Nowe zasady prenumeraty!
Informacje wewnątrz numeru

7 1982

informatyka

WYDAWNICTWO NOT SIGMA

Do autorów

Od dłuższego już czasu nie zamieszczamy na łamach pojawiających się dawniej regularnie — odredakcyjnych zaleceń dla potencjalnych autorów. Doszliśmy bowiem do wniosku, że precyzyjne wskazówki dotyczące objętości, formy czy konstrukcji raczej zniechęcają do pisania. Brak takich informacji okazał się jednak również wyjściem nie najlepszym, autorzy bowiem często pytają o podstawowe warunki przyjęcia materiału do druku.

Spróbujemy zatem w mniej zawiły sposób przedstawić zasady, jakimi kierujemy się przy kwalifikacji artykułu oraz warunki formalne, jakie powinien spełniać maszynopis.

Podstawowym kryterium merytorycznym materiału (nie licząc głównego kryterium — oceny z punktu widzenia dotychczasowej wiedzy informatycznej) jest społeczna użyteczność zawartych w tekście rozważań i informacji. To w końcu oczywiste — jesteśmy powołani przede wszystkim do tego, aby służyć czytelnikom. W pierwszej kolejności liczyć się musi zatem interes społeczny. O tym winni zresztą pamiętać przede wszystkim autorzy.

Objętość tekstu powinna być wyznaczona przez społeczną wagę tematu, jego konstrukcja — przez wymóg czytelnego przekazu, zaś forma — przez dostosowanie języka i sposobu argumentacji do możliwości percepcyjnych przeciętnego odbiorcy. To truizmy, niemniej — jak praktyka wskazuje — warte przypomnienia. Autorzy najczęściej nie zadają sobie trudu, by wprowadzić czytelnika w zawiłości swoich specjalistycznych wywodów, nie biorą pod uwagę miejsca publikacji artykułu, tzn. charakteru naszego czasopisma.

Jeśli zaś chodzi o wymogi formalne — nie są duże. Co do objętości — maszynopis nie powinien przekraczać 12 znormalizowanych stron (30 wierszy z 60 znakami w wierszu). Jeśli jednak interesujący, zgodny z profilem pisma temat wymaga większej objętości, to oczywiście względy formalne nie stoją na przeszkodzie. Rysunki techniczne, dołączane do tekstu, są w redakcji przygotowywane przez wykwalifikowanego kreślarza, nie muszą być zatem dostarczane na kalce; ważne natomiast, aby były całkowicie czytelne, przejrzyste i zachowujące konieczne proporcje.

Pożądany jest również dodatkowy materiał ilustracyjny (kontrastowe zdjęcia czarno-białe, wydruki, itp.), jeśli wzbogaca on bądź precyzuje wywód autora.

Maszynopis powinien być nadsyłany w dwóch egzemplarzach (oryginał plus kopia), z załączeniem aktualnych adresów, a także numerów telefonów, umożliwiających szybki kontakt. Autorów artykułów problemowych, którzy nie zamieszczali swych prac w ciągu dwóch ostatnich lat, prosimy ponadto o załączenie zdjęcia oraz życiorysu.

Autorzy, którzy chcieliby nawiązać bliższy kontakt z Czytelnikami, proszeni są o zamieszczenie adnotacji, iż godzą się na podanie na naszych łamach swojego adresu bądź telefonu.

Redakcja

NACZELNA ORGANIZACJA TECHNICZNA

WYDAWNICTWO SIGMA

CZASOPISM I KSIĄŻEK TECHNICZNYCH

ul. Świętokrzyska 14a
00-950 Warszawa
skrytka pocztowa 1004

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pok. 131 i 133, tel. 27-71-40, dyżury redakcji 10.00—12.00

Zakł. Graf. „Tamka". Zam. 229. Obj. 4,5 ark. druk. Nakład 5000 egz. Z-47.

Cena egzemplarza zł 50.— INDEKS 36124 Prenumerata roczna zł 600.—

Kreczmar A., Salwicki A.: Język programowania LOGLAN. Część 1

INFORMATYKA 1982, nr 7, s. 4

Pierwsza część prezentacji nowego języka programowania LOGLAN opracowanego w Uniwersytecie Warszawskim. Podano zwięzłą charakterystykę języka oraz omówiono część jego konstrukcji odpowiadającą możliwościom języka ALGOL 60.

Кречмар А., Сальвицки А.: LOGLAN — язык программирования. Часть I

ИНФОРМАТИКА 1982, № 7, стр. 4

Первая часть представления нового языка программирования LOGLAN разработанного Варшавским Университетом. Представлено сжатую характеристику языка и обсуждено часть его конструкции, отвечающую возможностям языка ALGOL 60.

Bulandra M., Kwiatkowski M., Maj M.: Łączenie maszyn cyfrowych ODRA 1305/ICL 1904

INFORMATYKA 1982, nr 7, s. 9

Charakterystyka rozwiązań sprzętowych i programowych łączenia lokalnego i zdalnego komputerów ODRA 1305 oraz ICL 1904, zrealizowanych w Centralnym Ośrodku Informatyki Górnictwa.

Буляндра М., Квятковски М., Май М.: Слияние цифровых вычислительных машин ODRA 1305/ICL 1904

ИНФОРМАТИКА 1982, № 7, стр. 9

Характеристика оборудовательных и программных решений локального слияния и телекоммутации вычислительных машин ODRA 1305 и ICL 1904, осуществляемых в Центре информатики Министерства горной промышленности.

Zębala A.: Weryfikacja struktury logicznej bazy danych

INFORMATYKA 1982, nr 7, s. 12

Prezentacja wyników badań nad weryfikacją struktury logicznej bazy danych za pomocą analitycznej metody wyliczania liczby dostępów logicznych. Podano zwięzłą charakterystykę metody oraz sposób jej zastosowania w konkretnych warunkach eksploatacji, a także wnioski wynikające z przeprowadzonych badań.

Зембаля А.: Проверка истинности логической структуры базы данных

ИНФОРМАТИКА 1982, № 7, стр. 12

Представление результатов исследований проверки истинности логической структуры базы данных, с помощью аналитического метода перечисляния количества логических доступов. Указано сжатую характеристику метода и способ его употребления в конкретных условиях эксплуатации, а также предложения вытекающие из проведённых исследований.

Snowacka M., Ziębiński M.: Automatyczne przetwarzanie tekstów

INFORMATYKA 1982, nr 7, s. 15

Charakterystyka automatycznego przygotowania korespondencji i innych dokumentów handlowych, realizowanego w przedsiębiorstwie handlu zagranicznego POLIMEX-CEKOP. Omówiono zastosowany sprzęt, organizację prac, zagadnienia wdrożenia oraz uzyskane korzyści.

Сновацка М., Зембиньски М.: Автоматическая обработка текстов

ИНФОРМАТИКА 1982, № 7, стр. 15

Характеристика автоматической подготовки корреспонденции и других коммерческих документов, осуществляемого Внешнеторгового Предприятия Полимэкс — Цекоп. Охарактеризовано употребляемое оборудование, организацию труда, проблемы внедрения и полученные эффекты.

Smereczyński A.: Automatyzacja prac wydawniczych na IBM/370

INFORMATYKA 1982 nr 7, s. 18

Charakterystyka programu SCRIPT wchodzącego w skład oprogramowania firmowego komputerów serii IBM/370. Omówiono konstrukcję programu oraz sposób i możliwości jego wykorzystywania.

Смеречиньски А.: Автоматизация издательских работ для IBM/370

ИНФОРМАТИКА 1982, № 7, стр. 18

Характеристика программы SCRIPT, входящей в состав фирменного программного обеспечения вычислительных машин серии IBM/370. Определено конструкцию программы, способ и возможность её использования.

Zawiła-Niedźwiecki J.: CHANGE — uniwersalny pakiet korygowania zbiorów

INFORMATYKA 1982, nr 7, s. 20

Ogólna charakterystyka rozwiązań uniwersalnego pakietu do korygowania zbiorów przeniesionych na komputery Jednolitego Systemu. Pakiet ten został opracowany w oparciu o koncepcję języka DML oraz pakietu DMS w Centrum Elektronicznym Narodowego Banku Polskiego.

Завила-Недзвецки Я.: CHANGE — универсальный пакет корректировки массивов данных

ИНФОРМАТИКА 1982, № 7, стр. 20

Общая характеристика решений универсального пакета для корректировки массивов данных, перенесенных на вычислительные машины Однородной системы. Пакет этот был разработан опираясь на концепцию языка DML и пакета DMS в Электроническом центре Польского национального банка.

Świąć S., Weiner R.: DISCAR — preprocesor języka FORTRAN na maszynę cyfrową CYBER

INFORMATYKA 1982, nr 7, s. 23

Charakterystyka programu zwiększającego stopień wykorzystania komputera CYBER 72-16 w sytuacji stosowania programów napisanych w języku FORTRAN EXTENDED. Podano parametry eksploatacyjne oraz istniejące ograniczenia, a także wskazano na możliwość wykorzystania koncepcji DISCAR do opracowania podobnego programu dla innych wersji języka FORTRAN.

Свионць С., Вейнар Р.: DISCAR — препроцессор языка FORTRAN для вычислительной машины CYBER

ИНФОРМАТИКА 1982, № 7, стр. 23

Характеристика программы повышающей ступень использования вычислительной машины CYBER 72-16 при обстановке применяния программ написанных на языке FORTRAN EXTENDED. Представлено параметры эксплуатации и существующие ограничения, а также указано возможность использования концепции DISCAR для обработки подобной программы других вариантов языка FORTRAN.

Kreczmar A., Salwicki A.: The LOGLAN programming language. Part 1

INFORMATYKA 1982, No. 7, p. 4

First part of presentation of the LOGLAN, a new programming language, which was elaborated at Warsaw University. A concise characteristics of the language and a part of its structure, similar to possibilities of the ALGOL 60, are discussed.

Kreczmar J., Salwicki A.: Die Programmiersprache LOGLAN. Teil 1

INFORMATYKA 1982, Nr. 7, S. 4

Erster Teil der Vorstellung einer neuen Programmiersprache LOGLAN, die an der Warschauer Universität erarbeitet wurde. Es wurden eine kurze Charakteristik dieser Sprache und ein Teil ihrer Konstruktion, die den ALGOL 60-Möglichkeiten entspricht, angegeben.

Bulandra M., Kwiatkowski M., Maj M.: A connection of ODRA 1305 and ICL 1904 computers

INFORMATYKA 1982, No. 7, p. 9

Characteristics of hardware and software solutions for local and remote connection of ODRA 1305 and ICL 1904 computers, which were realized in the Central Data Processing Center of Mining Industry.

Bulandra M., Kwiatkowski M., Maj M.: Verbindung der ODRA 1305 und ICL 1904 Rechenanlagen

INFORMATYKA 1982, Nr. 7, S. 9

Eine Charakteristik der Hard- und Softwarelösungen von Lokal- und Fernverbindung zwischen ODRA 1305 und ICL 1904 Rechenanlagen, die im Zentralen EDV-Rechenzentrum der Bergbauindustrie realisert wurden.

Zębala A.: A data base logical structure verification

INFORMATYKA 1982, No. 7, p. 12

A presentation of data base logical structure research results, using analytic method of logical access counting. Concise characteristics of the method and the way of its application in real operation conditions, as well as conclusions from the research, are discussed.

Zębala A.: Verifikation der logischen Struktur von Datenbasis

INFORMATYKA 1982, Nr. 7, S. 12

Eine Vorstellung der Forschungsergebnisse über die Verifikation der logischen Struktur von Datenbasis mit Verwendung einer analytischen Methode zur Berechnung der logischen Zutritte. Es wurden eine kurze Charakteristik der Methode und ihre Anwendunsweise bei konkreten Betriebsbedingungen, sowie die aus den durchgeführten Forschungen gezogenen Beschlüsse, angegeben.

Snowacka M., Ziębiński M.: Automatic text processing

INFORMATYKA 1982, No. 7, p. 15

Characteristics of automatic preparation of correspondence and other commercial documents, which is realized in the foreign trade enterprise POLIMEX-CEKOP. Applied hardware, work organization, problems of implementation and achieved advantages are discussed.

Snowacka M., Ziębiński M.: Automatisierte Textverarbeitung

INFORMATYKA 1982, Nr. 7, S. 15

Eine Charakteristik der automatisierten Vorbereitung der Korrespondenz und anderer Handelsdokumenten, die im Aussenhandelunternehmen POLIMEX-CEKOP realisiert wird. Es wurden die angewendete Hardware, die Arbeitsorganisation, die Einsatzprobleme und die festgestellten Vorteile besprochen.

Smereczyński A.: Editing software of IBM/370

INFORMATYKA 1982, No. 7, p. 18

Characteristics of the SCRIPT editing program, which is the part of IBM/370 software offered by the manufacturer. Structure of the program, as well as the way and possibilities of its application, are discussed.

Smereczyński A.: Software für Redigierensarbeiten auf IBM/370

INFORMATYKA 1982, Nr. 7, S. 18

Eine Charakteristik des SCRIPT-Programms, das ein Teil der Firmensoftware von IBM/370 ist. Es wurden die Programmkonstruktion und Möglichkeiten der Anwendung dieses Programms besprochen.

Zawiła-Niedźwiecki J.: CHANGE — a universal program package for file correcting

INFORMATYKA 1982, No. 7, p. 20

General characteristics of a universal program package solution for correcting of files, which are transferred on Unified System computers. The package was elaborated in the Electronic Center of Polish National Bank on the base of DML language and DMS package.

Zawiła-Niedźwiecki J.: CHANGE — ein universelles Programmpaket zur Dateikorrektur

INFORMATYKA 1982, Nr. 7, S. 20

Eine allgemeine Charakteristik der Lösungen eines universellen Programmpakets zur Korrektur der Dateien, die auf die ESER-Rechenanlagen übertragen werden. Das Paket wurde im Elektronischen Zentrum der Nationalbank von Polen auf Grund der DML-Sprache und des DMS-Pakets-Konzeption erarbeitet.

Świąć S., Weinar R.: DISCAR — a FORTRAN preprocessor for CYBER computer

INFORMATYKA 1982, No. 7, p. 23

Characteristics of a program, which increases the capacity of CYBER 72-16 computer when programs are written in FORTRAN EXTENDED language. Operation parameters and actual limitations, as well as use possibility of the DISCAR concept for elaborating similar programs for other FORTRAN versions are presented.

Świąć S., Weinar R.: DISCAR — ein FORTRAN-Preprozessor für CYBER-Rechenanlage

INFORMATYKA 1982, Nr. 7, S. 23

Eine Charakteristik des Programms, das die Kapazität der CYBER 72-16 Rechenanlage steigert, wenn die Programme in FORTRAN-EXTENDED geschrieben wurden. Es wurden die Betriebsparameter und bestehende Begrenzungen angegeben, sowie eine Möglichkeit der Ausnutzung von DISCAR-Konzept zur Erarbeitung eines ähnlichen Programms für andere FORTRAN-Versionen, betont.

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 7
MIESIĘCZNIK
1 9 8 2
ROK XVII
Październik

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

W NUMERZE: Strona

Po zakończeniu cyklu artykułów o ADZIE zdecydowaliśmy się na zaprezentowanie kolejnego języka programowania. Jest nim LOGLAN, opracowany w Uniwersytecie Warszawskim.

Dlaczego właśnie ten język? Oto podstawowe przyczyny, które skłoniły nas do tego wyboru:

- semantyka LOGLANU jest bardzo bogata — jego możliwości są porównywalne z możliwościami ADY
- język ten ma nowoczesną i przejrzystą składnię, wzorowaną na składni PASCALA; nie odbiega ona od typowych przyzwyczajeń programistów, jest więc łatwa do opanowania
- kompilator języka jest dostępny na bardzo powszechnym w kraju minikomputerze MERA 400.

ANTONI KRECZMAR
ANDRZEJ SALWICKI
Instytut Informatyki
Uniwersytet Warszawski

Język programowania

Język programowania LOGLAN został opracowany w Instytucie Informatyki Uniwersytetu Warszawskiego. LOGLAN wykorzystuje osiągnięcia języków programowania należących do linii rozwojowej: ALGOL 60 — PASCAL — SIMULA 67, a także korzysta z doświadczeń ośrodków obliczeniowych.

Język wyposażono we wszystkie niezbędne narzędzia programowania. Starano się zwłaszcza zapewnić narzędzia dla definiowania struktur danych. Praktyka wskazuje, że w wielu przypadkach, ta sama koncepcja jest wykorzystywana wielokrotnie, np.: tablice rozproszone (ang. *hashing*), B-drzewa etc. Elementami bibliotek oprogramowania powinny być zatem nie tylko procedury, ale także pakiety reprezentujące strukturę danych. Pierwszym językiem, który umożliwiał taką organizację oprogramowania była SIMULA 67, wywierająca coraz większy wpływ na nowe koncepcje w programowaniu. SIMULA posiada jednak bardzo istotne ograniczenie: pakiet opisujący strukturę danych musi być zadeklarowany na tym samym poziomie struktury blokowej co instrukcja (blok) wykorzystująca strukturę danych. Próby takiego rozwiązania problemu, które odrzucałoby to ograniczenie, doprowadziły wielu ekspertów do przekonania, że nie istnieje rozwiązanie poprawne i zarazem efektywne. Stąd w językach CLU, MESA, ALPHARD, ADA i in. poczyniono odstępstwa od oryginalnej koncepcji simulowskiej i przyjęto rozwiązania efektywne, lecz o ograniczonym zasięgu — statyczne.

Istotnym elementem tej sprawy jest też niewątpliwie potrzeba takiej organizacji języka, która umożliwiałaby niezależną kompilację modułów. Jedynym do tej pory językiem, który taką niezależną kompilację umożliwiał, był FORTRAN. Jest to jednak język tylko nieco wyższego poziomu niż asembler, nie jest możliwe definiowanie w nim struktur danych. Ostatnio zagadnienie odrębnej kompilacji modułów zostało podniesione w związku z językiem ADA.

LOGLAN proponuje powrót do koncepcji składania deklaracji klas. Jest to czytelna dla programisty-użytkownika propozycja tekstowej kontatenacji typów. Umożliwia ona osiągnięcie wielu różnorodnych celów, m.in.:

- hierarchiczną organizację typów
- stopniowe rozwijanie oprogramowania z góry w dół, z wykorzystaniem przez wielu użytkowników wspólnych elementów (prefiksów) w wielu kontekstach
- tworzenie języków problemowo zorientowanych wg jednolitej zasady i z możliwością rozwijania wielu różnych gałęzi wywodzących się z jednego, wspólnego pnia
- wielokrotną reinterpretację funktorów i predykatów.

W Uniwersytecie Warszawskim udało się rozwiązać zadanie kompilacji modułów powiązanych w strukturę drzewiastą ze względu na prefiksowanie (uwaga — ta struktura drzewiasta nie jest tożsama z drzewiastą strukturą bloków w programie). Rozwiązanie okazało się efektywne (koszt najczęściej występującej operacji to zaledwie trzy instrukcje maszyny) i wnosi nowe propozycje do zasobu metod translacji. Dzięki wprowadzeniu składanych deklaracji typów, programiści uzyskują nie tylko narzędzie opisu struktur danych, lecz także możliwości wykonywania algebraicznych operacji na nich, m.in.: unii struktur, dzielenia struktur przez relacje kongruencji, definiowania rodzin struktur i wyróżniania podstruktur.

Nowe możliwości uzyskano dzięki dopuszczeniu typów jako parametrów. LOGLAN stał się w rezultacie bardzo mocnym narzędziem tworzenia systemów.

Współprogramy (ang. *coroutines*) wprowadzone do języka stwarzają nowe możliwości tworzenia zmodularyzowanych systemów. Wykorzystywane dotąd głównie dla zadań symulacji, współprogramy są wciąż nie docenianym narzędziem programowania. Istotną cechą LOGLANU jest możliwość potraktowania współprogramów jako elementów struktur danych. Z jednej strony uzyskujemy stąd wszystkie korzyści płynące ze stopniowego, hierarchicznego rozwoju systemu, rozwoju, który jest możliwy dzięki prefiksowaniu, z drugiej zaś — struktury danych uzyskują dodatkową dynamikę; ich elementami mogą być obiekty dynamiczne — tzn. współprogramy, a nie tylko rekordy.

LOGLAN jest wyposażony również w język obsługi procesów współbieżnych. Jego prostota zapewnia efektywną implementację. System ten jest wyposażony w kilka podstawowych operacji zawieszania i uaktywniania procesów o łatwej do zrozumienia, jednoznacznej semantyce. Dokonano formalnej analizy semantyki obliczeń współbieżnych w systemach wieloprocesorowych i przedstawiono zupełnie nowy ich model. Okazało się, że podział czasu nie może w sposób całkowicie wierny modelować obliczeń w systemach wieloprocesorowych. Bardzo istotną cechą LOGLANU jest zatem możliwość definiowania nowych struktur synchronizacji (np. monitory) lub komunikacji zadań, np. „podawania rąk" w ADZIE.

HISTORIA LOGLANU

Inspirację do podjęcia prac nad nowym językiem programowania pochodziły z dwóch źródeł: z przekonania o potrzebie nowego języka dla nowej wieloprocesorowej generacji maszyn, która już niedługo stanie się powszechna, oraz z dydaktycznej potrzeby ilustracji różnorodnych metod programowania w jednym spójnym języku programowania. Ten drugi aspekt ma też znaczenie praktyczne — być może uda się nam skonstruować jeden język dla wyrażenia całego oprogramowania danej maszyny.

LOGLAN zostanie przedstawiony w trzech odcinkach. Pierwszy z nich, publikowany poniżej, stanowi opis fragmentu języka odpowiadającego ALGOLOWI 60 (bez tablic). W kolejnym artykule zaprezentowane zostaną dalsze konstrukcje występujące w LOGLANIE: tablice dynamiczne, klasy i współprogramy oraz mechanizm prefiksowania. Ta część języka odpowiada w pewnym sensie SIMULI 67, jest jednak istotnie od niej bogatsza (np. o prefiksowanie wielopoziomowe). Wreszcie trzeci artykuł — zawierać będzie opis koncepcji programowania współbieżnego, obsługi sygnałów, gospodarki pamięcią oraz oddzielnej kompilacji modułów. (Red.)

LOGLAN. Część 1

Pierwszy etap prac został ukończony w 1977, gdy pojawił się raport języka LOGLAN 77 [2]. W etapie tym wykonano analizę istniejących narzędzi programowania, porównano wiele istniejących języków programowania, przeanalizowano semantykę struktur danych i funktorów programotwórczych. W tym ostatnim zakresie badań korzystano z wyników i metod uzyskanych w Uniwersytecie Warszawskim w związku z pracami dotyczącymi logiki algorytmicznej. LOGLAN 77 był zaprojektowany tak, by obejmował możliwie szeroki zakres narzędzi programistycznych, np. siedem sposobów przekazywania parametrów. Wniósł wiele nowych rozwiązań, np.: w pełni dynamiczne traktowanie tablic (nazwa tablicy nie jest związana z obiektem tablicowym na sztywno), nowy matematyczny model obliczeń współbieżnych. Ogłoszony raport stał się podstawą do podpisania porozumienia o budowie kompilatora LOGLANU 79 na minikomputerze MERA 400. Porozumienie to zawarły Zjednoczenie MERA i Uniwersytet Warszawski.

Drobiazgowa analiza projektu i programy użytkowe zapisane w LOGLANIE poprzedziły wstępny projekt implementacji. Był nim interpreter na komputerze CYBER. Zespół kierowany przez Antoniego Kreczmara rozpoczął prace wdrożeniowe od rozwiązania najważniejszego, a jednocześnie bardzo trudnego problemu poprawnej i taniej (nie tekstowej) implementacji składania deklaracji typów (prefiksowania), zawartych w różnych blokach. Mechanizm uogólnionego wektora DISPLAY został opisany i zbadany w [1]. Doświadczenia uzyskane w czasie prac nad interpreterem pozwoliły stwierdzić realność budowy kompilatora języka LOGLAN na minikomputerze MERA 400. Materiały uzyskane w trakcie pracy nad interpreterem wpłynęły na modyfikację składni języka LOGLAN. Prace nad kompilatorem rozpoczęto w kwietniu 1980 r. i zakończono w grudniu 1981 r., przedstawiając pierwszą wersję kompilatora. Obecnie (lipiec 1982) działa już nowa wersja, zawierająca m.in. obsługę sygnałów.

Kompilator współpracuje z systemem operacyjnym SOM-3, jest niemal całkowicie napisany w FORTRANIE i wymaga 40 K słów pamięci operacyjnej oraz ok. 120 K słów pamięci dyskowej. W Uniwersytecie Warszawskim jest wykorzystywany w konfiguracji dwuprocesorowej i trzyzadaniowej. Doświadczenie eksploatacyjne minionego półrocza to przeszło setka programów i systemów. Warto wspomnieć, że opracowano dwie wersje języka problemowo zorientowanego dla celów symulacji. Posługiwanie się tym językiem jest bardzo proste — wystarczy bowiem własny program symulacyjny poprzedzić zwrotem:
pref SIMULATION
który udostępnia użytkownikowi złożone struktury danych niezbędne dla symulacji. W ten sposób LOGLAN zapewnia te same możliwości co SIMULA.

OGÓLNA CHARAKTERYSTYKA JĘZYKA

Wyliczmy teraz podstawowe konstrukcje i cechy charakterystyczne języka LOGLAN:

- wygodny zestaw instrukcji strukturalnych
- modularna struktura (z możliwością rozszerzania i zagnieżdżania modułów)
- procedury i funkcje (z pełną rekurencją), a także procedury i funkcje jako parametry formalne
- tablice dynamiczne (rozmiary tablic wyznaczane są w trakcie wykonywania programu, dzięki czemu tablice wielowskaźnikowe mogą być różnych kształtów — trójkątne, pasmowe itp.)
- klasy (uogólnienie pojęcia rekordu), pozwalające definiować nowe typy i struktury danych
- współprogramy
- prefiksowanie — mechanizm pochodzący od SIMULI 67, znacznie w LOGLANIE uogólniony, który pozwala budować hierarchię typów i struktur danych, języki zorientowane problemowo itp.
- typy formalne pozwalające na parametryzację modułów (procedur, funkcji, klas, współprogramów i procesów) ze względu na dowolny, złożony typ danych
- mechanizmy ochrony, które zabezpieczają moduły przed niewłaściwym ich użyciem
- równoległość, łatwo adaptująca się do systemu operacyjnego i umożliwiająca programowanie współbieżne w sposób prosty i efektywny
- programowane „odśmiecanie" (ang. *garbage collection*) — narzędzie pozwalające w sposób skuteczny, a jednocześnie w pełni bezpieczny, realizować optymalną strategię zarządzania pamięcią przydzieloną użytkownikowi w trakcie wykonywania programu
- obsługa sygnałów, tzn. obsługa przerwań systemowych, takich jak przekroczenie zakresu liczb zmiennopozycyjnych lub przekroczenie zakresu indeksu tablicy, a także przerwań świadomie spowodowanych przez użytkownika
- oddzielna kompilacja.

Postaramy się teraz, z konieczności skrótowo, opisać i zilustrować na przykładach wyżej wymienione konstrukcje i cechy charakterystyczne języka.

INSTRUKCJE STRUKTURALNE

Instrukcje złożone w LOGLANIE konstruuje się z instrukcji pierwotnych (przypisania — np. $x:=y+0.5$, wywołania procedury — np. *call* $P(7,x+5)$ i in.) za pomocą instrukcji warunkowej, instrukcji pętli i instrukcji wyboru.

Instrukcja warunkowa ma postać:

```
if wyrażenie boolowskie then ciąg instrukcji
else ciąg instrukcji fi
```

natomiast instrukcja warunkowa bez części *else* ma postać:

```
if wyrażenie boolowskie then ciąg instrukcji fi
```

Semantyka instrukcji warunkowej jest standardowa. Użycie słowa kluczowego *fi* pozwala zagnieżdżać takie instrukcje bez narażania się na niejednoznaczność. Poszczególne instrukcje w ciągu instrukcji rozdzielane są średnikiem. (Średnik może także kończyć ciąg instrukcji i wówczas ostatnia instrukcja ciągu jest instrukcją pustą).

Niekiedy wygodnie jest rozłożyć wyrażenie boolowskie w instrukcji warunkowej na części w taki sposób, że znajomość wartości logicznej początkowych części wyrażenia wystarcza do wyznaczenia wartości całego wyrażenia, bez względu na wartość logiczną pozostałych części, a nawet bez względu na to, czy są one określone czy nie. W LOGLANIE konstrukcje takie wyraża się za pomocą instrukcji warunkowych z listą *orif* lub z listą *andif*. Instrukcja warunkowa z listą *orif* ma postć:

```
if wb1 orif wb2 ... orif wbk
then ciąg instrukcji
else ciąg instrukcji fi
```

i w pewnym sensie odpowiada instrukcji warunkowej. Wykonanie instrukcji warunkowej ze zwykłym wyrażeniem boolowskim wymaga jednak wyznaczenia wartości wszystkich składników alternatywy, nawet wówczas, kiedy jeden z początkowych składników jest już spełniony. Natomiast dla instrukcji warunkowej z listą *orif* proces wyznaczania wartości kolejnych składników alternatywy zostaje przerwany z chwilą natrafienia na pierwszy spełniony składnik. Na przykład — instrukcji:

```
if wb1 orif wb2 then I1 else I2 fi
```

odpowiada instrukcja:

```
if wb1 then I1 else
if wb2 then I1 else I2 fi
fi
```

Instrukcje z *orif* ułatwiają programowanie w sytuacji, gdy wyznaczenie wartości całego wyrażenia boolowskiego w instrukcji warunkowej może spowodować błąd semantyczny. Na przykład wykonanie instrukcji:

```
if i> n or A (i) = 0 then i:=i—1 else A(i) :=1 fi
```

gdy wartość *i* jest większa od *n* oraz gdy *A* jest tablicą o górnym zakresie wskaźnika *n*, spowoduje błąd semantyczny. Można wówczas zastosować instrukcję z *orif*:

```
if i>n orif A(i)=0 then i:=i—1 else A(i) :—1 fi
```

Pozwala to uniknąć błędu semantycznego przy wyznaczaniu wartości wyrażenia boolowskiego *A(i)=0*, a jednocześnie jest zgodne z naszym zamierzeniem.

Instrukcja warunkowa z listą *andif* ma postać:

```
if wb1 andif wb2 ... andif wbk
then ciąg instrukcji
else ciąg instrukcji fi
```

W powyższej instrukcji, z chwilą natrafienia na pierwszy niespełniony czynnik *wbj* ($1 \leqslant j \leqslant k$), proces wyznaczania wartości kolejnych czynników koniunkcji zostaje przerwany.

Instrukcja pętli w LOGLANIE ma postać:

```
do ciąg instrukcji od
```

Wykonanie instrukcji pętli polega na iterowaniu ciągu instrukcji zawartego pomiędzy *do* i *od* oraz kończy się z chwilą wykonania w takiej pętli instrukcji pierwotnej *exit*, np.

```
s:=1; t:=1; i:=1;
do
  i:=i+1; t:=t*x/i;
  if abs(t)<1.0E-10 then exit fi;
  s:=s+t
od;
```

Jeżeli pętle są zagnieżdżone i chcemy w wewnętrznej pętli zakończyć iterowanie obu pętli jednocześnie, to piszemy podwójne *exit*, np.

```
r,x:=0;
do
  s,t:=1; i:=1; x:=x+0.2;
  do
    i:=i+1; t:=t*x/i;
    if i> n then exit exit fi; (* jednoczesne
                                  zakończenie obu pętli *)
    if t< 1 then exit fi; (* zakończenie
                             tylko pętli wewnętrznej *)
    s:=s+t
  od
od
```

Podobnie, potrójne *exit* pozwala zakończyć trzy zagnieżdżone instrukcje pętli jednocześnie itd. W przykładzie tym użyliśmy także instrukcji jednoczesnego przypisania (np. *r,x:=0*), oraz komentarza, który rozpoczynamy lewym nawiasem i gwiazdką, a kończymy gwiazdką i prawym nawiasem.

Instrukcja pętli z początkowym warunkiem boolowskim ma postać:

```
while wyrażenie boolowskie do ciąg instrukcji od
```

i jest równoważna instrukcji:

```
do
  if not wyrażenie boolowskie then exit fi;
  ciąg instrukcji
od
```

Instrukcje pętli ze zmienną sterującą mają postać:

```
for j:=wa1 step wa2 to wa3 do ciąg instrukcji od
```

oraz

```
for j:=wa1 step wa2 downto wa3 do ciąg instrukcji od
```

Ich semantyka jest następująca. Wartość zmiennej sterującej *j*, poczynając od wartości *wa1* i z krokiem o wartości *wa2*, w pierwszej pętli jest zwiększana, a w drugiej zmniejszana. Wykonanie pierwszej pętli kończy się wtedy, gdy wartość zmiennej sterującej i staje się po raz pierwszy większa niż wartość *wa3*, a wykonanie drugiej — wtedy, gdy wartość zmiennej sterującej *j* staje się po raz pierwszy mniejsza niż wartość *wa3*. Wartości wyrażeń *wa1*, *wa2*, *wa3* definiujących początek, krok i koniec pętli są wyznaczane na początku, przed wejściem do pętli. Po wyjściu z pętli wartość zmiennej sterującej jest określona i równa się pierwszej wartości wychodzącej poza wyznaczony zakres. Wyrażenie *wa2* (wraz ze słowem kluczowym *step*) definiujące krok może być pominięte i wówczas wartość kroku jest równa *1*.

W instrukcji pętli może się pojawić instrukcja pierwotna repeat. Używamy jej wtedy, kiedy chcemy przerwać wykonanie kolejnej iteracji pętli i rozpocząć następną (odpowiednik skoku do *„continue"* kończącego instrukcję pętli *do* w FORTRANIE).

Na przykład:

```
i:=0; s:=0;
do
  i:=i+1;
  if A(i)<0 then repeat fi; (skok na koniec pętli,
                             w pętli pozostajemy *)
  if i>m then exit fi; (* zakończenie pętli *)
  s:=s+sqrt (A (i) );
od;
```

Pętla *for* lub *while* może być połączona w dowolny sposób z instrukcją *exit*. Podobnie *repeat* może występować

wewnątrz tych pętli i wówczas następna iteracja rozpoczyna się albo od zmiany wartości zmiennej sterującej (dla *for*), albo od sprawdzenia warunku boolowskiego (dla *while*).

W języku LOGLAN istnieje także instrukcja z wyborem (*case*).
W swojej najprostszej postaci wygląda ona następująco:

```
case wa
  when Li : I1
  when L2 : I2
  ...
  when Lk : Ik
  otherwise I
esac
```

gdzie *wa* jest wyrażeniem, *Li,...,Lk* są stałymi całkowitymi lub znakowymi, a *I1,...,Ik, I* są ciągami instrukcji.
Jeżeli wartość wyrażenia *wa* równa się *Lj*, dla pewnego *j*, $1 \leqslant j \leqslant k$, to wykonuje się ciąg instrukcji *Ij*, w przeciwnym razie wykonuje się ciąg instrukcji występujący po *otherwise*.

STRUKTURA MODULARNA

Struktura modularna języka LOGLAN jest uzyskana dzięki szerokiemu zestawowi środków do tworzenia modułów oraz łączenia ich w większe jednostki. Modułem jest blok, procedura, funkcja, klasa, współprogram lub proces. Najprostszym rodzajem modułu jest blok. Jego struktura syntaktyczna jest następująca:

```
block ciąg deklaracji
begin ciąg instrukcji
end
```

Słowo kluczowe *begin* rozpoczyna ciąg instrukcji. (Słowo to można pominąć, jeżeli ciąg instrukcji jest pusty). W ciągu deklaracji definiujemy jednostki syntaktyczne (stałe, zmienne, inne moduły itp.), których zakresem działania jest ten blok. Jednostki te są identyfikowane w ciągu instrukcji poprzez nazwy (identyfikatory). Nie nakłada się żadnych ograniczeń na kolejność występowania poszczególnych deklaracji w ciągu deklaracji bloku.

Każda lista deklaracji zmiennych rozpoczyna się od słowa kluczowego *var*, każda lista deklaracji stałych rozpoczyna się od słowa kluczowego *const*. Wyrażenie definiujące stałą musi mieć taką strukturę, aby jego wartość można było wyliczyć w trakcie kompilacji.
Program w języku LOGLAN może być blokiem lub też może mieć taką samą strukturę co blok, po zastąpieniu słowa kluczowego *block* sekwencją:

```
program nazwa
```

Wówczas cały program może być identyfikowany poprzez tę nazwę (np. po zakatalogowaniu binarnej wersji programu w bibliotece użytkownika lub bibliotece systemowej).

Blok jest instrukcją, a zatem w ciągu instrukcji bloku może pojawić się także inny blok. Wykonanie instrukcji bloku polega na zarezerwowaniu w pamięci komputera odpowiedniego pola, a następnie wykonaniu ciągu instrukcji tego bloku. W polu tym rozmieszczone są jednostki syntaktyczne zadeklarowane w bloku. Po wykonaniu instrukcji bloku, pole to zostaje zwolnione i może być użyte do innych celów.

Struktura modularna języka funkcjonuje w pełni dopiero wówczas, kiedy używamy także innych rodzajów modułów, a nie tylko bloków. W szczególności dotyczy to możliwości zagnieżdżania modułów. (Różne rodzaje modułów omówimy pełniej w następnych punktach).

Zagnieżdżanie modułów pozwala tworzyć ich hierarchię, a także zwiększa bezpieczeństwo programowania dużych zadań.

PROCEDURY I FUNKCJE

Procedury i funkcje są ogólnie znanymi narzędziami programowania. Ich składnia jest zbliżona do PASCALOWEJ, np.

```
unit Newton: function (n,m:integer) : integer;
var i:integer;
begin
  if m>n then return fi;
  result:=n;
  for i:=2 to m do result:=result*(n—i+1) div i od
end Newton;
```

Nagłówek funkcji lub procedury rozpoczyna słowo kluczowe *unit*. Po identyfikatorze modułu, słowie kluczowym *procedure* lub *function*, wykazie parametrów oraz deklaracji typu (tylko dla funkcji), występuje ciąg deklaracji lokalnych tego modułu. Słowo kluczowe *begin* rozpoczyna ciąg instrukcji modułu, tak jak w bloku. Wartością funkcji jest wyjściowa wartość standardowej zmiennej *„result"*, *implicite* zadeklarowanej w każdej funkcji. Zmienna ta jest typu deklarowanej funkcji. Można ją również używać jako lokalnej zmiennej roboczej.

Instrukcja *return* w procedurze lub funkcji powoduje zakończenie wykonywania modułu i powrót do miejsca jego wywołania. (Jeżeli *return* nie występuje *explicite*, to powrót wykonywany jest po natrafieniu na końcowy symbol *end* danego modułu). Jednocześnie pole tego modułu jest usuwane z pamięci (tak jak pole zakończonego bloku).

Ponieważ wszystkie zmienne lokalne każdego modułu są standardowo inicjowane (*real* na *0,0*, *integer* na *0*, *boolean* na *false*, *char* na symbol pusty), zatem w przykładzie funkcji *Newton* dla $m>n$ wartość funkcji będzie *0* (gdyż inicjalizacja dotyczy także zmiennej *result*).

Po końcowym *end* procedury lub funkcji (a także wszystkich innych modułów z wyjątkiem bloku) może wystąpić nazwa. Translator dokona wówczas kontroli, czy ten symbol end odpowiada właściwemu modułowi. Kontrola ta ułatwia programiście szybkie znajdowanie błędów wynikających z niewłaściwego ustawienia symboli kluczowych *unit*, *begin* lub *end* w programie.

Wywołanie funkcji polega na użyciu jej identyfikatora w wyrażeniu z odpowiednim wykazem parametrów aktualnych, np.:

```
i:=1 — Newton (n,m+1) ;
```

Wywołanie procedury jest to instrukcja postaci np.:

```
call P(x,y+3) ;
```

Parametrem formalnym modułu może być zmienna, procedura, funkcja lub typ. Parametr formalny, który jest zmienną, jest zwykłą lokalną zmienną modułu. W języku LOGLAN istnieją trzy podstawowe sposoby transmisji takich parametrów. Rozróżniamy mianowicie parametry wejściowe (*input*), parametry wyjściowe (*output*) oraz parametry wejściowo-wyjściowe (*inout*). Przyjmuje się standardowo jako rodzaj transmisji *input*.
Na przykład — w nagłówku:

```
unit p: procedure (x,y:real, b:boolean;output c:char,
                                  i:integer; inout j:integer);
```

x,y,b są parametrami wejściowymi, *c,i* są parametrami wyjściowymi, natomiast *j* jest parametrem wejściowo-wyjściowym.

W LOGLANIE nie wprowadzono żadnych ograniczeń na stosowanie rekurencji. Nie zachodzi w szczególności konieczność wstępnego sygnalizowania nagłówka modułu przed wystąpieniem pełnej jego deklaracji (w odróżnieniu od PASCALA, gdzie taka zapowiedź jest wymagana), w przypadku wywołań wzajemnie rekurencyjnych takich modułów.

Na przykład dla ciągów określonych wzorami:

$a(n) = b(n-1)+n+2 \qquad n>0$

$b(n) = a(n-1)+(n-1) * n \qquad n>0$

$a(0) = b(0) = 0$

można napisać dwie funkcje:

```
unit a: function (n:integer):integer;
begin
  if n>0 then result:=b(n—1) +n+2 fi
end a;
unit b: function (n:integer):integer;
begin
  if n>0 then result:=a(n—1)+(n—1) * n fi
end b;
```

i wywołać np.:

```
k:=a(100) * b(50)+a(15);
```

LOGLAN dopuszcza także w sposób nieograniczony procedury i funkcje jako parametry formalne. Na przykład:

```
unit Bisec: procedure(a,b,eps;real; output x:real;
                      function f(x:real):real); ...;
```

W powyższym nagłówku procedury, po parametrach wejściowych *a, b, eps* oraz parametrze wyjściowym *x*, występuje parametr funkcyjny *f*. Zauważmy, że parametr ten występuje wraz z pełnym nagłówkiem. Umożliwia to wykonanie pełnej kontroli zgodności parametru aktualnego z parametrem formalnym w trakcie translacji programu (a nie w trakcie wykonania).

Możliwość taka ze względów syntaktycznych znika w momencie wprowadzenia procedury formalnej jako parametru procedury formalnej. Sytuacja taka może pojawić się rzadko, niemniej w języku LOGLAN konstrukcja taka jest dopuszczalna. Nie piszemy wówczas parametru formalnego wraz z całym nagłówkiem, a jedynie sam identyfikator. Pełna kontrola będzie natomiast dokonana w trakcie wykonywania programu. Syntaktycznie taka konstrukcja może wyglądać następująco:

```
unit P:procedure (i:integer; procedure G
                                (:integer; procedure H));
  ...
begin
  ...
  call G (j,P);
end P;
```

W powyższym przykładzie *G* jest parametrem proceduralnym pierwszego rzędu, ma zatem pełny nagłówek. Natomiast *H* jest parametrem proceduralnym drugiego rzędu, tzn. jest parametrem formalnym parametru formalnego w związku z czym występuje już bez nagłówka.
Wywołanie takiej procedury może mieć silnie rekurencyjną postać np.:

```
call P(i+10,P);
```

LITERATURA

[1] Bartol W. M., Kreczmar A., Litwiniuk W., Oktaba H.: Implementation of Prefixing at Many Levels. IIUW Report nr 94, Warszawa, 1980

[2] Müldner T., Oktaba H., Ratajczak W., Salwicki A.: Raport języka LOGLAN. Warszawa, IMM, 1977.

Konferencje specjalistyczne IFIP

Obserwatora coraz bogatszego kalendarza międznarodowych konferencji informatycznych zaskoczyło niezwykłe wzmożenie w tej dziedzinie aktywności Międzynarodowej Federacji Przetwarzania Informacji IFIP. Oto mimo konieczności ogromnej koncentracji wysiłków związanych z przygotowaniem przyszłorocznego jubileuszowego Światowego Kongresu IFIP w Paryżu (szczegółowe informacje na temat tej imprezy zamieściliśmy w poprzednim numerze INFORMATYKI), organizacja ta zapowiedziała w 1983 r. dwie międzynarodowe konferencje specjalistyczne. Fakty te świadczą, że po prawie ćwierćwieczu istnienia IFIP przeżywa jakby drugą młodość. Można przypuszczać, że aktywność organizacja zawdzięcza nowemu kierownictwu, które w ten sposób pragnie zaakcentować i ugruntować dotychczasową pozycję IFIP wśród coraz liczniejszego grona międzynarodowych organizacji informatycznych.

Obie wspomniane konferencje specjalistyczne anonsowane są jako pierwsze (pod auspicjami IFIP), co oczywiście nie przeczy faktom, że już w latach ubiegłych inne organizacje informatyczne organizowały imprezy międzynarodowe o identycznej tematyce.

Pierwszą z anonsowanych przez IFIP imprez przyszłorocznych jest konferencja na temat zastosowań informatyki w administracji publicznej. Odbędzie się ona w Wiedniu w dniach 23—25 lutego 1983 r. pod hasłem „Wpływ nowej technologii na systemy informatyczne administracji publicznej w latach osiemdziesiątych". Bezpośrednim organizatorem tej konferencji jest stowarzyszenie informatyków austriackich ADV, zaś jej stronę merytoryczną przygotowuje 24-osobowy Komitet Programowy, skupiający przedstawicieli 20 krajów.

Zlokalizowane na terenie Politechniki Wiedeńskiej obrady obejmować będą posiedzenia plenarne, 4 sesje tematyczne oraz 2 dyskusje panelowe. Trzy spośród sesji tematycznych ukierunkowano wg profilu zawodowego, a więc zainteresowań różnych grup uczestników (dla działaczy administracji publicznej, dla użytkowników, dla informatyków), natomiast sesja czwarta będzie przeznaczona na prezentację przykładów konkretnych zastosowań. Opublikowany szczegółowy program konferencji przewiduje wygłoszenie w ciągu trzech dni 4 referatów głównych na posiedzeniach plenarnych i 32 referatów na sesjach tematycznych.

Szczegółową treść konferencji najlepiej scharakteryzuje przytoczenie pełnych tytułów referatów posiedzeń plenarnych („Wpływ nowej technologii na systemy informacyjne", „Wpływ ustawodawstwa informatycznego na przetwarzanie informacji w administracji publicznej", „Polityka w zakresie zaawansowanej technologii przetwarzania informacji w ramach EWG", „Wpływ nowej technologii na organy władzy i społeczeństwo"); a także tematów dyskusji panelowych („Współpraca międzynarodowa polityków, kadry kierowniczej i informatyków w dziedzinie administracji publicznej", „Jak administracja publiczna powinna reagować na ustawodawstwo informatyczne?").

Drugą z zapowiedzianych imprez będzie konferencja na temat bezpieczeństwa przetwarzania. Konferencja ta o kryptonimie IFIP/SEC'83 (Security Conference) odbędzie się w dniach 16—19 maja 1983 r. w Sztokholmie. Organizatorami są tym razem stowarzyszenie informatyków szwedzkich SSI, Szwedzka Agencja ds. Rozwoju Administracji oraz szwedzka filia firmy HONEYWELL-BULL.

Głównymi celami konferencji są: poparcie dla rozwoju specjalizacji w zakresie bezpieczeństwa przetwarzania jako wyodrębnionego zawodu, zaakcentowanie znaczenia bezpiecznego przetwarzania oraz ochrony sfery prywatnej człowieka dla jednostek organizacyjnych oraz społeczeństwa oraz stworzenie międzynarodowego forum dla omówienia problemów bezpieczeństwa przetwarzania.

Tematykę konferencji podzielono na dwa następujące obszary problemowe:

1. Problemy bezpieczeństwa (Security Problems) — polityka i organizacja, zagadnienie jakości, kontrola przetwarzania, szkolenie personelu ds. bezpieczeństwa przetwarzania, bezpieczeństwo i architektura systemów, przepływ danych poza granice kraju.
2. Metody zabezpieczeń (Techniques for Protection) — zabezpieczenia kryptograficzne, narzędzia identyfikacji, oprogramowanie zabezpieczające, ochrona urządzeń, przewidywanie zniszczeń i odtwarzanie informacji, metody zabezpieczeń w warunkach przetwarzania rozproszonego.

MAGDALENA BULANDRA
MAREK KWIATKOWSKI
WALDEMAR MAJ
Centralny Ośrodek Informatyki Górnictwa
Katowice

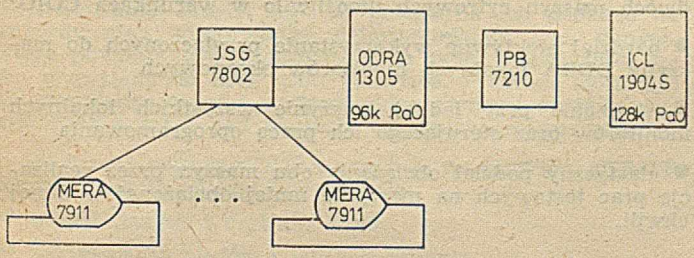
Łączenie maszyn cyfrowych ODRA / ICL 1904

Sprzęt komputerowy eksploatowany obecnie w ośrodkach obliczeniowych przemysłu węglowego jest w zasadzie jednolity i obejmuje dużą liczbę autonomicznie pracujących maszyn serii ODRA 1300 i kompatybilnych z nimi programowo maszyn serii ICL 1900. W celu efektywnego wykorzystania sprzętu poszczególnych ośrodków konieczne jest wzajemne łączenie wspomnianych maszyn. Prace w tym zakresie prowadzone są w Centralnym Ośrodku Informatyki Górnictwa (COIG) w Katowicach. Obejmują one m.in. tworzenie lokalnych i zdalnych połączeń maszyn ODRA 1305 i ICL 1904S. W niniejszym artykule przedstawiono charakterystyki oprogramowania komunikacyjnego dla obu typów połączeń.

POŁĄCZENIE LOKALNE MASZYN ODRA 1305 I ICL 1904S

W odróżnieniu od rozwiązania prezentowanego w [1] rozważa się tu połączenie lokalne maszyn ODRA 1305 i ICL 1904S, pracujących pod nadzorem systemu operacyjnego GEORGE-3. Zostało ono zrealizowane przy wykorzystaniu bufora międzyprocesorowego firmy ICL IPB 7210 [2] (polski odpowiednik nosi nazwę ADM 305/1. [1, 3], umożliwiającego przesyłanie danych z szybkością 100 K znaków/s.

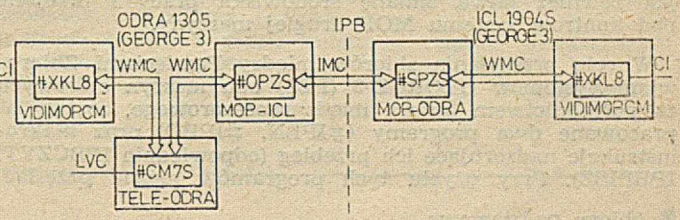
W oparciu o bazę techniczną połączenia, przedstawionego na rysunku 1, stworzono w COIG oprogramowanie komunikacyjne wykorzystujące własności tzw. kanałów komunikacji międzymaszynowej (skrót ang. IMC) oraz wewnątrzmaszynowej (skrót ang. WMC). Oznacza to, że dwa programy rezydujące w różnych maszynach cyfrowych mogą wymieniać dane za pośrednictwem bufora przy wykorzystaniu kanałów IMC. Wymiana danych między dwoma programami znajdującymi się w jednej maszynie cyfrowej odbywa się przy wykorzystaniu kanałów WMC. Przesyłanie danych za pośrednictwem obu typów kanałów jest podobne i odbywa się przy użyciu instrukcji PERI dla urządzenia typu 28. Szczegóły dotyczące postaci instrukcji PERI dla kanałów WMC i IMC znaleźć można w pracach [1, 5].



Rys. 1. Konfiguracja sprzętowa połączenia lokalnego maszyny cyfrowej ODRA 1305 z maszyną cyfrową ICL 1904S

Przygotowane oprogramowanie komunikacyjne przeznaczone jest do eksploatacji w warunkach systemu operacyjnego GEORGE-3. Umożliwia ono:

- wykorzystywanie lokalnych monitorów ekranowych maszyny ODRA 1305 do pracy w systemie MOP jednej spośród dwóch maszyn
- przesyłanie zbiorów między Pamięciami Zbiorowymi Systemów GEORGE-3 (PZS) obu maszyn.



Rys. 2. Struktura oprogramowania umożliwiającego pracę monitorów lokalnych w systemie MOP jednej z dwu maszyn cyfrowych LVC — sterownik monitorów lokalnych, CI — wydawca komend

Strukturę oprogramowania realizującego pierwszą z wymienionych możliwości prezentuje rysunek 2. W rozwiązaniu tym wykorzystano:

- program obsługi jednostki sterującej monitorów lokalnych CM7S, wygenerowany z pakietu COMMUNICATIONS MANAGER firmy ICL (8K) [4]
- standardowy program #XKL8 firmy ICL udostępniający funkcje MOP lokalnym monitorom ekranowym (3K)
- własne programy #OPZS i #SPZS (każdy po 2K) umożliwiające wymianę danych pomiędzy programami # CM7S a #XKL8 za pośrednictwem bufora ICL 7210 znajdującego się po stronie ICL 1904S.

Mgr inż. MAGDALENA BULANDRA ukończyła w 1977 r. studia na Wydziale Automatyki i Informatyki Politechniki Śląskiej. Obecnie zatrudniona jest w Centralnym Ośrodku Informatyki Górnictwa w Katowicach, gdzie zajmuje się oprogramowaniem standardowym na potrzeby teletransmisji.

Mgr inż. MAREK KWIATKOWSKI ukończył studia na Wydziale Automatyki i Informatyki Politechniki Śląskiej. Od 1978 r. pracuje w Centralnym Ośrodku Informatyki Górnictwa w Katowicach, gdzie zajmuje się problematyką sieci komputerowych.

Powyższe programy są uruchamiane i nadzorowane przez makroinstrukcje o nazwach podanych na rysunku 2.

W przypadku autonomicznej pracy maszyny ODRA, programem umożliwiającym pracę monitorów lokalnych w systemie MOP jest program standardowy firmy ICL o nazwie # XKL4. Łączy się on z systemem operacyjnych GEORGE-3 za pomocą mechanizmu tzw. programowego wydawcy komend (ang. *Command Issuer-CI*). Z drugiej strony program # XKL4 nadzoruje pracę jednostki sterującej monitorów lokalnych umożliwiając jej współużytkowanie innym programom znajdującym się w tym samym czasie w pamięci operacyjnej. W opisywanym rozwiązaniu nastąpił podział powyższych funkcji. Dostęp do systemu operacyjnego GEORGE-3 realizuje program # XKL8, natomiast zarządzanie jednostką sterującą przejął program # CM7S, który dodatkowo na możliwość łączenia się (przy użyciu kanałów WMC) z kilkoma tzw. programami użytkowymi np. # XKL8. Każdy monitor można niezależnie przydzielić do wybranego programu użytkowego oraz odłączyć i przydzielić do innego programu użytkowanego poprzez wysłanie z monitora do programu # CM7S komunikatów o standardowej postaci.

Jak pokazano na rysunku 2, program # CM7S łączy się z dwoma programami # XKL8, z których jeden działa na maszynie ODRA 1305, a drugi na ICL 1904S. Ze względu na pewne różnice we własnościach kanałów WMC i IMC nie można bezpośrednio łączyć programów # CM7S i # XKL8 działających na różnych maszynach cyfrowych. Połączenie to zostało zrealizowane przy wykorzystaniu dwóch programów pośredniczących # OPZS i # SPZS. Dodatkową funkcję tych programów jest wykrywanie i eliminacja niektórych typów błędów związanych z pracą bufora ICL 7210. Przedstawione rozwiązanie umożliwia użytkownikowi lokalnego monitora ekranowego pracę pod kontrolą systemu MOP jednej z dwu maszyn cyfrowych, jak również łatwą zmianę środowiska pracy i przejście pod kontrolę systemu MOP drugiej maszyny.

W celu przesyłania zbiorów między Pamięciami Zbiorowymi Systemów GEORGE-3 (PZS) obu maszyn cyfrowych za pośrednictwem bufora międzyprocesorowego, zostały opracowane dwa programy (#IPBN, #IPBO) oraz makroinstrukcje nadzorujące ich przebieg (odpowiednio IPBCZYT, IPBPISZ). Przy użyciu tych programów można przesłać:

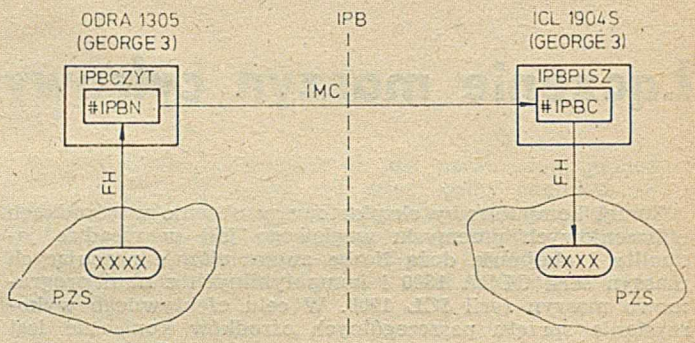
- zbiory podstawowe
- zbiory amorficzne
- programy binarne typu dyskowego.

Pierwszy z wymienionych programów (#IPBN) jest programem nadawczym, który czyta dane ze zbioru w PZS i wysyła je do bufora ICL 7210. Funkcją drugiego programu (#IPBO) jest odbieranie danych z bufora i zapisywanie ich do zbioru w PZS drugiej maszyny. Programy te mają mechanizm obsługi sytuacji błędnych związanych z pracą bufora. Operacje (czytania i zapisywania) na zbiorach PZS odbywają się przy wykorzystaniu manipulatora zbiorów (ang. *File Handler — FH*).

Schemat oprogramowania dla przykładowej transmisji zbioru XXXX z PZS maszyny ODRA 1305 do PZS maszyny ICL 1904S przedstawiono na rysunku 3. Ogólna postać wywołania makroinstrukcji nadzorujących pracę #IPBN i #IPBO jest następująca:

```
IPBCZYT %A, %B, %C .....
IPBPISZ %A, %B, %C .....
```

gdzie:
%A — nazwa kanału IMC (taka sama dla obu makroinstrukcji)
%B, %C — nazwy przesyłanych zbiorów.



Rys. 3. Schemat oprogramowania dla transmisji zbioru

W wyniku następującego, przykładowego wywołania powyższych makroinstrukcji:

```
IPBCZYT * LINK, # SOURCEPALFA, # PROGRAM
BETA
IPBPISZ * LINK, # SOURCEPALFA, # PROGRAM
GAMA
```

zostanie ustanowiony kanał IMC o nazwie LINK, za pośrednictwem którego realizowana będzie kolejno transmisja zbiorów SOURCEPALFA oraz PROGRAM BETA z PZS jednej maszyny cyfrowej do PZS drugiej maszyny cyfrowej, gdzie zbiory te zostaną zapisane odpowiednio pod nazwami SOURCEPALFA i PROGRAM GAMA.

Istnieje możliwość prowadzenia równoczesnych transmisji przez kilka kanałów IMC. W tym celu dla każdego z kanałów należy uruchomić parę makroinstrukcji IPBCZYT i IPBPISZ, przy czym nazwy poszczególnych kanałów muszą być różne. Obecnie prowadzone są prace nad oprogramowaniem, które umożliwi przesyłanie zbiorów taśmowych i dyskowych.

W porównaniu z pracą autonomiczną, opisane połączenie dwóch maszyn cyfrowych umożliwiło w warunkach COIG:

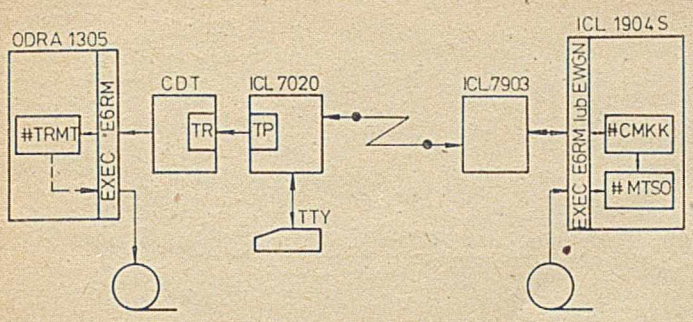
- bardziej efektywne wykorzystanie przyłączonych do maszyny ODRA lokalnych monitorów ekranowych
- skupienie przy jednej maszynie wszystkich lokalnych monitorów oraz sterującego ich pracą oprogramowania
- elastyczny podział obciążenia obu maszyn przez realizację prac testowych na maszynie mniej obciążonej w danej chwili.

POŁĄCZENIE ZDALNE MASZYN ODRA 1305 I ICL 1904S

W COIG zaprojektowano i wykonano elektroniczne połączenie stacji abonenckiej 7020 stanowiącej zdalny terminal maszyny ICL 1904S, z urządzeniem peryferyjnym CDT (czytnik — perforator taśmy papierowej) maszyny ODRA 1305. Połączenie to polega na sprzężeniu czytnika taśmy papierowej (TR) urządzenia CDT z perforatorem taśmy papierowej (TP) stacji 7020, co stwarza możliwość przesyłania danych z maszyny ICL 1904S do maszyny ODRA 1305.

Konfigurację sprzętową omawianego połączenia — z uwzględnieniem elementów oprogramowania, jakie zostało opracowane dla celów transmisji zbiorów — przedstawia rysunek 4. Oprogramowanie to umożliwia przesyłanie zbiorów z maszyny ICL 1904S znajdującej się w COIG, do maszyn typu ODRA zainstalowanych w ośrodkach obliczeniowych posiadających zdalne stacje abonenckie ICL 7020 połączone z maszyną ICL 1904S za pośrednictwem procesora komunikacyjnego ICL 7903.

Mgr inż. WALDEMAR MAJ ukończył studia na Wydziale Automatyki i Informatyki Politechniki Śląskiej. Od 1979 r. pracuje w Centralnym Ośrodku Informatyki Górnictwa w Katowicach. Zajmuje się standardowym oprogramowaniem komunikacyjnym.



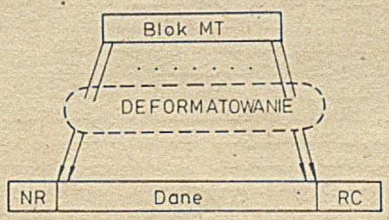
Rys. 4. Konfiguracja sprzętowa wraz z elementami oprogramowania do zdalnego przyłączenia maszyny cyfrowej ODRA 1305 do maszyny cyfrowej ICL 1904S

W skład oprogramowania wchodzą:

- własne programy #MTS0 i #TRMT, realizujące transmisję zbiorów taśmowych (5K)
- program obsługi zdalnych stacji 7020 — #CMKK, wygenerowany ze standardowego pakietu firmy ICL COMMUNICATIONS MANAGER (12K).

Program #MTS0 przeznaczony jest do uruchamiania na maszynie ICL 1904S (strona wysyłająca) i realizuje następujące funkcje:

- czytanie danych ze zbioru taśmowego
- formatowanie danych
- wysyłanie danych do stacji ICL 7020.



Rys. 5. Tworzenia rekordu przeznaczonego dla transmisji NR — numer rekordu, RC — część kontrolna rekordu

Formatowanie danych (rys. 5) polega na ich podziale na rekordy oraz odpowiednim deformatowaniu tych danych, które powinny być zinterpretowane przez stację 7020, jako przesyłane do niej znaki sterujące. Program może być uruchamiany zarówno w środowisku systemu operacyjnego GEORGE-3 jak i pod kontrolą egzakutora manualnego.

Po stronie maszyny ODRA znajduje się program #TRMT, który realizuje następujące funkcje:

- odbiór danych z czytnika CDT
- wykrywanie błędów i ich sygnalizacja
- deformatowanie danych
- zapis danych na taśmie magnetycznej.

Deformatowanie danych jest operacją odwrotną do operacji formatowania przeprowadzonej przez program #MTS0 i powoduje przywrócenie danym ich pierwotnej postaci.

Program #TRMT wykrywa zagubienie lub przekłamanie danych oraz sygnalizuje na pulpicie operatora maszyny ODRA konieczność powtórzenia transmisji błędnego rekordu. W procedurze obsługi sytuacji błędnych wykorzystywany jest pulpit (TTY) stacji 7020 znajdujący się pod kontrolą programu #MTS0.

Program #CMKK nadzoruje za pośrednictwem procesora komunikacyjnego pracę stacji abonenckich 7020 i umożliwia przydzielenie wybranej stacji do programu #MTS0. Podobnie jak program #CM7S, może on współpracować w tym samym czasie z kilkoma programami użytkowymi. Właściwość ta wykorzystywana jest w przypadku prowadzenia transmisji przez kilka połączeń równocześnie. W takim przypadku, dla każdego połączenia należy uruchomić odpowiednią wersję programu #MTS0 (tzn. stosującą numer identyfikacyjny odpowiedniej stacji).

Opisane zdalne połączenie dwu maszyn cyfrowych umożliwia w przypadku zastosowania modemów 1200 B/s transmisję danych z szybkością 100 znaków/s. Zwiększenie szybkości transmisji można osiągnąć poprzez zastosowanie szybszych modemów.

Obecnie trwają prace nad połączeniem umożliwiającym transmisję danych między maszynami ODRA 1305 i ICL 1904S w obu kierunkach przy wykorzystaniu tej samej stacji 7020. Za pomocą opisanego rozwiązania możliwe jest przesyłanie do odległych ośrodków obliczeniowych zbiorów taśmowych, głównie bibliotek programowych i zagregowanych danych do centralnego przetwarzania, które dotąd przewożone były środkami transportowymi.

* * *

Przedstawione w niniejszym artykule lokalne i zdalne połączenia maszyn ODRA 1305 i ICL 1904S stwarzają możliwość lepszego wykorzystania posiadanych zasobów informatycznych oraz istotne przyspieszenie obiegu informacji w przemyśle węglowym. Istnieje możliwość wykorzystania tego typu połączeń w dowolnych ośrodkach obliczeniowych przy założeniu posiadania przez nie odpowiedniego sprzętu.

LITERATURA

[1] Brady P., Rudak B., Rudakowa B.: Połączenie lokalne dwu maszyn cyfrowych ODRA 1305. INFORMATYKA, nr 10/1980
[2] Central Processors. Publikacja firmy ICL nr 4095, 1972
[3] Janyszek J.: Adapter międzymaszynowy ADM-305, jego budowa i zastosowania. Raporty Centrum Obliczeniowego Politechniki Wrocławskiej, seria PRE nr 3/1979
[4] Komunikacyjny Menedżer. Publikacja Centrum MERA-ELWRO, nr 1351804, Wrocław, 1977
[5] System operacyjny GEORGE 3. Publikacja Centrum MERA-ELWRO, nr 1300203, Wrocław, 1977.

W trakcie eksploatacji programu użytkowego uzyskano następujące rzeczywiste czasy wprowadzania poszczególnych typów rekordów:

- rekordu *PROFILOWYMIAR-WYR-WALC*:

$K_1 = 14{,}18$ s

- rekordu *GAT-STALI*:

$K_2 = 6{,}58$ s

- rekordu *ASORTYMENT*:

$K_3 = 861{,}34$ s

Z porównania czasów obliczonych z czasami uzyskanymi w trakcie rzeczywistej eksploatacji wynika, że popełniany błąd oceny wynosi około 10%.

* * *

Wyżej przedstawione wyniki potwierdzają pełną przydatność analitycznej metody wyliczania logicznych dostępów do bazy danych (wraz z jej rozwinięciem, dotyczącym obliczania czasu trwania poszczególnych operacji na bazie danych do oceny alternatywnych rozwiązań struktury logicznej bazy danych). Występujące różnice pomiędzy wynikami wyliczeń a uzyskanymi w trakcie rzeczywistej eksploatacji są niewielkie i wynikają głównie z niedostatecznej dokładności czasu trwania jednego fizycznego dostępu do dysku.

Na etapie projektowania struktury logicznej bazy danych dysponujemy jedynie wielkościami podawanymi przez producenta sprzętu. Podawany w dokumentacji technicznej jednostki dyskowej czas fizycznego dostępu dotyczy jednakże sytuacji niespotykanej w normalnej eksploatacji (praca jednej jednostki dyskowej poprzez jeden kanał we/wy oraz brak kolejki wejściowej do tego kanału). Urealnienie wielkości czasu dostępu fizycznego może więc odbywać się w zasadzie w trakcie przeprowadzania badań na tej konfiguracji sprzętowej, na której w przyszłości eksploatowana będzie projektowana baza danych. Pomocne mogą być również informacje zebrane z eksploatacji podobnych baz danych na porównywalnym sprzęcie komputerowym. Jednakże brak dokładnej znajomości czasu trwania fizycznego dostępu nie ma zasadniczego wpływu na użyteczność omawianej metody oceny baz danych. Wynika to z przeprowadzonej weryfikacji, gdyż uzyskane wyniki (dla średniego teoretycznego czasu trwania fizycznego dostępu) są tylko ok. trzy razy gorsze od przewidywanych na etapie projektowania struktury logicznej bazy danych, a jedynie o 10% po uwzględnieniu średniego czasu dostępu wynikającego z doświadczeń eksploatacyjnych.

Z dotychczas zebranych doświadczeń wynikają następujące wnioski:

- **Proponowana metoda wyliczania dostępów logicznych do bazy wraz z jej rozwinięciem jest wystarczającą do dokonywania porównań alternatywnych rozwiązań struktur logicznych bazy danych. Ponadto pozwala ona uzyskać odpowiedź na pytanie, czy zaprojektowana baza danych będzie możliwa do realizacji na dostępnej konfiguracji sprzętowej. Ma to kapitalne znaczenie ze względu na fakt, że odpowiedź uzyskujemy na początkowym etapie projektowania, co w przypadku odpowiedzi negatywnej pozwala uniknąć poważnych strat.**

- **Proponowana metoda uzmysławia wagę etapu analizy systemu, szczególnie pod kątem wiarygodnych oszacowań populacyjnych i niezbędności informacji zawartych w bazie danych. Oczywista jest również konieczność uwzględniania sposobu korzystania z informacji zawartych w projektowanej bazie danych.**

LITERATURA

[1] Fry J. P., Teoray T. J.: Design and performance tools for improving database usability and responsiveness. Red. D. Shneiderman. Academic Press, 1978

[2] Kapuścik W.: Liczba logicznych dostępów w bazach danych typu CODASYL. Podstawy Sterowania. Tom 10/1980

[3] Zębala A.: Opracowanie technik oceny za pomocą metod analitycznych struktur w sieciowych bazach danych ze względu na czas realizacji podstawowych operacji na bazie. Bytom, grudzień 1981.

KONFERENCJE

CAPE'83

W dniach 25—28 kwietnia 1983 r. odbędzie się w Amsterdamie międzynarodowa konferencja na temat zastosowania komputerów w produkcji i pracach inżynierskich CAPE'83 (Computer Applications in Production and Engineering). Organizatorem konferencji jest Międzynarodowe Biuro Informatyki IBI (Intergouvernmental Bureau for Informatics) w Rzymie, a protektorat nad nią objęły trzy organizacje: Międzynarodowa Federacja Przetwarzania Informacji IFIP (International Federation for Information Processing), Międzynarodowa Federacja Stowarzyszeń Użytkowników Komputerów w Pracach Inżynierskich i Architektonicznych FACE (International Federation of Associations of Computer Nsers in Engineering and Architecture) oraz Międzynarodowa Federacja Stowarzyszeń Badań Operacyjnych IFORS (International Federation of Operatiotional Research Societies).

Organizatorzy podkreślają, że CAPE'83 będzie pierwszą międzynarodową konferencją poświęconą zastosowaniu komputerów do automatyzacji całego procesu przemysłowej realizacji wyrobu, który obejmuje:

- określenie elementów składowych wyrobu
- opracowanie projektu koncepcyjnego
- weryfikację projektu
- opracowanie projektu szczegółowego
- przygotowanie produkcji i metod kontroli
- opracowanie technologii wytwarzania elementów oraz montażu
- zarządzanie produkcją
- planowanie zaopatrzenia materiałowego
- kontrolę zapasów materiałowych
- sterowanie procesów technologicznych
- emitowanie, rozdział i archiwowanie dokumentacji produkcyjnej wyrobu.

Organizatorzy konferencji akcentują szczególną wagę wzajemnych powiązań poszczególnych etapów realizacji wyrobu. Stwarza to możliwość integracji większości, a w dalszej kolejności również wszystkich etapów, a także bardziej skutecznej kontroli całego „cyklu życia" procesu realizacji wyrobu. Jest to określane często jako „zintegrowane CAD/CAM" (Computer Aided Design/Computer Aided Manufacturing — projektowanie i wytwarzanie wspomagane komputerem).

Miejscem obrad konferencji będzie amsterdamskie Centrum Kongresowe, a jej program obejmuje wygłoszenie 98 referatów przez reprezentantów 22 krajów.

MIROSŁAWA SNOWACKA
MICHAŁ ZIĘBIŃSKI
POLIMEX–CEKOP
Warszawa

Automatyczne przetwarzanie tekstów

Skuteczność działania wielu przedsiębiorstw, a w szczególności przedsiębiorstw handlu zagranicznego zależy m.in. od terminowości, poziomu, poprawności językowej oraz szaty zewnętrznej sporządzanej korespondencji. W przypadku opracowywanych dużych ilości korespondencji zagadnienie to stanowi poważny problem techniczny, organizacyjny i ekonomiczny. Wiadomo, że korespondencja handlowa opiera się w znacznym stopniu na wielokrotnie powtarzanych tekstach. Właściwości te pozwalają na zastosowanie do sporządzania korespondencji handlowej szybkopiszących automatów korespondencyjnych.

W Przedsiębiorstwie Handlu Zagranicznego POLIMEX-CEKOP Sp. z o.o. już w końcu 1975 roku podjęto prace nad automatyzacją korespondencji instalując do tych celów dwa automaty: FRIDEN 2301 i OPTIMA-528. W 1976 roku uruchomiono automat korespondencyjny EDITOR S-24, a w 1977 — następny tego typu. W roku tym zainstalowano również nowoczesny system do przetwarzania tekstów TES-501. Dalsze trzy automaty TES-501 zainstalowano w roku 1980. Kolejne wprowadzanie nowych, bardziej wydajnych urządzeń pozwoliło na znaczne rozszerzenie usług w zakresie automatycznego przetwarzania tekstów (APT).

Wieloletnie doświadczenia PHZ POLIMEX-CEKOP w dziedzinie wdrażania nowoczesnych technik i form opracowywania korespondencji stwarzają podstawę do przekazania na ten temat szerszej informacji. Sądzimy, że nasze doświadczenia i pozytywne wyniki uzyskane w trakcie wieloletniego stosowania APT przyczynią się do szerszego rozpropagowania tej techniki i wprowadzenia jej do innych przedsiębiorstw i instytucji.

ZAKRES ZASTOSOWAŃ

W każdym przedsiębiorstwie handlu zagranicznego podstawowym elementem pracy jest wymiana informacji — w formie listów i innych dokumentów — prowadzona w wielu językach (polskim, rosyjskim, angielskim, niemieckim, francuskim i hiszpańskim). Analizy korespondencji i dokumentów handlowych w PHZ POLIMEX-CEKOP wykazały, że przeważająca ich część może być wykonywana w sposób zautomatyzowany. Możliwości techniczne wspomnianego na wstępie sprzętu pozwalają na różne rodzaje przetwarzania tekstów z wyjątkiem tych, które wymagają przetwarzania danych cyfrowych (faktury statystyczne, dokumenty księgowe, itp.). Utworzony w przedsiębiorstwie Zespół APT zdołał do chwili obecnej opracować i wdrożyć do eksploatacji ponad 280 różnych wzorów dokumentów i korespondencji w różnych językach.

Podstawowe prace w zakresie automatyzacji korespondencji i dokumentów dotyczą:

w eksporcie:

- listów akwizycyjnych
- informacji technicznych
- programów eksportowych
- ofert
- dokumentów przedkontraktowych
- kontraktów
- dokumentów wynikających z bieżącej realizacji kontraktu (np. wniosków wyjazdowych, dokumentów wysyłkowych: faktur handlowych, wniosków do PIHZ o wydanie świadectwa pochodzenia, świadectw pochodzenia itp.)

w imporcie:

- dokumentów przedkontraktowych, w tym przede wszystkim zapytań ofertowych
- potwierdzeń zamówień
- kontraktów
- dokumentów wynikających z bieżącej realizacji kontraktu (np. zlecenie otwarcia akredytywy, monity bankowe, oferta, podziękowanie za ofertę, noty debetowo-kredytowe i inne).

Obszerną pozycją w pracach Zespołu APT jest korespondencja sporządzana okolicznościowo, na zasadzie dużych akcji, np. zaproszenia na targi i wystawy, życzenia z okazji świąt narodowych, przeszeregowania itp.

MIROSŁAWA SNOWACKA jest absolwentką Studium Ekonomicznego Handlu Zagranicznego. Od 1974 r. pracuje w PHZ POLIMEX-CEKOP jako koordynator prac w zakresie automatyzacji korespondencji handlowej przedsiębiorstwa. Od czasu powstania Sekcji Automatycznego Przetwarzania Tekstów w Ośrodku Informatyki POLIMEX-CEKOP jest jej kierownikiem.

Mgr inż. MICHAŁ ZIĘBIŃSKI ukończył Politechnikę Warszawską (1962). W latach 1967—1971 pracował we Francji i Stanach Zjednoczonych jako konstruktor maszyn budowlanych i urządzeń elektrowni atomowych. W 1971 r. uzyskał tytuł Haster of Science in Engineering Management w Northeastern University w Bostonie (USA). Od 1968 r. zajmował się informatyką w zastosowaniach inżynierskich i sterowaniu inwestycjami. W latach 1978—1982 pracował w PHZ POLIMEX-CEKOP na stanowisku doradcy dyrektora naczelnego ds. Informatyki oraz kierownika Ośrodka Informatyki. Jest autorem ponad 30 publikacji krajowych i zagranicznych, związanych z organizacją i informatyką.

Należy jednak pamiętać, że oszczędności czasowe leżą nie tylko po stronie wykonującego dokument, ale także po stronie zamawiającego. Proces automatyzacji korespondencji eliminuje bowiem tradycyjny brudnopis całego dokumentu.

WPŁYW APT NA ZMIANY W ORGANIZACJI PRACY ORAZ ELEMENTY SOCJOLOGII I PSYCHOLOGII PRACY

Omówione wcześniej korzyści z zastosowania APT mają szereg konsekwencji natury organizacyjnej, socjologicznej i psychologicznej, które w istotny sposób wpływają na poprawę warunków pracy i zwiększenie jej efektywności oraz to, co można niewątpliwie nazwać humanizacją pracy.

Na ogół sądzi się, że wprowadzenie maszyn i automatów prowadzi do dehumanizacji pracy, do stwarzania sytuacji, w której człowiek staje się dodatkiem do maszyny. W przypadku automatów do przetwarzania tekstów i podobnych im urządzeń sytuacja wygląda jednak inaczej.

Wszyscy zainteresowani, a więc zarówno handlowcy przygotowujący w tradycyjnym trybie brudnopis pisma, jak i maszynistki piszące na tradycyjnych maszynach — zostają odciążeni od dosyć niewdzięcznych powtarzalnych czynności, niezależnie czy jest to sporządzenie brudnopisu prostego nawet pisma, a następnie jego weryfikacja, czy też ręczne pisanie na maszynie. Czynności te w tradycyjnym procesie przygotowania tekstów wymagają dużego wysiłku. W systemie APT natomiast stanowią niewielki wkład w stosunku do otrzymywanych efektów.

Pomijając stronę ekonomiczną, popatrzmy na pracownika, który dziennie musi przygotować kilka lub kilkanaście pism o bardzo podobnej lub zbliżonej treści. Nie jest to ani interesujące, ani sensowne zajęcie dla pracownika legitymującego się najczęściej wyższym wykształceniem. Pomyślmy z kolei o maszynistkach przepisujących stale prawie identyczne teksty. Wiadomo również, jak trudno jest rozwiązać problem maszynopisania w dużym przedsiębiorstwie.

Na podstawie wieloletnich obserwacji stwierdziliśmy, że wprowadzenie APT radykalnie poprawia współpracę pomiędzy przygotowywującym teksty, a maszynistkami. Po pierwsze — dzięki dużej szybkości i wydajności, znika w znacznym stopniu problem pilnego pisania dużych listów — po prostu nawet duży tekst może być napisany w ciągu kilkunastu minut. Po drugie — w zasadniczym stopniu znika problem czytelności rękopisów, gdyż stanowią one teraz niewielki ułamek prac, a wiemy jak często zły rękopis jest powodem dyskusji i napięć. Po trzecie — zanika również problem błędów maszynopisania, które są zmorą, zwłaszcza początkujących i niezbyt wprawnych maszynistek.

Z praktyki wiemy, że te niewątpliwie drobne problemy stają się przyczyną wielu ostrych nieporozumień obu stron. Można więc stwierdzić, że APT w znacznym stopniu wpływa na poprawę warunków pracy i stosunków międzyludzkich. A przecież warto pomyśleć o tym, że pracować należy nie tylko dobrze i wydajnie, ale również w dobrym samopoczuciu, które pozwala polubić pracę i nie traktować jej jako uciążliwej konieczności.

Wchodzimy tu w sferę daleką od rozważań techniczno-organizacyjnych. Pamiętajmy jednak, że sensem każdej organizacji jest jej wpływ na życie człowieka, nie tylko tego, który decyduje, ale i tego, który te decyzje realizuje. Z myślą o nim kończymy ten artykuł. APT przynosi bowiem nie tylko oszczędność czasu i pieniędzy, lecz również lepsze stosunku między ludźmi i więcej uśmiechu na ich twarzach.

ANDRZEJ SMERECZYŃSKI
Instytut Organizacji Zarządzania i Doskonalenia Kadr
Warszawa

Automatyzacja prac wydawniczych na IBM/370

W Instytucie Organizacji Zarządzania i Doskonalenia Kadr, który w ramach współpracy z IBM korzysta z komputera IBM/370-148, sprawdzono i przygotowano do użytkowej eksploatacji SCRIPT/370 w wersji 3 — program przeznaczony do przetwarzania tekstów.

Ogólnie rzecz ujmując — przetwarzanie tekstów polega na: czytaniu danych, ich formatowaniu do żądanej postaci oraz wyprowadzeniu wyników. Dane są wprowadzane do maszyny poprzez terminal (np. ekranowy z klawiaturą) lub zapisane w zbiorze dyskowym. Formatowanie tekstu — to jego rozmieszczanie w poszczególnych wierszach, kolumnach, na stronach i w całym tomie według przyjętych zasad redakcyjnych. Wyniki w postaci sformatowanego tekstu wyprowadzane są na drukarkę, ekran monitora (terminala) bądź do zbioru dyskowego. SCRIPT realizuje wszystkie funkcje związane z formatowaniem tekstu za pomocą tzw. słów kontrolnych. Zapewnia on mianowicie:

- odpowiednie odstępy między wierszami lub rozdziałami, wielkość wcięcia wierszy, sposób organizacji wyliczeń w tekście
- ustaloną liczbę wierszy na stronie, długość wiersza, a także sposób umieszczania tytułu strony i jej numerowania
- zapis tekstu w kilku kolumnach na stronie
- tworzenie spisu treści, zawierającego nagłówki rozdziałów i numery ich stron
- właściwą długość tekstu, podział tekstu na oddzielne części i łączenie ich w czasie przetwarzania
- numerowanie ilustracji i tworzenie dla nich miejsc w tekście, tworzenie tablic, pustych stron lub pustych ich fragmentów
- użycie zmiennych informacji, poprzez wykorzystanie nazw symbolicznych dla tych informacji
- obsługę interaktywnego przetwarzania tych fragmentów tekstu, które mają być wprowadzane z terminala w czasie działania programu
- obsługę makroinstrukcji tworzonych z najczęściej używanych grup słów kontrolnych.

Podstawową funkcją SCRIPTU jest tworzenie tekstu z wyrównanym prawym marginesem. Funkcję tę można usunąć, jeżeli w żądanym miejscu tekstu wstawimy — jako pierwsze znaki w wierszu — słowo kontrolne w postaci:

.fo off

Od tego miejsca SCRIPT wyprowadzi tekst w tej postaci, w jakiej jest on zapisany w zbiorze danych, aż do końca zbioru lub do wiersza, którego pierwsze znaki będą zawierać słowo kontrolne:

.fo on

Przykładem wykorzystania słów kontrolnych jest także użycie słowa kontrolnego wprowadzającego wcięcia tekstu:
.11 25
Tych kilka linii tekstu zostało sformatowanych z dostateczną liczbą słów
.in 5
tak, aby można było zauważyć, jak SCRIPT formatuje tekst
.in +3
oraz, że wielkość wcięcia
.in —6
może być odwrócona przez użycie wartości ujemnej.

Powyższe linie po przetworzeniu będą umieszczone w liniach 25-znakowych i będą miały następującą postać:

Tych kilka linii tekstu
zostało sformatowanych z
dostateczną liczbą słów
    tak, aby można było
    zauważyć, jak SCRIPT
    formatuje tekst
        oraz, że wielkość
        wcięcia
może być odwrócona
przez użycie wartości
ujemnej.

Oto postać i funkcje wybranych słów kontrolnych SCRIPTU, wymienionych w alfabetycznej kolejności:

.ap nazwa-zbioru

umożliwia dołączenie zbioru do zbioru, który jest aktualnie przetwarzany

.be ON/OFF

powoduje wyrównanie kolumn przed zakończeniem przetwarzania strony i zapisuje tekst od nowego wiersza

.cd n d1 d2...d9

określa liczbę kolumn (n) na stronie; kolumny zaczynają się od d1, d2,...,d9 znaku w wierszu

.cl n

określa liczbę znaków w każdym wierszu kolumny

.cp n

powoduje pisanie tekstu od nowej strony, jeśli mniej niż n linii pozostaje na stronie

.ds

powoduje pisanie tekstu z odstępami na dwa wiersze

.hO wiersz-nagłówka
.hl wiersz-nagłówka
. . . . . .
.h6 wiersz-nagłówka

tworzą z wierszy-nagłówka nagłówki i pozycje spisu treści

.mc

pisanie tekstu w wielu kolumnach

.pl n

określa liczbę wierszy na stronie

.rd n

czyta n wierszy z terminala i umieszcza je w zbiorze wyjściowym

.ri n|ON|OFF|wiersz

ustawia lub kasuje ustawienie n wierszy tekstu do prawego marginesu

.sc

powrót do umieszczania tekstu w jednej kolumnie

.sy komenda-CP-lub-CMS

traktuje wiersz jako komendę systemu CP lub CMS

.te

powoduje utworzenie spisu treści z nagłówków określonych słowami kontrolnymi .hO h6

.te n|ON|OFF

umożliwia wprowadzenie n wierszy z terminala w czasie przetwarzania tekstu programem SCRIPT

.tr s s

określa ostateczną wyjściową postać dowolnego znaku

.uc wiersz

podkreśla wiersz i wypisuje go wielkimi literami.

Brak miejsca nie pozwala na szczegółowe opisanie wszystkich 85 słów kontrolnych SCRIPTU i pokazanie choćby na krótkich przykładach sposobu ich funkcjonowania. Warto jednak podkreślić, że funkcje te spełniają wszelkie wymagania, jakie stawiane są przy redagowaniu tekstów.

W ramach programu SCRIPT zawarty jest podprogram o nazwie EasySCRIPT (łatwySCRIPT). Wywoływany jest on wówczas, gdy w tekście umieszczone jest słowo kontrolne .ez on . Od tego miejsca wystarczy wykorzystywać tylko pięć instrukcji, tj.:

& Hx — oznaczanie i numerowanie nagłówków wg numeracji Deweya
& P — wyznaczanie początku rozdziału
&N i & B — tworzenie wyliczeń numerowanych i punktowanych
& toc-tworzenie spisu treści

Stosowanie EasySCRIPTU jest wskazane przy uproszczonych wymaganiach redakcyjnych i dla początkujących użytkowników SCRIPTU.

Istotną cechą SCRIPTU, wykorzystywaną dla bardziej optymalnego formatowania tekstu, jest automatyczne przenoszenie wyrazów na podstawie informacji podanych w zbiorze pomocniczym SCRIPTU lub na podstawie algorytmów określających zasady przenoszenia wyrazów (np najdłuższy fragment wyrazu wykraczający poza przyjętą długość wiersza). Algorytmy te są wykorzystywane wówczas, gdy SCRIPT nie znajdzie zbioru pomocniczego.

Użytkownikami SCRIPTU mogą być osoby zajmujące się profesjonalnie pisaniem, korektą, drukowaniem korespondencji, opracowań, wydawnictw dokumentacji projektowej, programowej lub innymi czynnościami tego typu. Wykorzystywać go zatem można w pracach sekretarskich, projektowo-programowych, wydawniczych, administracyjnych itp.

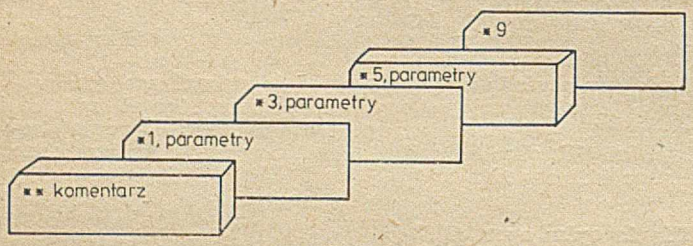
Istnieją pewne uwarunkowania dla działania SCRIPTU. Po pierwsze — musi być on zainstalowany na komputerze IBM/370 pracującym pod kontrolą systemu operacyjnego VM, którego integralną częścią jest tzw. Konwersacyjny System Monitorowy (Conversational Monitor System — CMS). Jest to potrzebne dla samego procesu przetwarzania. Natomiast przygotowanie zbiorów danych dla programu SCRIPT może odbywać się na dowolnym urządzeniu do przygotowania danych, zachowującym odpowiedni standard zapisu w zakresie dopuszczanym przez IBM. Po drugie — drukarka, na którą wyprowadzony będzie tekst po przetworzeniu powinna mieć także znaki specyficzne dla

- w przypadku nałożenia się kilku tego samego typu operacji na tym samym polu wynik jest nieznany, tzn. zależny od przebiegu sortowania (losowy lub klucze sortowania identyczne)
- hierarchia operacji będzie zgodna z rosnącym posortowaniem kodów operacji
- możliwe jest użycie operacji WPR jako kompletnego zastąpienia istniejącego w zbiorze rekordu, ale pamiętać trzeba o niebezpieczeństwach tego rozwiązania.

RODZAJE OPERACJI

UPR — usunięcie pojedynczego rekordu
UPC, UCZ — usunięcie ciągu rekordów
WPR — wprowadzenie nowego rekordu
ZPR — zmiana wartości w polu pojedynczego rekordu
ZCP, ZCZ — zmiana wartości w polu ciągu rekordów
ZWZ — zmiana wartości w polu całego zbioru
DPR — dodanie określonej wartości do pola pojedynczego rekordu
DCP, DCZ — dodanie określonej wartości do pola ciągu rekordów
DWZ — dodanie określonej wartości do pola całego zbioru
OPR — odjęcie określonej wartości od pola pojedynczego rekordu
OCP, OCZ — odjęcie określonej wartości od pola ciągu rekordów
OWZ — odjęcie określonej wartości od pola całego zbioru
PPR — wykonanie operacji określonej procedurą we wskazanym polu pojedynczego rekordu
PCP, PCZ — wykonanie operacji określonej procedurą we wskazanym polu ciągu rekordów
PWZ — wykonanie operacji określonej procedurą we wskaaznym polu całego zbioru
PZU — wywołanie procedury użytkownika

KONSTRUKCJA ZADANIA CHANGE

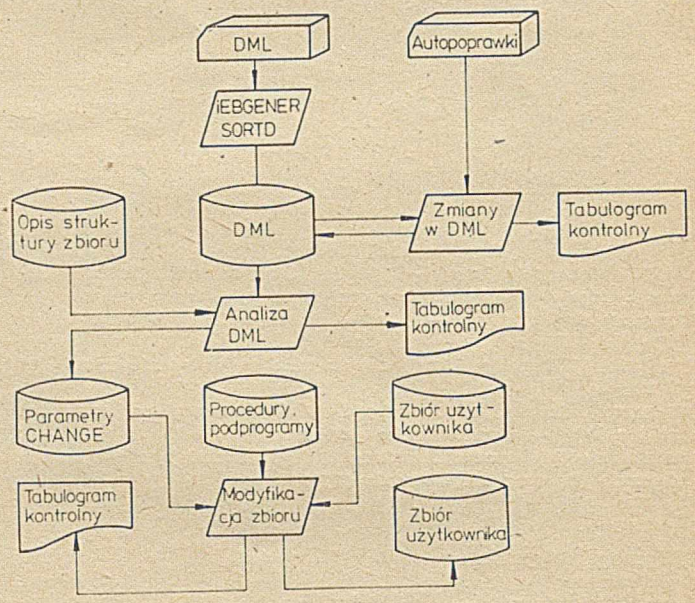


Rys. 2. Zadanie w języku DML

PRZYKŁAD:

Zdanie DML	Interpretacja
**1TEST # CHANGE	
**2DNIA 10 STYCZNIA	
1 ,,21 , 03 , 211 ,	— zbiór do zmodyfikowania nie będzie specjalnie badany, klucz złożony z trzech bajtów zaczyna się od 21 bajtu, a maksymalna długość rekordu wynosi 211 bajtów
*3 , NONE ,	— nie przewiduje się specjalnej kontroli w zbiorze
*5 , 003 , UPR	— usunięcie rekordu o kluczu 003
*5 , 008,UCP,011 ,	— usunięcie ciągu rekordów
*5 , 008,UCZ ,	o kluczach od 008 do 011
*5 ,019,ZPR,003,12	— w rekordzie o kluczu 0,19, do pola dwuznakowego, zaczynającego się w trzecim bajcie, wprowadza się znaki: „12”
*9	— koniec zadania.

PLAN OPERACYJNY PAKIETU CHANGE



Rys. 3. Schemat operacyjny pakietu CHANGE

Proces korygowania zbioru prezentowanego rozwiązania przebiega w pełnym zakresie w następujących etapach:

- Przygotowanie, wczytanie i zapisanie na nośniku magnetycznym zdań DML.
- Sortowanie zbioru zdań DML do postaci zadania CHANGE.
- Analiza formalna zadania obejmująca badanie poprawności i kompletności składniowej zadania oraz kontrolę poprawności odwołań do struktury korygowanego zbioru. W drugiej części analizy chodzi o sprawdzenie zgodności identyfikatorów podanych w zdaniach DML z zasadami budowy identyfikatorów w zbiorze korygowanym oraz o sprawdzenie odpowiedniości adresów względnych pól, ich długości i typów z podanymi explicite lub wynikającymi z kontekstu parametrami zdań DML. Służy temu specjalny zbiór opisu struktury, tworzony standardowo, a określający: typy rekordów zbioru korygowanego, ich wyróżniki, położenie i charakter klucza, charakterystykę pól dla każdego typu rekordu itp. Wyniki analizy są dokumentowane tabulogramem kontrolnym.
- Właściwe korygowanie zbioru użytkowego z wykorzystaniem procedur standardowych oraz podprogramów użytkownika. Przebieg korekty jest dokumentowany tabulogramem.

* * *

Potrzebę takiego opracowania zrodziła praktyka uruchomiania systemów bankowych. Systemy te — opracowywane częstokroć przez jednostki usługowe — z różnych względów nie pasują do innych istniejących już rozwiązań i wówczas doraźną pomocą służyć może pakiet CHANGE. Zwłaszcza w fazie testowania, gdy uruchomiane są wieloprogramowe cykle przetwarzania, daje on możliwość równoczesnego „szlifowania” programów. Oczywiste jest niebezpieczeństwo nadużyć w korzystaniu z tego pakietu, stąd też konieczność stworzenia organizacyjnych ograniczeń w jego stosowaniu.

STEFAN ŚWIĄĆ
RYSZARD WEINAR
Środowiskowe Centrum Obliczeniowe CYFRONET
Kraków

DISCAR – preprocesor języka FORTRAN na maszynę cyfrową CYBER

Maszyna cyfrowa CYBER 72-16 firmy CONTROL DATA zainstalowana w Środowiskowym Centrum Obliczeniowym CYFRONET w Krakowie zawiera m.in.:

— pamięć operacyjną o pojemności 96 K słów 60-bitowych, z czego około 72 K słów przeznacza się na prace użytkowników (maksymalnie 36 K słów dla jednej pracy)
— pamięć dyskową o pojemności 3×100 M bajtów

— sieć terminali wsadowych i dalekopisowych

i pracuje pod kontrolą wieloprogramowego systemu operacyjnego SCOPE 3.4. Mimo dużej mocy obliczeniowej (w ciągu doby wykonywanych jest około 1000 prac) oraz niezawodności sprzętu, potrzeby użytkowników nie są w pełni realizowane.

Od pewnego czasu prowadzone są pomiary i obserwacje, mające na celu ustalenie optymalnych parametrów pracy systemu. Podczas badań stwierdzono, że:

- zdecydowaną większość prac wykonywanych w ośrodku stanowią programy napisane w języku FORTRAN
- po upływie godziny od wyłączenia sieci terminali wykorzystanie komputera spada do tego stopnia, że staje się on w praktyce maszyną jednoprogramową.

Przyczyną tak dużej popularności języka FORTRAN jest numeryczny charakter obliczeń, realizowanych w ośrodku. Natomiast zmniejszenie wykorzystania komputera, po wyłączeniu sieci terminali, spowodowane jest brakiem dopływu prac o krótkim czasie realizacji i małym wykorzystaniu pamięci. Takie prace mają wyższy priorytet niż prace o długim czasie realizacji lub dużym zapotrzebowaniu na pamięć operacyjną i są realizowane w pierwszej kolejności. Po ich wykonaniu pozostają jedynie prace wymagające więcej niż 36 K słów pamięci operacyjnej, lecz dwóch tak dużych prac nie można wykonywać jednocześnie.

Zmniejszenie zapotrzebowania programów napisanych w języku FORTRAN na pamięć operacyjną można osiągnąć stosując nakładanie i segmentowanie, a także — wykorzystując inne programy (np. pakiet ZBOOK, opracowany w Europejskiej Organizacji Badań Jądrowych CERN, pod Genewą).

Niewielki zakres stosowalności tych metod skłonił nas do zastanowienia się nad możliwością stworzenia nowego narzędzia, wolnego od wad i niedogodności oprogramowania znajdującego się w ośrodku. Powinno ono być proste w użyciu i łatwo stosowalne — bez konieczności wnikania w istotę działania programów.

Założono, że zasadniczym czynnikiem wpływającym na wielkość pamięci operacyjnej potrzebnej do wykonania programu napisanego w języku FORTRAN są deklaracje tablic. Na tej podstawie opracowano preprocesor DISCAR, który umożliwia automatyczne przenoszenia na dysk wartości tablic wskazanych przez użytkownika.

PROGRAM DISCAR

DISCAR jest programem modyfikującym segmenty napisane w języku FORTRAN EXTENDED w taki sposób, że są one równoważne co do wyników obliczeń, a różnią się umiejscowieniem niektórych tablic i zmiennych. W segmentach oryginalnych wszystkie tablice i zmienne znajdują się w pamięci operacyjnej. W segmentach utworzonych przez preprocesor wartości wskazanych tablic i zmiennych przechowywane są w pliku dyskowym o dostępie bezpośrednim, a ściąganie ich do pamięci operacyjnej następuje dopiero wtedy, gdy są potrzebne. Taka organizacja pracy umożliwia zredukowanie wielkości pamięci operacyjnej, koniecznej do wykonania programu użytkowego, choć następuje to kosztem czasu jego realizacji.

Jedyną informacją niezbędną do wykonania modyfikacji jest lista nazw tablic i zmiennych, które należy przenieść na dysk. Do przekazania tych informacji służą instrukcje postaci:

LEVEL 1, $\text{nazwa}_1 \ldots, \text{nazwa}_n$

które umieszcza się w obszarze deklaracji segmentu.

Instrukcja LEVEL 1 ma znaczenie tylko wtedy, gdy komputer jest wyposażony w odpowiedni sprzęt. W przypadku braku takiego sprzętu jest ona ignorowana przez kompilator. Dzięki tej właściwości, segmenty zawierające instrukcje LEVEL 1 mogą być kompilowane bezpośrednio przez kompilator języka FORTRAN EXTENDED. W takim przypadku wszystkie wymienione w tej instrukcji tablice i zmienne zostaną umieszczone w pamięci operacyjnej.

Oprócz zmiennych i tablic podanych w instrukcji LEVEL 1, na dysk zostaną przeniesione wszystkie zmienne i tablice związane z nimi instrukcjami COMMON i EQUIVALENCE.

Mgr Stefan ŚWIĄC ukończył w 1974 r. studia na Wydziale Matematyczno-fizyczno-chemicznym UJ, kierunek — matematyka, specjalizacja — metody numeryczne. Po studiach podjął pracę w Środowiskowym Centrum Obliczeniowym CYFRONET w Krakowie. Zajmuje się rozwojem oprogramowania użytkowego.

Mgr Ryszard WEINAR ukończył w 1974 r. studia na Wydziale Matematyczno-fizyczno-chemicznym UJ, kierunek — matematyka, specjalizacja — metody numeryczne. Po studiach podjął pracę w Środowiskowym Centrum Obliczeniowym CYFRONET w Krakowie. Zajmuje się zagadnieniami komputerowo wspartego konstruowania map i rozwojem oprogramowania użytkowego.

```
            end;
        u:=ul-j+1;
        if u>kk then ll:=i
        else begin u:=u+z-ll;
            if u ⩾ kk then begin kk:=l; ul:=ll end
            else begin kk:=kk-u; ll:=z+1; ul :=j end
        end
    end reorganize;
    procedure push;
    begin STACK [zero+1] :=ll; STACK [zero+2] :=ul;
        STACK [zero+3] :=kk; zero:=zero+3
    end push;
    procedure pop;
    begin zero:=zero-3; ll:=STACK [zero+1];
        ul:=STACK [zero+2]; kk:=STACK [zero+3]
    end pop;
    procedure sort (ll,ul); integer ll,ul;
    comment sortuje elementy A[ll], A[ll+1],..., A[ul];
    begin integer i,j,ind; real pom;
    if ul>ll then
    begin ind:=ll+1;
        for i:=ind step 1 until ul do
        begin pom:=A[i]; j:=i-1;
            while A[j]>pom do
            begin A[j+1] :=A[j]; j:=j-1;
                if j<ll then go to endwhile
            end;
        endwhile: A[j+1] :=pom
        end for;
            end
            end sort;
        ll:=low; ul:=up; kk:=k; one:=zero;
        while true do
        begin
            while ul-ll⩾10 do
            begin j:=ll-1;
                for i:=ll step 5 until ul-2 do
                begin sort(i,i+2); j:=j+1; pom:=A[j];
                    A[j] :=A[i+2]; A[i+2] :=pom
                end;
                push; ul:=j; kk:=l+(j-ll) div 2
            end while;
            sort (ll,ul); r:=A[ll+kk-1];
            if zero = one then begin Vk:=r; go to exit end;
            pop; reorganize
        end;
        exit:
    end Vk;
```

Uwaga:

Jeżeli wszystkie elementy zbioru *E* są różne, to procedura *reorganize* może przyjąć nieco prostszą postać:

```
procedure reorganize;
comment wersja, gdy wszystkie elementy są różne;
begin boolean wh;
  i:=ll; j:=ul; wh:=true;
  while wh do
  begin
    while A[i]<r do i:=i+1;
    while A[j]>r do j:=j-1;
    if i>j then wh:=false
    else begin pom:=A[i]; A[i]:=A[j]; A[j]:=pom end
  end;
  if i-ll⩾kk then ul:=j else begin kk:=kk-i+ll; ll:=i end
end reorganize;
```

Poniżej prezentujemy procedurę funkcyjną *RT*, która stanowi dokładne rozwiązanie postawionego problemu. Parametrami *RT* są: tablica *A*, jej zakres *n* oraz liczba naturalna *k*.

```
real procedure RT(A,n,k); real array A; integer n,k;
comment wartością jest k-ty co do wielkości element z tablicy A[1:n]
begin integer m,zero;
if n>10 then m:=3*entier(log(n)/log(5))+3 else m:=1;
zero:=0;
if 1⩽k and k⩽n then
begin integer array STACK[1:m];
  RT:=Vk(A,1,n,k,zero,STACK)
end
end RT;
```

ANDRZEJ SZAŁAS
ZBIGNIEW ŚWIRSKI

Rubryka ALGORYTMY pojawia się na naszych łamach od listopada ub. r. W tym czasie zebraliśmy na jej temat trochę opinii, z których wynika, że obecna jej postać dla wielu Czytelników jest zbyt trudna w odbiorze. Sama zaś zasada — prezentacji „chytrych" algorytmów — nie została przez nikogo podważona. Chcielibyśmy zatem w przyszłym roku zmienić nieco formułę rubryki, zgodnie z przekazanymi nam postulatami. Prosimy o dalsze uwagi.
(Red.)

Dlaczego nie korzystamy z oprogramowania powtarzalnego

Ze społecznego punktu widzenia nie sposób uzasadnić wielokrotnej pracy nad oprogramowaniem obsługującym zbliżone funkcje użytkowe. Faktem jest jednak istnienie w Polsce wielu systemów gospodarki materiałowej, list płac, systemów kadrowych itd. Celem zanalizowania przyczyn tego stanu, przynoszącego znaczne straty gospodarcze, Zakład Obsługi Informacyjno-Komputerowej CPiZI przeprowadził na zlecenie Sekretariatu Komitetu Informatyki odpowiednie badania sondażowe w krajowych ośrodkach obliczeniowych. Uzyskane w wyniku tych badań materiały [1]) stanowią podstawę niniejszego artykułu.

Struktura oprogramowania

Stosowane w ośrodkach oprogramowanie analizowano w podziale na:

- indywidualne — użytkowe, rozwiązujące konkretne zadanie, nie dające się na ogół wykorzystać do rozwiązywania innych, nawet zbliżonych zadań
- powtarzalne — użytkowe, posiadające pewne cechy uniwersalności zastosowań
- narzędziowe — służące do wykonania oprogramowania użytkowego (indywidualnego lub powtarzalnego).

Z przeprowadzonych badań wynika, że technologia pracy ośrodków opiera się głównie na tworzeniu oprogramowania indywidualnego. Zaledwie 30% ośrodków opiera swoją działalność na oprogramowaniu powtarzalnym, pozostałe korzystają z niego sporadycznie lub nie korzystają wcale (tab.).

Ośrodki stosujące oprogramowanie powtarzalne deklarowały najczęściej stosunek wykorzystania oprogramowania indywidualnego do powtarzalnego — 4 : 1. Bardzo niewielki jest przy tym udział oprogramowania narzędziowego (5—10%).

Do najczęściej stosowanych systemów powtarzalnych należą: FK (z ZETO Bydgoszcz), ASKOP (z ZETO Szczecin), niektóre systemy gospodarki materiałowej oraz systemy kadrowe wykonane w ośrodkach branżowych.

Przyczyny wyboru oprogramowania

Ośrodki informatyki stosują taki rodzaj oprogramowania, jakiego wymaga organizacja ich pracy i jaką wymusza sposób finansowania, a także metoda szkolenia kadry.

Ośrodki ogólnodostępne (ZETO, ETOB itp.) w stopniu większym niż inne ośrodki informatyki opierają się na oprogramowaniu powtarzalnym. Proporcje rodzajów stosowanego oprogramowania są następujące — 70 : 25 : 5 (powtarzalne, indywidualne, narzędziowe). W dużych ośrodkach zakładowych i branżowych sytuacja przedstawia się odmiennie. Mają one na ogół własny sprzęt oraz liczną kadrę projektantów i programistów, co stwarza sprzyjające warunki dla tworzenia oprogramowania indywidualnego. Ośrodki te nie są nastawione na świadczenie usług komputerowych dla zleceniodawców spoza zakładu macierzystego. Ponadto duże ambicje zawodowe zatrudnionych tu informatyków nie sprzyjają stosowaniu gotowego oprogramowania.

Brak zainteresowania oprogramowaniem powtarzalnym wynika także ze sposobu finansowania poszczególnych ośrodków. W ośrodkach ogólnodostępnych możliwość wielokrotnego użycia i sprzedaży gotowego systemu jest ceniona ze względu na opłacalność takiej praktyki. W innych ośrodkach, nie będących na pełnym rozrachunku ekonomicznym (branżowych lub zakładowych), zainteresowanie to jest znacznie mniejsze. Ośrodki branżowe finansowane są bowiem z funduszu postępu technicznego, natomiast ośrodki zakładowe — z budżetu macierzystego zakładu. Sprzyja to bardziej prowadzeniu prac projektowo-programowych, niż prac o charakterze eksploatacyjnym. Ponadto pracochłonność dokonań projektowych i programowych jest trudno wymierna, zwłaszcza w przypadku programowania indywidualnego — łatwiej więc tu uzasadnić konieczność istnienia ośrodka.

Ważnym czynnikiem wyboru technologii programowania przez ośrodek jest metoda przygotowania kadry. Dotąd nastawiano się przede wszystkim na projektowanie i oprogramowanie, dlatego też dzisiaj brakuje tzw. eksploatatorów systemów, natomiast bardzo liczna jest grupa „projektantów jednego systemu". W efekcie oprogramowanie indywidualne stało się powszechną technologią pracy w ośrodkach.

Ograniczenia oprogramowania powtarzalnego

Wybór odpowiedniego gotowego oprogramowania utrudnia użytkownikowi w znacznej mierze brak wymiany pomiędzy ośrodkami informacji o powstających systemach oraz wymiany doświadczeń związanych z ich opracowaniem. Ośrodki postulują stworzenie informatora przedstawiającego wdrożone i wypróbowane systemy, a także systemy nowopowstające.

W ośrodkach informatyki brakuje też działalności marketingowej. Zaledwie 20% badanych ośrodków prowadzi taką działalność (ośrodki ogólnodostępne — 100%, branżowe — 25%, pozostałe — tylko sporadycznie). Większość produktów programowych nie jest przystosowana do rozpowszechniania i sprzedaży.

Do pogłębiania się tych niekorzystnych zjawisk przyczynia się jeszcze nieodpowiedni system cen oprogramowania. W aktualnie stosowanych cennikach nie uwzględniono, lub tylko znikomo zaznaczono, działy dotyczące użytkowej eksploatacji gotowego oprogramowania, jego wdrażania i konserwacji. A przecież koszty tych prac wynoszą ponad 50% wszystkich nakładów w przypadku prawidłowego stosowania informatyki!

Dodatkową przyczyną trudności rozpowszechniania gotowego oprogramowania jest jednolitość cen systemów na terenie całego kraju. W przypadku wdrażania i konserwacji systemu w macierzystym mieście ośrodka koszty mogą być 2—3-krotnie niższe niż w przypadku wykonywania tych prac w odległej miejscowości. Dotychczaso-

Procentowy udział ośrodków stosujących lub sprzedających oprogramowanie powtarzalne (w poszczególnych grupach ośrodków)

Rodzaj ośrodka	Część ośrodków stosujących oprogramowanie powtarzalne (%)
Ogólnodostępne (ZETO, ETOB)	100
Branżowe	15
Uczelniane	20
Zakładowe duże (liczba zatrudnionych powyżej 30 osób)	15
Zakładowe małe (liczba zatrudnionych do 30 osób)	50
Stacje przygotowania danych	50

[1]) Elżbieta Ożóg-Skolimowska: Raport o stanie ośrodków obliczeniowych, część II — Metoda oprogramowania, czerwiec 1982

wy system wysokich jednolitych cen odstraszał więc najliczniejszą grupę potencjalnych odbiorców gotowego oprogramowania, jaką stanowią zakładowe ośrodki informatyki.

* * *

Wydaje się, że przyczyną obserwowanego stanu jest brak rynku usług informatycznych. Rynek ten nie może powstać z trzech powodów. Po pierwsze — ośrodki informatyczne i zatrudniona w nich kadra nie są w istocie zainteresowane oferowaniem i sprzedażą usług; system ekonomiczny większości ośrodków nie zachęca do działań rynkowych, a system płac nie rekompensuje zwiększonego wysiłku. Po drugie — kadra ośrodków nie jest dobierana i przygotowana do działań usługowych. Po trzecie — istnienie rynku jest integralnie związane z systemem informacji, wiążącym dostawców i odbiorców. Skuteczne funkcjonowanie tego systemu, poza ujednoliceniem informacji umożliwiającym porozumienie, wymaga istnienia środków i struktur zapewniających powstawanie i krążenie informacji handlowej.

Uważam, że nie możemy czekać, aż w sposób naturalny rynek wytworzy struktury i środki informacyjne (powstaną struktury biurokratyczne). Władze centralne, a w tym SKI, lub inny podobny organ — muszą się tu stać inicjatorem i mecenasem.

GRAŻYNA KLAJN-ZIENKIEWICZ
Sekretariat Komitetu Informatyki

54 Międzynarodowe Targi Poznańskie

Inwazja mikroprocesorów

54 Międzynarodowe Targi Poznańskie (13—22 VI 82) zwiedziłem poszukując nowości informatyki i elektroniki oraz automatyki i pomiarów. Uważam bowiem, że dziedziny te coraz silniej przenikają się wzajemnie i nie powinno się ich rozdzielać. Spotkanie z informatyką na wyjątkowych pod wieloma względami [1]) Targach Poznańskich'82 było dla mnie miłym zaskoczeniem. Podzielałem pogląd, że wytwórcy krajowego sprzętu informatycznego załamią się — jako jedni z pierwszych — pod ciężarem kryzysu. Byłoby to logicznym następstwem niekorzystnej sytuacji tej branży na tle ogólnych trudności gospodarczych. Trudno powiedzieć, jak będzie wyglądała realna produkcja, postęp techniczny jest jednak wyraźny i to pozwala na odrobinę optymizmu.

Poniższa relacja z Targów nie stanowi ścisłego „urzędowego" spisu wystawców ani — tym bardziej — poszczególnych wyrobów. Nie udało mi się zdobyć danych technicznych niektórych eksponatów, o kilku zaś brak było w ogóle informacji. Podstawą niniejszego tekstu są materiały udostępnione podczas 54 MTP przez producentów i wystawców.

Przegląd Targów zacznijmy od pawilonów krajów RWPG. Będzie on stanowił dobre tło dla prezentacji oferty krajowej.

ZWIĄZEK RADZIECKI przedstawił:

● **Minikomputer typu SM-1800,** przeznaczony do sterowania procesów produkcyjnych, automatyzacji badań doświadczalnych oraz do prac obliczeniowych. Jest on zbudowany w oparciu o mikroprocesor typu K580IK80 [2]). W skład zestawu — oprócz jednostki centralnej, zawierającej m.in. procesor i półprzewodnikową pamięć operacyjną — wchodzą: monitor ekranowy, pamięć na dysku elastycznym oraz drukarka mozaikowa. Przewidziano trzy wersje systemu operacyjnego, w tym jedną do pracy w czasie rzeczywistym. Oprogramowanie składa się ponadto z asemblera, języka PL/M i najprostszej wersji BASICA.

● **Zestaw SM-2M,** należący do systemu małych maszyn cyfrowych (SM EMC). Powstał on w wyniku modernizacji SM-2 i ma podobne przeznaczenie (sterowanie procesami technologicznymi, opracowywanie wyników doświadczeń, obliczenia naukowo-techniczne). Zachowano pełną zgodność oprogramowania z systemami M-7000 i SM-2, a jednostronną — z M-6000. Wspólny jest również standardowy sprzęg typu 2K. Nowy procesor typu A131-15 nawiązuje do architektury M-7000. Jest on jednak bardziej wydajny [3]) i ma bogatszą listę rozkazów, m.in. operacji zmiennoprzecinkowych. Może tworzyć zestawy dwuprocesorowe. Pamięć mikroprogramów ma 4 K słów 36-bitowych i czas dostępu 260 ns. Pamięć operacyjna typu A211-20 ma pojemność 32 K słów 22-bitowych, z czego 6 bitów jest bitami kontrolnymi. Czas cyklu wynosi 1 μs. Maksymalna konfiguracja SM-2M może zawierać 4 bloki takiej pamięci, 2 bloki procesora A131-15 (zawierające również kanały bezpośredniego dostępu do pamięci) oraz 3 bloki we/wy umożliwiające przyłączenie do 52 urządzeń peryferyjnych, w tym 4 działających selektorowo. Jest oferowane bogate oprogramowanie: systemy operacyjne — dyskowy i bezdyskowy oraz czasu rzeczywistego, dość duża biblioteka programów. Użytkownik może korzystać z FORTRANU II i IV, dialektu ALGOLU-60 oraz BASICU.

WĘGRY wystawiły:

● **Nową wersję znanego programowanego kalkulatora stołowego 666,** oznaczoną jako 666B. Ma on możliwość współpracy z takimi urządzeniami zewnętrznymi, jak: czytniki i dziurkarki taśmy papierowej, drukarki pamięci na dysku elastycznym i pisaki x-y, a także możliwość sprzęgania z aparaturą pomiarową wg standardu IEC 625. Kalkulator 666B jest programowany w sposób klasyczny, typowy dla starszych kalkulatorów stołowych. Liczby, na których operuje, mają 12-cyfrową mantysę i 2-cyfrowy wykładnik. Dodawanie i odejmowanie trwa 0,15—0,25 ms, mnożenie i dzielenie 1—3 ms, a obliczanie wartości funkcji standardowych — od 20 do 80 ms. Ma wbudowaną pamięć kasetową o prędkości 240 bajtów/s oraz jest wyposażony w pamięć operacyjną o pojemności równoważnej 1008 rejestrów danych lub 8000 instrukcji programu. Model ten pomimo poprawy niektórych parametrów należy ocenić jako przestarzały, a cały system jako nieperspektywiczny.

[1]) Zjawiły się tylko nieliczne firmy zachodnie. Nie zauważyłem przy tym takiej, w której dominowałaby informatyka

[2]) Odpowiednik znanego mikroprocesora 8080

[3]) Czas wykonywania operacji: a) stałoprzecinkowych — dodawanie 2,2 μs, mnożesnie — 10 μs, b) zmiennoprzecinkowych — dodawanie 13—26 μs, mnożenie — 20 μs

BUŁGARIA zaprezentowała:

● **System do obróbki tekstów IZOT 1002S.** Pozwala on na tworzenie, redagowanie, przechowywanie i drukowanie tekstów napisanych zarówno alfabetem łacińskim, jak i cyrylicą. Centralną częścią systemu są dwa mikroprocesory produkcji bułgarskiej typu SM 601. Mikroprocesor podporządkowany steruje wydrukiem. Pamięć RAM, dynamiczna, ma pojemność 48 K bajtów a pamięć ROM — 10 K bajtów. Zestaw obejmuje 2 jednostki pamięci na dysku elastycznym 5074, monitor ekranowy (24 wiersze po 80 znaków) oraz drukarkę mozaikową z klawiaturą.

● **System minikomputerowy IZOT 1003S** przeznaczony jest do EPD, ze szczególnym uwzględnieniem specyfiki problemów gospodarki magazynowej.

● **Elektroniczne kasy rejestrujące** m.in. ELKA 81, elektroniczne taksometry itp.

NRD pokazała:

● **Komputer biurowy ROBOTRON 5110.**

● **System do przetwarzania danych ROBOTRON 6401.**
Niestety, nie udało się uzyskać bliższych danych technicznych na temat tego sprzętu.

CSRS nie wystawiała sprzętu informatycznego.

KRAJOWĄ OFERTĘ można podzielić na kilka charakterystycznych działów.

Oprogramowanie

Dla MERY 400:

● System wielodostępny — SOM 3.1 z monitorem wielodostępu został opracowany przez producenta sprzętu (ZWPPiSM), Przedsiębiorstwo Systemów Komputerowych MERA-SYSTEM, Uniwersytet Warszawski oraz Instytut Maszyn Matematycznych.

● Język programowania LOGLAN, zaprojektowany i wykonany w Instytucie Informatyki UW, jest nowoczesnym, uniwersalnym językiem o składni wzorowanej na PASCALU oraz bogatej semantyce.

● System programowania BASIC-EXT umożliwia wielostronną pracę w rozszerzonym języku BASIC. System ten opracowano w Instytucie Okrętowym Politechniki Gdańskiej.

● Język PSG (Procedury Syntezy Grafiki) jest przeznaczony do programowania graficznych urządzeń wyjściowych komputerów. PSG opracowano w Instytucie Maszyn Matematycznych.

ZETO Wrocław oferowało m.in.:

● system informacji o zapasach BOMIS,

● System planowania i zarządzania produkcją oraz system obsługi imprez kongresowych.

Po latach zastoju oferta targowa w tej dziedzinie była zaskakująco bogata. Przedstawiono:

Grafika komputerowa

● MERA 7954 — graficzny monitor ekranowy, w którym wykorzystano wyświetlanie rastrowe. Wyświetlania alfanumeryczne i graficzne są niezależne od siebie, a oba obrazy nakładają się na ekranie. Matryca punktowa grafiki ma wymiary: 256 linii i 512 kolumn, natomiast obraz alfanumeryczny składa się z 32 linii po 64 znaki. Istnieje możliwość dodatkowego wyświetlania 128 symboli graficznych. Monitor ma pamięć zewnętrzną na dyskach elastycznych SP 45 DE (1,6 M bajta). Przyłączenie do systemu cyfrowego następuje poprzez standardowy sprzęg RS 232 C/V 24.

● DGM-180 — model drukarki graficzno-mozaikowej, pozwalającej na tworzenie obrazów na papierze metodą drukowania punktowego.

● UG-80 — mikrokomputerowy system graficzny. Może być wykorzystywany jako niezależny system mikrokomputerowy lub jako inteligentny terminal konwersacyjny. Część centralna zawiera mikrokomputer z mikroprocesorem Z-80 oraz układy sterowania. Urządzenia zewnętrzne stanowią: 2 jednostki pamięci na dysku elastycznym PLx45, monitor ekranowy, klawiatura z piórem świetlnym (ewentualnie drukarka), digitizer lub ploter (autokreślarka). Oprogramowanie obejmuje: dyskowy system operacyjny FDOS-80, makroasembler, edytor, debuger, programy testujące, pakiet funkcji graficznych, a także FORTRAN IV i interpretator BASICA.

Powyższy sprzęt był wystawiany przez Zakłady Urządzeń Komputerowych MERA-ELZAB z Zabrza.

Monitory ekranowe

Rodzina monitorów ekranowych produkowana przez MERA-ELZAB wzbogaciła się o kilka nowych modeli.

● Podstawowy monitor MERA 7952 — pokazano w wersji MERA 7952 vgd. Jego ekran o przekątnej ok. 30 cm mieści 24 wiersze po 80 znaków. Klawiatura ma wydzieloną część numeryczną. Transmisja danych szeregowa, asynchroniczna, 75—9600 bitów/s. Sprzęg RS 232 C/V 24.Kod ASCII, pełny repertuar znaków (128)

● MERA 7953 vgd — udoskonalona wersja modelu 7952 vgd o identycznych podstawowych parametrach z dodaniem sprzęgu równoległego oraz licznych funkcji sterujących, umożliwiających pełną kontrolę ekranu i wydruk jego zawartości oraz wygodne posługiwanie się kursorem. MERA 7953 vgd jest funkcjonalnym odpowiednikiem monitora DEC VT 52 z układem klawiszy (wg standardu ANSI).

GIEŁDA INFORMACJI

Publikujemy bezpłatnie:

● **krótkie prezentacje nowych, nie wykorzystanych dostatecznie systemów**

● **doniesienia o innowacjach programowych**

● **ogłoszenia dotyczące nowych rozwiązań technicznych sprzętu informatycznego**

● **indywidualne oferty informatyków poszukujących pracy oraz informacje o wolnych etatach**

● **ogłoszenia w sprawie szkoleń informatycznych**

● **inne informacje ułatwiające doraźną organizację informatyki w Polsce.**

IV Szkoła Mikroprocesorowa

Centrum Szkolenia Informatycznego (CSI) przy ZETO-Łódź organizuje w dn. 26—28 stycznia 1983 r. — wspólnie z Instytutem Technologii Elektronowej „UNITRA-CEMI" (Warszawa) — IV Szkołę Mikroprocesorową Łódź'83. Głównym jej tematem będą systemy wieloprocesorowe i wielomaszynowe.

Zasadą Szkół jest prezentowanie prac autorskich. Organizatorzy proszą o przesyłanie do 31 grudnia 1982 r. do CSI: referatów i komunikatów oraz zgłaszanie do prezentacji systemów mikroprocesorowych, które będą wystawione podczas trwania sympozjum (atestacja, marketing, doradztwo techniczne i oprogramowanie). Tematyka Szkoły, z podziałem na kolejne dni obrad, jest następująca:

26.01. — Wielosystemy mikroprocesorowe

27.01. — Mikrokomputery jednostrukturowe

28.01. — Pokazy programów z biblioteki „B.ITE.M" oraz mikrosystemów autorskich.

Dodatkowych informacji udziela w Łodzi mgr inż. Stanisław Rybarczyk — tel. 647-70, teleks 885208.

Wszyscy prelegenci — autorzy referatów i komunikatów oraz osoby prezentujące systemy mikroprocesorowe są zapraszani na koszt organizatora. Natomiast uczestnicy sympozjum muszą przesłać — pod adresem: **Centrum Szkolenia Informatycznego, ZETO-Łódź, ul. Hutora 69, 90-558 Łódź** — pisemne, imienne zgłoszenie, zawierające oświadczenie o wpłaceniu na konto: ZETO-Łódź w NBP I o/m Łódź nr 47018-2219 kwoty 2800 zł od osoby. Uczestnicy zostaną zakwaterowani w bazie noclegowej przy ul. Hutora 69 w Łodzi już od doby 25/26 stycznia (płatne na miejscu) lub w pobliskim hotelu miejskim.

GIEŁDA INFORMACJI

DLA MERY-100

ZETO Wrocław oferuje użytkownikom minikomputera MERA-100:

— **Dyskowy system operacyjny FDOS** — moduł programowy ułatwiający obsługę urządzeń przy pracach nad oprogramowaniem użytkowym, zapewniający realizację standardowych funkcji manipulowania zbiorami, konwersję zbiorów między nośnikami itp. System daje możliwość organizacji wyszukiwania informacji w dużych zbiorach danych (dostęp indeksowy) oraz możliwość wykorzystania języka wyższego rzędu BASCOF (zbliżonego do COBOLU).

— **Translator języka MAKROASSEMBLER MAS1**, pracujący pod systemem operacyjnym FDOS. Język ten zapewnia realizację oprogramowania aplikacyjnego i systemowego: systemów pracujących pod kontrolą FDOS i samodzielnie, systemów nakładkowych, a także oprogramowania komunikacyjnego (emulatory). Zagadnienia, które nie mogą być programowane w języku BASCOF, można oprogramować za pomocą tego translatora.

— **Emulator stacji wsadowej ICL 7020 (ICL 7503).** Umożliwia on wykorzystanie MERY-100 jako zdalnej stacji wsadowej — współpracującej on-line z ODRĄ 1305 lub off-line z inną stacją ICL 7020. Może być stosowany we wszystkich przypadkach, w których wymagane jest przesyłanie z dalszych odległości dużych partii informacji do/z komputera ODRA 1305.

Bliższych informacji udziela: kierownik Działu Marketingu, mgr Tadeusz Czerniewski; ZETO Wrocław, ul. Ofiar Oświęcimskich 7/13, 50-069 Wrocław, tel. centr. 445-431 do 37, teleks 0712533 ZETO PL.

PTI

Polskie Towarzystwo Informatyczne ma możliwość organizowania zespołów specjalistów wysokiej klasy, gwarantujących szybkie i wielokrotnie tańsze wykonanie zleceń w zakresie:

- pomocy przy rozpakowaniu systemów operacyjnych lub pakietów programowych wraz ze szkoleniem ich użytkowników
- pomocy w wyszkoleniu obsługi technicznej sprzętu, zapewniającej wysoką jego dyspozycyjność (np. dysków)
- wykonywania ekspertyz projektów informatycznych
- pomocy przy organizowaniu ajencyjnego serwisu minikomputerów.

Adres do korespondencji: **Sekretarz Generalny PTI, Andrzej Wiśniewski, Zarząd Główny PTI, ul. Jasna 14/16 pok. 338, 00-041 Warszawa.**

- MERA 7954 monitor graficzny, który został już powyżej scharakteryzowany.

- MERA 7956 — kolorowy monitor „półgraficzny", przeznaczony do systemów minikomputerowych. Wyświetla 95 znaków kodu ASCII oraz 96 dodatkowych znaków graficznych. Znak jest tworzony w polu 8 × 8 kropek, 22-calowy ekran mieści 32 wiersze po 64 znaki. Możliwe jest niezależne sterowanie kolorem znaku i tła. Do wyboru mamy kolory: biały, fioletowy, żółty, czerwony, niebieski, zielony, niebiesko-zielony, czarny. Pełne sterowanie kursorem. Pole atrybutów pozwala m.in. na migotanie tekstu, jego ochronę itp. Szybkość transmisji do 9600 bodów, sprzęg RS 232C/V 24 lub pętla prądowa.

Systemy terminalowe

Modułowy system terminalowy MST-8000 jest wprowadzany do produkcji w MERA-ELWRO. Składa się on ze sprzętu i oprogramowania umożliwiającego budowę specjalizowanych terminali. Pozwala również na tworzenie systemów terminalowych o działaniu bezpośrednim. Zestaw modułów obejmuje: jednostki sterujące, klawiatury, urządzenia wejścia, monitory, drukarki, zasilacze, pamięci zewnętrzne oraz urządzenia transmisji danych.

Pracę terminali organizują koncentratory, które zapewniają ich współdziałanie z systemem komputerowym. Transmisja danych szeregowa. W wersji lokalnej (do 500 m) szybkość transmisji wynosi 56 Kb/s, natomiast praca zdalna wymaga modemu 1200/2400 b/s. Obowiązuje protokół BSC. Podstawowym modułem terminali i koncentratorów jest jednostka sterująca, będąca w istocie mikrokomputerem z rodziny 8080.

Systemy mikroprocesorowe

System wspomagający uruchamianie systemów mikrokomputerowych o nazwie RTDS-85 był wystawiany przez MERA-ELZAB oraz Zakład Systemów Automatyki Kompleksowej PAN w Gliwicach. System jest modularny, pozwala projektować i uruchamiać systemy na bazie 8080/8085. Najistotniejsze jest to, że pozwala on na emulację i śladowanie przebiegu programu w czasie rzeczywistym. Część sprzętowa RTDS-85 obejmuje: procesor, pamięć operacyjną o pojemności 16KB, sterownik dysków elastycznych, monitor ekranowy, drukarkę, stację taśmy papierowej, pamięć śladu 1 K słów 32-bitowych, pamięć emulowaną 16 KB, sondy emulujące i odpowiednie do nich adaptery. Oprogramowanie RTDS-85: dyskowy system operacyjny z biblioteką zawierającą asembler, edytor, debuder, testy, FORTH, BASIC 4 KB i BASIC 24 KB. Przewiduje się rozszerzenie o PL/M i FORTRAN.

Prosty sprzęt wspomagający uruchamianie systemów mikroprocesorowych był reprezentowany przez analizatory stanów logicznych typu ANISS-10 i ANISS-20 (Instytut Systemów Sterowania) oraz wielofunkcyjny przyrząd mikroprocesorowy WMP-2 (UNITRA-CEMI, Zakład w Szczytnie).

Uniwersalne, modułowe systemy mikroprocesorowe były oferowane przez kilku wystawców i można spodziewać się, że w bliskiej przyszłości wystąpi tu konkurencyjna walka. A oto kolejne propozycje tych systemów.

- Modułowy system mikroprocesorowy MSM — wystawiony przez Przedsiębiorstwo polonijno-zagraniczne IMPOL z Warszawy. Jest to uniwersalny system mikroprocesorowy, z którego modułów można zestawiać mikrokomputery do sterowania procesami, regulacji automatycznej, pomiarów, obliczeń inżynierskich i prac biurowych. Standard sygnałów w kasecie systemowej BUSMAT pozwala na adresowanie do 1 MB pamięci oraz na pracę wieloprocesorową. Można wybrać jednostkę centralną z mikroprocesorem 8080A, 8085, Z 80, 6800, 8088, 6809. Istnieją ponadto moduły pamięci, wejścia i wyjścia, klawiatury. W przygotowaniu jest szereg modułów sprzężenia z otoczeniem oraz... moduł syntetyzera mowy! Całość uzupełnia oprogramowanie, obejmujące również BASIC.

- Sterownik mikroprocesorowy ZSA-MIKRO-80 był oferowany po raz kolejny na Targach) przez Zakłady Systemów Automatyki w Poznaniu. Jest on przeznaczony do sterowania dowolnego obiektu, automatyzacji pomiarów itp. Może on składać się z następujących pakietów: procesora, pamięci ROM i RAM, zegarów, transmisji, wejść i wyjść dwustanowych, przetworników A/C i S/A itp. Całość uzupełnia oprogramowanie: monitor, edytor, asembler.

- Uniwersalny, modułowy system mikroprocesorowy oferowało również Naukowo-Produkcyjne Centrum Półprzewodników w Warszawie. Eksponat w gablocie nie był niestety uzupełniony żadnymi informacjami.

- Mikroprocesorowy sterownik do modularnego systemu CAMAC (blok 180) oferował również POLON Warszawa. Brak było konkretnych danych technicznych.

- Sterownik przemysłowy SUM-40 oferowało PPZ MARCO-ELECTRONIC z Krakowa.

- Uniwersalny system mikroprocesorowy USM-80 był wystawiony przez PPZ DANPOL z Gdyni. Niestety, i tu brak było istotnych informacji.

Gdybym mógł przedstawiać do nagrody za nowoczesność wystawionych na tegorocznych MTP opracowań i urządzeń to wybrałbym następujące:

- w dziedzinie oprogramowania — język LOGLAN (Uniwersytet Warszawski)

- w dziedzinie sprzętu oferowanego przez przedsiębiorstwa państwowe — system wspomagający RTDS-85 (MERA-ELZAB i ZSAK, Zabrze-Gliwice) [4])
- w dziedzinie sprzętu oferowanego przez przedsiębiorstwa polonijno-zagraniczne: modułowy system mikroprocesorowy MSM (IMPOL, Warszawa)
- w dziedzinie zastosowań (za wybitne podwyższenie walorów użytkowych nowoczesnego sprzętu przez wprowadzenie informatyki) — SPC-100 centrala telefoniczna z rodziny modułowych central systemu PENTACONTA, sterowana zestawem mikroprocesora typu 8080; wystawiana była przez Zakłady Wytwórcze Urządzeń Telefonicznych w Warszawie.

* * *

Na koniec kilka ogólniejszych refleksji.

Cieszy fakt, że producenci zaczynają posługiwać się pojęciem ceny, czego dotychczas nie było. Cieszy również i to, że pomimo znanych trudności poligraficznych było dość dużo dobrych opisów katalogowych. Co prawda — są jeszcze firmy (powiedzmy — co piąta), które nie bardzo wiedzą po co przybyły na targi. Objawia się to m.in. lekceważeniem potencjalnego klienta. Ta przykra choroba nie ominęła nawet niektórych przedsiębiorstw polonijno-zagranicznych.

Co chciałbym zobaczyć na przyszłorocznych Targach? Przede wszystkim więcej oprogramowania, i to lepiej pokazanego. Sprzęt jest wystawiać łatwo: klient może go dotknąć, reszty dopełnia ulotka, plansza. Oprogramowanie nie jest towarem łatwym do prezentacji. Wymaga specjalnych środków, aby trafiło do odbiorcy. Opracowanie i rozpowszechnianie oprogramowania nie wymaga dewiz na import składników, nie wymaga surowców i wielkich ilości energii... Mamy zdolnych informatyków. Czekam, kiedy oprogramowanie stanie się normalnym towarem, kiedy będzie można w nim wybierać. Może wkrótce zobaczymy u nas małych, „niezależnych" producentów oprogramowania.

JACEK ŻEBROWSKI

[4])Nie jest to jedyny w Polsce dobry system wspomagania tej klasy. Mniej więcej rok temu Instytut Maszyn Matematycznych informował, że w 1982 r. przewidywana jest produkcja Mikroprocesorowego Systemu Wspomagania Projektowania będącego odpowiednikiem systemu MDS-800 firmy INTEL. Nie znalazłem tego systemu na Targach'82.

SICOB'81

Poniższa relacja jest bardzo spóźniona. Postanowiliśmy jednak nie rezygnować z niej — tak, by opis wielkich spotkań informatycznych SICOB był na naszych łamach możliwie kompletny. Bogatszą relację z SICOBU'82 przedstawimy w jednym z najbliższych numerów. (Red.)

W dniach od 23 września do 2 października 1981 r. w Paryżu po raz 32 odbyła się doroczna międzynarodowa wystawa-targi SICOB (Salon International d'Informatique, Télématique Communication, Organisation du Bureau, et Bureautique). Zgodnie z tradycją wystawa miała trzy dni zarezerwowane dla profesjonalistów (23—25 września), natomiast dni pozostałe przeznaczone były dla szerokiej publiczności. Podobnie jak w ostatnich latach wystawie nadano wysoką rangę państwową, zaakcentowaną udziałem François Mitteranda w ceremonii otwarcia. W przemówieniu inauguracyjnym prezydent Francji podkreślił aspekty humanistyczne, polityczne i ekonomiczne informatyki, zwracając uwagę na jej szczególnie dynamiczny rozwój, a zwłaszcza szybko wzrastającą liczbę wdrożeń systemów informatycznych w małych przedsiębiorstwach.

W olbrzymim pawilonie wystawowym dzielnicy Defanse prezentowały swoje oferty handlowe wszystkie największe firmy, takie jak: IBM, CII-HB, ICL, BURROUGHS, UNIVAC, OLIVETTI, SIEMENS, HEWLETT PACKARD, THOMSON-CSF MAI. Z krajów RWPG obecna była jedynie firma ROBOTRON z NRD.

Zgodnie z obserwowaną od kilku lat tendencją, szczególnie dużo było systemów przystosowanych do potrzeb małych i średnich przedsiębiorstw. Cechą charakterystyczną ekspozycji był prawie całkowity brak prezentacji dużych systemów komputerowych. Asortyment prezentowanego sprzętu i oprogramowania był wyjątkowo bogaty. I tak, między innymi, firma IBM przedstawiła minikomputer IBM 5120, minikomputer IBM 34, komputery Serii 1, system IBM 5280 dla przetwarzania rozproszonego oraz system terminali dla przedsiębiorstw handlowych typu IBM 5260.

Firma CII-HB prezentowała minikomputer MINI 6, mikrokomputer QUESTAR M, systemy średniej wielkości PPS 4-62-61/DPS. Firma ICL proponowała system 25, DRS 20 z modelami 10, 40 i 50 oraz w zakresie oprogramowania — cieszący się dużym powodzeniem — system MASTOC, przeznaczony do sterowania magazynami, zakupem i sprzedażą.

Firma MAI oferowała systemy MAI 80, MAI 210, MAI 510, MAI 610, MAI 730, firma OLIVETTI — system BCS 2025, firma BURROUGHS — system B 920, a firma HEWLETT PACKARD — minikomputer HP-85.

Znaczną część ekspozycji stanowiły mikro i minikomputery zapewniające możliwość zarówno pracy samodzielnej, jak i współpracy z innymi systemami poprzez sieć telefoniczną. Sprzęt tego typu, odznaczający się szczególnie małymi gabarytami, niską ceną, a jednocześnie stosunkowo wysokimi parametrami technicznymi zdobywa sobie coraz więcej zwolenników. Przykładem takim może być produkt firmy SMT o nazwie GOUPIL 2, który przy cenie zaledwie 6995 F (ok. 1200 $) budził znaczne zainteresowanie wśród zwiedzających.

Jak wiadomo, wystawa SICOB stwarza nie tylko dogodne warunki dla reklamy możliwości technologicznych, ale i sprzedaży prezentowanych produktów. Z uwagi na fakt, że każdy z potencjalnych użytkowników patrzy na produkty te przez pryzmat swoich potrzeb, ekspozycja SICOB'81 organizowana została pod hasłem zaspokajania rzeczywistych potrzeb użytkowników. W związku z tym prezentowano nie tylko możliwości sprzętu, lecz w coraz większym stopniu akcentowano jego oprogramowanie.

Oprócz firm produkujących sprzęt komputerowy, na wystawie obecne były także firmy typowo usługowe, takie jak: STERIA, CISI, CGI, SEMA, oferujące usługi w zakresie doradztwa oraz metod i narzędzi projektowania systemów informatycznych.

Swoistym novum tegorocznych targów był fakt reklamowania przez niektóre firmy systemów firm konkurencyjnych. Nierzadkie były fakty kierowania potencjalnego użytkownika do konkurenta, w przypadku nie dysponowania systemem adekwatnym do zgłaszanych potrzeb. Czyżby, jak podkreśla francuska prasa, nadszedł nowy okres w historii międzynarodowych imprez handlowych, preferujący uczciwość zawodową?

DANUTA SEGIET

Japonia

Piąta generacja

Piąta generacja komputerów jest obecnie jednym z największych kompleksowych projektów światowej informatyki. Japońskie przedsięwzięcie zakłada stworzenie w ciągu najbliższego dziesięciolecia rodziny inteligentnych systemów, nieporównywalnych z obecnymi komputerami. Prace i badania japońskich naukowców nastawione są głównie na stworzenie nowej relacji człowiek-maszyna, czyli takiego „inteligentnego interfejsu", który akceptowałby wszystkie rodzaje danych: numeryczne, tekstowe, graficzne, ludzki głos itd. Nowe systemy, dostępne dla każdego człowieka, byłyby stosowane we wszystkich niemal dziedzinach życia społecznego.

Prace nad systemem piątej generacji już rozpoczęto. Niedaleko od centrum Tokio, w 22-piętrowym budynku, znajduje się siedziba Instytutu Komputerów Nowej Generacji ICOT. Instytut ten skupia zaledwie 40 pracowników — najwybitniejszych specjalistów czołowych firm informatycznych i elektronicznych.

SUPERKOMPUTERY

Jednym z celów piątej generacji są superkomputery, których konstrukcją zajmuje się aż pięć grup pracujących w oparciu o różne projekty. Trzy spośród nich nastawione są na osiągnięcie konkretnych handlowych rezultatów w ciągu najbliższych 18 miesięcy. Czwarty projekt stanowią prace badawcze nad możliwościami rozszerzania klasycznej architektury opartej na ideach Von Neumanna. Ostatni z tej piątki ma przynieść rezultaty dopiero w końcu bieżącego dziesięciolecia.

Wiele przedsiębiorstw — nie czekając na realizację głównych projektów — współzawodniczy między sobą, prowadząc własne badania. HITACHI proponuje zastosowanie w komputerach uniwersalnych największej mocy procesora macierzystego (ang. *array processor*), NEC pracuje nad maszyną o strukturze strumieniowej (ang. *pipeline*), a FUJITSU proponuje wprowadzenie na rynek — zimą 1983/1984 — systemu o mocy obliczeniowej równej mocy komputera CRAY 1. Trwają też prace nad maszyną, której moc obliczeniowa byłaby 100-krotnie większa od mocy CRAY 1.

Warto zauważyć, że to właśnie firma FUJITSU uruchomiła uniwersalny system 380 o prędkości 25 mln operacji na sekundę, a więc jeden z najszybszych komputerów świata. Nowa maszyna nie opiera się na zasadach architektury Von Neumanna, lecz tworzy system równoległy używający jednostek logicznych z trzema tysiącami bramek i czasem reakcji 10 pikosekund, jednostkami pamięci, o pojemności 16 K bajtów i czasem dostępu mniejszym od 10 nanosekund.

Wyrazem szczególnie szybkiego postępu jest wprowadzany obecnie do produkcji w Japonii procesor wyspecjalizowany w obliczaniu transformacji Fouriera (zbudowany oczywiście z podzespołów VLSI). Ten typ procesora odegra szczególną rolę w rozpoznawaniu i syntezie mowy — zagadnieniu, które można uznać za serce projektu piątej generacji. A pamiętajmy, że język japoński używa ok. 2 mln różnych znaków, co wyklucza np. budowę maszyny do pisania według europejskich wzorów.

Mówiąc o superkomputerach nie można nie wymienić systemu skonstruowanego przez studentów uniwersytetu w Tokio, pod kierunkiem profesora Eiichi Goto. System ten o prędkości 10 mln operacji na sekundę jest najbardziej zaawansowanym rozwiązaniem w dziedzinie badań mikrokodu. Na przykład — wszystkie instrukcje skoku w systemie są odwzorowane przez odpowiednie układy VLSI. Trzeba również podkreślić, że projekt studentów z Tokio nie korzystał z żadnych specjalnych dotacji — sfinansowano go z normalnego budżetu uczelni.

UKŁADY VLSI

Piąta generacja to świat układów VLSI. Ale z czego będą robione te układy? Obecnie wszystkie środowiska japońskich elektroników debatują nad tym problemem.

Maszyna-pilot, która zostanie zrealizowana w ciągu najbliższych trzech lat, będzie prawdopodobnie oparta na układach z konwencjonalnego krzemu. Nic nie wiadomo natomiast na temat „budulca" przyszłych układów VLSI. Po pierwsze — jaki materiał będzie następcą krzemu? Po drugie — z jednej strony mamy zwolenników elementów z efektem Josephsona, a więc z koniecznością super-chłodzenia, z drugiej — zwolenników arsenku galu (arsenek galu wymaga mniejszego chłodzenia i może funkcjonować z ograniczoną prędkością w temperaturze otoczenia).

Profesor Tohru Oka z uniwersytetu w Tokio, przewodniczący Komitetu do Spraw Rozwoju Superkomputerów, wyjaśnia, że dla obliczeń numerycznych oraz szybkiego przetwarzania danych wybór dokonają się pomiędzy elementami z efektem Josephsona i Hemta, natomiast arsenek galu jest doskonały dla zastosowań związanych z przetwarzaniem obrazów.

Dr Katakoa z laboratorium elektroniki MITI (Ministerstwo Handlu Zagranicznego i Przemysłu) ocenia, że prędkość superkomputerów będzie około 10-krotnie większa od osiąganej obecnie. Ten ogromny postęp zapewni nowa architektura superkomputera oraz wykorzystanie możliwości tkwiących w istocie układów VLSI. Prawdopodobnie nie jest to jeszcze granica możliwości wynikłych z kombinacji tych dwóch czynników. Być może ograniczenia VLSI wynikają bardziej z niedostatków ludzkiej wyobraźni niż ograniczeń technologicznych.

TRZECI WYMIAR

Czy technologia elektroniczna musi być ograniczona do dwóch wymiarów płaszczyzny? W październiku 1981 r. ministerstwo MITI włączyło trójwymiarową elektronikę do dziesięcioletniego programu rozwoju. Trzeci wymiar pozwoli konstruować elementy nowego typu, co zdaniem Japończyków znacznie rozszerzy możliwości integracji. Technologia planarna z trudem dopuszcza więcej niż 16 M bitów pamięci w jednym układzie, podczas gdy dołożenie trzeciego wymiaru daje szanse uzyskania pojemności 256 M bitów.

DLACZEGO PROLOG

Po wielu dyskusjach zdecydowano, że językiem przyszłych systemów piątej generacji będzie PROLOG. Dlaczego właśnie on, a nie np. LISP? LISP ma liczne ograniczenia — jest językiem zbudowanym w oparciu o logikę deterministyczną, natomiast PROLOG zawiera wszelkie potrzebne dla systemów piątej generacji narzędzia programowania logicznego. PROLOGU używa się obecnie we Francji, Wielkiej Brytanii, Japonii, a także na Węgrzech. Według pana M. Fuchi, dyrektora wspomnianego Instytutu Komputerów Nowej Generacji: *„PROLOG realizuje syntezę osiągnięć w dziedzinach logiki, systemów baz danych, sztucznej inteligencji, a także przetwarzania języka naturalnego oraz architektury systemów. Programowanie logiczne możemy uważać za połączenie programowania funkcjonalnego i relacyjnych baz danych".*

Decydując się na PROLOG, konstrukturzoy piątej generacji określili nową jednostkę przetwarzania — operację dedukcji. Miarą mocy obliczeniowej nowego systemu będzie liczba wykonywanych logicznych operacji dedukcji na sekundę (fran. Lips — *inférences logiques par seconde*). Jednostka nowej miary równoważna jest od 100 do 1 mln instrukcji maszynowych na sekundę. Oczekuje się, że komputery piątej generacji osiągną moc 100 mln, a nawet miliarda Lips.

Oprac. MARIANNA SOBCZYK
na podstawie „01 Informatique"
nr 716-717

Bibliografia wydawnictw polskich z dziedziny informatyki

● **Postępy komputeryzacji w okrętownictwie — JAGODA J., ZIELIŃSKI S., Wydawnictwo Morskie, Gdańsk 1980, s. 128, cena 18 zł Biblioteka Okrętownictwa, seria: Postęp w okrętownictwie**

Ogólne zagadnienie komputeryzacji. Systemy komputerowe w zarządzaniu. Zastosowanie komputerów w projektowaniu i konstruowaniu w instytucjach klasyfikacyjnych i procesach technologicznych. Komputerowe systemy do projektowania i produkcji. Książka przeznaczona jest dla krajowego środowiska okrętowców zainteresowanych zagadnieniami komputeryzacji okrętownictwa. Opracowano ją na podstawie materiałów trzeciej międzynarodowej konferencji ICCAS-79 w Glasgow.

● **Projektowanie systemów informatycznych — BUCHTA D. (red.), Wyd. Akademii Ekonomicznej im. Karola Akademickiego, Katowice 1980, s. 315, cena 25,50 zł**

Wiadomości wstępne o projektowaniu systemów informatycznych. Projektowanie wstępne i szczegółowe systemu informatycznego. Projektowanie wejść i wyjść. Zbiory systemu i technologia ich przetwarzania. Opracowanie założeń do programowania i dokumentacji eksploatacyjnej systemu informatycznego. Załączniki. Skrypt przeznaczony jest dla studentów wyższych uczelni ekonomicznych.

● **Środki bezpieczeństwa i plany działań w sytuacjach awaryjnych. Tłum. wyd. ang. z 1980 r. Wyd. Zjednoczenia Informatyki. Centrum Projektowania i Zastosowań Informatyki, Warszawa 1981, s. 171, cena 126 zł**

„Europejski Program Badawczy Diebolda". Zeszyt 120 (180 M 54) Podsumowanie wyników badań. Obecne kierunki rozwoju: fizyczne środki bezpieczeństwa, środki zabezpieczenia danych, administracyjne środki bezpieczeństwa. Ocena stanu bezpieczeństwa. Obecnie praktykowane metody planowania bezpieczeństwa. Udoskonalenie programu bezpieczeństwa. Problemy przyszłego planowania bezpieczeństwa.

Materiały przeznaczone są dla użytkowników i projektantów systemów informatycznych.

● **Kierunki rozwoju usług informacyjnych 1980—1985. Tłum. wyd. ang. z 1979 r. Wyd. Zjednoczenia Informatyki. Centrum Projektowania i Zastosowań Informatyki. Warszawa 1981, s. 108, cena 126 zł**

„Europejski Program Badawczy Diebolda" Zeszyt 119 (E 176).

Usługi informacyjne. Tendencje w usługach informacyjnych. Wpływ usług informacyjnych na zarządzanie systemami informacji kierownictwa. Wpływ kierunków rozwoju na dostawców usług informacyjnych.

Materiały przeznaczone są dla kadry kierowniczej przedsiębiorstw i ośrodków obliczeniowych oraz użytkowników usług informacyjnych.

● **Zastosowanie metod optymalizacyjnych w planowaniu produkcji — PLESZCZYŃSKI S. Wyd. Zjednoczenia Informatyki. Centrum Projektowania i Zastosowań Informatyki, Warszawa 1981, s. 124, cena 82 zł**

„Problemy Informatyki"

Ogólne zagadnienia programowania liniowego. Optymalizacja asortymentowych planów produkcji. Wybrane zagadnienia optymalizacji planów rozwoju produkcji i bazy.
Materiały przeznaczone są dla kadry kierowniczej przedsiębiorstw przemysłowych i projektantów systemów informatycznych.

● **Efektywne kierowanie projektowaniem systemów EPD. Tłum. wyd. ang. z 1978 r. Wyd. Zjednoczenia Informatyki Centrum Projektowania i Zastosowań Informatyki, Warszawa 1981, s. 159, cena 126 zł**

„Europejski Program Badawczy Diebolda". Zeszyt 114 (E 169). Tendencje w kierowaniu projektami EPD. Efektywne kierowanie projektami. System MAP (dieboldowski podręcznik kierowania projektami EPD — Diebold Manual of ADP Project Management). Załączniki.

Materiały przeznaczone są dla kierowników projektów EPD.

● **Doroczny przegląd technologiczny Diebolda. Tłum. wyd. ang. z 1979 r. Wyd. Zjednoczenia Informatyki. Centrum Projektowania i Zastosowań Informatyki. Warszawa 1981, s. 146, cena 126 zł „Europejski Program Badawczy Diebolda". Zeszyt 116 (E 177).**

Architektura systemów informacyjnych. Pomoce do tworzenia i wdrażania programów użytkowych. Oprogramowanie systemowe i sprzętowe. Sprzęt i oprogramowanie sprzętowe systemów. Inżynieria programowa.
Materiały przeznaczone są dla kadry kierowniczej przedsiębiorstw oraz dla projektantów systemów informatycznych.

● **Zastosowanie informatyki w zarządzaniu procesami inwestycyjnymi — GRABSKI A., PWN, Warszawa 1981, s. 224 cena 30 zł**

Cz. 1. Podstawy teoretyczne: Model procesu inwestycyjnego. Cybernetyczny model procesu zarządzania. Przepływ i przetwarzanie informacji w zarządzaniu. Podstawowe wiadomości z zakresu teorii podejmowania decyzji. Techniczne środki przetwarzania informacji. Cz. 2. Systemy informatyczne: Wiadomości ogólne o systemach informatycznych. Wybrane aspekty projektowania systemów. Systemy dla poszczególnych przedsięwzięć i zadań inwestycyjnych. Systemy dla jednostek wykonawstwa inwestycyjnego.
Materiały przeznaczone są dla inżynierów budowlanych specjalizujących się w zakresie organizacji i zarządzania budownictwem.

● **Użytkowanie elektronicznych maszyn cyfrowych — SZAFNICKI B., Wyd. Szkolne i Pedagogiczne, Warszawa 1981, s. 295, cena 20 zł**

Wprowadzenie. Zarys metod numerycznych. Podstawy programowania maszyn cyfrowych. Maszyny cyfrowe serii ODRA 1300. Język algorytmiczny ALGOL. Zastosowanie elektronicznych maszyn cyfrowych.

Podręcznik przeznaczony jest dla uczniów kl. V technikum elektronicznego o specjalności elektroniczne maszyny matematyczne.

● **Organizacja przetwarzania danych. Cz. 1. Narzędzia. Metody — NOWICKI A. (kier. pr. zb.), Wyd. Akademii Ekonomicznej im. Oskara Langego we Wrocławiu, Wrocław 1981, s. 273, cena 40 zł**

Podstawy organizacji przetwarzania danych. Środki techniczne informatyki. Projektowanie i efektywność systemów informatycznych. Wdrażanie systemu informatycznego w jednostce gospodarczej.

Skrypt stanowi pomoc dydaktyczną dla studentów wszystkich kierunków studiów stacjonarnych i zaocznych uczelni ekonomicznych, na których wykładany jest przedmiot „Organizacja przetwarzania danych".

● **Organizacja przetwarzania danych. Cz. 2. Ćwiczenia — MACIASZEK L. (kier. pr. zb.), Wyd. Akademii Ekonomicznej im. Osakra Langego we Wrocławiu, Wrocław 1981, s. 207, cena 30 zł**

Tytuły rozdziałów są identyczne jak w cz. 1, a zadania, testy i pytania odpowiadają zakresem tematycznym wiadomościom zawartym w cz. 1.

● **Podstawy programowania EMC. Zbiór zadań — LECH D., SOIŃSKA A., Wyd. Politechniki Częstochowskiej, Częstochowa 1981, s. 155, cena 12 zł**

Wprowadzenie. Schematy blokowe. Zadania elementarne. Organizacja rozgałęzień. Instrukcje cyklu. Podprogramy. Zadania różne. Wskazówki i odpowiedzi zadań.

Skrypt przeznaczony jest dla studentów uczelni technicznych rozpoczynających naukę podstaw informatyki.

● **Matematyczne podstawy informatyki — ANDRZEJEWSKI B., Wyd. Politechnik Lubelskiej, Lublin 1981, s. 297, cena 28 zł**

Maszyny Turinga. ALGOL 60. Obliczanie wartości wyrażeń Boole'owskich. Schematy blokowe. Elementy budowy komputera i jego działanie. Relacje i systemy relacyjne. Lingwistyka matematyczna. System informacyjny przedsiębiorstwa produkcyjnego. Zarys podstaw teorii informacji.

Skrypt przeznaczony jest dla studentów uczelni technicznych specjalizujących się w organizacji i zarządzaniu przedsiębiorstwami produkcyjnymi.

Oprac. A.K.

O oprogramowaniu

Wraz z rozwojem teorii i praktyki programowania komputerów wyodrębnia się wiele pojęć, w większości znanych wcześniej, lecz nie mających jeszcze ustalonego zakresu znaczeniowego. Poniżej zajmiemy się tymi pojęciami, których powstanie wiąże się przede wszystkim z dążeniem do zapewnienia wymienności oprogramowania, nie tyle użytkowego (do czego wystarczają języki wysokiego poziomu), lecz — głównie — podstawowego, co jest uzasadnione ogromnymi kosztami ponoszonymi na jego tworzenie, rozbudowę i utrzymanie.

Drugą istotną przyczyną powstawania całej serii nowych pojęć w dziedzinie oprogramowania podstawowego jest niebywały rozwój techniki mikroprocesorowej. Masowe wykorzystywanie mikroprocesorów, a więc także — przeznaczonych dla nich programów, spowodowało w ciągu ostatnich dziesięciu lat duże zmiany w technologii produkcji oprogramowania. Przykładowo — jako jedną z głównych tendencji można wymienić tworzenie oprogramowania dla mikroprocesorów na komputerach większych, np. na minikomputerach (tzw. *cross software*), co jest bardziej opłacalne.

Zjawiska takie towarzyszą w szczególności powstawaniu i upowszechnianiu się języka ADA (p. INFORMATYKA, nr 11-12/1981). Podstawowe dla posługiwania się tym językiem jest pojęcie tzw. *environment* (z ang., *środowisko, otoczenie*). Wraz z rozwojem prac nad językiem zainicjowano przedsięwzięcie polegające na stworzeniu tzw. APSE (ang. *Ada Programming Support Environment*). Według jednego z określeń *environment* jest to zbiór zintegrowanych narzędzi programowych, wspomagających programowanie.

O ile z wyborem polskiego odpowiednika słowa *environment* w zasadzie nie powinno być trudności, to mogą być wątpliwości dotyczące znaczenia tego pojęcia. Środowiskiem programowym języka nazywa się bądź całe oprogramowanie podstawowe, bądź oprogramowanie podstawowe bez kompilatora. Dotychczasowa praktyka wskazuje, że przyjmie się raczej drugie znaczenie, choć obecnie trudno jeszcze rozstrzygnąć to ostatecznie.

Oczywiście, najistotniejszy dla skutecznego posługiwania się językiem programowania jest kompilator. Sam termin kompilator nie powinien wzbudzać żadnych wątpliwości terminologicznych. Niemniej określenie podane w normie PN-71/T-01016 — *„program tłumaczący, przeznaczony do przekształcania programów napisanych w języku źródłowym na programy w języku wynikowym"* — jest nieprecyzyjne. Według normy ISO 2382 kompilator jest programem tłumaczącym programy wyrażone w języku problemowym na programy w języku maszyny. Pozostaje kwestią otwartą, czy na podstawie tego określenia można odróżnić kompilatory od innych programów, tłumaczących, np. od interpreterów i — czy w ogóle granice podziału są ostre (możliwe do przeprowadzenia). Być może w celu bardziej szczegółowego wyjaśnienia tych zagadnień należałoby głębiej wniknąć w budowę kompilatora.

Z całą pewnością należy to zrobić wtedy, gdy mamy do czynienia z kompilatorem przenośnym, określanym po angielsku *portable*. Program (a więc także kompilator) przenośny jest programem, który może być wykonywany na różnych komputerach. Choć niekiedy tę cechę programu nazywa się portabilnością, a sam program — portabilnym, nie sposób jednak tej formy uzasadnić. Nazwa portabilny nie wnosi do języka informatyki niczego więcej niż nazwa przenośny, a więc pod tym względem nie jest od niej lepsza. Ponadto nazwanie programem przenośnym jest naturalną konsekwencją faktu, że program można przenosić z jednego komputera na drugi. Trudo natomiast doszukać się takiego związku językowego dla przymiotnika portabilny — ze względu na jego pochodzenie.

W celu pełniejszego wyjaśnienia zagadnień związanych z pojęciem przenośności, należy odwołać się do terminów odnoszących się do tzw. *cross software* (częściowo omawialiśmy to już w tej rubryce, p. INFORMATYKA, nr 10/1980, nr 2/1981). Kompilator zewnętrzny (ang. *cross compiler*) działa na komputerze zwanym po angielsku *host* i tłumaczy program do wykonania na komputerze zwanym *target*. *Target computer* (*target machine*) jest komputerem, który wykonuje przetłumaczony program, a więc jest kompute-

Jak rozruszać informatykę

Sytuacja ekonomiczna i techniczna naszych ośrodków obliczeniowych coraz bardziej pogarsza się. Przy dalszym utrzymywaniu się obecnego trendu grozi nam znaczne obniżenie — i tak przecież niewysokiego — poziomu informatycznego kraju. Jestem wprawdzie zwolennikiem wdrażania zasad reformy gospodarczej również w informatyce, sądzę jednak, że w naszych warunkach nie jest ona w stanie utrzymać nawet dotychczasowego poziomu bez udzielenia jej zewnętrznego poparcia (które jest zresztą regułą we współczesnym świecie).

Użytkownicy wywierają dużą presję na obniżanie cen usług informatycznych. Wynikiem tego jest (mimo kilkakrotnego wzrostu kosztów) fakt, że tylko nielicznym ośrodkom udaje się przeprowadzić zwyżkę cen większą niż 40—50% w stosunku do dotychczas obowiązującego cennika Zjednoczenia Informatyki (nr 1-U/80). Niektóre ośrodki, działając pod presją klientów, uciekają się do ukrytych podwyżek cen (wykazując w tym względzie wyjątkową inwencję), jednakże nie jest to chyba metoda, która mogłaby okazać się skuteczną w dłuższym okresie. Dochodzi więc do znacznej relatywnie obniżki cen usług obliczeniowych (w stosunku do zwyżki cen dóbr produkowanych przez przedsiębiorstwa konsumujące te usługi).

Analizując koszty ośrodków obliczeniowych, można zauważyć, że znaczną pozycję zajmują w nich płace personelu (30—40%) oraz amortyzacja użytkowanego sprzętu (20—40%). Pierwszy składnik nie może raczej ulec zmniejszeniu (płace informatyków nie są zbyt wysokie w porównaniu z innymi grupami zawodowymi); trzeba się dokładniej zatem przyjrzeć drugiemu składnikowi. Jest on —

podstawowym

rem, wykonawczym. Natomiast, *host computer (host machine)* jest komputerem, który wytworzył program wynikowy, a więc — komputerem macierzystym (dla tego programu). To bardzo trafne określenie, które proponujemy używać, pochodzi od H. Orłowskiego (Komputerowe układy automatyki, WNT, Warszawa, 1980). Choć równie często (równie rzadko?) mówi się komputer docelowy (ang. *target*) i komputer gościnny (ang. *host*), trudniej znaleźć uzasadnienie tego sposobu nazywania komputerów.

Patrząc z punktu widzenia kompilatora (a nie, jak dotąd, z punktu widzenia tłumaczonego programu, *host* jest komputerem, na którym wykonuje się kompilację (ang. *on which compiler runs*), a *target* — komputerem, dla którego jest generowany kod wynikowy (ang. *for which code is generated*). Kompilator może działać na różnych komputerach i generować kod wynikowy dla różnymh maszyn wykonawczych. Terminy *host* i *target* dotyczą więc w równym stopniu programu tłumaczącego, jak i tłumaczonego.

Skoro przenoszenie programów może odnosić się zarówno do komputera macierzystego, jak i wykonawczego, to należałoby uwzględnić tę dwoistość i rozszerzyć pojęcie przenośności. W rzeczywistości postępuje się inaczej nazywając przenoszenie w pierwszym przypadku *rehosting*, a w drugim — *retargeting*. Kompilator, który można przenosić z komputera X na komputer Y nazywa się *rehostable*, natomiast kompilator generujący kod wynikowy dla różnych komputerów nazywa się *retargetable*. Kto potrafi znaleźć polskie odpowiedniki dla tych określeń?

Jeśli zapytamy, w jaki sposób możliwa jest zmiana komputera macierzystego (ang. *rehosting*) lub zmiana komputera docelowego (ang. *retargeting*) dla tego samego kompilatora, to napotkamy dalsze trudności terminologiczne.

Taki kompilator składa się z części zwanej *front-end* (na ogół jest to analizator składni), niezależnej od typu komputera wykonawczego, oraz części zwanej *back-end* — niezależnej od języka programowania. Kompilator typu *front-end* tłumaczy program w języku źródłowym na program w języku pośrednim (ang. *intermediate language*), natomiast kompilator typu *back-end* generuje kod wynikowy dla komputera wykonawczego na podstawie programu w języku pośrednim.

Często mówi się także *compiler front-end* i *compiler back-end* dla zaznaczenia, że są to dwie części jednego kompilatora. Pierwszy składnik nazywa się niekiedy *root compiler* (kompilator rdzenny, rdzeń?), a drugi — *code generator*.

Program skompilowany na komputerze macierzystym powinien zostać załadowany do pamięci komputera wykonawczego. Czynność tę określa się jako *downloading* (a nie *loading*) dla podkreślenia, że ładowanie nie następuje do pamięci tego samego komputera. Przyznaję, że nie wiem, jak ten szczegół uwzględnić w polskim odpowiedniku terminu angielskiego.

Podobnie zresztą ma się sprawa z ważnym i nie nowym określeniem, jakim jest tzw. *run-time system* (ang. *run-time package, run-time support*; niem. *Laufzeitsystem*), tj. zespół podprogramów, do których odwołują się wszystkie programy podczas wykonywania.

Trudno jest także nazwać po polsku oprogramowanie określane ogólnym mianem *hosted* (ang. *hosted software*). Przykładowo — program służący do łączenia na komputerze macierzystym programów przetłumaczonych na tym komputerze nazywa się *hosted linker*.

Omawiane pojęcia wkraczają intensywnie do języka polskiego wraz z rozwojem obu wymienionych na początku dziedzin, tj. techniki mikroprocesorowej i nowych technik programowania. Jednakże w większości przypadków odpowiednie nazwy angielskie są trudne do przetłumaczenia na język polski. Wydaje się, że ważną rolę w upowszechnianiu właściwych określeń mogliby odegrać Czytelnicy INFORMATYKI, gdyby zaproponowali polskie terminy.

JANUSZ ZALEWSKI

moim zdaniem — zbyt duży, a ma swe źródło w nienaturalnie wysokich cenach sprzętu informatycznego, jakie stosowane są w naszym kraju. Pomińmy już fakt, że producenci sprzętu żądają często za swoje wyroby tzw. wkładu dewizowego. Same ceny złotówkowe sprzętu są astronomiczne; przedsiębiorstw informatycznych nie stać po prostu na zakup nowych maszyn, co w konsekwencji powoduje systematyczne starzenie się stosowanych urządzeń.

Biorąc pod uwagę dzisiejsze uwarunkowania gospodarki w Polsce, uważam za konieczne rozsądne dotowanie przez państwo silnych ośrodków informatycznych, które powinny stać się motorem napędzającym całą informatykę. Warto się przy tym zastanowić, czy informatyka nie powinna znaleźć się w planach operacyjnych „drugiego rzutu", które gwarantują należyte zainteresowanie ze strony rządu. Potrzebne jest dotowanie rozsądne, tj. takie, które nie będzie hamowało inicjatywy w ośrodkach, lecz przeciwnie — będzie ją inspirowało. Proste dopłacanie do działalności setek ośrodków informatycznych nie byłoby dobrym rozwiązaniem. Lepsze będzie — moim zdaniem — działanie zmniejszające koszty ośrodka, a pozostawiające zasadę swobodnej gry cen i samodzielności przedsiębiorstwa.

Jedyną możliwą do zmniejszenia przez państwo pozycją kosztową jest — powtarzam — amortyzacja. Aby nie pozbawiać bodźców ekonomicznych ani producentów sprzętu, ani jego użytkowników, można by dotować w określonym procencie zakupy sprzętu informatycznego. Można zacząć od ośrodków obliczeniowych, które wykazują się dobrym wykorzystaniem sprzętu. Silne ośrodki nadal umocniałyby swoją pozycję, niemniej w naszych warunkach najbardziej opłacalne mogą być duże centra obliczeniowe, posiadające dobry sprzęt oraz silną kadrę. Ośrodki te powinny być preferowane bez względu na strukturę administracyjną, w której zostały umieszczone. Są wśród nich także ośrodki zakładowe, które potrafią wykazać swą pełną przydatność; w wielu przypadkach nie sprostają one jednak zadaniom prawidłowego wykorzystania dużych maszyn i dlatego będą dążyły do przestawienia się na sprzęt minikomputerowy, powiązany ewentualnie siecią transmisji danych. Tendencja ta powinna wyjść na korzyść zarówno zainteresowanym przedsiębiorstwom, jak i całej polskiej informatyce. A jest to cel, który powinien być dla naszego środowiska najważniejszy.

RYSZARD GRZESIAK
Łódź

Kombinatoryka dla programistów

Kolejna pozycja [1]) z serii Biblioteki Inżynierii Oprogramowania, która ostatnio ukazała się w księgarniach omawia pewne zagadnienia kombinatoryki i teorii grafów. Są to działy, których roli w rozwoju metod informatycznych nie da się przecenić. Tymczasem na krajowym rynku nie była dotychczas dostępna praca o metodach kombinatorycznych pisana z myślą o programistach. Dlatego też pomysł wydania książki systematycznie wykładającej tę dziedzinę nauki jest przysłowiowym „strzałem w dziesiątkę".

Pierwsza część pracy zawiera elementarne wiadomości z zakresu kombinatoryki. Czytelnik obok podstaw matematycznych znajdzie algorytmy generowania permutacji, kombinacji czy podziałów zbioru. Dobór tematyczny jest tu bardzo dobry, choć moim zdaniem brakuje głębszego metodologicznego uzasadnienia dla możliwości stosowania szeregów formalnych. Programista nie zaznajomiony bliżej z podstawami kombinatoryki będzie zapewne zdumiony stosowaniem operacji na szeregach, które nierzadko są rozbieżne i chyba nie zadowoli go krótkie wyjaśnienie zawarte w książce. Twierdzenie Doubileta-Roty-Stanleya uzasadniające ten fakt byłoby według mnie dobrym uzupełnieniem rozdziału o funkcjach tworzących. W tej części pracy odczuć się daje także brak choćby najprostszych formuł inwersyjnych.

Kolejne trzy części książki stanowią wprowadzenie w tematykę algorytmów związanych z grafami. Poruszane tu są problemy o rozmaitej skali trudności, począwszy od maszynowej reprezentacji grafów, a skończywczy na dyskusji zadania maksymalnego przepływu i pewnych jego zastosowań. Czytelnik znajdzie też takie algorytmy, jak przeszukiwanie grafu, znajdowanie drzew rozpinających czy najkrótszych dróg w grafie. Przedstawione zagadnienia są reprezentatywne dla bogatej problematyki związanej z grafami. Ich lektura, warta polecenia zwłaszcza tym programistom, którzy dotychczas nie mieli do czynienia z podobnymi dziedzinami programowania, stanowi doskonałe powtórzenie dla bardziej doświadczonych informatyków.

Znakomita jest ostatnia część książki poświęcona algorytmom zachłannym i matroidom. Algorytmy zachłanne, to pewna klasa algorytmów optymalizacyjnych. Działają one w ten sposób, że na każdym etapie obliczeń wybierają najlepszą z możliwości, która spełnia zadane warunki. Oczywiście taka strategia nie zawsze doprowadza do optymalnego rozwiązania. W książce podane jest twierdzenie Edmondsa-Rado wiążące rozważaną klasę algorytmów z matroidami. Stanowi ono bardzo silne kryterium poprawności stosowania zachłannej strategii w algorytmach optymalizacyjnych. Prostym wnioskiem z tego twierdzenia jest na przykład poprawność algorytmu Kruskala, budującego dla danego grafu drzewo rozpinające o najmniejszym koszcie.

Szkoda, że w książce zabrakło podstawowych funkcji „przeliczających" grafy. Jest to dobrze rozwinięty dział kombinatoryki o dużych zastosowaniach w rozmaitych metodach informatycznych. Pouczające byłyby także oszacowania asymptotyczne dla takich wielkości, jak liczby Stirlinga czy Bella. Wprawdzie załączone są tabele zawierające czasem nawet po kilkadziesiąt z tych liczb, ale zazwyczaj lepsze wyobrażenie o tych wielkościach daje wzór funkcji przybliżającej, podany choćby informacyjnie, bez dowodu.

Podsumowując: uważam, że pomimo wspomnianych, niewielkich zresztą mankamentów, książka jest bardzo dobra. Napisana w interesujący sposób prezentuje przejrzysty tok rozumowania. Czytelna notacja (PASCAL), poparta zwartymi dowodami, znakomicie ułatwia zrozumienie problematyki. Treść każdej z części uzupełniona jest ciekawymi zadaniami stanowiącymi dobre pogłębienie materiału. Myślę, iż praca W. Lipskiego będzie cennym nabytkiem dla każdego programisty.

ANDRZEJ SZAŁAS

[1]) Witold Lipski: Kombinatoryka dla programistów, WNT, Warszawa, 1982, s. 187, cena 120 zł.

Przedpłaty na 1983 r

TYTUŁ CZASOPISMA	Cena 1 egz w zł	PRENUMERATA w zł kwart.	półr.	roczna
Aura, m	50	150	300	600
Budownictwo Okrętowe dm	150	–	450	900
Budownictwo Rolnicze, m	50	150	300	600
Cement-Wapno-Gips, m	100	300	600	1200
Chemik, m	100	300	600	1200
Chłodnictwo, m	100	300	600	1200
Ciepłownictwo-Ogrzewnictwo-Wentylacja, m	75	225	450	900
Doświadczony Mistrz, 2xm	40	240	480	960
Dozór Techniczny, kw	150	–	300	600
Eksploatacja Maszyn, m	80	240	480	960
Elektronika, m	95	285	570	1140
Elektronizacja, m	100	300	600	1200
Energetyka, m	75	225	450	900
Gaz, Woda, Technika Sanitarna, m	65	195	390	780
Gazeta Cukrownicza, m	60	180	360	720
Giełda Rezerw, 2xm	40	240	480	960
Gospodarka Mięsna, m	80	240	480	960
Gospodarka Paliwami i Energią, m	60	180	360	720
Gospodarka Wodna, m	65	195	390	780
Hutnik, m	50	150	300	600
Informatyka, m	75	225	450	900
Inżynieria i Aparatura Chemiczna, dm	150	–	450	900
Inżynieria i Budownictwo, m	75	225	450	900
Inżynieria Materiałowa, dm	150	–	450	900
Inżynieria Morska, dm	150	–	450	900
Koks-Smoła-Gaz, m	40	120	240	480
Maszyny i Ciągniki Rolnicze, m	80	240	480	960
Materiały Budowlane, m	100	300	600	1200
Mechanik, m	80	240	480	960
Nafta, m	50	150	300	600
Nowator – Problemy Wynalazczości i Racjonalizacji, m	95	285	570	1140
Ochrona Powietrza, dm	60	–	180	360
Ochrona Pracy, m	40	120	240	480
Ochrona przed Korozją, m	55	165	330	660
Odzież, m	95	285	570	1140
Opakowanie, dm	150	–	450	900
Poligrafika, dm	150	–	450	900
Pomiary-Automatyka-Kontrola, m	75	225	450	900
Prasa Techniczna, kw	150	–	300	600
Problemy Jakości, dm	150	–	450	900
Przegląd Budowlany, m	80	240	480	960
Przegląd Elektrotechniczny, m	70	210	420	840
Przegląd Gastronomiczny, dm	70	–	210	420
Przegląd Geodezyjny, m	100	300	600	1200
Przegląd Górniczy, m	50	150	300	600
Przegląd Mechaniczny, 2xm	60	360	720	1440
Przegląd Odlewnictwa, dm	150	–	450	900
Przegląd Papierniczy, m	100	300	600	1200
Przegląd Piekarski i Cukierniczy, m	75	225	450	900
Przegląd Skórzany, m	100	300	600	1200
Przegląd Spawalnictwa, m	90	270	540	1080
Przegląd Telekomunikacyjny, m	100	300	600	1200
Przegląd Włókienniczy, m	100	300	600	1200
Przegląd Zbożowo-Młynarski, kw	150	–	300	600
Przemysł Chemiczny, m	100	300	600	1200
Przemysł Drobny i Usługi, m	60	180	360	720
Przemysł Drzewny, m	100	300	600	1200
Przemysł Fermentacyjny i Owocowo-Warzywny, m	80	240	480	960
Przemysł Spożywczy, m	100	300	600	1200
Rudy i Metale Nieżelazne, m	50	150	300	600
Szkło i Ceramika, dm	150	–	450	900
Technik Włókienniczy, m	80	240	480	960
Technika Lotnicza i Astronautyczna, m	60	180	360	720
Technika Motoryzacyjna, m	60	180	360	720
Trybologia, dm	150	–	450	900
Wiadomości Elektrotechniczne, 2xm	60	360	720	1440
Wiadomości Górnicze, m	40	120	240	480
Wiadomości Hutnicze, m	40	120	240	480
Wiadomości Produkcyjne, 2xm	35	210	420	840
Wiadomości Telekomunikacyjne, m	80	240	480	960
Wiadomości Warsztatowe, 2xm	20	120	240	480

Oznaczenie skrótów przy tytułach: 2xm – 2 razy w miesiącu, m – miesięcznik, dm – dwumiesięcznik, kw – kwartalnik

Prenumerata ze zleceniem wysyłki za granicę jest dwukrotnie droższa.

Dodatkowych informacji udziela: Dział Handlowy Wydawnictwa SIGMA – Warszawa, ul. Mazowiecka 12, tel. 26-80-16.

Isotimpex

EC 9004 – URZĄDZENIE DO PRZYGOTOWANIA DANYCH NA TAŚMIE MAGNETYCZNEJ

Urządzenie EC 9004 jest przeznaczone do bezpośredniego zapisu na taśmie magnetycznej informacji wprowadzanych za pośrednictwem klawiatury. Zapewnia ono możliwość wyszukania określonego bloku danych celem jego sprawdzenia i jeśli jest to konieczne – skorygowania zapisanych na taśmie danych.

EC 9004 ma wbudowaną pamięć buforową, w której mieści się cały blok danych do chwili jego zapisania na taśmie magnetycznej. Pozwala to korygować błędy powstające podczas wprowadzania danych natychmiast po ich zauważeniu przez operatora. Odbywa się to przez odwołanie się do określonego adresu pamięci oraz wprowadzenia z klawiatury prawidłowego znaku.

Dane są zapisywane na taśmie magnetycznej w postaci bloków o długości 80 lub 160 znaków. W czasie zapisu na taśmie i następującego bezpośrednio po nim odczytu kontrolnego, klawiatura jest automatycznie blokowana, co pozwala operatorowi zachować stały rytm wprowadzania danych.

Przez czas trwania cyklu zapisu nie ulega zniszczeniu zawartość pamięci buforowej. Zawarte w niej dane przechowywane są w celu sprawdzenia, które przebiega w czasie odczytu realizowanego bezpośrednio po operacji zapisu. Odczytane z taśmy dane są porównywane bit po bicie z danymi zawartymi w pamięci.

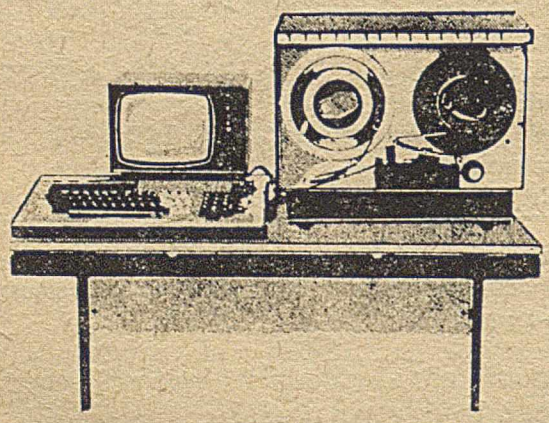
W trybie pracy „KONTROLA" przewidziane do sprawdzenia bloki danych są kolejno odczytywane i wprowadzane do pamięci. Następnie z dokumentu źródłowego wprowadzane są ponownie za pośrednictwem klawiatury kolejne znaki bloku. Kod każdego wprowadzanego znaku jest porównywany automatycznie z kodem znajdującym się już w pamięci. Jeśli są one jednakowe, to operator może kontynuować sprawdzanie.

W trybie pracy „WYSZUKIWANIE" odbywa się automatyczne odszukanie określonego bloku danych drogą porównania każdego zapisanego na taśmie bloku z wprowadzonym uprzednio do pamięci za pośrednictwem klawiatury identyfikatorem. Wyszukiwanie przebiega z szybkością ok. 1000 bloków/min.

W każdym urządzeniu EC 9004 może być zainstalowany wskaźnik, wykazujący kolejne numery bieżąco zapisywanych bloków danych.

Na żądanie zamawiającego urządzenie EC 9004 może realizować następujące dodatkowe operacje:

- przepisywanie danych z jednego urządzenia na inne
- drukowanie zapisanych na taśmie magnetycznej danych
- wprowadzanie danych z czytnika kart dziurkowanych



DANE TECHNICZNE:

● gęstość zapisu	32 bity/mm
● metoda zapisu	NRZ-1
● szybkość przesuwu taśmy	39,6 cm/s
● pojemność pamięci buforowej	160 bajtów
● długość bloku	80 lub 160 bajtów
● format zapisu	wg norm ISO
● podstawowe tryby pracy	wprowadzanie, kontrola i wyszukiwanie danych
● dodatkowe tryby pracy	wprowadzanie programu, kontrola programowana, drukowanie, wprowadzanie z czytnika kart dziurkowanych
● programy	2, działające niezależnie
● wskaźniki	1, rejestrujący zapis kolejnych bloków
● operacje automatyczne	kopiowanie, przeskakiwanie
● zasilanie	220V +10% –15%

Eksporter: PHZ „ISOTIMPEX", ul. Czapajewa 51, Sofia, Bułgaria, tel. 73-61, teleks 022731, 022732

EO 403 K 81-A