskryptory semantycznie szersze lub węższe. Ma to również duży wpływ na podniesienie kompletności odpowiedzi. Język został zaprojektowany głównie z myślą o przetwarzaniu trybem konwersacyjnym, co szczególnie widoczne jest w funkcji pytań specjalnych.

Na zakończenie należy stwierdzić, że aczkolwiek założenia języka sprawdziły się na stosunkowo niewielkich zbiorach założonych do celów testowania systemu (ok. 2 000 dokumentów), to jednak na całkowite potwierdzenie przydatności języka trzeba poczekać, aż zbiory dokumentów zaindeksowanych w języku JWO osiągną dostatecznie duże rozmiary.

LITERATURA:

[1] Meadow Ch. T.: Analiza systemów informatycznych. WNT, Warszawa 1972
[2] Turski W. M.: Propedeutyka informatyki. PWN, Warszawa 1975
[3] Dąbrowski M., Laus-Mączyńska K.: Metody wyszukiwania i klasyfikacja informacji. WNT, Warszawa 1978

W blasku korzyści,

Warszawską „przemysłową Wolę" reprezentują przede wszystkim cztery nowoczesne zakłady, zwłaszcza gdy ich nowoczesność określa się stopniem informatyzacji procesów technologicznych i zarządzania.

Będzie to więc relacja o informatyce w Zakładach Mechanicznych im. M. Nowotki, Zakładach Radiowych im. M. Kasprzaka, Kombinacie Przemysłu Narzędziowego „VIS" — Fabryce Wyrobów Precyzyjnych im. gen. Świerczewskiego i Zakładach Wytwórczych Lamp Elektrycznych im. Róży Luksemburg „Polam".

Struktura tych zakładów, wielkość zatrudnienia, złożoność produkcji, ilość i jakość produkowanych asortymentów wymusiły niejako uznanie informatyki za instrument nie tylko potrzebny, ale wręcz niezbędny. Toteż wszystkie te zakłady skwapliwie skorzystały z możliwości płynących z zastosowania informatyki w procesach organizacji, technologii i zarządzania. Rozmaitymi drogami zakłady wolskie osiągnęły obecny stan informatyki — różny w poszczególnych zakładach, tak jak różne są w każdym z nich efekty wynikające ze stosowania komputerów — ale zawsze bardzo istotny dla dalszego rozwoju produkcji i informatyki.

Historia informatyzacji wolskich zakładów ma tyle cech specyficznych, co i wspólnych. Do tych ostatnich należy zaliczyć cechy charakterystyczne dla okresu pionierskiego.

Informatykę na Woli zaczęto stosować wcześnie. Na dużą skalę najwcześniej zaczął „Kasprzak", ale pierwszeństwo w instalacji sprzętu przypadło **„Nowotce"**. O takim sprzęcie mówi się dziś z pobłażaniem, ale w początkach lat sześćdziesiątych maszyny liczące — analityczne SAM (importowane z ZSRR) uważane były za nowoczesne. W „Nowotce", który już wówczas w sieci przedsiębiorstw zrzeszonych w PZL należał do największych producentów (specjalność: silniki, agregaty napędowe, zespoły prądotwórcze dla silników i pojazdów), maszyny te stanowiły poważną wyrękę, eliminując nadmierne zatrudnienie w pionie księgowości. Pierwsze efekty płynące z mechanizacji obliczeń wytworzyły klimat sprzyjający usprawnieniom. Trzeba było rozejrzeć się za czymś lepszym. ZOWAR otrzymał właśnie IBM 1440. Wśród pakietów oprogramowania użytkowego — akurat dla takiego zakładu jak „Nowotko" — znajdował się BOMP. Rychło rozpoczęto adaptację do polskiej specyfiki; jak się później okazało z niezłym skutkiem. Oprócz ewidentnych korzyści dla zakładów, autorzy adaptacji uzyskali w 1973 r. nagrodę ministra Nauki, Szkolnictwa Wyższego i Techniki. A nieco wcześniej (również w oparciu o sprzęt ZOWAR-u) przeniesiono system na większy komputer IBM 360/50.

Pomyślne zakończenie przygody z BOMP-em skłoniło „Nowotkę" do zintensyfikowania zabiegów o własny komputer. Zwłaszcza że w sieci PZL (w Rzeszowie i Kaliszu) pojawiły się pierwsze komputery Jednolitego Systemu R-20. We wrześniu 1976 r. także na Woli zainstalowano R-20. Opóźnienie w stosunku do innych zakładów rekompensowała lepsza konfiguracja komputera — pamięć operacyjna 256 KB, większa liczba pamięci dyskowych (8 jednostek po 7,25 KB), zdublowane urządzenia wyjścia i wejścia. Zanim jednak uporano się z niestabilnym napięciem, jeszcze pewien czas eksploatowano systemy na komputerach w ZOWARZE i POLMO. A ponieważ niespełna półtora roku później „Nowotko" otrzymał drugi komputer, nagromadzone systemy można było poprzydzielać optymalnie — BOMP zainstalowano na polskim komputerze R-32 (z pamięcią operacyjną 256 KB i 8 jednostkami dyskowymi po 30 MB firmy MEMOREX oraz 3 jednostkami po 30 MB — produkcji bułgarskiej), pozostawiając R-20 prostsze systemy ewidencyjne (np. płace) oraz funkcje testowe prac programowych i dublera R-32.

Wydawać by się mogło, że zaprezentowany wyżej potencjał sprzętu zaspokoi potrzeby Zakładów nawet w dość odległej przyszłości. Tymczasem okazuje się, że obydwa komputery pracują co najmniej na dwie zmiany i to tylko dlatego, że zakładowy Ośrodek Informatyki nie ma możliwości uruchomienia drugiej zmiany przygotowania danych. Gdyby 26 posiadanych SOEMTRONÓW eksploatowano w ruchu ciągłym, to przygotowałyby taką ilość danych, że byłyby kłopoty z ich przetworzeniem.

w cieniu niedostatków

Zapowiada się, że proces przygotowania danych uda się usprawnić już w przyszłym roku, przede wszystkim dzięki zainstalowaniu SEECHECKA — najpierw w ośrodku, a potem na poszczególnych wydziałach „Nowotki". Dlatego też równocześnie myśli się o rozbudowaniu konfiguracji R-32, głównie o dodatkowe 256 KB pamięci operacyjnej. W dalszej perspektywie przewiduje się zakup procesora komunikacyjnego i terminali ekranowych, co w konsekwencji sprowadzałoby się do zmiany metod przetwarzania danych.

Troska o uzupełnienie sprzętu i kadry jest jak najbardziej na czasie, bowiem wzrósł popyt poszczególnych służb „Nowotki" na usługi informatyczne. Trzeba tu jeszcze dodać, ze wspomniany Ośrodek Informatyki obsługuje filię „Nowotki" w Siedlcach oraz Ośrodek Badawczo-Rozwojowy. Zachodzi obawa, czy dalsza informatyzacja Zakładów siłami 80-osobowej kadry Ośrodka nie ulegnie zahamowaniu.

Wprawdzie większość systemów już eksploatowanych i przewidzianych do eksploatacji to systemy opracowywane na użytek całej branży, ale wdrażanie, eksploatacja i konserwacja też pochłaniają niemało czasu i energii.

Postanowiłem pisać raczej o kłopotach niż o sukcesach. Tych pierwszych **„Kasprzak"** ma chyba więcej niż „Nowotko". Ma też bogatszą tradycję. Ośrodek Informatyki powstał już w 1961 r., jako jeden z trzech przewidzianych w Uchwale Komitetu Ekonomicznego Rady Ministrów ośrodków pilotowych zastosowań informatyki. Jako taki skorzystał z wielu preferencji. Między innymi bardzo wcześnie otrzymał nowoczesny wówczas komputer ICL 1904E (1968 rok) z PAO 32 K, sześcioma jednostkami pamięci taśmowej, a jeszcze wcześniej szkolono kadrę i przygotowywano systemy w CODKK.

W początkach lat siedemdziesiątych szybka modernizacja Zakładów (powstałych w 1951 r.) dyktowała rozliczne zadania informatyce. Rozwinęła się produkcja oparta na nowoczesnych licencjach Marconi-Thompson i Grundig. „Kasprzak" rozbudował się i zwielokrotnił zatrudnienie.

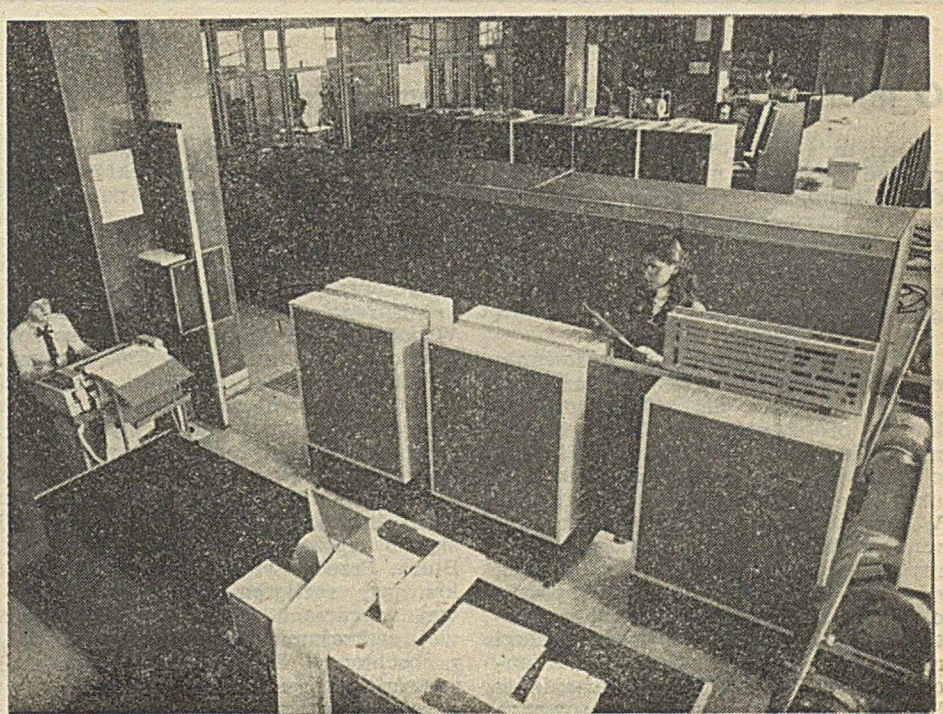
Branżowe Biuro UNIPRO opracowało projekt rozwoju Ośrodka Informatyki na najbliższe lata. Rozbudowano konfigurację ICL (w 1975 r. już 128 K pamięci operacyjnej i 6 jednostek dyskowych po 60 MB). „Kasprzak" jako pierwszy w Polsce użytkownik sprzętu ICL otrzymuje jednostki pamięci dyskowej. Na terenie Zakładów powstaje Rada Sterująca (zastosowaniem informatyki — przyp. red.), złożona z przedstawicieli wszystkich służb. Ona przejmuje troskę o sprawy organizacyjne, pozostawiając Ośrodkowi ETO tylko dbałość o wdrażanie postanowień. Niestety, żywot Rady Sterującej był krótki, a szkoda.

Obecnie Ośrodek jest zdany na siebie. Tymczasem rozwój Zakładów nakłada na informatyków coraz to nowe obowiązki, którym w warunkach manewru gospodarczego coraz trudniej jest sprostać.

Dożywa okresu świetności komputer ICL 1904E. W roku ubiegłym 4 miesiące pauzował. Główna przyczyna — brak części zamiennych. W tej sytuacji jedynym wyjściem są zakupy sprzętu krajowego. Postanowiono, że dalszy rozwój informatyki w „Kasprzaku" będzie odbywał się w oparciu o system dwukomputerowy (dwie jednostki centralne ODRA 1305, każda o pojemności 128 K PAO, sprzężone adapterem międzykomputerowym ADM 305-1). Docelowy rozwój systemu informatycznego musi być ukierunkowany na tzw. przetwarzanie rozproszone. W systemie tym będą organizowane lokalne bazy danych, w których będą gromadzone masowe informacje z ewidencji produkcji i gospodarki materiałowej. Taki system organizacji przetwarzania danych wynika z potrzeby operatywnego sterowania produkcją masową. Specyfiką „Kasprzaka" jest bowiem masowa produkcja na potrzeby rynku i eksport o bardzo krótkich cyklach wytwarzania i wielu skomplikowanych powiązaniach kooperacyjnych. Wydaje się, że realizacja systemu przewarzania rozproszonego jest tym bardziej realna, że producent krajowego sprzętu minikomputerowego podjął udaną produkcję mikroprocesorów, których konfiguracja spełnia oczekiwania „Kasprzaka".

Tymczasem realizowany jest projekt modernizacji i rozbudowy Ośrodka Obliczeniowego w ZRK (projekt powstał w Pracowni Projektowania Systemów MERA ELWRO). W trakcie uruchamiania jest pierwsza z dwóch jednostek centralnych ODRA 1305. Ponadto jeszcze w tym roku „Kasprzak" otrzyma urządzenia bezpośredniego dostępu do centralnej bazy danych (monitory ekranowe MERA 7911 z jednostką sterującą MERA 7802 oraz drukarki znakowo-mozaikowe DZM 180 KSRS, które poprzez system skanerowy i procesor komunikacyjny pracują jak urządzenia monitorowe).

W oparciu o procesor komunikacyjny i system skanerowy od trzech lat funkcjonuje system teletransmisji zdalnej z zakładów ZPE UNITRA (opartej na terminalach ICL 7503) oraz eksploatowane są lokalnie monitory ekranowe ICL 7181/2 (VDU), głównie do prac programowych w ramach systemu operacyjnego GEORGE 3. Są już pierwsze wdrożenia z zakresu Systemu Informacji Kierownictwa.



Docelowy rozwój Ośrodka i wdrożenie systemu dwukomputerowego uwarunkowane jest przebudową pomieszczeń w Ośrodku, ponieważ projekt sprzed 25 lat nie spełnia wymogów bezpieczeństwa przeciwpożarowego. W tej sytuacji uruchomiona ODRA 1305 może jedynie wspomagać wysłużoną ICL-kę.

Inną bolączką jest chroniczny brak kadry. Ośrodek zatrudnia 66 ludzi, co w kontekście wyżej relacjonowanych uzupełnień sprzętu i radykalnej zmiany technologii przetwarzania wzbudza lekki niepokój. Podobno etaty są, tylko nie wiadomo skąd brać ludzi. Na przekór istniejącym trudnościom tradycyjnie przychylny informatycę klimat pozwala budzić nadzieję na dalsze wdrożenia systemów w Zakładach Radiowych im. M. Kasprzaka.

Podobnie jak w „Nowotce" Kombinatowy Ośrodek Informatyczny (dawniej ZOPI) rozpoczynał od maszyn liczącо-analitycznych. W 30 rocznicę PRL, w 1974 roku zainaugurowano uroczyście działalność Ośrodka — po 9 miesiącach prac nad pospieszną adaptacją pomieszczeń i przenoszeniem systemów z MLA (przy pomocy OBRI, IOK i UW) na zainstalowaną ODRĘ 1305 (PAO 96 K, 3 jednostki dyskowe po 60. MB, 2 drukarki, 3 terminale ekranowe). Ostatnio udało się wspomóc park tradycyjnych drukarek (10 maszyn) 9-stanowiskowym SEECHECKIEM, który w pierwszym rzędzie zaspokoił potrzeby działu księgowości.

ODRA 1305 rychło weszła w rytm pracy 2-zmianowej, głównie dzięki temu, że swego czasu nie zmarnowano możliwości tak MLA, jak i eksploatowanej przejściowo ODRY 1204.

Szybko objęto automatyzacją podstawowe sfery działalności Fabryki Wyrobów Precyzyjnych — gospodarkę materiałową, gospodarkę wyrobami gotowymi i kadry, planowanie produkcji, normatywny rachunek kosztów, system finansowo-księgowy i obliczenia inżynierskie. Czego nie dało się zrobić szczupłymi siłami kilkunastoosobowej kadry projektantów-programistów (z ogólnej liczby 50 zatrudnionych), nabywano u innych. Dziś można powiedzieć, że może nie tyle Kombinat („Świerczewski" posiada wiele zakładów — od Szczecina po Siedlce) co sama Fabryka na Woli jest wysoce zinformatyzowana. Stało się tak głównie dzięki dobrej propagandzie możliwości płynących z zastosowań informatyki. Dziś do informatyki nikogo w Fabryce nie trzeba namawiać.

Skwapliwość, z jaką poszczególne wydziały ulegają pokusie informatyzacji, pozwala na takie usprawnienia (z punktu widzenia potrzeb ośrodka), jak decentralizacja przygotowania danych. Tak na przykład zanosi się, że SEECHECK będzie eksploatowany przez najbardziej zainteresowany dział księgowości, a następne zestawy też będą lokowane bezpośrednio na wydziałach.

Stopień zainteresowania informatyką nie pozostaje bez wpływu na kurczenie się możliwości Ośrodka. Już obecnie wystąpiła potrzeba zainstalowania drugiego komputera. Problem tylko sprowadza się do dylematu: druga ODRA 1305 czy R-35. Zanim zapadnie decyzja, Ośrodek „wydusza" z pracującej obecnie ODRY jej wszystkie możliwości — przystępuje do instalowania systemu operacyjnego GEORGE 3 i zamyśla się instalowanie większej liczby terminali.

O informatyce na studenckiej naradzie

Jednym z wielu tematów, nad którymi dyskutowali na Krajowej Naradzie (Chełm, 9—12 maja br.) przedstawiciele aktywu Socjalistycznego Związku Studentów Polskich, było wprowadzenie elektronicznej techniki obliczeniowej w pionie turystycznym organizacji.

Pewne doświadczenia w stosowaniu eto ma Biuro Podróży i Turystyki „Almatur" SZSP, dla którego od kilku lat projektują i realizują programy (głównie na potrzeby administracji) członkowie studenckiego Koła Naukowego Informatyków Uniwersytetu Warszawskiego. W ubiegłym roku powstały bardziej ambitne plany: stworzyć sieć teleinformatyczną łączącą oddziały z centralą Biura. W oparciu o tę sieć planuje się wdrożenie pakietu systemów kompleksowej obsługi ruchu turystycznego. Obecnie studenci z Koła UW i z Koła Mechaników Politechniki Warszawskiej prowadzą już prace projektowe. Wstępne oceny wskazują na osiągnięcie efektów ekonomicznych pozwalających na znaczne rozszerzenie oferty „Almaturu".

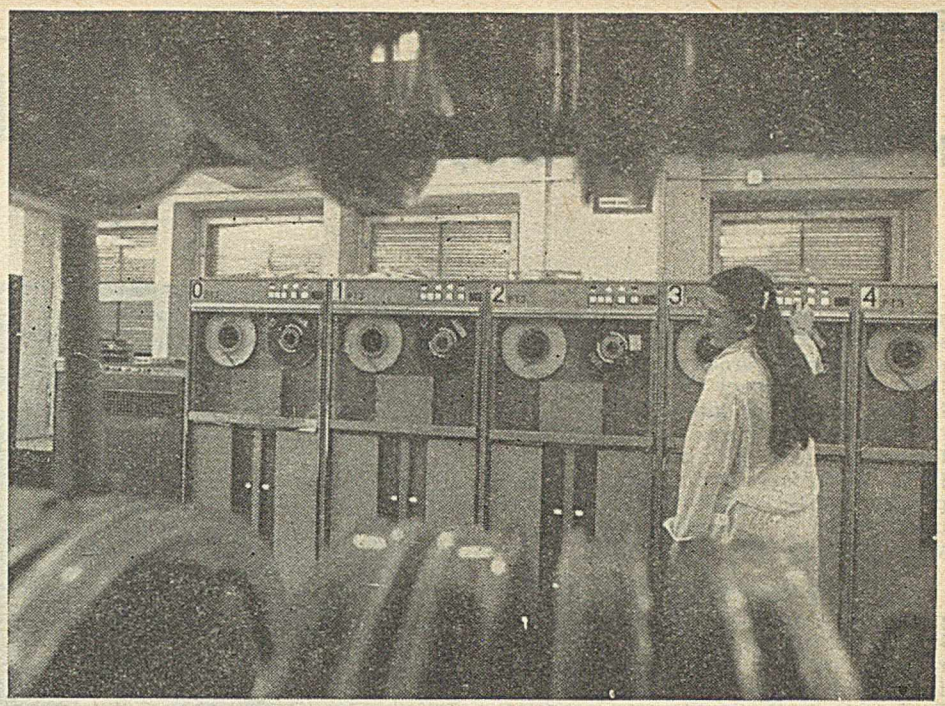
Plany te spotkały się z dużym zainteresowaniem uczestników tegorocznej Narady. Dzięki uprzejmości ZETO Wrocław, które udostępniło sprzęt (jednostka centralna RC 2200, 2 terminale ekranowe i drukarka mozaikowa), zaprezentowano niektóre systemy. Najbardziej atrakcyjny był system rezerwacji miejsc na wycieczki zagraniczne. Wdrożenie tego systemu znacznie podniesie jakość usług świadczonych przez „Almatur" i przyczyni się do lepszego wykorzystania oferty Biura. Przedstawiono też program dokonujący rozliczenia grupy wyjazdowej w ramach wymiany bezdewizowej z krajami socjalistycznymi. Ta pracochłonna dotąd czynność została znacznie uproszczona i skrócona przy użyciu komputera. Jest to jeszcze jeden argument przemawiający za stosowaniem informatyki — zwłaszcza że turystyka studencka cierpi na dotkliwy brak etatowych pracowników.

W czasie dyskusji pod hasłem „Informatyka na potrzeby organizacji turystyki studenckiej" zwrócono uwagę na konieczność szybkiego dostępu do aktualnych informacji w pionie turystycznym SZSP — oczywiście przy użyciu komputerów. Bardzo atrakcyjna wydaje się propozycja wdrożenia systemu informacyjnego na potrzeby organizatorów turystyki. Sieć teleksów bądź terminali zastąpiłaby szybko dezaktualizujące się informatory. Odpowiedzi na konkretne pytania można by było otrzymać z zawsze aktualnego centralnego banku informacji.

Warto wykorzystać wiedzę i zapał studentów w podnoszeniu kultury technicznej, we wprowadzaniu informatyki nie tylko w organizacjach turystycznych. **(P.K.)**

Jeśli chodzi o sprzęt, Ośrodek Informatyki Zakładów Wytwórczych Lamp Elektrycznych im. Róży Luksemburg „Polam" można by nazwać ubogim krewnym wolskich potentatów. Historię ma prawie tak samo długą — powstał w 1965 roku, korzystał blisko 9 lat z życzliwości i usług „Kasprzaka". W 1974 roku poszedł „na własne". Cały jego sprzęt to ODRA 1304 ze skromną pamięcią operacyjną 32 K i 8 jednostkami pamięci taśmowej. Nieco lepsze jest wyposażenie w urządzenia wejścia — wyjścia (po 2 czytniki i 2 drukarki), pozwalające na maksymalne wykorzystanie możliwości jednostki centralnej. Również skromne jest wyposażenie stacji przygotowania danych — 4 dziurkarki oraz 3 sprawdzarki. A przecież „Róża Luksemburg" to nowoczesny zakład o europejskiej renomie, główny producent RWPG pełnego asortymentu świetlówek, zakład o przedwojennej tradycji (Zakłady Philipsa), zatrudniający dziś parotysięczną załogę.

Skromny sprzęt, szczupła kadra, a już dorobek informatyczny Zakładów spory. Poza najpowszechniejszymi zastosowaniami informatyki, w gospodarce materiałowej, płacach, kadrach i obrocie towarowym — przetwarza się na komputerze dane o kapitalnym znaczeniu dla doskonalenia procesów technologicznych, śledzi się przyczyny powstawania braków, usprawnia procesy przygotowania produkcji. Zanosi się zresztą na rychłą poprawę wyposażenia Ośrodka. I tak już w roku przyszłym Ośrodek uzyska ODRĘ 1305 (z pamięcią operacyjną 128 K i pamięciami dyskowymi), z wyposażeniem umożliwiającym zdalny dostęp do komputera. Pozwoli to „przybliżyć" do Ośrodka wydziały Zakładu. Stacja przygotowania danych będzie dysponować SEECHECKIEM, a nadzieję na uzupełnienia kadrowe wiąże się z absolwentami warszawskich kierunków informatycznych.



Tak w ogromnym skrócie można przedstawić informatykę na Woli — warszawskiej dzielnicy nowoczesnego przemysłu. To, co jest wspólne informatyce przedstawionych zakładów, to stosunkowo wczesny start, okres stagnacji na przełomie lat sześćdziesiątych i siedemdziesiątych, okres kłopotów ze sprzętem i kadrą w połowie lat siedemdziesiątych.

Bez informatyki obyć się już nie sposób. Stała się ona nieodzowna w procesach organizacji i zarządzania. Najbliższe rokowania wydają się więc pomyślne. Świadczą o tym plany uzupełnienia sprzętu, modernizacji systemu przetwarzania danych oraz jakościowego i ilościowego wzmocnienia kadry informatyków.

Tekst przygotował Krystyn BERNATOWICZ, zdjęcia: Ireneusz RADKIEWICZ

O konferencji w Żaganiu

Nie dopisała frekwencja, zawiedli referenci, a mimo to konferencję AMPIG-79 (Żagań, 17—18 czerwca br.), organizowaną przez Towarzystwo Naukowe Organizacji i Kierownictwa w Zielonej Górze, należy uznać za udaną.

Założeniem konferencji była prezentacja problemów i osiągnięć informatyki na ziemi lubuskiej, co po pierwsze pozwoliło na dokonanie bilansu, a po drugie umożliwiło konfrontację z doświadczeniami informatyków spoza ziemi lubuskiej, głównie z Warszawy.

W ramach problematyki ogólnokrajowej

- budowę systemów państwowych i wynikające z nich konsekwencje dla ośrodków terenowych omówił dr Józef Oleński z OBR GUS
- możliwości zastosowania komputerów do procesów sterowania produkcją przedstawił mgr inż. Wincenty Łada z Centrum ETOB
- możliwości przenoszenia oprogramowania z komputerów ODRA na komputery Jednolitego Systemu (głównie R-32) zaprezentował mgr Stanisław Mrozik ze Zjednoczenia Informatyki.

Pozostałe referaty bazowały na doświadczeniach ośrodków macierzystych autorów. W grupie tej na szczególną uwagę zasługiwały referaty mgr. inż. Ryszarda Jankowskiego i mgr. Janusza Czekaja z zielonogórskiego ZETO („Sterowanie zapasami materiałowymi przy wykorzystaniu komputera współpracującego z zestawem monitorów") oraz mgr. inż. Jerzego S. Nowaka z Kombinatu Urządzeń Mechanicznych „Bumar" w Łabędach („System obiektowy Kombinatu").

Absencja kilku referentów (wśród nich niestety także miejscowych) zmusiła organizatorów do improwizacji — jak się zresztą okazało bardzo udanych. Przede wszystkim przedłużono dyskusję. Wiele pytań skierowano pod adresem referentów, co przemawia tylko za tym, że w referatach poruszono tematy niewątpliwie istotne dla praktyków z ziemi lubuskiej.

Przy okazji konferencji miała też miejsce impreza o nieco innym charakterze. Z inicjatywy Instytutu Informacji Naukowej, Technicznej i Ekonomicznej (a zwłaszcza dyrektora Instytutu — prof. dr. hab. inż. Konrada Fiałkowskiego), przy pomocy ZOPI w zielonogórskim „Zastalu", odbyło się inauguracyjne spotkanie klubu użytkowników systemu automatycznego gromadzenia i udostępniania informacji naukowej, technicznej i ekonomicznej — SINFO 1300. Na spotkaniu nie zabrakło nikogo z 9 użytkowników systemu. Przekazano doświadczenia eksploatacyjne, omówiono dokonane modyfikacje systemu. Wszyscy uczestnicy tego konwersatorium uznali, że należy regularnie organizować takie spotkania i to nie rzadziej niż co pół roku. **(K.B.)**

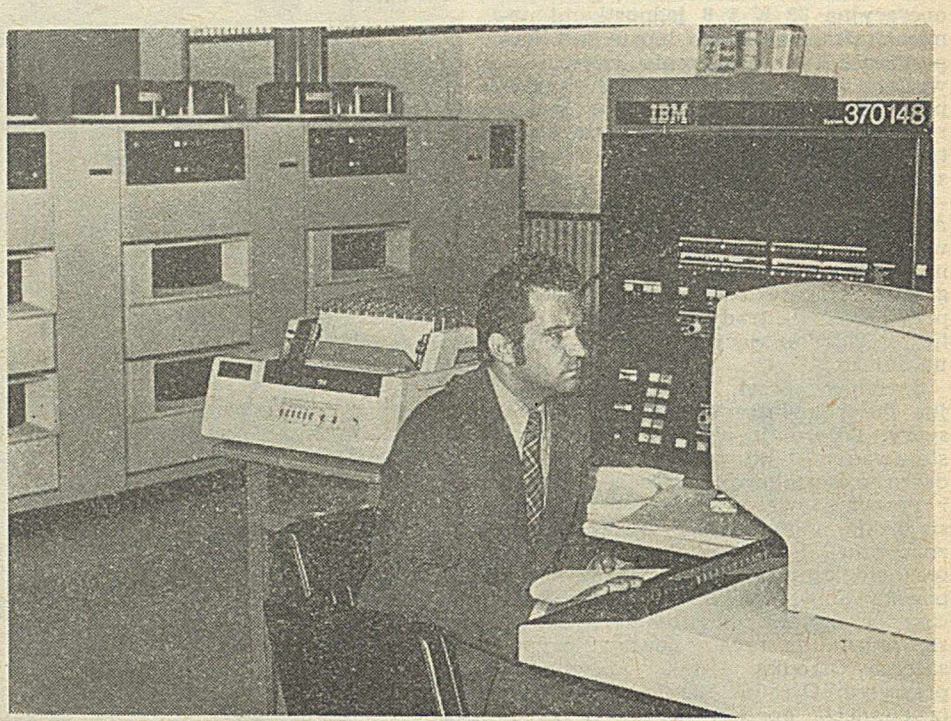
Kolejna instalacja komputera IBM

W dniu 30 maja br. w gmachu Instytutu Organizacji Zarządzania i Doskonalenia Kadr w Warszawie nastąpiło otwarcie ośrodka pomocniczego IBM (IBM Support Center), wyposażonego w system komputerowy IBM/ /370 model 148 wraz z urządzeniami towarzyszącymi i oprogramowaniem IBM. W uroczystości wzięli udział przedstawiciele nauki, administracji, użytkownicy systemów komputerowych IBM.

Witając gości, dyrektor Instytutu, prof. dr Bronisław Ostapczuk, podkreślił, że już od kilku lat intensywnie rozwija się współpraca między Instytutem Organizacji Zarządzania i Doskonalenia Kadr a koncernem IBM, przynosząc korzyści dla obu stron. Zainstalowanie nowego komputera IBM w polskiej placówce naukowej umożliwi popularyzację najbardziej nowoczesnych rozwiązań systemowych i odpowiednie przygotowanie kolejnych użytkowników komputerów IBM.

Warto przypomnieć, że pierwszy komputer, IBM 360 model 50 wraz z wyposażeniem, urządzeniami peryferyjnymi, częściami zamiennymi i podstawową dokumentacją techniczną i eksploatacyjną systemu, koncern IBM przekazał jako dar dla Centrum Zdrowia Dziecka. Dzięki tej instalacji przeszkolono ok. 20 tysięcy polskich użytkowników komputerów IBM. (Ponadto firma IBM szkoliła polskich specjalistów i wykładowców w swoich ośrodkach na terenie Austrii, Wiekiej Brytanii, Francji, RFN).

Otwarcia ośrodka IBM dokonał prof. dr Bronisław Ostapczuk



Nowy komputer IBM umożliwi odpowiednie przygotowanie kolejnych użytkowników systemów komputerowych IBM

Zdj. St. Porowski

Dyrektor generalny IBM ROECE, Otto Weideli, wyraził zadowolenie z otwarcia ośrodka, co — zapewnił — traktuje jako dowód uznania dla wieloletniego partnera i zapowiedź dalszej współpracy. O. Weideli dodał, że osiągnięcia firmy IBM, chętnie udostępniane stronie polskiej w ramach powiązań kooperacyjnych wynikających z dwustronnych porozumień i umów, mogą również przyczynić się do podniesienia poziomu polskiej informatyki.

W ośrodku, który będzie pracował na dwie zmiany, został zainstalowany system komputerowy IBM/370 model 148, z pamięcią operacyjną o pojemności 1 MB. W zestaw systemu wchodzą urządzenia pozwalające na jego wykorzystanie także w trybie konwersacyjnym:

- jednostka sterowania teletransmisją — IBM 3705
- nowoczesne monitory ekranowe — IBM 3278 i 3277.

IBM 370 model 148 umożliwia korzystanie z następujących systemów operacyjnych IBM: VM/370, OS/VS1, DOS/VS, VM/VSE.

Aktualnie ośrodek dysponuje następującym oprogramowaniem:
VS/APL — konwersacyjny system programowania w języku APL
DL/I i IMS/VS — uniwersalne systemy zarządzania bazą danych
CICS/VS — system sterujący siecią teleprzetwarzania
DMS/VS — zestaw programów ułatwiających wdrażanie systemów informacyjnych w trybie *on-line*
GIS/VS — system umożliwiający szybkie uzyskanie informacji z bazy danych przez użytkowników nie będących specjalistami w przetwarzaniu danych
ATMS-II — system przetwarzania informacji tekstowych
STAIRS/VS — system gromadzenia i wyszukiwania informacji (technicznej, naukowej, ekonomicznej, kadrowej)
MPSX/370 — rozszerzony pakiet oprogramowania w zakresie metod matematycznych
PLANCODE — interaktywny system przygotowania i weryfikacji planów
COPICS — szereg programów licencyjnych realizujących systemy sterowania i zarządzania produkcją w trybie *on-line*
INTERPERS — konwersacyjny system kadrowy.

Szczegółowych informacji o działalności i możliwości współpracy z ośrodkiem IBM udzielają:
IBM — WTE/ME/A Corp., Oddział w Polsce
ul. Szpitalna 6/18, 00-031 Warszawa
tel. 27-76-64, telex: 81-27-35 IBM PL
oraz IBM Support Center c/o IOZiDK
ul. Wawelska 56, 02-267 Warszawa
(tel. 25-86-70). **(L.G.)**

Isotimpex

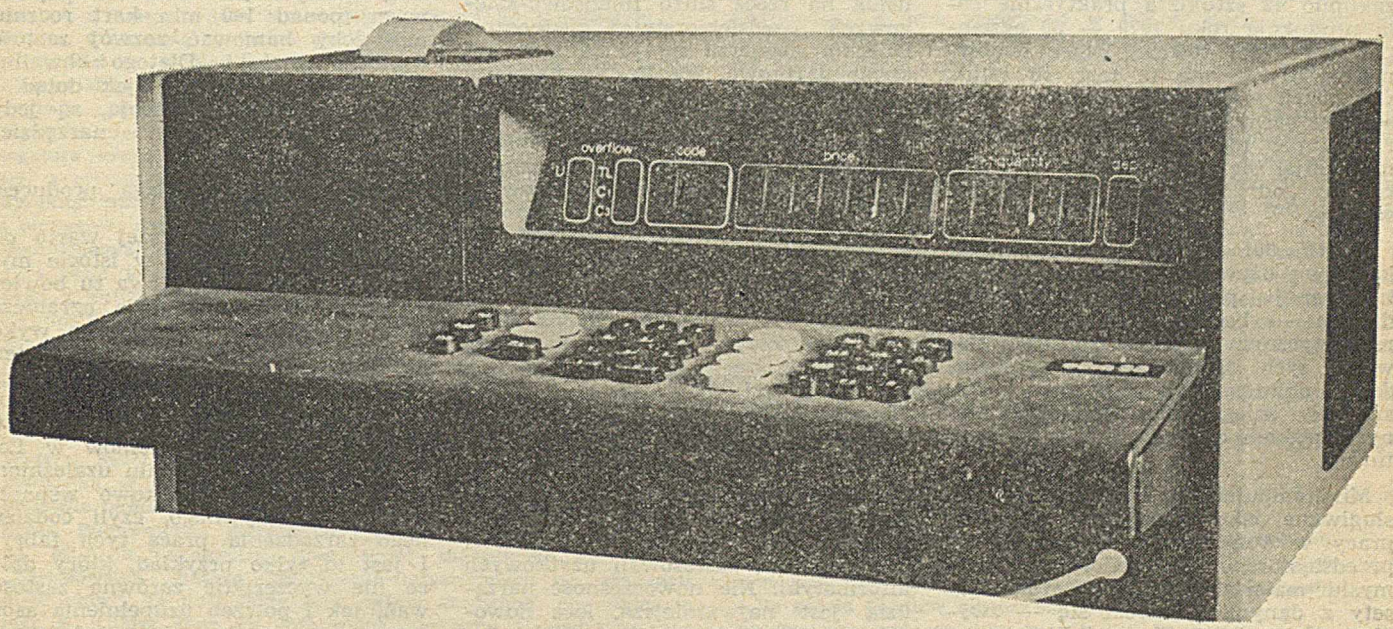
Elektroniczne rejestrujące urządzenie kasowe „ELKA 88"

Dokładna, szybka, niezawodna obsługa i rozliczenie w zakładach zbiorowego żywienia.

Nowoczesna technologia produkcji oparta na wykorzystaniu mikroprocesorów i ferrytowej pamięci operacyjnej 4 k × 4 bity — oto podstawa dużych możliwości operacyjnych rejestracji kasowej za pomocą urządzenia „ELKA 88".

DANE TECHNICZNE:

— 200 rejestrów pamięci ceny jednostkowej
— 200 rejestrów pozycji rachunku
— na użytek 7 grup kelnerów, dla każdej po 4 rejestry pamięci — kuchnia, bufet, anulowanie, liczba obsłużonych klientów
— magnetycznie zakodowane klucze dla każdego kelnera
— anulowanie błędnego zamówienia o równowartość zwracanych pozycji
— rozrachunek według brygad, asortymentu i całej kasy
— drukuje 3 czeki — dla klienta, dla kuchni i dla bufetu
— automatyczna blokada przy przekroczeniu pojemności rejestrów
— drukarka SEIKO —2,5 wierszy/s
— wyświetlanie — 17-pozycyjne, o kolorze zielonym; wyświetlacze segmentowe
— wymiary — 230 × 530 × 560 mm
— masa — 31 kg



Eksporter:

Przedsiębiorstwo Handlu Zagranicznego ISOTIMPEX
Bułgaria, Sofia, ul. Czapajewa 51. Telefon: 73-61. Teleks: 022731, 022732

WCT/733/K/79

Pochwała dla MERY 300

Losy komputerów mogą niekiedy przypominać losy artystów. Zwłaszcza w sferze ocen jednych i drugich. Artyści mają impresariów i odbiorców, bywają późno doceniani. Komputery również.

Impresariowie minikomputerów serii MERA 300 są dobrze znani krajowemu środowisku informatycznemu. Napotykali oni — nie bez powodów zresztą — prawie zgodną niechęć odbiorców do lansowanych przez siebie urządzeń. I oto, tuż przed dokonującym się nieomal w zapomnieniu końcem kariery tej serii sprzętu informatycznego, pragnę wyrazić pogląd, iż mógłby się to okazać koniec przedwczesny. MERY 300 mają bowiem jeszcze całkiem realną szansę spłacić gospodarce narodowej dług swego powstania i zejść z polskiej sceny informatycznej bardziej honorowo, niż na to się zanosi.

Gdański ETOB, na przełomie lat 1976/77, dostrzegł w MERZE 300 możliwość radykalnego unowocześnienia trybu obsługi informatycznej budownictwa na swym terenie. Wypada przyznać, że wobec ogólnego w tym względzie sceptycyzmu, dość nieufnie przyglądano się gdańskim poczynaniom. Powody do nieufności wprawdzie były, ale też producent stopniowo udoskonalał swoje wyroby. Wynik jest jednak taki, że na czterdzieści dziewięć MER 301 wyprodukowanych w 1978 roku budownictwo zakupiło 42 sztuki, a praktycznie cała produkcja roku 1979 — 50 egzemplarzy — jest również przez budownictwo wykupiona (z tego 10 sztuk, w ramach współpracy, odstąpiono resortowi górnictwa). Chcieliśmy w 1979 r. zakupić 100 sztuk, ale producent nie podjął się takiej dostawy. A potrzeby „od zaraz" są parokrotnie większe.

MERY 301 z podwójną pamięcią kasetową używa się głównie do kontrolowanej programowo rejestracji danych — bezpośrednio na nośniku magnetycznym i w miejscu powstawania tych danych oraz do sporządzania dokumentów źródłowych (zwłaszcza w księgowości), z równoczesnym zapisem danych na taśmie kasetowej.

Minikomputery są instalowane i obsługiwane jako podręczne narzędzie pracy w komórkach funkcjonalnych przedsiębiorstw budownictwa i przemysłu materiałów budowlanych. Kasety z danymi przewozi się — zwykle codziennie — do najbliższego ośrodka obliczeniowego ETOB, gdzie następuje konwersja ich zawartości na typową taśmę półcalową. Konwerterami są minikomputery MERA 305 lub MERA 306, w odpowiedniej konfiguracji, z pamięcią PT 105. Uzyskuje się w ten sposób stosunkowo wysoką — jak na potrzeby budownictwa — operatywność świadczenia usług informatycznych. Wyniki przetwarzania transakcji z dnia poprzedniego mogą być bowiem dostarczane użytkownikowi nazajutrz rano.

W porównaniu z tradycyjną techniką dziurkowania kart w ośrodku ETOB, pożytki są tu oczywiste: radykalne skrócenie cyklu przetwarzania, eliminacja kart i ich dziurkowania, główna weryfikacja danych w miejscu ich powstawania, niewydawanie na zewnątrz dokumentów źródłowych, odciążenie komputerów w ośrodkach od stosowania podstawowych procedur kontrolnych. Mimo kosztów zakupu MER 301, tańsze stają się usługi informatyczne.

MERY 301 są własnością użytkowników, konwertery — własnością ośrodków obliczeniowych (na jeden konwerter przypada ok. 20 MER 301). Usługi serwisowe świadczą ośrodki ETOB, w oparciu o stosowne porozumienie z producentem sprzętu.

Takim trybem usług są objęte regiony działania ośrodków ETOB w Gdańsku, Olsztynie i Koszalinie. Dotyczy to prawie w całości Zjednoczenia Budownictwa Przemysłowego „Północ" oraz zjednoczeń budownictwa ogólnego — gdańskiego, koszalińskiego i olsztyńskiego. Pierwsze zastosowania odnosiły się głównie do usług na rzecz służb finansowo-księgowych przedsiębiorstw i zjednoczeń. Obecnie rozszerza się je na inne systemy użytkowe. Wkrótce — najpierw w rejonie działania ETOBU Gdańsk — będą uruchamiane połączenia teletransmisyjne, umożliwiające kablowe przesyłanie zawartości kaset magnetycznych do ośrodka obliczeniowego oraz wyników przetwarzania — do użytkowników. Zwiększać się też będzie zakres autonomicznego wykorzystania minikomputerów.

Tyle relacji. Pora na refleksje końcowe.

Nie ulega wątpliwości, iż u progu lat osiemdziesiątych, minikomputery MERA 300 — z leciwym już „Momikiem" w środku — nie mogą być, zwłaszcza perspektywicznie, wizytówką nowoczesności rozwiązań technicznych. Lecz chyba jednak rzecz nie w tym, ale w osiąganiu powszechności nowoczesnych zastosowań użytkowych informatyki. Nie nowoczesność narzędzia jest najważniejsza, lecz nowoczesne stosowanie takich narzędzi, jakie są akurat potrzebne. Warto więc pomyśleć i o takich przykładach, jak „garbus" Volkswagena, czy karabin Kałasznikowa, choćby nawet stopień adekwatności tych przykładów był w

tym miejscu wysoce wątpliwy. A MERY 300 są nieźle opanowane produkcyjnie, nie wymagają wkładów dewizowych i — jak informacja powyższa wskazuje — są w stanie dobrze służyć gospodarce narodowej. Nietrudno też ulepszyć technologię ich produkcji.

Istnieje, owszem, bardziej pociągająca i nowocześniejsza alternatywa, zwłaszcza w postaci MERY 100. Póki jednak będzie ona osiągalna tylko za dewizy (nie za zwrotem uzasadnionego wkładu dewizowego), póty nie będzie to alternatywa na serio przez budownictwo traktowana. Nie tylko ze względów ekonomicznych — te bardziej pryncypialne na równi traktować wypada.

Pilna jest potrzeba zwielokrotnienia zastosowań zdecentralizowanej rejestracji i przetwarzania danych. Bez tego nie będzie operatywnych zastosowań informatyki na szerszą skalę. Bez tego budownictwo, które już osiągnęło pułap możliwości wzrostu usług opartych na nośnikach papierowych (ponad 140 mln kart rocznie), musiałoby hamować rozwój zastosowań informatyki. Dlatego chwalimy MERY 300, bo one — jak dotąd — istnieją. A dopóki istnieją, są jedynym realnie dostępnym narzędziem do tych celów.

Wnioski niech wyciąga producent. A może nie tylko on?

W informacji powyższej warto odnaleźć podtekst, który w istocie myśl przewodnią stanowi. Mowa tu bowiem o najważniejszym obecnie ograniczeniu, utrudniającym efektywne wykorzystanie już zainwestowanego potencjału kadrowego i sprzętowego informatyki w budownictwie. A wzrost efektywności wykorzystania blisko stu pięćdziesięciu fabryk domów w Polsce jest w dużym stopniu uzależniony od nowoczesnego użytkowo wsparcia właśnie operatywnego, czyli codziennego zarządzania pracą tych fabryk. I jest to tylko przykład, który dalece nie wyczerpuje zarówno zastosowań, jak i potrzeb uzupełnienia asortymentu sprzętu informatycznego, zwłaszcza w odniesieniu do stosunkowo tanich, a najbardziej efektotwórczych dopełnień tego sprzętu.

Wincenty ŁADA

Eugeniusz Kubica

Na Śląsku i w Zagłębiu — bardziej chyba niż gdzie indziej — obserwuje się personifikowanie instytucji. Najpierw jest szef, potem długo nic i reszta. Może to przykład z góry, może wypraktykowany model stylu menadżerskiego. W takim szefie trudno się na ogół odnajduje cechy osobowości, które choć trochę mogłyby wykraczać poza rytuał sprawowanej funkcji. Chyba, że zna się go całkiem prywatnie, ale takie z kolei obserwacje nie bardzo publicznie zdradzać wypada.

Dyrektor katowickiego ETOBU — mgr Eugeniusz Kubica — jest nie tylko szefem instytucji informatycznej. On sam wydaje się być taką instytucją, od lat około piętnastu. Niemal wszystko, co się dotąd działo w zastosowaniach informatyki dla śląsko-zagłębiowskiego budownictwa, działo się przy jego obecności, a przeważnie — pod jego kierownictwem. Z powodów podanych we wstępie trudno jednak pisać atrakcyjnie o osobie E. Kubicy. Łatwej — o kolejach losu budowlanych zastosowań informatyki na Śląsku. Co zresztą na jedno wychodzi.

Pochodzi E. Kubica z Bielska-Białej. Tam też, w 1951 r. po Liceum Ekonomicznym, już w pierwszej swej pracy, trafił do insytuacji i napotkał człowieka prowadzącego go prostą drogą do tego, co informatyką później nazwano. Instytucją była bielska delegatura krakowsko-katowickiego Biura Organizacji Rachunkowości, a człowiekiem — mgr Kazimierz Sowa (późniejszy profesor), dyrektor BOR, który już za dwa lata miał zostać współtwórcą i pierwszym dyrektorem warszawsko-krakowskiego Biura Rozliczeń Budownictwa, protoplasty obecnej sieci ETOB.

Na razie jednak praca naszego bohatera nie miała nic wspólnego z automatyzacją przetwarzania danych. W 1952 r. przenosi się do Bielskiego Przedsiębiorstwa Budownictwa Przemysłowego, gdzie wytrwale awansuje w służbach o organizacyjnym głównie charakterze: od referenta do kierownika działu kontroli wewnętrznej. Twierdzi, że poznał wówczas budownictwo „od podszewki".

W międzyczasie wypada służba wojskowa w szkole oficerskiej. Opuszcza wojsko pod warunkiem podjęcia studiów. Czyni to na Wyższej Szkole Ekonomicznej w Katowicach, kontynuując pracę zawodową w bielskiej „przemysłówce" budowlanej. W przedsiębiorstwie uczestniczy przy wdrażaniu przez krakowskie Biuro Rozliczeń Budownictwa pierwszych systemów przetwarzania danych na MLA, a na uczelni pisze pracę magisterską na temat: „Zastosowanie maszyn analityczno-statystycznych do ewidencji materiałów". Te dwa fakty oraz ogólna aktywność powodują, że mgr E. Kubica staje się dość znaną śląskiemu budownictwu postacią w nowej wówczas dziedzinie zautomatyzowanego przetwarzania danych.

Kiedy więc w 1963 roku, głównie z inicjatywy dyrektora M. Johanna, Śląskie Zjednoczenie Budownictwa Przemysłowego postanowiło utworzyć własny ośrodek rozliczeń na MLA, nic dziwnego, że właśnie E. Kubicy to zadanie powierzono. Od stycznia 1964 r. zostaje kierownikiem, a w rok później — dyrektorem Biura Rozliczeń tego zjednoczenia. Zaczyna od 4 zestawów MLA. Wkrótce będzie ich 12, a usługi zostaną rozszerzone także na budownictwo miejskie.

W połowie lat sześćdziesiątych coraz konkretniej jawi się konieczność przechodzenia na elektroniczną technikę obliczeniową. W Katowicach działają już dwa silne ośrodki przetwarzania danych: ZETO oraz Centralne Biuro Rozliczeń Przemysłu Węglowego (obecnie COIG), które coraz wyraźniej wyprzedzają budownictwo w nowej technice. Dyr. Kubica stawia na współpracę z tymi ośrodkami, tym bardziej, że chce osobiście szybko i praktycznie opanować nową wiedzę. Ukończył właśnie studium podyplomowe na SGPiS, co bardziej rozbudziło apetyt niż go zaspokoiło. Zabiera się w 1966 r. wraz z ekipą ZETO na półroczny kurs do Mińska, a z górnikami poznaje dokładnie kalkulator EW-83 M (2444 lampy), rodem z Wilna. Wspólnie też organizują wprowadzenie specjalizacji ETO w katowickich średnich szkołach zawodowych.

Na przełomie lat 1970/71 Biuro Rozliczeń kupuje dwa MIŃSKI 32, które już po roku pracują na trzy zmiany. Oprogramowanie użytkowe pozyskują katowiczanie z ETOBU w Warszawie. Podejmują też własne opracowania — w tym zwłaszcza (przy inspiracji Głównego Księgowego Zjednoczenia — E. Kucharczyka) systemu planowania i rozliczania budownictwa metodą rachunku kosztów normatywnych, do dziś stosowanego.

W 1969 r. następuje zmiana nazwy przedsiębiorstwa na Zakład Elektronicznej Techniki Obliczeniowej Budownictwa „ETOB" w Katowicach. Trzy lata później słowo „budownictwo" w nazwie zmienia się na „przemysł budowlany", a w 1976 roku zjednoczeniowa placówka informatyczna zostaje włączona do krajowej sieci ETOB, i w randze przedsiębiorstwa podporządkowana Centrum ETOB w Warszawie. Obecnie jest to Katowickie Przedsiębiorstwo Informatyki Przemysłu Budowlanego „ETOB".

Oczywiście, przez cały czas dyrektorem tej placówki jest mgr Eugeniusz Kubica. I m.in. dlatego trudno jest odróżnić życiorys bohatera portretu od „życiorysu" jego przedsiębiorstwa.

A oto dzieje najnowsze. Przejęcie w 1977 r. — pod zarząd ETOBU-Katowice — Zakładu Obliczeniowego we Wrocławiu (usamodzielniony w br.), utworzenie w 1977 r. stacji przygotowania danych w Bielsku-Białej, przekształconej w br. w filialny Ośrodek Obliczeniowy z ODRĄ 1305. W 1977 r. przejęto z Bydgoszczy trzeci MIŃSK 32, a w 1978 r. uruchomiono pierwszy w sieci ETOB komputer R 32, z obowiązkiem sprawowania roli wiodącej w opanowywaniu i wykorzystaniu użytkowym tych maszyn na rzecz całej sieci. Ważną specjalizacją katowickiego ETOBU jest świadczenie usług informatycznych dla resortu w utrzymaniu, aktualizacji i dystrybucji jednolitej bazy normatywnej budownictwa.

Dyrektor Kubica dba o to, aby aktywnie działać na swym terenie w niemal wszystkich organizacjach mających związek z informatyką, jej zastosowaniami, rozwojem kadr w tej dziedzinie, PZPR, Związki Zawodowe, NOT, TNOiK, Akademia Ekonomiczna — to główne, lecz nie wszystkie tereny jego aktywności. Był wielokrotnie i z różnych tytułów odznaczany. Ostatnio — Krzyżem Kawalerskim Orderu Odrodzenia Polski.

Próbką jego osobistego samookreślenia w zawodzie informatyka-dyrektora może być poniższy dialog.

— Wygląda na to, że dość gładko rozwijała Ci się kariera zawodowa.

Może i tak to wygląda, ale miałem po prostu szczęście do dobrych szefów i pedagogów.

— Masz opinię indywidualisty i szefa z dużym dystansem.

Taka opinia nawet mi schlebia, ale ważniejsze jest, że stwarzając wymagania dla otoczenia, stwarzam je również dla siebie.

— Ile godzin na dobę czujesz się dyrektorem?

Dwadzieścia cztery godziny. Po tylu latach pracy przedsiębiorstwo jest dla mnie także czymś na podobieństwo rodziny.

— W takim razie bardziej przypada mi do gustu portret twego przedsiębiorstwa, ale to tylko moje zdanie.

(wład)

Konferencje IFIP na temat kształcenia

Komitet techniczny do spraw kształcenia *(Technical Committee on Education)*, TC-3, jest jednym z komitetów technicznych Międzynarodowej Federacji Przetwarzania Informacji IFIP, których działanie obejmuje najważniejsze dziedziny współczesnej informatyki. Komitety techniczne IFIP są organami powoływanymi przez tę organizację dla popierania, pobudzania i organizowania międzynarodowej współpracy naukowej w dziedzinach stanowiących ich zakres działania. Do wykonania konkretnych prac badawczych i rozwojowych komitety techniczne powołują grupy robocze *(Working Groups)* złożone ze specjalistów odpowiedniej dziedziny.

Działalność komitetu do spraw kształcenia w ostatnich latach koncentrowała się przede wszystkim wokół metod stosowania środków informatyki w procesach kształcenia, wpływu rozwoju nowych metod informatycznych na kształtowanie się zapotrzebowania na specjalistów o określonych kwalifikacjach oraz informatycznego kształcenia zawodowego i nauczania informatyki w szkolnictwie średnim. Tym problemom były poświęcone konferencje i narady robocze, organizowane przez komitet TC-3: „Potrzeby kształcenia spowodowane przez wielkie systemy informacyjne" (Haga, kwiecień 1977), „Informatyka a matematyka w szkołach średnich, wpływy i powiązania" (Warna, wrzesień 1977), „Zastosowania komputerów w nauczaniu i procesach przekazywania wiedzy" (Budapeszt, marzec 1979), „Kształcenie pomaturalne i zawodowe informatyki: potrzeby przemysłu, handlu i administracji" (Amsterdam, kwiecień 1979), „Nauczanie wspomagane komputerem: zakres, rozwój, ograniczenia" (Londyn, wrzesień 1979). Materiały z tych konferencji są publikowane przez North Holland — niektóre tomy już się ukazały.

Od 15 do 18 kwietnia 1980 r. odbędzie się w Paryżu konferencja robocza „Mikrokomputery a nauczanie w szkołach średnich", której program przewiduje m.in. następujące tematy:

- Urządzenia specjalizowane czy ogólnego przeznaczenia? (dyskusja panelowa)
- Dostarczanie oprogramowania: problemy i sposoby ich rozwiązywania
- Metody nauczania i oceny wyników
- Dostęp lokalny czy zdalny, urządzenia indywidualne czy wspólne?
- Przygotowanie nauczycieli
- Koszty
- Zastosowania administracyjne
- Sytuacja i strategia rozwoju poszczególnych krajów lub regionów (dyskusja panelowa).

Do problemów wpływu rozwoju zastosowań mikrokomputerów w przetwarzaniu informacji na kształcenie oraz wykorzystanie minikomputerów w kształceniu będzie się na pewno wracać na następnych konferencjach, organizowanych przez TC-3.

Również w Paryżu i w kwietniu odbędzie się konferencja robocza „Rola programowania w nauczaniu informatyki".

Jej program obejmuje następujące tematy:

- Związki między ogólnymi pojęciami informatyki a programowaniem
  Systematyczne ujęcie sposobów formułowania problemów
  Środki i metody opisywania problemów
  Dobieranie struktur danych a metody rozwiązywania problemów
  Algorytmy i programy
  Rozwiązywanie problemów a programowanie
- Programowania dla nie-informatyków
  Określanie kursów programowania poprzez cele
  Wstępne wiadomości niezbędne dla uczestników kursów programowania
  Jakie wiadomości programowania są rzeczywiście konieczne?
  Programowanie w szkołach średnich
  Wpływ upowszechnienia kalkulatorów programowanych
- Programowanie w programach studiów
  Co powinien wiedzieć programista o strukturze komputera?
  Ile musi umieć z programowania projektant sprzętu?
  Względna wartość formalnego nauczania programowania w porównaniu z ćwiczeniami praktycznymi
  Przykłady-zabawki a problemy rzeczywiste
  Czy można uczyć przedmiotu „tworzenie oprogramowania"?

Jak zwykle na konferencjach roboczych tego typu, program wypełnią referaty zaproszonych autorów i dyskusja panelowa. Akceptacja nadesłanych materiałów i ustalenie ostatecznego programu przewidziane są dla obu tych konferencji na początek 1980 r.

27—31 lipca 1981 roku w Lozannie odbędzie się największa z konferencji organizowanych przez TC-3: III Światowa Konferencja „Informatyka w nauczaniu", WCCE 81. W odróżnieniu od konferencji roboczych, grupujących zwykle nie więcej niż 40—50 uczestników, konferencje te są wielkimi zjazdami naukowymi, a liczbą uczestników i liczbą wygłaszanych referatów ustępują tylko odbywającym się co 3 lata światowym kongresom IFIP.

Pierwsza światowa konferencja „Informatyka w nauczaniu" odbyła się w Amsterdamie w 1970 roku, druga w Marsylii w 1975. Kolejną trzeba było zaplanować dopiero na 1982, gdyż w lecie 1980 wypada kongres IFIP w Tokio i Sydney. Ponieważ ostatnie kongresy IFIP odbywają się poza Europą, konferencja w Lozannie ma wszelkie szanse stać się największym zjazdem IFIP, jaki odbędzie się w najbliższych latach na terenie Europy.

Pierwsza z tych konferencji, amsterdamska, dała możliwość konfrontacji poglądów na istniejące wtedy problemy nauczania informatyki i zastosowań informatyki w nauczaniu oraz wyciągnięcia wniosków o celach i metodach dalszego działania w tej dziedzinie. Konferencja w Marsylii była przede wszystkim okazją do wymiany doświadczeń w skali światowej. Organizatorzy trzeciej konferencji w Lozannie stawiają sobie za cel stworzenie możliwości dokonania oceny aktualnego stanu, osiągniętego po kilkunastu latach rozwoju informatyki w nauczaniu, a także sformułowania zaleceń na przyszłość.

Przewiduje się następujący podział tematyczny referatów:

- Informatyka a inne nauki (matematyka, nauki przyrodnicze, nauki humanistyczne i społeczne, nauki o zarządzaniu i administracji, nauki techniczne)
- Metody nauczania (nauczanie wspomagane komputerem, komputerowe kierowanie nauczaniem, kształcenie masowe)
- Wpływ nowych technik (mikrokomputery, wideodyski, komputery domowe, sieci, dalsze tendencje rozwojowe)
- Skutki społeczne (zmiana roli nauczycieli w kształceniu)
- Przegląd strategii rozwojowej różnych krajów i rozwiązań dotyczących komputeryzacji nauczania, ze szczególnym uwzględnieniem potrzeb krajów rozwijających się
- Cele, strategie i programy nauczania informatycznego.

Zgłaszane referaty mogą dotyczyć wszystkich poziomów i form nauczania, mogą być poświęcone ocenie aktualnego stanu i kierunków rozwoju albo przedstawiać nowe metody, koncepcje lub środki techniczne. W każdym jednak przypadku prezentowane rozwiązania muszą być poparte pozytywnymi rezultatami dostatecznie długiego i skutecznego stosowania ich w kształceniu. Wszystkie nadesłane i zaakceptowane przez Komitet Programowy referaty zostaną opublikowane w materiałach konferencyjnych. Wstępne zgłoszenia referatów — do połowy lipca 1980 r., termin nadsyłania prac — do 30 października 1980 r.

Organizatorzy konferencji pragną zapewnić jej uczestnikom wiele okazji do przedstawienia i skonfrontowania swoich poglądów, toteż tematy, które nie będą poruszone na sesjach, zostaną uwzględnione w referatach przeglądowych stanowiących wprowadzenie do dyskusji. Program konferencji obejmie więc zarówno referaty zaproszonych specjalistów, referaty przeglądowe z następującymi po nich dyskusjami panelowymi oraz nadesłane i przyjęte do wygłoszenia jak też dyskusje o wybranych problemach lub o tendencjach rozwojowych. Z okazji konferencji odbędzie się również tradycyjna wystawa, poświęcona zastosowaniom informatyki w kształceniu.

Nadzór nad organizacją konferencji sprawuje Association Suisse pour l'Automatique (ASSPA). Przewodniczącym Komitetu Programowego jest prof. B. Levrat z Uniwersytetu w Genewie, przewodniczącym Komitetu Organizacyjnego — prof. Immer.

W 1981 roku odbędzie się również w Holandii kolejna konferencja robocza TC-3 „Kształcenie a wielkie systemy informacyjne", następna po podobnej tematycznie konferencji w 1977 r. Dokładne miejsce, termin i program konferencji nie są jeszcze znane.

Ponadto planuje się następujące konferencje robocze:

- **„Kształcenie i dokształcanie informatyczne na potrzeby rozwoju narodowego" (1982, przypuszczalnie Malawi)**
- **„Kształcenie informatyczne na poziomie szkół wyższych dla nie-informatyków" (1982, prawdopodobnie w Bułgarii)**
- **„Kształcenie uwzględniające potrzeby współdziałania projektantów systemów i użytkowników" (1983, przypuszczalnie Austria).**

Czwarta konferencja światowa „Informatyka w nauczaniu" jest planowana na rok 1985.

Konferencje te są co prawda dopiero w fazie projektów i konsultacji z zainteresowanymi organizacjami, ich tematyka może dać jednak pewne pojęcie o kierunkach zainteresowań i prac Komitetu TC-3. Jak widać, działalność Komitetu jest dość intensywna — przeciętnie dwie konferencje rocznie w różnych krajach, przy współpracy z innymi komitetami technicznymi IFIP, z UNESCO oraz Międzynarodowym Biurem Informatyki IBI. I chociaż działalność ta nie obejmuje w jednakowym stopniu wszystkich dziedzin kształcenia informatycznego i zastosowań informatyki w nauczaniu (do słabiej reprezentowanych należą na przykład problemy kształcenia specjalistów informatyki), to jednak jej wyniki mogą być z pożytkiem wykorzystywane — w miarę możliwości — także przez polskie środowisko informatyczne.

Stanisław WALIGÓRSKI

800-megabajtowy dysk

System masowej pamięci dyskowej nowej generacji powstał wysiłkiem kilku firm japońskich. System ten opiera się na pakiecie dysków o rekordowej pojemności, przekraczającej 800 megabajtów.

Pojemność pakietu, opracowanego w laboratorium MUSASHINO, wynosi 804,39 megabajtów. Porównywalne osiągnięcia firm amerykańskich to system STC 8650 o pojemności jednostkowej 635 megabajtów firmy STORAGE TECHNOLOGY CORP. oraz urządzenie 33 502 o tej samej pojemności firmy CDC, które otrzymano w wyniku podwojenia gęstości ścieżek na dysku w stosunku do wcześniejszych modeli. Jeśli chodzi o IBM, to największy aktualnie system dyskowy tej firmy (typ 3 350) ma pakiety o pojemności 317,5 megabajtów.

Gęstość zapisu w omawianym pakiecie japońskim wynosi 8 500 bitów na cal oraz 650 ścieżek na cal. Kluczem rekordowych osiągnięć jest podwójnie nakładana na dysk warstwa nośnika z ferrytu gamma-żelazowego. Dla uzyskania wysokiej gęstości zapisu ferrytowe pokrycie musi być jak najcieńsze i bardzo gładkie. W tak cienkiej warstwie łatwo o defekty w postaci pęknięć, dlatego zastosowano podwójnie nakładaną warstwę nośnika o łącznej grubości 0,7—0,8 mikrona. Druga warstwa przykrywa defekty pierwszej, a prawdopodobieństwo nałożenia się defektów z dwu warstw jest niewielkie.

Z uzyskaniem wysokiej gęstości związany jest także nowy schemat kodowania, który poprawia pewność odczytu danych. Bit znacznikowy (ang. *flag bit)* jest umieszczony między czwartym i piątym bitem każdego 8-bitowego bajtu, a jego wartość jest określana analogicznie, jak w konwencjonalnych metodach wyznaczania bitu parzystości.

W metodzie zapisu bez powrotu do zera (NRZ) każda wartość „1" reprezentuje skok napięciowy, czyli przejście z jednego poziomu napięciowego do drugiego, a wartość „0" reprezentuje brak zmiany stanu napięciowego. W nowych pamięciach dyskowych zastosowano schemat kodowania, w którym liczba zapisywanych „1" jest zmaksymalizowana, co powoduje wzrost liczby skoków napięciowych, dzięki czemu poprawia się regeneracja zegarowa i pewność odczytu.

Do uzyskania podwyższonej gęstości zapisu przyczyniło się także zastosowanie nowego rdzenia magnetycznego, z którego wykonana jest głowica zapisująco-odczytująca. Dzięki temu uzyskano zmniejszenie przesłuchów i indukcyjności, które zwykle ograniczają gęstość zapisu.

W opisywanym pakiecie dysków dane rejestrowane są na 20 powierzchniach (w najnowszym systemie CDC jest też 20 takich powierzchni, a w systemie IBM — 15). Przy maksymalnej konfiguracji 32 pakietów w systemie, daje to ogólną pojemność systemu 25,6 gigabajtów.

Szybka drukarko—kopiarka

W praktyce biurowej kopiarka była dotąd zawsze urządzeniem odrębnym od drukarki komputerowej. Specjaliści firmy WANG zaprojektowali urządzenie kombinowane, o nazwie IMAGE-PRINTER, spełniające funkcje obu wspomnianych urządzeń.

Drukarka ma szybkość 18 stron na minutę (ok. 4 500 znaków). Podobne drukarki nieuderzeniowe (elektrograficzne) są znacznie szybsze, ale kosztują ok. 200 000 dolarów, podczas gdy opisywane urządzenie kombinowane kosztuje tylko 35 000 dolarów.

Mikroprocesorowa jednostka sterująca urządzenia jest sprzężona z kserograficzną kopiarką o procesie suchym. Ponadto w urządzeniu znajduje się monitor ekranowy, wyświetlający obraz wiersza, który ma być aktualnie drukowany. Obraz ten jest przesyłany za pomocą zespołu światłowodów do nośnika pośredniego, w którym powstaje utajony obraz ładunków elektrycznych. Nośnik ten znajduje się w tzw. wzorniku fotoelektrycznym (ang. *photomaster),* który jest sercem kopiarki kserograficznej.

W urządzeniu znajdują się dwa wzorniki odbierające na przemian obraz strony, każdy z nich przekazuje powstały w nim obraz elektrostatyczny na przylegający doń arkusz papieru. Na papierze powstaje obraz z proszku, który zostaje utrwalony przez podgrzanie.

Zespół światłowodów zapewnia rozdzielczość ok. 90 000 elementarnych punktów na cal kwadratowy, co daje bardzo dobrą jakość druku, taką, jaką otrzymuje się z maszyny do pisania, z drukarki uderzeniowej z tarczą drukującą, czy też z drukarki natryskowej, a więc z urządzeń najczęściej stosowanych do drukowania dokumentów w systemach przetwarzania tekstów. Oprócz znacznej wydajności i dobrej jakości drukowania urządzenie umożliwia wykonywanie kopii dokumentów.

Mikroprocesor Z80 firmy ZILOG steruje współpracą drukarki z systemem przetwarzania tekstów oraz nadzoruje drukowanie w jednym z czterech repertuarów czcionek i w jednej z sześciu odmian kroju czcionki. Drukarkę wyposażono ponadto w dwa podajniki papieru, aby dostosować ją do różnych rozmiarów papieru.

Drukarka może być połączona za pomocą konwencjonalnego kabla współosiowego z Biurowym Systemem Informacyjnym *(Office Information System — OIS)* serii 100 lub komputerami serii 2200 firmy WANG, przy czym przesyłanie danych do drukarki odbywa się z prędkością do 400 000 bodów.

Opracował Z. NAOTYŃSKI na podstawie czasopisma „ELECTRONICS", grudzień 1978

Szukamy nowych technologii na eksport

„Szukamy nowych technologii na eksport"

— pod tym hasłem przebiega konkurs organizowany przez PHZ „Polimex-Cekop" oraz Wydawnictwo Czasopism i Książek Technicznych SIGMA, pod patronatem prezesa Naczelnej Organizacji Technicznej, dr inż. Aleksandra Kopcia.

Na konkurs „Szukamy nowych technologii na eksport — 1978" wpłynęło łącznie 387 prac. Nagrodzono dziesięć prac i dziesięć prac wyróżniono.

Nagroda I (50 000 zł)

Technologia produkcji metanolu metodą średniociśnieniową

Autorzy: mgr inż. Karol Cempiel, inż. Ignacy Lachman, mgr inż. Zbigniew Kunstler, mgr inż. Andrzej Heber, mgr inż. Jan Szuba, mgr inż. Antoni Masiewicz — Biuro Projektów i Realizacji Inwestycji Przemysłu Syntezy Chemicznej „Prosynchem" w Gliwicach, mgr inż. Werner Kuszka — Zakłady Azotowe w Chorzowie, prof. dr hab. inż. Włodzimierz Kotowski — Instytut Ciężkiej Syntezy Organicznej „Blachownia", mgr inż. Bolesław Grochowski — Zakłady Chemiczne „Oświęcim"

Nagroda II (40 000 zł)

Technologia wytwarzania dwumetylotereftalenu

Autorzy: inż. Ignacy Lachman, mgr inż. Stanisław Młynarczyk, mgr inż. Jerzy Orliński, inż. Jerzy Wiśniewski — Biuro Projektów i Realizacji Inwestycji Przemysłu Syntezy Chemicznej „Prosynchem" w Gliwicach, mgr inż. Renata Fiszer, prof. dr hab. inż. Włodzimierz Kotowski, mgr inż. Zofia Pokorska — Instytut Ciężkiej Syntezy Organicznej „Blachownia"

Dwie nagrody III (po 30 000 zł)

Ciągła odcukrzalnia melasu

Autorzy: dr inż. Witold Łękawski, dr inż. Andrzej Kubasiewicz, mgr inż. Marek Wolff, mgr inż. Krzysztof Dals — Przedsiębiorstwo Projektowania i Dostaw Kompletnych Obiektów Przemysłowych „Chemadex" w Warszawie

Instalacja mocznika

Autorzy: mgr inż. Jerzy Baranowski, technik Jerzy Iwańczuk, inż. Andrzej Kłapouszczak, mgr inż. Leszek Korczyński, mgr inż. Jan Marciniak, mgr inż. Rafał Świnarski, mgr inż. Grzegorz Szyszkowski, mgr inż. Leszek Wawrzonek — Instytut Nawozów Sztucznych w Puławach

Nagroda IV (25 000 zł)

Piec cyklonowy do intensywnej przeróbki lub spalania zdyspergowanych surowców mineralnych

Autorzy: prof. dr hab. inż. Wiesław Kurdowski, mgr inż. Edmund Nowak, mgr inż. Andrzej Parda — Instytut Przemysłu Wiążących Materiałów Budowlanych Kraków — Nowa Huta, mgr inż. Stefan Jankowski, doc. dr inż. Rudolf Żamojdo — Instytut Techniki Cieplnej i Mechaniki Płynów Politechniki Wrocławskiej

Dwie nagrody V (po 20 000 zł)

Technologia produkcji płytek posadzkowych klejonych z drewna skrawanego bezwiórowo

Autorzy: dr inż. Włodzimierz Poskrobko, inż. Aleksy Zin, technik Mieczysław Komoń, inż. Jerzy Sahajdak — Hajnowskie Przedsiębiorstwo Przemysłu Drzewnego

Oczyszczalnia ścieków z przemysłu azotowego przy zastosowaniu żywic jonowymiennych

Autorzy: mgr inż. Jerzy Jaros, mgr inż. Andrzej Cichocki, mgr inż. Longin Zagulski, inż. Henryk Pawlak — Biuro Projektów Gospodarki Wodnej i Ściekowej „Biprowod", prof. dr Janusz Barcicki, dr Lucjan Pawłowski — Zakład Technologii Chemicznej UMCS w Lublinie, mgr inż. Jerzy Studencki, mgr inż. Zbigniew Schimmelpfennig — Zakłady Azotowe „Puławy"

Nagroda VI (15 000 zł)

Technologia wytwarzania furfuralu z bagassy

Autorzy: inż. Ignacy Lachman, inż. Jan Łobacz, dr inż. Andrzej Plaskura, mgr inż. Krzysztof Bronikowski — Biuro Projektów i Realizacji Inwestycji Przemysłu Syntezy Chemicznej „Prosynchem" w Gliwicach, mgr inż. Zdzisław Maciejewski, mgr inż. Elżbieta Rutkowska, mgr inż. Jerzy Miszka — Instytut Ciężkiej Syntezy Organicznej „Blachownia", mgr inż. Zygfryd Makowski, inż. Zygmunt Kaliszewski, mgr Aleksander Stawski, mgr inż. Eugeniusz Kolonko — Zakłady Sklejek i Chemicznego Przerobu Drewna w Bydgoszczy

Dwie nagrody VII (po 10 000 zł)

Bezodpadowa technologia i urządzenie do oczyszczania gazów z hut szkła ołowianego

Autorzy: dr inż. Piotr Kabsch, mgr inż. Henryk Meloch, technik Eugeniusz Bielec, technik Janusz Robaszkiewicz — Instytut Inżynierii Ochrony Środowiska Politechniki Wrocławskiej

Anteny paraboliczne (radioteleskopy) dla łączności satelitarnej oraz dla badań radioastronomicznych

Autorzy: mgr inż. Zygmunt Bujakowski — Biuro Studiów i Projektów Urządzeń Hutniczych „Hutomaszprojekt" w Gliwicach, mgr inż. Andrzej Bujakowski — Centrum Naukowo-Produkcyjne Systemów Sterowania „MERA-CNPSS" w Katowicach

Wyróżnienia

Technologia wytwarzania dianu metodą jonitową

Autorzy: dr inż. Maciej Kiedik, prof. dr inż. Edward Grzywa, dr inż. Kazimierz Terelak, mgr inż. Sylwester Chybowski — Instytut Ciężkiej Syntezy Organicznej „Blachownia", mgr inż. Józef Kolt, inż. Jerzy Czyż, mgr inż. Anna Niezgoda, mgr inż. Eugeniusz Zając, inż. Roman Jarawka — Zakłady Chemiczne „Blachownia"

Stypizowany proces technologiczny produkcji ozdób choinkowych oraz zunifikowanie urządzenia do automatyzacji i mechanizacji tego procesu

Autorzy: inż. Jan Stępkowski, inż. Ryszard Balwierz, technik Kazimierz Kiełpiński, technik Zygmunt Bujnowicz, technik Zbigniew Karolak — Koszalińskie Biuro Konstrukcyjno-Technologiczne Krajowego Związku Szklarsko-Mineralnych Spółdzielni Pracy we Wrocławiu

Naftogum — instalacja utylizacji zużytych opon samochodowych

Autorzy: mgr Helena Masiarczykowa — Instytut Technologii Nafty, Oddział Trzebinia, mgr inż. Tomasz Szczurek — „Naftochem" Kraków, mgr inż. Adam Hamarski — Rafineria Nafty w Trzebini, mgr inż. Hanna Noworytko, inż. Władysław Swierz — „Bipronaft" Kraków

Linia malowania zanurzeniowego farbami wodorozcieńczalnymi

Autorzy: inż. Roman Partyka, inż. Maria Rajtar, inż. Henryk Wasiluk, mgr inż. Czesław Janikowski, inż. Julian Zygmunt, inż. Marek Rychter — Fabryka Samochodów Ciężarowych w Lublinie

SOCOR — system produkcji odlewów ze stopów aluminiowych

Autorzy: doc. dr inż. Paweł Hurza-Mucha, mgr inż. Zbigniew Koszarewski, technik Wojciech Ostaszewski — Zakład Odlewnictwa Instytutu Technologii Bezwiórowych Politechniki Warszawskiej, mgr inż. Marian Zając — Zakłady Wytwórcze Aparatury Wysokiego Napięcia ZWAR w Ostrołęce

Szukamy nowych technologii na eksport

Rozwiązanie technologiczne i konstrukcyjne „Flotoflok" do uzdatniania wód, oczyszczania ścieków i zagęszczania osadów za pomocą równoczesnej koagulacji i flotacji

Autorzy: mgr inż. Ryszard Przybyłowicz — Biuro Studiów, Projektów i Realizacji Przemysłu Mleczarskiego „Promlecz" w Warszawie, dr inż. Wojciech Królikowski, mgr inż. Ryszard Kęliszek — Centrum Techniki Komunalnej w Warszawie

Technologia odzysku wody i związków chromu ze ścieków galwanizerskich

Autorzy: dr Lucjan Pawłowski, prof. dr hab. Janusz Barcicki, mgr inż. Ryszard Próchniak — Instytut Chemii UMCS w Lublinie, inż. Zbigniew Dąbrowski, mgr inż. Józef Koziarski — Huta „Stalowa Wola", inż. Feliks Wyczółkowski — Zespół Usług Technicznych OW NOT w Lublinie, mgr inż. Longin Zagulski — Biuro Projektów Gospodarki Wodno-Ściekowej w Warszawie

Wieżowe zbiorniki na wodę

Autorzy: mgr inż. Zbigniew Antoszewski, inż. Bogusław Hejdukiewicz, mgr inż. Stanisław Szprynger, mgr inż. Czesław Zieliński — Biuro Projektów Przemysłu Hutniczego „Biprohut" w Gliwicach

Projekt i technologia budowy wolnostojących kominów stalowych o dużych wysokościach

Autorzy: mgr inż. Zbigniew Antoszewski, mgr inż. Adam Biskup, mgr inż. Jan Kałyniak, inż. Ignacy Żak — Biuro Projektów Przemysłu Hutniczego „Biprohut" w Gliwicach

Koparki hydrauliczne z osprzętem wysuwnym

Autorzy: prof. dr inż. Ignacy Brach — Politechnika Warszawska, mgr inż. Ryszard Walczewski — Kombinat Maszyn Budowlanych im. L. Waryńskiego.

Uroczyste ogłoszenie wyników konkursu „Szukamy nowych technologii na eksport — 1978" odbyło się na tegorocznych Międzynarodowych Targach Technicznych w Poznaniu. Nagrody wręczył laureatom prezes NOT, minister Przemysłu Maszynowego, dr inż. Aleksander Kopeć.

Konkurs przyniósł niezmiernie bogaty plon oryginalnej myśli technicznej. Pod zgłoszonymi pracami złożyło swoje podpisy około 2 000 inżynierów zatrudnionych w przemyśle, w biurach projektowo-konstrukcyjnych, ośrodkach naukowo-badawczych i na wyższych uczelniach. Stanowi to niejako deklarację osobistego zaangażowania przedstawicieli środowiska technicznego w niełatwe sprawy eksportu polskiej myśli technicznej, oferowanej w jej najbardziej złożonej, ale też i najbardziej opłacalnej formie, tj. w postaci obiektów przemysłowych.

Prace zgłoszone i spełniające warunki konkursu, a nie nagrodzone będą przedmiotem dalszej analizy i konsultacji z autorami, aby nadawały się na eksport. Prace te będą mogły ponownie ubiegać się o nagrodę w konkursie.

Przypominamy, że konkurs trwa. Zachęcamy naszych Czytelników do udziału w konkursie „Szukamy nowych technologii na eksport — 1979". Warunki konkursu (zamieszczamy obok) nie uległy zmianie.

WARUNKI KONKURSU

Celem konkursu jest wyszukanie nowych rozwiązań technologicznych lub konstrukcyjnych, oryginalnych polskich lub opartych na zakupionej i przetworzonej licencji, które mogłyby być przedmiotem eksportu w postaci kompletnego ciągu produkcyjnego, obiektu przemysłowego.

Uczestnikami mogą być pracownicy i współpracownicy prasy technicznej, a więc dziennikarze, pracownicy przemysłu, biur projektowych i placówek naukowo-badawczych oraz pracownicy handlu zagranicznego. Chodzi głównie o zgłoszenia z następujących przemysłów: chemicznego, drzewnego, elektromaszynowego, papierniczego, spożywczego, materiałów budowlanych i jednostek organizacyjnych zajmujących się ochroną środowiska.

Zgłoszone rozwiązanie technologiczne lub konstrukcyjne:
— musi wykazywać cechy oryginalności i nowości, przynajmniej w skali krajowej
— może dotyczyć technologii konstrukcji wdrożonych i nie wdrożonych
— musi wykazywać cechy przydatności eksportowej
— nie może być przedmiotem aktualnego eksportu.

Zgłoszenie rozwiązania powinno zawierać:
— imię, nazwisko, adres, telefon zgłaszającego lub zgłaszających (w przypadku zgłoszenia zespołowego należy podać procentowy udział uczestników)
— jasny i precyzyjny opis rozwiązania
— określenie zasadniczych walorów technicznych i ekonomicznych
— wskazanie aktualnego lub potencjalnego producenta (dostawcy) lub kompletatora (generalnego dostawcy)
— informację o wdrożeniu lub możliwości wdrożenia rozwiązań w przemyśle
— stan ochrony prawnej rozwiązania w kraju i za granicą
— podpis zgłaszającego (lub zgłaszających).

Termin nadsyłania prac upływa z dniem 31 stycznia 1980 r. Zgłoszone na konkurs rozwiązanie będzie oceniane według kryteriów uwzględniających jego wartość techniczną, oryginalność i cechy nowości, konkurencyjność wobec dotychczas stosowanych rozwiązań, stopień kompletności opracowania i (lub) wdrażania, atrakcyjność w aspekcie przydatności eksportowej, inicjatywę zgłaszającego rozwiązanie oraz sposób przedstawienia problemu. Ponadto sąd konkursowy może uwzględnić inne kryteria oceny, charakterystyczne dla zgłoszonych opracowań.

Nagrody: zespołowe — I stopnia (do 50 000 zł), II stopnia (do 30 000 zł), III stopnia (do 20 000 zł) — oraz indywidualne.

W przypadku nagrody zespołowej rozdziału przyznanych kwot dokonuje sąd konkursowy; może on dokonać innego podziału nagród zgodnie z obowiązującymi przepisami. Decyzje sądu konkursowego są ostateczne i nie podlegają zaskarżeniu.

Ogłoszenie wyników nastąpi na Międzynarodowych Targach Technicznych w Poznaniu w 1980 r.

Analityczna weryfikacja programów

Coraz bardziej rozpowszechnione jest już u nas przekonanie, że programowania należy nauczać jako specyficznej umiejętności niezależnej nie tylko od dziedziny zastosowań, ale i od języka programowania. Ponieważ jednak do niedawna za naukę programowania uważało się naukę konkretnego języka programowania, przy sporej liczbie podręczników o poszczególnych językach brak jest pozycji metodycznych. Takim właśnie podręcznikiem programowania jest **„Wstęp do programowania systematycznego" N. Wirtha** *).

Zdaniem autora *nauka programowania w sposób dokładny, metodyczny powinna opierać się na problemach i technikach typowych dla programowania, ale niezależnych od dziedziny zastosowań, z której się konkretne zadanie wywodzi (...) Podobnie należy traktować język programowania: powinien on być narzędziem, nie zaś celem samym w sobie.* Programowanie trzeba więc traktować *jako samodzielną dyscyplinę zajmującą się systematycznym konstruowaniem i formułowaniem algorytmów.*

Jest to jednak samodzielność względna, gdyż Wirth w istocie rzeczy traktuje programowanie jako dział matematyki. Wyraźnie to mówi, określając krąg czytelników. Adresuje on swoją książkę do tych, *którzy metodyczne konstruowanie algorytmów programów komputerowych traktują jako część swojego wykształcenia matematycznego, a nie tylko czasami chcieliby zakodować problem i powierzyć go komputerowi do rozwiązania.*

Innymi słowy, książka jest przeznaczona dla profesjonalnych programistów, czyli twórców oprogramowań podstawowych, firmowych, bibliotecznych itp. Ale i liczniejsza grupa programistów-użytkowników znajdzie w niej bardzo solidnie sformułowane podstawy programowania.

Ten matematyczny rodowód programowania ma określone konsekwencje w sposobie traktowania omawianych problemów. Odnosi się wręcz wrażenie, że dla Wirtha najważniejsze są piękno i elegancja samego algorytmu. A czytelnikowi brakuje uzasadnienia, czyżby warto tyle wysiłku poświęcać dla osiągnięcia tejże elegancji. Jest to jednak sprawa drugorzędna, aczkolwiek dla czytelnika — niematematyka nieco drażniąca.

Z owego matematycznego podejścia do programowania wypływa też inny motyw książki, o kapitalnym dla praktyki programistycznej znaczeniu. Chodzi mianowicie o sprawdzanie poprawności programu. Tok rozumowania autora można streścić następująco: wychodząc z przyjętej we wstępnych rozdziałach definicji procesu zakłada się istnienie pewnej klasy procesów. Ponadto zakłada się istnienie programu realizującego procesy danej klasy. Aby stwierdzić, czy program jest poprawny, należy zbadać, czy wszystkie procesy dające się wykonać wg danego programu dają wyniki poprawne.

W sformułowaniu tym warto zwrócić uwagę na słowo „wszystkie"; podobny sens zawarty jest w innej uwadze Wirtha, *że sumienny konstruktor programu musi umieć dowieść, że jego wytwór wykazuje żądane właściwości w każdych warunkach.*

Trzeba zdawać sobie sprawę, że żądania te są bardzo ostrym kryterium i w praktyce programistycznej rzadko są formułowane w tak jawnej postaci. Częściej sprawdza się jedynie, czy program działa poprawnie w warunkach średnich, lub inaczej mówiąc — bardziej typowych. Wirth zdaje sobie z tego sprawę, gdyż jako pierwszą metodę sprawdzania poprawności programu omawia testowanie, aczkolwiek zaraz potem zdecydowanie się od niej odcina. Można by nawet powiedzieć, że zbyt zdecydowanie.

Testowanie jest to, jak czytamy, wybranie dozwolonych wartości początkowych, wykonanie dla nich programu oraz porównanie otrzymanych wyników ze znanymi uprzednio wynikami poprawnymi. Czynność taka jest powtarzana dla pewnej liczby różnych wartości. Metoda jest jednak kosztowna, zwłaszcza gdy przeprowadza się odpowiednio dużo testów. Ale co gorsze, jest to metoda niezadowalająca, gdyż nie sposób wykonać takich operacji dla wszystkich możliwych wartości początkowych. Tak więc należy uzmysłowić sobie, że *empiryczne testowanie co najwyżej wykaże obecność błędów, nigdy natomiast nie zagwarantuje poprawności programu* oraz że *empiryczne metody sprawdzania programów używane w praktyce dają podstawę do wypowiadania się tylko o pojedynczych obliczeniach.* Natomiast *aby móc wypowiadać się o programie, niezbędna jest weryfikacja analityczna.* Jest ona, zdaniem autora, możliwa, gdyż *programowanie (...) jest podatne na stosowanie systematycznych matematycznych metod analizy.* Należy jedynie konsekwentnie *abstrahować od pojedynczych procesów, a przyjmować pewne zawsze spełnione warunki, które można wywieść z wzorców zachowania się procesów.* W tym miejscu czytelnik może mieć wątpliwości, co kryje się za takim, dość ogólnikowym sformułowaniem. I trzeba stwierdzić, że lektura książki nie rozwiewa do końca tych wątpliwości, aczkolwiek spokojne przemyślenie licznych przykładów znacznie problem wyjaśnia. W *przeciwieństwie do testowania* (stwierdza Wirth), *gdzie bada się własności poszczególnych procesów, weryfikacja bada własności samego programu.*

Jako podstawowe w tym zakresie wprowadza Wirth pojęcia: asercji i niezmiennika; stwierdzając ponadto, że *bez nich nie można mówić o nauce programowania,* oraz że *dopiero one dają właściwe podstawy do głębszego rozumienia algorytmów.*

Istotę podanej przez Wirtha metody analitycznej weryfikacji można ująć następująco. Dla każdej instrukcji podaje się jeden lub kilka warunków zwanych asercjami, które są zawsze spełnione. Warunki występujące przed instrukcją nazywa się przesłankami, po instrukcji — wnioskami. Tak więc dla każdej instrukcji musi się określić zarówno przesłanki, jak i wnioski. Jeżeli zdarzy się, że pewna asercja jest przesłanką i wnioskiem, jest to niezmiennik danej instrukcji. Pojęcie niezmiennika ma szczególne znaczenie przy pętlach, przy których należy dokonywać przecięcia pętli w najbardziej celowym miejscu. Jest to jednak bardzo trudne zadanie i Wirth przyznaje, że dlatego *podanie niezmiennika każdej pętli stanowi najcenniejszy (!) element dokumentacji programu.* Zwraca również uwagę, że pętle są najbardziej charakterystycznym elementem programów, gdyż opisują *powtarzanie się akcji, do czego automaty są szczególnie dobre, gdyż nie męczą się.* Wymaga to jednak szczególnej czujności programisty, który musi zadbać, aby wszystkie procesy wykonywane według danego programu kończyły się po skończonej liczbie powtórzeń.

Jako jeden z zabiegów skutecznie przeciwdziałający możliwości zapętlania się programu zaleca Wirth zadbanie, by instrukcja, zwykle złożona, wykonywana w pętli, zmieniała wartość co najmniej jednej zmiennej wchodzącej w skład warunku, którego spełnienie decyduje o realizacji tej instrukcji, tak, aby po skończonej liczbie powtórzeń warunek nie musiał być spełniony.

Do analitycznej weryfikacji programu zalicza też Wirth *precyzyjne wskazanie zakresów wartości zmiennych,* a zwłaszcza dopuszczalnych wartości zmiennych początkowych.

Tak szerokie omówienie problemu analitycznej weryfikacji programu zostało podyktowane przeświadczeniem, że to właśnie zagadnienie jest w książce Wirtha najciekawszym i najważniejszym dla czytelnika motywem. Jest to motyw, gdyż po wprowadzeniu podstawowych pojęć w pierwszych sześciu, bardzo krótkich rozdziałach Wirth konsekwentnie przez cały dalszy ciąg wykładów — bo i tak

*) Niklaus Wirth: Wstęp do programowania systematycznego. WNT, „Biblioteka Inżynierii Oprogramowania", Warszawa 1978 (oryginał 1975). Z niemieckiego tłumaczyli M. Grzymkowski i D. Prus-Grzymkowska, 15 rozdziałów, 2 dodatki, str. 117, rys. 34, tabl. 12, poz. lit. 6, nakład 6000 + 250 egz., cena 42 zł

można nazwać kolejne partie książki — stosuje tę analityczną weryfikację i na coraz to bardziej rozbudowanych przykładach wykazuje jej użyteczność oraz istotę działania.

Warto również zdać sobie sprawę, że dla programisty prawdziwie twórczego, o silnie rozwiniętej odpowiedzialności nie jest to nic nowego. Formalizuje jednak tok rozumowania i postępowania, porządkuje go i nadaje mu rangę obowiązującego fragmentu prac nad programem.

Na zakończenie należy powiedzieć, że zasadniczą treścią książki Wirtha jest zgodnie z tytułem systematyczny i solidny wykład podstawowych zagadnień związanych z programowaniem. Tak więc, po wprowadzeniu w początkowych rozdziałach wszystkich podstawowych pojęć, z których w dalszych częściach korzysta, oraz omówieniu tych elementów budowy komputerów, które są niezbędne do analizy samego procesu programowania, przechodzi autor w następnych rozdziałach do omawiania kolejnych problemów związanych z tym procesem. W rozdziale siódmym daje przegląd pojęć związanych z językami programowania, w rozdziale ósmym z pojęciem „typ danych", aby w rozdziałach dziesiątym i jedenastym omówić sposoby przechowywania danych w komputerze, a w rozdziałach trzynastym i czternastym — sposoby działania na tych danych. Rozdziały dziewiąty i dwunasty są poświęcone pewnym specyficznym konstrukcjom programowym. Dopiero w ostatnim — piętnastym — rozdziale można odnaleźć pewne elementy pogłębiające sposób patrzenia na tworzenie programów, proponując tu tzw. programowanie metodą kolejnych ulepszeń. Ale i tu, jak w całej książce, w zasadzie prezentowane są konkretne techniki postępowania. Ta właśnie „konkretność", wyrażająca się wyjaśnianiem wszelkich problemów na przykładach, jest bardzo charakterystyczna dla omawianej książki. Czytelnikowi znającemu prace W. M. Turskiego czy D. Van Tassela może brakować jakiejś refleksji i dystansu do omawianych zagadnień, znajdzie w niej jednak solidne podstawy dla własnych przemyśleń.

Stanisława BONKOWICZ-SITTAUER

Informatyka w transporcie

System transportowy w gospodarce narodowej przypomina system krwionośny w żywym organiźmie. Warunkiem koniecznym dla właściwego działania takiego „organizmu" jest poprawne funkcjonowanie obsługującego go systemu transportowego.

Z uwagi na skalę wielkości, złożoność powiązań wewnętrznych oraz nieustanny rozwój systemu transportowego, kierowanie tym systemem jest niezwykle trudne, pracochłonne i wymaga szczególnej sprawności działania. Nieodzowne staje się zastosowanie informatyki.

W krajowej literaturze przedmiotu zaprezentowano już wiele poglądów i doświadczeń na temat zastosowań informatyki w przemyśle, budownictwie, czy obrocie towarowym, natomiast nader mało powiedziano na temat zastosowań informatyki w transporcie. Toteż z dużą satysfakcją należy powitać dwie najnowsze pozycje przygotowane przez Wydawnictwa Komunikacji i Łączności.*)

A. Wielądek w książce „Informatyka w zarządzaniu koleją" na wstępie ogólnie charakteryzuje całokształt zagadnień dotyczących zarządzania transportem, a następnie koncentruje się na problemach zastosowania informatyki na potrzeby zarządzania transportem kolejowym — prezentując aktualny stan i wskazując kierunki dalszego rozwoju zastosowań.

Na szczególną uwagę zasługują trzy spośród opisanych systemów:

- scentralizowany system obliczeń i rozliczeń należności za przewóz ładunku koleją
- system operatywnego śledzenia pociągów wahadłowych
- system informowania kierownictwa o przewozach kontenerowych.

Należy podkreślić, że te duże i bardzo dynamiczne systemy informatyczne oparto na stosunkowo skromnej, krajowej bazie technicznej. Bardzo interesujące są doświadczenia organizacyjne i metodyczne uzyskane w trakcie opracowywania i wdrażania systemów. Wzbogacają one niewielkie jeszcze doświadczenia metodyczne w dziedzinie projektowania resortowych systemów informatycznych. Rozwiązania systemowe wdrażane w PKP również wspomagają systemy zarządzania resortów przemysłowych. Stąd więc wzajemne zainteresowania rozwiązaniami, które prowadzą do coraz szerszej integracji i koordynacji działań.

Bardzo ciekawe są też dane charakteryzujące zasady rozwiązania analogicznych zagadnień w zagranicznych zarządach kolejowych. Szkoda tylko, że dane te, podobnie zresztą jak i dane statystyczne charakteryzujące naszą kolej, są trochę przestarzałe.

O ile pierwsza z omawianych publikacji dotyczy wyłącznie zastosowań informatyki w zarządzaniu, to druga — „Zautomatyzowane systemy kierowania transportem w przemyśle" — prezentuje doświadczenia praktyczne i kierunki prac badawczych w dziedzinie zastosowań informatyki do operatywnego sterowania lub inaczej kierowania dyspozytorskiego transportem przemysłowym w ZSRR.

M. I. Szmulewicz omawia tu ogólne zasady budowy podsystemów kierowania transportem w ramach zautomatyzowanych systemów kierowania przedsiębiorstwami w podstawowych gałęziach przemysłu: górnictwie węglowym i rud żelaza, hutnictwie, przemyśle maszynowym i chemicznym oraz w międzyzakładowych kompleksach transportowo-technologicznych. Autor charakteryzuje również funkcje zautomatyzowanych systemów kierowania transportem oraz ich algorytmy i rozwiązania techniczne, a także poszczególne etapy wdrażania i strukturę wdrożonych systemów. Przedmiotem rozwiązań są nie tylko systemy transportu kolejowego, lecz również samochodowego i wewnątrzzakładowego, realizowanego za pomocą wózków elektrycznych.

Zaprezentowany dorobek jest obszernym przeglądem badań, doświadczeń i wdrożeń wykonanych przez wiele znanych radzieckich instytutów naukowych, fabryk, kopalń i hut.

Materiał zawarty w książce może być interesujący również dla polskiego czytelnika, a szczególnie dla kierownictwa i służb transportowych dużych kombinatów przemysłowych. Zasady automatyzacji poszczególnych zadań w dziedzinie sterowania pracą transportu przemysłowego oraz efekty ekonomiczne i organizacyjne wynikające z tej automatyzacji, jak też możliwości zdobywania informacji mogą być przedmiotem bezpośrednich kontaktów roboczych pomiędzy zainteresowanymi instytutami czy kombinatami przemysłowymi Polski i ZSRR.

Obydwie publikacje znakomicie wypełniają istniejącą do tej pory lukę w polskiej literaturze fachowej na temat zastosowań informatyki w transporcie i sądzę, że znajdą one uznanie zarówno teoretyków, jak i praktyków w tej dziedzinie gospodarki.

Zbigniew BIEŃKO

*) **A. Wielądek: Informatyka w zarządzaniu koleją. WKiŁ, Warszawa 1979; stron 248, nakład 3000 egz., cena 40 zł**
M. I. Szmulewicz: Zautomatyzowane systemy kierowania transportem w przemyśle; tłumaczenie z języka rosyjskiego. WKiŁ, Warszawa 1979; stron 306, nakład 2000 egz., cena 50 zł.

GRZEGORZ GRUCHMAN
ZETO Poznań

Technologia baz danych w praktyce

Problemy technologii baz danych — obok zagadnień przetwarzania rozproszonego — dominują dziś w fachowym piśmiennictwie zagranicznym poświęconym przetwarzaniu danych na potrzeby zarządzania. Technologia ta została uznana za jedyne narzędzie budowy kompleksowych (wielodziedzinowych) zastosowań eto w zarządzaniu. Jej historia rozpoczęła się w 1964 roku, kiedy to na amerykańskim rynku oprogramowania pojawił się system IDS (Integrated Data Store), prekursor nowoczesnych systemów zarządzania bazą danych. Kolejne kamienie milowe rozwoju technologii to raporty Grupy Roboczej ds. Baz Danych (DBTG) komitetu CODASYL, opublikowane w 1969 i 1971 roku, a także słynny artykuł Codda z 1970 roku o relacyjnym modelu danych. Wreszcie, w 1975 roku, ukazał się raport komitetu X3/SPARC Instytutu Standardów ANSI, spisujący pełną architekturę docelowego systemu zarządzania bazą danych (SZBD).

W natłoku publikacji poświęconych pracom badawczym i różnym aspektom technologii baz danych zaciera się jednak jej praktyczne oblicze. Niniejszy artykuł jest próbą przedstawienia obecnego stanu zastosowań technologii na świecie i ogólnej charakterystyki użytkowych SZBD. Artykuł omawia także pokrótce najbardziej znane systemy zagraniczne oraz polskie.

STOSOWANIE BAZ DANYCH

Użytkowe bazy danych są najbardziej rozpowszechnione w Stanach Zjednoczonych, gdzie głównym ich użytkownikiem jest przemysł. Możliwość modelowania powiązań świata rzeczywistego w bazie danych była dla przedsiębiorstw przemysłowych szczególnie atrakcyjna ze względu na występującą w nich dużą liczbę powiązanych danych. Technologia ta nie jest jednak tak szeroko stosowana jak mogłoby się wydawać. W 1978 roku szacowano, że w bazach danych obsługiwanych przez SZBD znajduje się od 10 do 20% danych zgromadzonych w zewnętrznych pamięciach komputerowych [1]. Szacunek ten dotyczył jedynie Stanów Zjednoczonych, w innych krajach procent ten jest z pewnością niższy.

Zaskakuje niewielka liczba wspólnych baz danych, wykorzystywanych w kilku zastosowaniach. Prawie wszystkie przedsiębiorstwa stosujące SZBD użytkują wiele baz danych, z których każda obsługuje jedno zastosowanie. Świadczą o tym opisy baz danych i badania ankietowe przeprowadzone w USA i Wielkiej Brytanii w ciągu ostatnich kilku lat [2, 3, 4, 5]. Tak więc podstawowy atut baz danych — możliwość integracji zastosowań na poziomie danych — jest bardzo rzadko wykorzystywany. Nie znaczy to, że budowa wspólnych baz danych jest niemożliwa, gdyż takowe istnieją. Przyczyn małej ich liczby należy szukać w braku pełnej metodologii projektowania logicznych struktur baz danych oraz w problemach natury organizacyjnej, związanych ze wspólnym użytkowaniem danych. Z reguły wybierano prostszą drogę budowy szeregu odrębnych, zorientowanych dziedzinowo baz danych niż stopniową rozbudowę jednej bazy celem użytkowania jej w szeregu zastosowań. Z kolei zbudowane niezależnie dziedzinowe bazy danych — jak wykazuje praktyka — są trudne do zintegrowania i proces ten jest odkładany na później. Pełna metodologia projektowania, zakładająca m.in. stosowanie słownika metadanych (definicji danych), dopiero się krystalizuje. Natomiast za jedyne lekarstwo na konflikty związane ze wspólnym wykorzystaniem danych uważa się powołanie stanowiska Administratora Danych na najwyższych szczeblach zarządzania przedsiębiorstwem.

CHARAKTERYSTYKA SZBD

Obecnie na dosłownie każdym typie dużego komputera i minikomputera można eksploatować co najmniej jeden SZBD. Powstaje także coraz więcej systemów dla minikomputerów. Ponieważ każdy producent sprzętu oferuje SZBD dla swych komputerów — a są jeszcze przecież niezależni producenci oprogramowania — liczba systemów jest bardzo duża. Podczas badań na jednym z uniwersytetów amerykańskich doliczono się około 50 systemów, a jest ich na pewno więcej [6].

Na światowym rynku oprogramowania liczą się właściwie tylko systemy amerykańskie. Jedynym wyjątkiem jest ADABAS (Adaptable DAta BAse System) produkcji niemieckiej. Liczbę instalacji najbardziej rozpowszechnionych szacowano z początkiem bieżącego roku na około 6 000 [7]. Rekordzistami są systemy TOTAL (2 000 instalacji) oraz IMS (Information Management System, 1 500 instalacji). Liczba zainstalowanych systemów typu CODASYL wynosi 1 050. Nie jest to jednak właściwy miernik popularności specyfikacji DBTG, gdyż pierwsze systemy na nich bazujące pojawiły się dopiero w 1971 roku.

Dla współczesnych SZBD charakterystyczne jest zatarcie granicy między systemami o hierarchicznym i sieciowym modelu danych. Najnowsze wersje pierwotnie hierarchicznych systemów — jak IMS lub SYSTEM 2000 — pozwalają obecnie na budowę struktur sieciowych. Co prawda w pierwszym z nich obowiązują poważne ograniczenia, w drugim z kolei można je tworzyć tylko programowo. Zawsze jednak takie możliwości są lepsze niż czysto hierarchiczny model danych. Zatarciu uległa także granica między systemami automatycznymi i systemami języka bazowego. Ze względu na ograniczone możliwości nieproceduralnych języków programowania dla systemów autonomicznych opracowano łącza z językami bazowymi. Łącza te działają w oparciu o makroinstrukcję CALL, a przykładem systemu może być ADABAS lub znowu SYSTEM 2000.

Równocześnie okazało się, że w wielu przypadkach pożądany jest nieproceduralny dostęp do baz danych obsługiwanych zasadniczo przez systemy języka bazowego. Problem ten został rozwiązany przez wbudowanie odpowiednich łączy w niektóre systemy zarządzania zbiorami konwencjonalnymi i generatory raportów. Na przykład w przetwarzaniu wsadowym baz danych typu IMS lub TOTAL do nieproceduralnego dostępu można wykorzystać systemy MARK-IV lub ASI-ST. Z kolei nieproceduralny dostęp w trybie *on-line* do baz danych IMS zapewnia system ASI/INQUIRY. Wiele pierwotnie bazowych SZBD ma także własne, opracowane w późniejszym okresie nieproceduralne środki dostępu.

SZBD mają szereg zalet i oczywiście wad. Ciekawe badania ich ocen z praktycznego punktu widzenia przeprowadzono w 1978 roku w USA [5]. Użytkownicy SZBD najwyżej cenili niezależność danych, wysoką integralność danych w połączeniu z teleprzetwarzaniem, centralizację opisu danych oraz łatwość restrukturalizacji bazy. Teleprzetwarzanie jest zaletą dość zaskakującą, gdyż do przetwarzania baz danych wcale nie jest potrzebny dostęp w trybie *on-line* i odwrotnie. Jednak fakt, że prawie wszyscy indagowani użytkownicy podkreślili teleprzetwarzanie jako zaletę SZBD, świadczy o ścisłym powiązaniu baz danych i teleprzetwarzania w praktyce.

O mniejszym zapotrzebowaniu na pamięć zewnętrzną — która miała być jedną z głównych zalet technologii — wyrażano się niechętnie. Integracja danych w obrębie zasto-

sowań ograniczyła co prawda liczbę znaków danych użytkowych. Jednak obciążenie ich łącznikami oraz indeksami wykorzystywanymi przez SZBD spowodowało, że zapotrzebowanie na pamięć z reguły wzrastało.

Do wad SZBD zaliczono spadek osiągów (wydajności) przetwarzania wskutek stosowanej bezpośredniej metody dostępu, wzrost kosztów eksploatacji, oprogramowania i sprzętu (dyski), a także problemy z bezpośrednim użytkownikiem oraz (tak jest!) z projektantami i programistami. Opór ludzki i niezrozumienie istoty technologii baz danych są zresztą głównymi przyczynami kłopotów i niepowodzeń związanych z instalowaniem SZBD [1].

Dla technologii baz danych charakterystyczny jest brak standardów. Z wielu przyczyn (godnych odrębnego rozpatrzenia) specyfikacje CODASYL-u nie zostały oficjalnie uznane za standard amerykański i/lub międzynarodowy i nie wiadomo zresztą czy zostaną. Nie pomogła ich popularność oraz naciski użytkowników i niektórych producentów sprzętu. Równocześnie brak na rynku oprogramowania relacyjnego SZBD. Istnieją co prawda dziesiątki takich systemów, ale wszystkie — nawet najbardziej kompleksowy System R firmy IBM — mają charakter wyłącznie eksperymentalny.

Fizyczne struktury danych stosowane przy konwencjonalnych pamięciach dyskowych są podobno adekwatne dla relacyjnych baz danych o wielkości nie większej niż 1 MB [8]. Powyżej tej wielkości spadek osiągów przetwarzania jest tak znaczny, że relacyjne bazy danych nie wytrzymują konkurencji z sieciowymi lub hierarchicznymi. Z kolei milion bajtów jest wielkością zbyt małą dla użytkowych zastosowań. Istnieją za to łącza umożliwiające konwersję sieciowego lub hierarchicznego modelu danych na relacyjny. Firma Honeywell oferuje takie łącze — o nazwie Multics Data Base Manager — już od 1976 roku. Jak głosi plotka zamieszczona w kwietniowym „DATAMATION", także IBM ma zamiar w najbliższym czasie przekazać do sprzedaży relacyjne łącze dla baz danych typu IMS.

PRZEGLĄD SZBD

Najbardziej popularnym SZBD na świecie jest TOTAL wyprodukowany przez Cincom Systems Inc., którego pierwsza wersja ukazała się na rynku oprogramowania w 1969 roku [1]. Przyczyną popularności systemu jest duża prostota, dzięki której bardzo łatwo go instalować i eksploatować. TOTAL był także pierwszym SZBD dla minikomputerów. Obecnie można go instalować na wszystkich ważniejszych typach dużych komputerów i oczywiście mini. System używa sieciowego modelu danych, zbliżonego bardzo do CODASYL-owskiego.

Za najlepsze systemy użytkownicy uważają IDMS (Integrated Database Management Systems), ADABAS oraz IMAGE/3000. System IMAGE — wyłącznie dla minikomputerów firmy Hewlett-Packard — jest stosunkowo prosty, zbliżony do systemu TOTAL i nie będzie szerzej omawiany. Z wymienionych systemów zdecydowanie wyróżnia się IDMS. Ten sztandarowy SZBD typu CODASYL jest bardzo wysoko oceniany przez użytkowników już trzeci rok z rzędu. Przewyższa wszystkie systemy ocenami przepustowości/wydajności, dokumentacji, szkolenia oraz serwisu sprzedawcy, który w krytycznych momentach przerzuca swoich specjalistów do użytkownika wynajętymi samolotami (reklamy milczą na czyj rachunek).

Ciekawa jest historia tego systemu. Jego pierwsza wersja — dość prymitywna zresztą — powstała w 1970 roku w przedsiębiorstwie przemysłowym Goodrich B. F. Chemicals CO. Firma ta, wobec braku na rynku systemu typu CODASYL dla maszyn IBM 360/370, zdecydowała się na wyprodukowanie własnego SZBD. IDMS powstał w oparciu o specyfikacje DBTG z 1969 roku, nakładem zaledwie 6 osobolat. Później system został zakupiony przez niezależnego producenta oprogramowania Culliname Corp., który jest obecnie jego głównym sprzedawcą. System jest rozpowszechniany także przez Scicon Ltd., ICL oraz Univac (pod nazwą DMS-90). Liczba instalacji IDMS-u wynosi obecnie około 400. Firma Culliname oferuje IDMS jako składnik kompleksowego systemu zarządzania danymi. Pozostałe jego elementy składowe to generator raportów CULPRIT, język zapytań ON-LINE QUERY, monitor teleprzetwarzania IDMS-DC oraz pakiet przetwarzania tekstów i programowania konwersacyjnego INTERACT. „Sercem" systemu, integrującym jego elementy składowe, jest słownik metadanych.

System ADABAS został wyprodukowany w niemieckiej firmie Software AG nakładem 40 osobolat. Na rynku oprogramowania ukazał się w 1971 roku, a liczba jego instalacji wynosi około 375. Przyjęte rozwiązania w dziedzinie fizycznej organizacji danych pozwalają na dynamiczne (programowe) tworzenie struktur hierarchicznych i sieciowych. Umożliwia to m.in. restrukturalizację bazy danych bez jej przeładowania (jedyny system!). Wszystkie dane podlegają standardowo bardzo wydajnej kompresji, a jedną z atrakcyjnych opcji systemu jest szyfrowanie/rozszyfrowanie danych. ADABAS ma także własny generator raportów ADAWRITER oraz język zapytań ADASCRIPT.

Spośród wszystkich SZBD stosunkowo najgorszą opinią wśród użytkowników „cieszy się" system IMS produkcji IBM, a właściwie grupa SZBD wykorzystujących język danych CL/1. IBM oferuje bowiem trzy systemy oparte na tym języku, który powstał w 1966 roku w firmie North American Aviation jako produkt uboczny księżycowego programu APOLLO. Najbardziej z nich znany IMS wszedł do sprzedaży w 1969 roku. Jest on właściwie systemem zarządzania danymi, gdyż stanowi kombinację oprogramowania bazy danych oraz transmisji danych. Duża liczba instalacji IMS-u jest raczej wynikiem agresywnej polityki koncernu, gdyż użytkownicy z reguły system ten krytykują. Głównym zarzutem jest nadmierna kompleksowość. Wskutek niej IMS jest trudny w instalowaniu i eksploatacji, „pożera" dużo pamięci operacyjnej i powoduje znaczny spadek osiągów przetwarzania. Nie można jednak zapominać, że był to w 1969 roku jedyny SZBD dla maszyn serii 360/370, a także pierwszy, który umożliwił teleprzetwarzanie bazy danych. W związku z systemem IMS nasuwa się ciekawe spostrzeżenie. Najbardziej dziś popularny SZBD TOTAL oraz 2 najlepsze — ADABAS i IDMS — pochodzą od niezależnych producentów oprogramowania.

W Polsce najbardziej znanym SZBD jest RODAN, wyprodukowany w OBRI (obecnie Centrum Projektowania i Zastosowań Informatyki ZETO). Jest to system typu CODASYL, którego bardzo dobry opis można znaleźć w [11]. Wyróżnia go odrębny język opisu pamięci o szerszych możliwościach deklarowania fizycznej struktury danych niż proponowany przez CODASYL język sterowania urządzeniami/nośnikami (DMCL). W opisach RODAN-u zwraca uwagę brak generatora raportów. Jeżeli istotnie prace nad nim nie są prowadzone, byłoby to poważne niedopatrzenie. Użytkownicy byliby pozbawieni prostego narzędzia nieproceduralnego generowania potrzebnych „na wczoraj" wydruków. Liczba instalacji RODAN-u wynosi obecnie 15. Dwie z nich omawiane były na konferencji w Kaliszu w maju br., poświęconej oprogramowaniu maszyn serii RIAD [12, 13].

Oprócz RODAN-u istnieją w Polsce jeszcze dwa SZBD. Pierwszy z nich to wyprodukowany w OBRI autonomiczny SYKON — SYstem KONwersacyjny [14, 15]. Drugim natomiast jest SAD (System Administrowania Danymi), opracowany w Instytucie Maszyn Matematycznych. Poza łączami z językami bazowymi system ten posiada ciekawy język zapytań KWINTET [16, 17].

* * *

Technologia baz danych jest dzisiaj najbardziej dynamiczną dziedziną informatyki, ale era jej praktycznych zastosowań dopiero się rozpoczęła. Szacuje się, że w połowie lat osiemdziesiątych na świecie będą działać dziesiątki tysięcy instalacji SZBD, a liczba danych w bazach wyniesie 70—80%. Rozpowszechnią się wspólne bazy danych i dopiero wówczas przedsiębiorstwa zaczną odczuwać ich wpływ na zarządzanie.

[1] Wyboru systemów dokonano na podstawie najnowszych rezultatów ankiety DATAMATION/Datapro [9]. Krótkie opisy systemów opracowane zostały wg [2] i [10], a także reklam omawianych produktów

W dziedzinie użytkowych SZBD należy jednak spodziewać się nie tyle rewolucji, co ewolucji. Względną stabilizację w ciągu najbliższych kilku lat gwarantują duże nakłady finansowe poniesione na istniejące systemy przez ich producentów i użytkowników. SZBD z ugruntowaną pozycją będą modernizowane i rozbudowywane, ale bez zasadniczych przeobrażeń ich podstaw. Dowodem długowieczności systemów może być liczba użytkowników piętnastoletniego IDS-u, która wynosi obecnie 500. Firma IBM będzie oferować z kolei IMS dla nowych maszyn serii 4300, co przedłuża egzystencję systemu o dobre kilka lat. Powszechnie uważa się, że poważniejsze zmiany nastąpią po 1985 roku. Według prognoz, dopiero wówczas powszechne użytkowe zastosowanie znajdą relacyjne i rozproszone bazy danych oraz wyspecjalizowany sprzęt do ich obsługi.

LITERATURA:

[1] Canning R.: Planning for DBMS conversions. EDP ANALYZER 5/1978
[2] Palmer I.: Dabase Systems — A practical Reference. CACI International 1975
[3] Boot R.: Doświadczenia użytkowników stosujących systemy gospodarowania bazami danych, w: „Baza danych, Europejski Program Badawczy Diebolda", Zeszyt 72, OBRI 1975, s. 104—130
[4] Davis B.: User experience with DBMS in the United Kingdom. DATA PROCESSING DIGEST 4/1978, s. 18—19
[5] Wiorkowski G., Wiorkowski J.: Does a Database Management System Pay off? DATAMATION 4/1978, s. 109—114
[3] Canning R.: How to prepare for the coming changes. EDP ANALYZER 4/1979
[7] Baker G.: Database software — a wide choice. DATA PROCESSING 4/1979, s. 28—29
[8] Canning R.: Toward the better management of data. EDP ANALYZER 12/1976
[9] Gepner H.: User ratings of software packages. DATAMATION 12/1978, s. 163—167
[10] Tsichritzis D., Lochovsky F.: Data Base Management Systems. Academic Press 1977
[11] Pasula J.: Uniwersalny system zarządzania bazą danych RODAN. INFORMATYKA 9/1973, s. 37—40
[12] Pawłowska A.: Badania techniczne systemu RODAN w ZETO Poznań, w: „Systemy zarządzania bazami danych i przegląd zastosowań oprogramowania komputerów JS RIAD", TNOiK, Kalisz 28—29 maja br.
[13] Jamontt W.: Zastosowanie systemu RODAN do tworzenia Regionalnego Banku Danych Statystycznych STATYW, w: „Systemy zarządzania bazami danych..."
[14] Bogucki W., Staniszkis W.: System zarządzania bazą danych SYKON. INFORMATYKA 11/1974, s. 47—50
[15] Wacławik A.: SYKON — system zarządzania bazą danych. INFORMATYKA 10/1978, s. 32—34
[16] Kołacka D.: System Administrowania Danymi — SAD. w: „Systemy zarządzania bazą danych..."
[17] Tarasiuk J., Zaborowska E.: System KWINTET. Konwersacyjne Wyszukiwanie INformacji TEle-Transmitowanych, w: „Systemy zarządzania bazą danych...".

ANDRZEJ BRANDT
Centrum Projektowania i Zastosowań Informatyki ZETO
Warszawa

Projekt struktury pamięci

Fizyczna struktura danych, przyjęta w określonym systemie zarządzania bazą danych, wyznacza zakres możliwości odwzorowania logicznej struktury danych na urządzeniach pamięci zewnętrznej i zdeterminowana jest w poważnym stopniu możliwościami technicznymi tych urządzeń.

Istotny wpływ na parametry eksploatacyjne zastosowań realizowanych w oparciu o bazę danych ma więc wybór odpowiednich odwzorowań (odpowiedniej fizycznej struktury danych) w projekcie struktury pamięci.

Projekt struktury pamięci dla bazy danych wykonywany jest przez Administratora Bazy Danych (ABD), często przy współpracy z administratorem (administratorami) zastosowań oraz technologiem oprogramowania (programistą systemowym). Proces projektowania jest wspomagany komputerem. Na przykład w SZBD RODAN do definiowania struktury pamięci służy Język Opisu Pamięci (JOP), zaś rolę postulowanego w literaturze Języka Sterowania Urządzeniami (DMCL — Device/Media Control Language) spełnia tu Job Control Language (JCL) systemu operacyjnego OS (OS/JS).

W niektórych SZBD możliwe jest daleko idące uproszczenie projektowania struktury pamięci. Pozostając przy przykładzie RODAN-u należy stwierdzić, że większość klauzul JOP jest opcjonalna, co powoduje przyjęcie przez system wartości parametrów struktury pamięci przez domniemanie. W skrajnym przypadku nawet całkowite opuszczenie opisu pamięci spowoduje przyjęcie przez domniemanie kompletnej struktury pamięci dla określonego schematu bazy danych. Oczywiście taka szablonowa struktura daleka będzie od optymalnej.

Wykonywanie projektu struktury pamięci, jak wykazuje praktyka, jest procesem iteracyjnym. Uzyskanie optymalnej w danych warunkach struktury pamięci od razu, „już w pierwszym podejściu", nie jest praktycznie możliwe.

Aby zapewnić sobie wpływ na efektywność projektowanej bazy danych, ABD (i jego zespół projektantów struktury pamięci) musi umiejętnie określić przede wszystkim:
— sposób reprezentacji zbiorów strukturalnych
— typ łączników adresowych zbiorów strukturalnych
— liczbę stron, ich wielkość i stopień wypełnienia
— sposób indeksowania zbiorów
— strategię poszukiwania wolnych miejsc w bazie danych.

Wydaje się, że proces projektowania struktury pamięci należy podzielić na trzy etapy:
etap I: odwzorowanie rekordów i zbiorów strukturalnych (setów)
etap II: określenie stron i obszarów bazy danych
etap III: weryfikacja projektu struktury pamięci.
Kolejność czynności wykonywanych w poszczególnych etapach przez ABD i jego zespół oraz sposób postępowania w kilku określonych sytuacjach ilustrują odpowiednio rysunki 1, 2 i 3.

Przy **ustalaniu długości rekordu** należy dążyć do:

- ograniczenia fizycznej długości pól zajmowanych przez poszczególne dane elementarne, co można spowodować 1° wybierając najbardziej oszczędną reprezentację danych arytmetycznych, a zwłaszcza rezygnując z przechowywania danych w postaci zredagowanej, 2° kodując i dekodując dane, 3° określając zmienną długość danych czy też definiując dane wirtualne

● ograniczenia długości części organizacyjnej rekordu — poprzez weryfikację uczestnictwa rekordu w zbiorach strukturalnych oraz wybór takiej metody organizacji zbioru, która przy zapewnieniu odpowiednich własności użytkowych wymaga najmniejszej liczby łączników adresowych.

Decyzje o **wyborze długości segmentu** różnej od długości rekordu winny być dokonywane ostrożnie i tylko w uzasadnionych przypadkach. Długość segmentu większą od długości rekordu należy stosować jedynie w sytuacjach, gdy przewidywane jest zwiększenie długości rekordu w trakcie eksploatacji bazy danych. Z kolei długość segmentu mniejsza od długości rekordu może być korzystna przy wprowadzaniu nowych wystąpień rekordów do bazy danych wypełnionej w znacznym stopniu (ponad 70%). Wobec małego prawdopodobieństwa znalezienia miejsca w bazie danych na duży segment, wystąpienie rekordu podzielonego na kilka segmentów zostanie znacznie szybciej zapisane niż postać nie podzielona.

Większość SZBD umożliwia **realizację zbioru strukturalnego** w postaci listy okrężnej (CHAIN) lub zbioru z tablicą łączników adresowych (POINTER-ARRAY).

Postać listy okrężnej wybierana jest do realizacji zbioru przede wszystkim w przypadku, gdy:
— liczba rekordów członkowskich zbioru w wystąpieniach zbioru jest niewielka
— proces przetwarzania zbioru będzie wymagał dostępu do wszystkich (lub niemal wszystkich) rekordów członkowskich zbioru.

Przyjmując tę metodę organizacji zbioru strukturalnego należy pamiętać, że czas przeszukiwania listy jest wprost proporcjonalny do jej długości.

Powyższe zalecenie można uszczegółowić uwzględniając różne warianty listy okrężnej:
● lista okrężna jednostronna winna być stosowana w przypadku, gdy proces przetwarzania zbioru strukturalnego będzie sekwencyjny według kolejności uporządkowania rekordów członkowskich w zbiorze; metoda ta pozwala ponadto na najoszczędniejsze wykorzystanie miejsca w bazie danych, co ma szczególne znaczenie przy bardzo licznych zbiorach
● lista okrężna dwustronna znajduje zastosowanie:
— w odniesieniu do zbiorów często modyfikowanych (istnienie dwóch list łączników adresowych przyspiesza usuwanie rekordów członkowskich ze zbioru)
— w przypadku, gdy proces przetwarzania zbioru strukturalnego będzie sekwencyjny zarówno w kolejności uporządkowania, jak i w kolejności odwrotnej do kolejności uporządkowania rekordów członkowskich w zbiorze strukturalnym
● lista okrężna z łącznikiem adresowym do rekordu — właściciela winna być stosowana, jeśli w procesie przetwarzania zajdzie potrzeba częstego wyszukiwania tego rekordu.

Postać zbioru z tablicą łączników adresowych wybierana jest do realizacji przede wszystkim w następujących przypadkach:
● gdy rekordy członkowskie będą często przetwarzane w kolejności różnej od określonego w zbiorze porządku logicznego
● gdy przewiduje się liczne operacje wyszukiwania w zbiorze sortowanym (związane jest to z możliwością zastosowania efektywniejszego algorytmu tablicy, np. cięcia binarnego zamiast pełnego przeglądu tablicy)
● gdy zbiór zawiera więcej niż jeden typ rekordów członkowskich — tablica łączników adresowych umożliwia zorganizowanie wyszukiwania w ramach tylko jednego typu rekordów członkowskich.

Topografią zbioru strukturalnego nazywa się tutaj sposób rozmieszczenia elementów tego zbioru w bazie danych.

Z samej istoty przetwarzania zbioru strukturalnego wynika potrzeba dostępu na przemian do rekordów-właścicieli i rekordów członkowskich oraz do tablicy łączników adresowych (jeżeli istnieje). Skrócenie czasu przetwarzania można osiągnąć poprzez umieszczenie powyższych elementów w oddzielnych obszarach bazy danych.

Jeśli z innych względów rekordy-właściciele i rekordy członkowskie zbioru muszą być przypisane do tych samych obszarów, należy zadbać, by rekordy te zostały umieszczone w fizycznej bliskości (na tych samych lub sąsiednich stronach bazy danych). Zwiększa się wtedy prawdopodobieństwo, że właściciel i członkowie tego samego wystąpienia zbioru mogą być przeniesieni do buforu SZBD za pomocą jednego transferu fizycznego. Jeśli niezbędne będą dodatkowe przesłania, czas ich też będzie krótki, gdyż sąsiadujące strony obszaru umieszczane są na sąsiednich ścieżkach cylindra na dysku.

Dla zbiorów strukturalnych posiadających tablicę łączników adresowych możliwe jest kształtowanie jeszcze jednego elementu ich topografii, a mianowicie miejsca zapamiętania tej tablicy oraz długości jej segmentu. Można tu wyróżnić trzy możliwości:

1) Tablica łączników adresowych w obszarze A, rekord-właściciel i rekordy członkowskie — w obszarze B
Tablica łączników adresowych winna być podzielona na segmenty w liczbie równej liczbie wystąpień zbioru. Długość segmentu winna być wystarczająca na pomieszczenie liczby łączników adresowych równej średniej liczbie rekordów-członków zbioru. Jeżeli przyjęto małą długość strony, niewystarczającą na pomieszczenie takiej długości segmentu tablicy, jego długość należy określić jako możliwie największą.

2) Tablica łączników adresowych i rekordy członkowskie w tym samym obszarze A
Długość segmentów tablicy łączników adresowych winna być tak określona, aby mogły one pomieścić liczbę łączników adresowych równą liczbie rekordów członkowskich na jednej stronie obszaru. Prawdopodobieństwo umieszczenia tablicy i odpowiadających jej rekordów fizycznie blisko siebie wzrasta wtedy znacznie.

3) Tablica łączników adresowych i rekord-właściciel w tym samym obszarze A.
Tablica łączników adresowych powinna być lokowana w tym samym miejscu co rekord-właściciel wyłącznie w przypadku, gdy liczba rekordów-członków w wystąpieniu zbioru jest niewielka. Wtedy tablica może być jednosegmentowa i może zajmować niewielki ułamek przestrzeni strony. Przy dużej liczbie rekordów-członków przetwarzanie zbioru o takiej topografii może być bardzo czasochłonne.

Projektowanie struktury fizycznej indeksów jest zagadnieniem skomplikowanym. Dla określonego w Języku Opisu Danych argumentu, dla którego zakładany będzie indeks, w Języku Opisu Pamięci określa się sposób tworzenia indeksu dla wystąpień zbioru strukturalnego oraz miejsce jego przechowywania, a także długość segmentu w tablicy indeksowej, tj. liczbę kluczy w rekordzie indeksu.

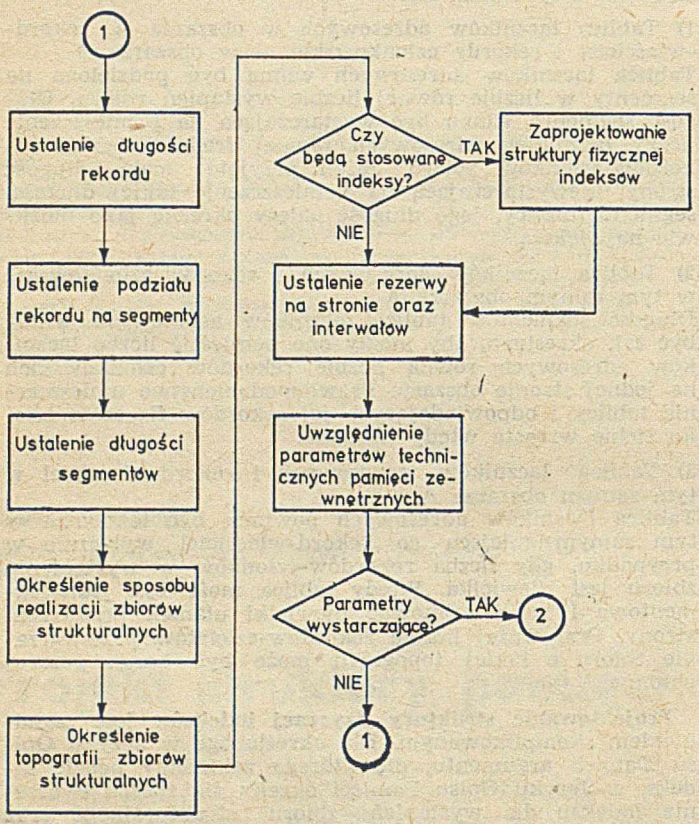
Problem wyboru miejsca zapamiętania indeksu w bazie danych zbliżony jest do omówionego wyżej zagadnienia określania sposobu rozlokowania tablic łączników adresowych w bazie danych. Najkorzystniej jest umieścić indeks w oddzielnym obszarze, zapewniając w ten sposób systemowi zarządzania bazą danych niezależny dostęp do indeksu i rekordów indeksowanych. Natomiast umieszczenie obu tych grup danych łącznie pogarsza strukturę fizyczną bazy danych i powoduje konieczność zwiększenia liczby transferów fizycznych między urządzeniami pamięci zewnętrznej a buforami SZBD.

Ustalenie rezerwy na stronie, czyli inaczej ustalenie stopnia wypełnienia strony, ma na celu ustalenie wielkości dopuszczalnego wypełnienia stron w obszarze bazy danych. Wielkość ta wyrażana jest zwykle w procentach lub w bajtach.

Określenie dopuszczalnego stopnia wypełnienia strony pozwala na zarezerwowania miejsca dla przyszłych modyfikacji rekordów (zwiększania ich długości). Do późniejszego wykorzystania można również zostawić całe wolne strony. Celowe jest to zwłaszcza w odniesieniu do rekordu-właściciela zbioru. Na określonych stronach zostaną umieszczone pojedyncze wystąpienia rekordu-właściciela i będzie zachowane wolne miejsce dla wystąpień rekordów członkowskich zbioru. Wokół wystąpień rekordu-właściciela zostaną utworzone skupienia rekordów-członków. Dzię-

ki takiej organizacji obszaru przy jednej operacji fizycznego dostępu do dysku otrzymuje się w buforze SZBD od razu pewną liczbę rekordów związanych logicznie, a pozostałe rekordy tego związku logicznego będą się znajdować w najbliższym sąsiedztwie fizycznym. Liczba stron interwału winna być tak określona, aby można było pomieścić bądź wszystkie wystąpienia rekordów-członków przedmiotowego zbioru, bądź taką ich liczbę, która wynika z przyjętej metody przetwarzania.

Przy projektowaniu fizycznej struktury danych należy również uwzględnić parametry techniczne pamięci zewnętrznej, a także ograniczenie związane z przewidzianymi do zastosowania środkami technicznymi (rys. 1).



Rys. 1. Projektowanie struktury pamięci — etap I: odwzorowanie rekordów i zbiorów strukturalnych

Wybór długości strony jest kompromisem między ograniczeniami wynikającymi z wpływu szeregu różnych czynników. Oto podstawowe ograniczenia:

- na jednej ścieżce dysku musi się zmieścić całkowita liczba stron; ogranicza to maksymalną długość strony obszaru do liczby bajtów możliwych do zapisania na jednej ścieżce
- SZBD określa minimalną długość strony (np. RODAN 128 bajtów, w tym nagłówek strony o długości 13 bajtów).

Nie zaleca się dokonywania wyboru długości strony większej niż 3520 bajtów (lub długości ścieżki najmniejszego modelu dysku przewidzianego dla bazy danych), gdyż utrudni to — lub wręcz uniemożliwi — późniejszą reorganizację bazy danych przez zwiększenie długości strony.

Ponadto wybór długości strony powinien uwzględniać:
— przewidywany sposób przetwarzania danych w obszarze
— rozmiar rekordu (długości wystąpień rekordów, które mają być zapisane do obszaru)
— liczbę wystąpień rekordów, które mają być zapisane do obszaru
— długość i liczbę wystąpień rekordów, które mają być umieszczone na jednej stronie obszaru
— sposób lokowania rekordów w obszarze bazy danych.

Jeśli chodzi o przewidywany sposób przetwarzania danych w obszarze, to:

— możliwie największa długość strony[1]) powinna być określana w przypadku, gdy przewiduje się sekwencyjne przetwarzanie rekordów w obszarze

— duża długość strony korzystna jest również w przypadku, gdy możliwe jest umieszczenie na jednej stronie obszaru całego wystąpienia zbioru strukturalnego wraz z ewentualną tablicą łączników adresowych

— względnie mała długość strony winna być wybierana wtedy, gdy przewidywane jest wyrywkowe przetwarzanie danych w obszarze.

Rozmiar rekordu ma szczególne znaczenie przy wyborze stron małych. Przy najmniejszych stronach zajętość miejsca na nagłówki zbliża się do 10%. Mała długość stron ogranicza ponadto rozmiar segmentów danych. Przy dużych rekordach użycie małej długości stron zwiększa niekorzystnie liczbę segmentów.

Przy ustalaniu liczby stron w obszarze należy wziąć pod uwagę, czy programy reorganizacji bazy danych zapewniają możliwość zwiększenia liczby stron w obszarze. Jeśli nie — określenie liczby stron w obszarze powinno zapewnić możliwość późniejszego rozwoju bazy danych.
Należy przyjąć, że liczba stron w obszarze *Ls* winna spełniać nierówność:

$$Ls \times (Ws - Wcos) > Lrek \times Drek$$

gdzie:
Ws — długość strony
Wcos — średnia długość części organizacyjnej strony
Lrek — liczba wystąpień rekordów w obszarze
Drek — średnia długość rekordu w obszarze

Wzór powyższy należy traktować orientacyjnie, jako górne oszacowanie.

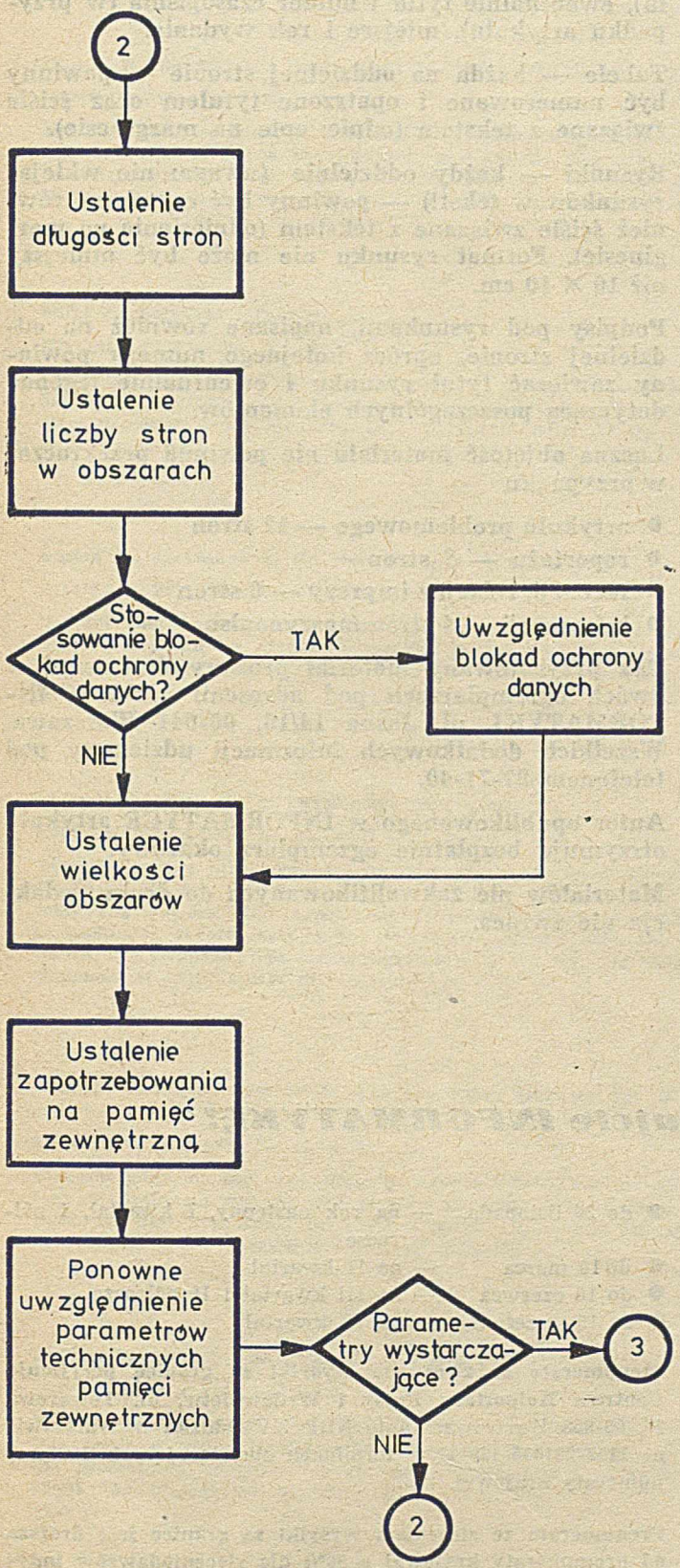
Po uwzględnieniu zapotrzebowania na powierzchnię dyskową związaną z wymogami ochrony danych można ustalić wielkość obszarów dla bazy danych oraz zapotrzebowanie na pamięć zewnętrzną dla projektowanej bazy danych (rys. 2). Następnie konieczne jest ponowne uwzględnienie parametrów technicznych pamięci zewnętrznej. W przypadku rozbieżności potrzeb z możliwościami ich zaspokojenia trzeba zweryfikować projekt struktury pamięci, poczynając od ponownego ustalenia długości strony obszaru bazy danych. Natomiast w przypadku, gdy urządzenia pamięci zewnętrznej mają pojemność wystarczającą na umieszczenie określonych obszarów bazy danych oraz możliwe są do spełnienia wymogi ochrony danych — można dokonać ostatecznej delimitacji obszarów bazy danych i związać je z konkretnymi obszarami dyskowymi (rys. 3). Kolejną czynnością jest opis ustalonej fizycznej struktury danych w odpowiednim dla SZBD Języku Opisu Pamięci.

Wersję źródłową opisu struktury pamięci, stanowiącą pełny opis fizycznej struktury bazy danych, poddaje się translacji. W zależności od wyniku translacji, tzn. w zależności od wykrytych błędów opisu, postępujemy jak niżej:

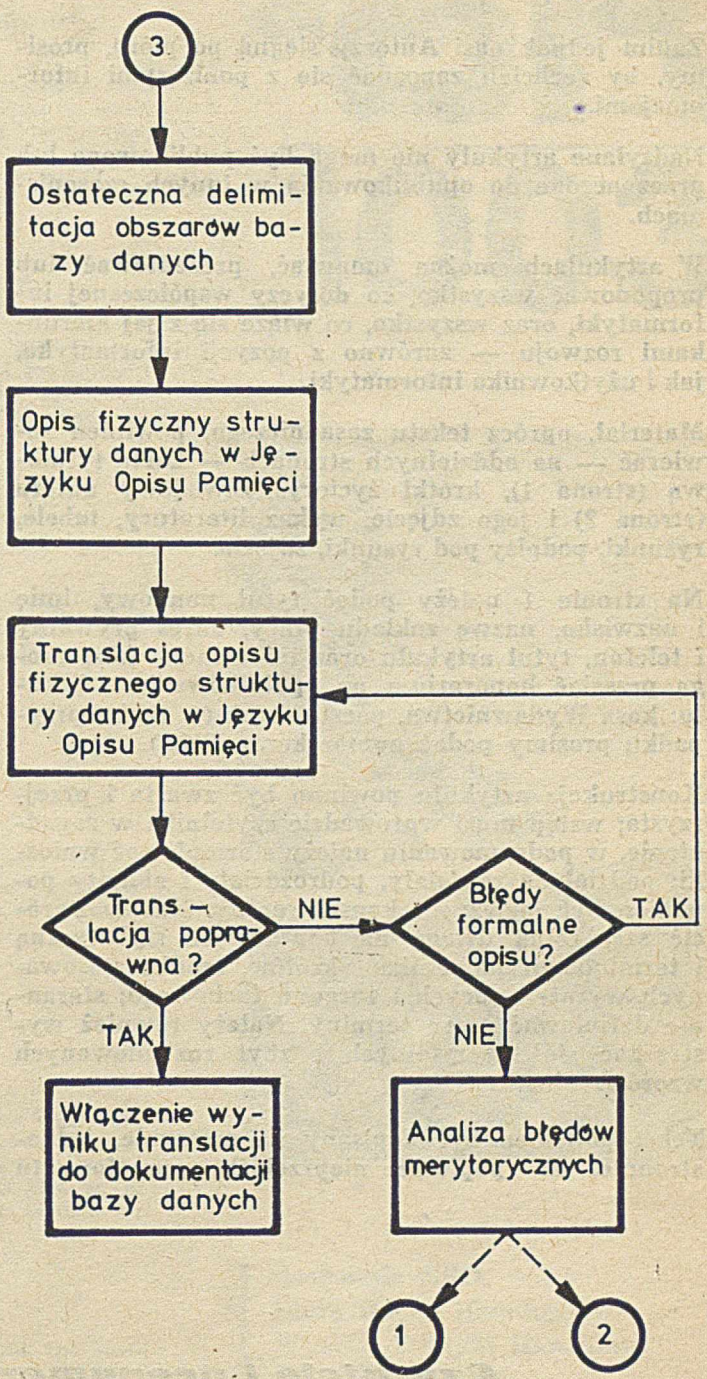
- w przypadku formalnych błędów opisu — korygujemy opis i dokonujemy ponownej translacji
- w przypadku ujawnienia wewnętrznych sprzeczności opisu jesteśmy zmuszeni do korekty projektu struktury pamięci; o ile sprzeczności dotyczą opisu obszarów, niezbędny jest powrót do ponownego wykonania czynności ustalenia długości strony oraz odpowiednio do dokonanych zmian wykonanie czynności następnych (rys. 2); o ile natomiast sprzeczności opisu dotyczą zbiorów strukturalnych bądź rekordów — nieunikniony jest najczęściej powrót do

[1]) W niniejszym artykule przyjęto, że mała długość strony oznacza stronę o długości 128—512 bajtów, średnia długość strony oznacza stronę o długości 512—1048 bajtów, duża długość strony oznacza stronę o długości ponad 1048 bajtów

wykonania czynności ustalenia długości rekordów bazy danych (rys. 1) i odpowiednio do dokonanych zmian powtórzenia praktycznie całego procesu projektowania struktury pamięci



Rys. 2. Projektowanie struktury pamięci — etap II: określenie stron i obszarów bazy danych



Rys. 3. Projektowanie struktury pamięci — etap III: weryfikacja projektu struktury pamięci

● w przypadku otrzymania bezbłędnego wyniku translacji — zwykle jednak dopiero w kolejnej iteracji — możemy włączyć poprawny wynik tej translacji do dokumentacji bazy danych.

Projektowanie struktury pamięci jest procesem, którego automatyzacja nie wydaje się być bliska. Mimo dość szerokiego zakresu komputerowego wspomagania tego procesu, wiele kluczowych decyzji, mających zasadniczy wpływ na efektywność zastosowań realizowanych w oparciu o bazę danych, pozostaje i pozostawać będzie jeszcze w gestii człowieka: Administratora Bazy Danych (aktualnie) czy też wyspecjalizowanego projektanta struktury pamięci, którego rolę opisuje przyszłościowa architektura SZBD—ANSI/X3/SPARC [2]).

[2]) ANSI/X3/SPARC Interim Report Study Group on Data Base Management System ANSI/SPARC February 1975.

Łamy INFORMATYKI otwarte dla wszystkich!

Zanim jednak nasi Autorzy sięgną po pióro, prosimy, by zechcieli zapoznać się z poniższymi informacjami.

Nadsyłane artykuły nie mogą być publikowane lub przeznaczone do opublikowania w innych czasopismach.

W artykułach można omawiać, prezentować lub proponować wszystko, co dotyczy współczesnej informatyki, oraz wszystko, co wiąże się z jej kierunkami rozwoju — zarówno z pozycji informatyka, jak i użytkownika informatyki.

Materiał, oprócz tekstu zasadniczego, powinien zawierać — na oddzielnych stronach — kartę tytułową (strona 1), krótki życiorys zawodowy autora (strona 2) i jego zdjęcie, wykaz literatury, tabele, rysunki, podpisy pod rysunki, zdjęcia.

Na stronie 1 należy podać tytuł naukowy, imię i nazwisko, nazwę zakładu pracy, adres prywatny i telefon, tytuł artykułu oraz informację, jaką drogą przesłać honorarium po opublikowaniu artykułu: kasa Wydawnictwa, poczta, bank (w takim przypadku prosimy podać numer konta PKO).

Konstrukcja artykułu powinno być zwarta i przejrzysta; wstęp musi wprowadzić czytelnika w zagadnienie, w podsumowaniu należy sformułować wnioski; podział na rozdziały, podrozdziały i akapity powinien być logiczny i konsekwentny. Należy zwrócić szczególną uwagę na poprawność stylistyczną i terminologiczną, unikać skrótów, rzadko stosowanych wyrażeń obcych i żargonu fachowego; starannie definiować nowe terminy. Należy również wystrzegać się nieczytelnych i zbyt rozbudowanych wzorów.

Tekst powinien być napisany na maszynie, jednostronnie, na papierze nieprzebitkowym formatu A-4, z marginesem 5 cm (30 wierszy na 1 stronie, 60 znaków w 1 wierszu).

Wykaz literatury powinien zawierać: kolejny numer pozycji (w nawiasie kwadratowym), nazwisko i imię autora, tytuł publikacji (książki lub artykułu), ewentualnie tytuł i numer czasopisma (w przypadku artykułu), miejsce i rok wydania.

Tabele — każda na oddzielnej stronie — powinny być numerowane i opatrzone tytułem oraz ściśle związane z tekstem (odniesienie na marginesie).

Rysunki — każdy oddzielnie (uwaga: nie wklejać rysunków w tekst!) — powinny być czytelne i również ściśle związane z tekstem (odniesienie na marginesie). Format rysunku nie może być mniejszy niż 10 × 10 cm.

Podpisy pod rysunkami, napisane również na oddzielnej stronie, oprócz kolejnego numeru powinny zawierać tytuł rysunku i ewentualnie legendę dotyczącą poszczególnych elementów.

Łączna objętość materiału nie powinna przekraczać w przypadku

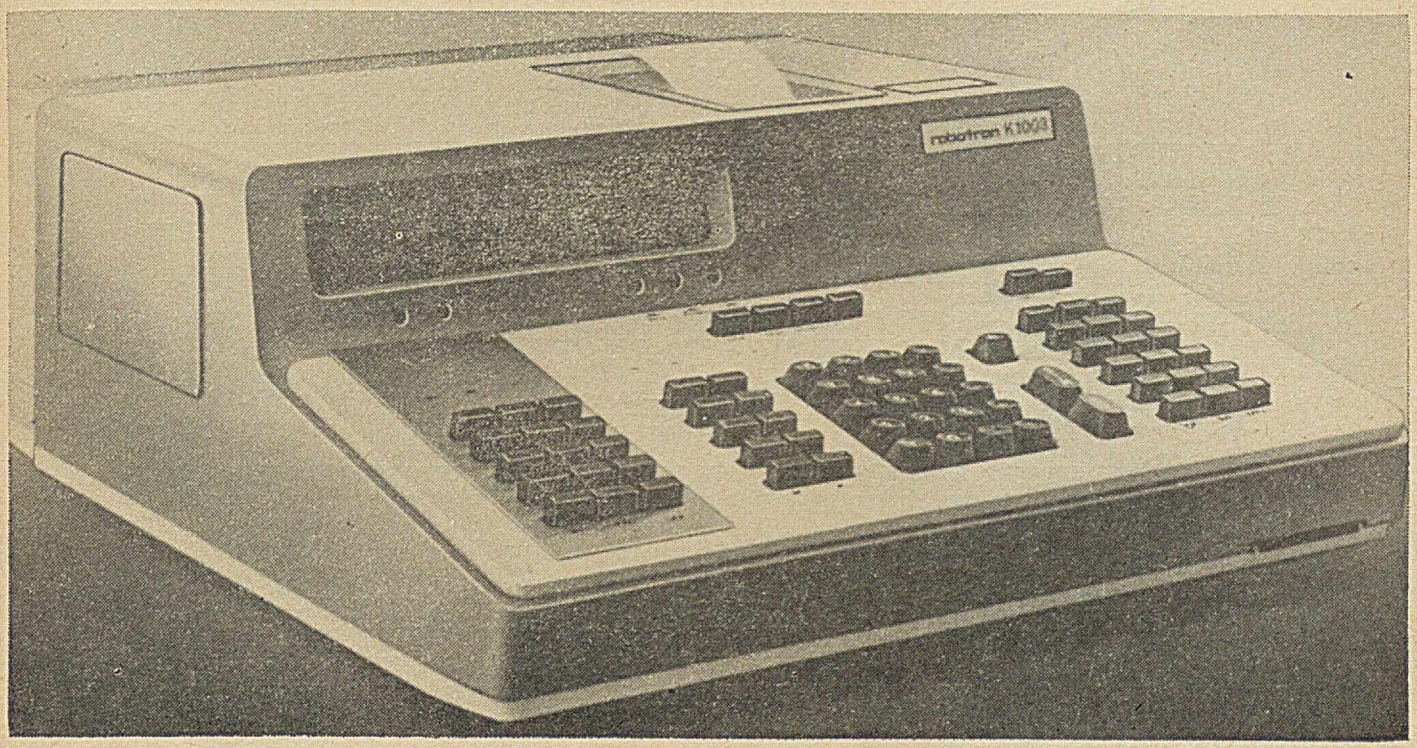
- artykułu problemowego — 12 stron
- reportażu — 8 stron
- recenzji, relacji z imprezy — 6 stron
- informacji — 4 stron maszynopisu

Tak przygotowany materiał prosimy dostarczyć w dwóch egzemplarzach pod adresem: redakcja INFORMATYKI, ul. Jasna 14/16, 00-041 Warszawa. Wszelkich dodatkowych informacji udzielamy pod telefonem 27-71-40.

Autor opublikowanego w INFORMATYCE artykułu otrzymuje bezpłatnie egzemplarz okazowy.

Materiałów nie zakwalifikowanych do druku redakcja nie zwraca.

robotron K 1003



Oto zalety, które można bardzo łatwo wymienić

1. Jest bardzo wydajnym minikomputerem z wieloma wbudowanymi funkcjami

2. Można go zaprogramować bez konieczności nauczenia się języka programowania

3. Zapamiętuje programy na kartach magnetycznych

4. Nawet bardzo specyficzne obliczenia można realizować za pomocą krótkiej sekwencji naciśnięć klawiszy

5. Wyposażony jest w układ automatycznego przełączania z obliczeń stałoprzecinkowych na zmiennoprzecinkowe

6. Do wyprowadzania wyników można w razie potrzeby zastosować drukarkę z zapisem na taśmie

Ambasada NRD
Biuro Radcy Handlowego
Wydział Polityki Handlowej
Dział Informatyki
i Maszyn Biurowych
Aleja I Armii
Wojska Polskiego 2-4
01-524 Warszawa

robotron

Robotron Export-Import
Przedsiębiorstwo
Handlu Zagranicznego
Niemieckiej Republiki
Demokratycznej
DDR 108 Berlin,
Friedrichstrasse 61

WCT/1057/K/79-B

SZÁMOK

Międzynarodowy Ośrodek Informacyjno-Szkoleniowy

ds. ETO (SZÁMOK)

H-1502 Budapest 112, pf. 146.,
Magyarország (Węgry) Telex: 224498

MIĘDZYNARODOWE KURSY W ZAKRESIE TECHNIKI OBLICZENIOWEJ W BUDAPESZCIE 1980

Kursy	Data	Język wykładowy
Oprogramowanie dla mikrokomputerów	5—9 maja	rosyjski
Zastosowania mikroprocesorów i mikrokomputerów	12—16 maja	rosyjski
Zastosowania minikomputerów	19—23 maja	rosyjski
Zastosowania statystyki	26—30 maja	rosyjski
Zastosowania minikomputerów	2—6 czerwca	angielski
Rozproszone systemy przetwarzania danych	9—13 czerwca	angielski
Pakiety oprogramowania dla skomputeryzowanego sterowania produkcją i gospodarką zapasami	22—26 września	angielski
Projektowanie bazy danych (ćwiczenia)	29 września—3 października	angielski
Strukturalna organizacja systemów (ćwiczenia)	6—10 października	angielski
Kierowanie projektowaniem skomputeryzowanych systemów informacyjnych (*Project Management*)	13—17 października	angielski
Projektowanie programów strukturalnych	20—24 października	angielski
Kierowanie pracami oprogramowania	27—31 października	angielski
Organizacja danych w systemie ES-DOS	10—14 listopada	rosyjski
Projektowanie bazy danych (ćwiczenia)	17—21 listopada	rosyjski

Ponadto Ośrodek organizuje inne kursy wspólnie z firmą Control Data Institute (USA) — język wykładowy: angielski.

Zainteresowanych takimi kursami prosimy wstawić krzyżyki w odpowiednie kwadraciki.

☐ Kursy SZÁMOK

☐ Kursy SZÁMOK-CDI

Dla słuchaczy uczestniczących w dwu lub kilku kolejnych kursach koszty zakwaterowania i utrzymania w czasie pomiędzy kursami pokryje SZÁMOK.

W celu otrzymania bliższych informacji oraz szczegółowego programu prosimy podać wymienione obok dane i przesłać pod adresem:

SZÁMOK

Tanfolyamszervezési csoport
H 1502—Budapest 112, pf. 146.

Imię i nazwisko .....

Stanowisko .....

Przedsiębiorstwo .....

Kraj ..... Kod pocztowy .....

Miasto ..... Ulica .....

Telex .....

WCT/1213/K/79